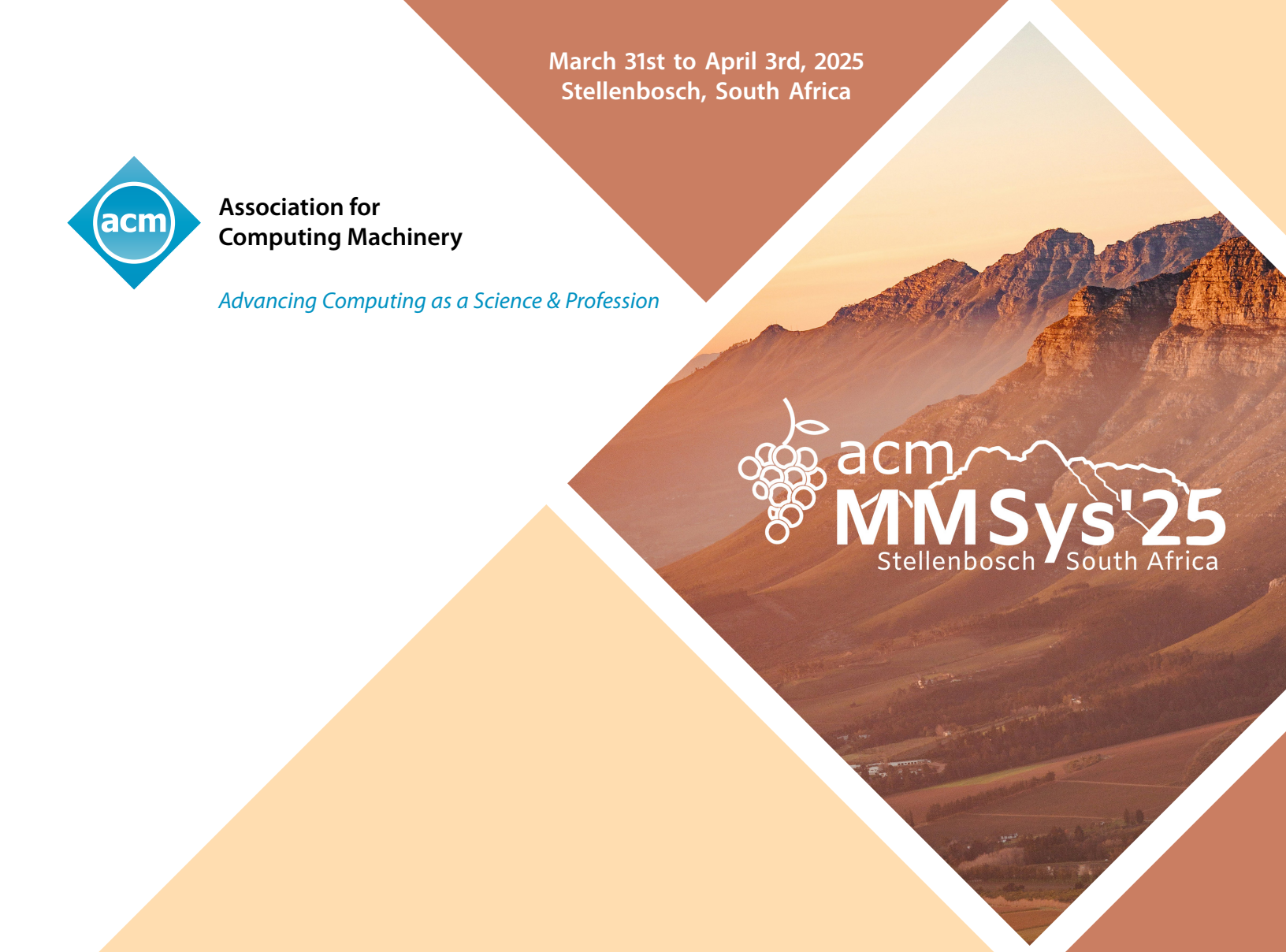
March 31st to April 3rd, 2025
Stellenbosch, South Africa



Association for
Computing Machinery

Advancing Computing as a Science & Profession

MMSys '25

Proceedings of the 16th ACM

Multimedia Systems Conference



Association for
Computing Machinery

Advancing Computing as a Science & Profession

The Association for Computing Machinery
2 Penn Plaza, Suite 701
New York, New York 10121-0701

Copyright © 2025 by the Association for Computing Machinery, Inc. (ACM). Permission to make digital or hard copies of portions of this work for personal or classroom use is granted without fee provided that copies are not made or distributed for profit or commercial advantage and that copies bear this notice and the full citation on the first page. Copyright for components of this work owned by others than ACM must be honored. Abstracting with credit is permitted. To copy otherwise, to republish, to post on servers or to redistribute to lists, requires prior specific permission and/or a fee. Request permission to republish from permissions@acm.org or Fax +1 212 869-0481.

For other copying of articles that carry a code at the bottom of the first or last page, copying is permitted provided that the per-copy fee indicated in the code is paid through www.copyright.com.

Notice to Past Authors of ACM-Published Articles
ACM intends to create a complete electronic archive of all articles and/or other material previously published by ACM. If you have written a work that has been previously published by ACM in any journal or conference proceedings prior to 1978, or any SIG Newsletter at any time, and you do NOT want this work to appear in the ACM Digital Library, please inform permissions@acm.org, stating the title of the work, the author(s), and where and when published.

ISBN: 979-8-4007-1467-2

Additional copies may be ordered prepaid from:
ACM Order Department
PO Box 30777
New York, NY 10087-0777, USA

Phone: +1 800 342-6626 (USA and Canada)
+1 212 626-0500 (Global)
Fax: +1 212 944-1318
Email: acmhelp@acm.org
Hours of Operation: 8:30 am–4:30 pm ET

Foreword

Dear MMSys 2025 Participants,

On behalf of the organizers, we are very pleased to welcome you to the 16th ACM Multimedia Systems Conference, taking place for the first time in Africa, in the university town of Stellenbosch (South Africa).

MMSys is a premier conference dedicated to the exciting and multidisciplinary field of multimedia, with a specific focus on its systems and applications. The conference provides a platform for researchers from both academia and industry to share their latest findings in the multimedia systems research area. Many international researchers, practitioners, engineers, and students from academia, industry, standardization bodies, and government agencies join the MMSys conference each year.

In addition, since its inception, MMSys has been:

- Among the first conferences to emphasize the long-term scientific and practical impact of research by promoting and offering Reproducibility Badges, publishing Open Datasets and Software, and arranging Industrial and Technical Demo Sessions.
- Fostering and supporting young researchers and students by providing mentoring opportunities, arranging Doctoral Symposia, and offering travel grants and discounted registration fees.
- Inviting multiple high-profile keynote speakers from academia and industry to share with the community their research thoughts and latest advancements in the multimedia systems area.
- Hosting multiple high-quality workshops that attract diverse and international research participants.

In continuity with the MMSys conference series and building upon its success, MMSys'25 provides a platform allowing prominent researchers from around the world to meet and interact. Additionally, two social events have been organized: the welcome reception taking place at the Stellenbosch Institute for Advanced Study (STIAS) in Stellenbosch, and the social dinner held at De Warenmarkt, in the central part of Stellenbosch.

MMSys'25 spans five days, featuring an exciting program that includes the MMSys Research Track, the Open Dataset and Software Track, the Technical Demos Track, and the Doctoral Symposium.

Furthermore, the program includes three exciting keynote speeches. The first speech, entitled "*On the Road to Scalable, Interoperable and Cost-Efficient Realistic Holographic Communications*", will be delivered by Mario Montagud Climent, a Consolidated Research Scientist at University of Valencia and the i2CAT Foundation in Spain. The second keynote, entitled "*Art & Tech - A multimedia journey from VR Ndebele painting to*

Creative Robotic performances" will be delivered by Vali Lalioti, the Director of Programmes, Creative Computing Institute (CCI), University of Arts London (UAL) and a renowned expert in human-robot interaction and designing robotic movement with culture in mind. The third keynote, "*Multi-CDN Streaming: Architectures and Optimization Problems*" will be presented by Yuriy Reznik, Technology Fellow and Vice President of Research at Brightcove, Inc, an expert in the field of several multimedia coding and delivery standards. In addition, the technical program contains a special session on Equality, Diversity and Inclusion (EDI). A panel, composed by experts from different industry/private sector/academia related to the topics of the conference, will discuss various EDI issues and how diverse and inclusive are the topics, methodologies and evaluations of the Multimedia Systems community.

The first and final days of MMSys'25 will be dedicated to two focused and engaging workshops. The workshop for the first day is the 17th International Workshop on Immersive Mixed and Virtual Environment Systems (MMVE). The workshop on the final day of MMSys'25 is the long-running 35th ACM SIGMM Workshop on Network and Operating Systems Support for Digital Audio and Video (NOSSDAV). The workshops are held on separate days so that participants can attend both workshops.

We wish to extend our sincere gratitude to all the members of the Organizing Committee and volunteers. The organization of MMSys '25 would not have been possible without your dedication and support!

Last but certainly not least, we wish to thank the generous support of our supporters: platinum supporters Adobe; silver supporters DASH-IF and the Electrical and Electronic Engineering department of Stellenbosch University; bronze supporters Alphawave.

We look forward to hosting you in Stellenbosch and we hope MMSys'25 will provide you with the opportunity for personal exchange of scientific results, facilitate the making of new acquaintances, and strengthen personal friendships among participants from different parts of the world.

Herman Engelbrecht (Stellenbosch University, South Africa)
ACM Multimedia Systems 2025 General Chair

Silvia Rossi (Centrum Wiskunde & Informatica, The Netherlands)
ACM Multimedia Systems 2025 Program Coordinator

A Word from the Technical Program Chairs

We are delighted to announce the technical program for the 16th ACM Multimedia Systems Conference (MMSys'25). While other venues focus on specific aspects related to multimedia systems, MMSys serves as an inclusive and interdisciplinary forum, spanning a broad range of topics related to multimedia systems such as networking, operating systems, real-time systems, databases, mobile computing, distributed systems, computer vision, and middleware communities. This provides a unique opportunity to investigate the intersections and the interplay of the various approaches and solutions developed across these domains.

For the 16th edition, MMSys received 79 research track submissions. Thanks to the hard and valuable work of our international Technical Program Committee (TPC) members, each submission underwent a rigorous double-blind review process and received at least three high-quality reviews. MMSys employs a two-round submission approach, with the first round in September and the second round in November. Authors from the first round had the opportunity to revise their work based on initial reviews before resubmitting in the second round. All submissions and revisions were carefully evaluated based on the review feedback with an emphasis on novelty, quality and relation to the conference. Further, all accepted papers were carefully shepherded before the final camera-ready submission. Ultimately, 19 submissions were accepted (including 2 from the first round), resulting in an acceptance rate of 24%.

In addition to the research track, MMSys'25 hosts three more tracks, which have their own chairs and program committees:

- The open-source software and datasets (ODS) track accepted 13 submissions. The ODS track enables the sharing of data and code to allow others to replicate research results and further advance the respective research field.
- The demo and industry track accepted 8 submissions with the aim to bridge the gap between research and industry by showcasing demonstrations of innovative multimedia systems concepts.
- The doctoral symposium accepted 2 submissions (only 1 with publication) and targets students being enrolled toward a doctoral degree in a field relevant to the themes of MMSys.

True to our scope of interdisciplinary topics on multimedia systems, the program covers a wide range of themes, including research on Virtual Reality and Immersive Experiences, Adaptive Streaming, Quality of Experience, Point Cloud and Volumetric Video Streaming, 3D Scene Representation and Compression, and Advances in Video Streaming and Security.

We thank all authors, the technical program committee, and everyone involved in the submission process. Your efforts in submitting and reviewing papers are greatly appreciated and resulted in a high quality program of ACM MMsys 2025.

We hope that you enjoy the 16th edition of the ACM MMSys conference and that it gives you new insights into the field of multimedia systems as well as many fruitful meetings and interactions with prominent researchers from around the world.

Program Chairs
Simon Gunkel (TNO, The Netherlands)
Silvia Rossi (CWI, The Netherlands)
Pan Hui (Hong Kong University of Science and Technology (Guangzhou), China)

Contents

Organising Committee . . . . . **ix**

Technical Program Committees . . . . . **xi**

Sponsors and Support . . . . . **xv**

360 Video Viewing with Virtual Reality Headsets: Connecting User Head Movements to Intentions **1**
Mohammed Metwaly (*Purdue University*); Alexander J. Quinn

Palantir: Towards Efficient Super Resolution for Ultra-high-definition Live Streaming **12**
Xinqi Jin, Zhui Zhu, Xikai Sun, Fan Dang (*Tsinghua University*); Jiangchuan Liu (*Simon Fraser University*); Jingao Xu (*Carnegie Mellon University*); Kebin Liu, Xinlei Chen, Yunhao Liu (*Tsinghua University*)

Spatial Visibility and Temporal Dynamics: Rethinking Field of View Prediction in Adaptive Point Cloud Video Streaming **24**
Chen Li, Tongyu Zong, Yueyu Hu, Yong Liu, Yao Wang (*New York University*)

StreamWise: An Intelligent Content Steering for DASH **35**
Chidambar Joshi (*Concordia University*); Jashanjot Singh Sidhu (*Concordia Univeristy*); Abdelhak Bentaleb (*Concordia University*)

SGSS: Streaming 6-DoF Navigation of Gaussian Splat Scenes **46**
Mufeng Zhu (*Rutgers University*); Mingju Liu, Cunxi Yu (*University of Maryland, College Park*); Cheng-Hsin Hsu (*National Tsing Hua University*); Yao Liu (*Rutgers University*)

LL-Sparse: Low-Latency 6-DoF Field of View Prediction **57**
Jérémy Ouellette, Abdelhak Bentaleb (*Concordia University*)

Decoupling Video Upscaling from Rendering for Cloud Gaming **68**
Deniz Ugur (*Simon Fraser University*); Ihab Amer (*AMD*); Mohamed Hefeeda (*Simon Fraser University*)

TVMC: Time-Varying Mesh Compression Using Volume-Tracked Reference Meshes **79**
Guodong Chen (*Northeastern University*); Filip Hácha, Libor Váša (*University of West Bohemia*); Mallesham Dasari (*Northeastern University*)

To Cap or not to Cap: Bandwidth Capping Effects on Network Interactions and QoE of Competing Short Video Streams **90**
Nikolas Wehner (*University of Würzburg*); Theo Karagioules, Emir Halepovic (*AT&T Labs - Research*); Filip Simonovski (*University of Augsburg*); Tobias Hossfeld (*University of Würzburg*); Michael Seufert (*University of Augsburg*)

RemoteVIO: Offloading Head Tracking in an End-to-End XR System **101**
Qinjun Jiang, Yihan Pang (*University of Illinois Urbana-Champaign*); William Sentosa (*University of Illinois Urbana Champaign*); Steven Gao, Muhammad Huzaifa, Jeffrey Zhang (*University of Illinois Urbana-Champaign*); Javier Perez-Ramirez (*Ofinno*); Dibakar Das (*Intel Corporation*); David Gonzalez-Aguirre (*Intel Labs*); Brighten Godfrey (*University of Illinois Urbana-Champaign and Broadcom*); Sarita Adve (*University of Illinois Urbana-Champaign*)

Low-Latency Volumetric Video Conferencing in Congested Networks Through L4S 113
Matthias De Fré, Jeroen van der Hooft (*Ghent University - imec*); Chia-Yu Chang, Koen De Schepper, Patrice Rondao Alface, Danny De Vleeschauwer (*Nokia Bell Labs*); Tim Wauters (*Ghent University - imec*); Peter Steenkiste (*CMU*); Filip De Turck (*Ghent University - imec*)

XRgo: Design and Evaluation of Rendering Offload for Low-Power Extended Reality Devices . . 124
Steven Gao, Jeffrey Liu, Qinjun Jiang, Finn Sinclair (*University of Illinois at Urbana-Champaign*); William Sentosa (*University of Illinois at Urbana Champaign*); Brighten Godfrey (*UIUC and Broadcom*); Sarita Adve (*University of Illinois at Urbana-Champaign*)

LTS: A DASH Streaming System for Dynamic Multi-Layer 3D Gaussian Splatting Scenes 136
Yuan-Chun Sun (*National Tsing Hua University*); Yuang Shi (*National University of Singapore*); Cheng-Tse Lee (*National Tsing Hua University*); Mufeng Zhu (*Rutgers University*); Wei Tsang Ooi (*National University of Singapore*); Yao Liu (*Rutgers University*); Chun-Ying Huang (*National Yang Ming Chiao Tung University*); Cheng-Hsin Hsu (*National Tsing Hua University*)

SAILS: A Synchronous Accessible Immersive Online Learning System for Young Learners 148
Yuran Sun, Zhuoying Zhang, Zhenxiao Luo, Alan William Dougherty, Man Ho Yip, Yi King Choi, Chuan Wu (*The University of Hong Kong*)

SAMPL: Self-Attention Modelled Patch Learning for Efficient Visual Understanding 159
Zhiming Hu (*Samsung AI Center, Toronto*); Salar Hosseini Khorasgani (*Tenstorrent*); Weiming Ren, Iqbal Mohomed (*Samsung AI Center, Toronto*)

Enablers of low-latency immersive interaction in future remote-rendered Mixed Reality applications 170
János Dóka, Bálint György Nagy (*Budapest University of Technology and Economics*); Dávid Jocha, Bence Formanek (*Ericsson Research*); Iván Viciedo, Adrian Rodrigo, David Gomez-Barquero (*UPV*); Balázs Sonkoly (*Budapest University of Technology and Economics*)

Accelerating Video Segment Access via Quality-Aware Multi-Source Selection 181
Dominik Winecki, Arnab Nandi (*The Ohio State University*)

Secure the Stream, Not the Hosts: Attribute-Based Encryption for DRM Enabled Video Streaming 190
Mohammad Waquas Usmani (*University of Massachusetts Amherst*); Susmit Shannigrahi (*Tennessee Technological University*); Michael Zink (*University of Massachusetts Amherst*)

COMPACT: Content-aware Multipath Live Video Streaming for Online Classes using Video Tiles . 201
Shubham Chaudhary, Navneet Mishra, Keshav Gambhir, Tanmay Rajore (*IIIT Delhi*); Arani Bhattacharya (*IIIT-Delhi*); Mukulika Maity (*IIIT Delhi*)

eCHFD : extended Ceasefire Hierarchical Firearm Dataset 214
Loubna Lechelek, Sylvie Chambon, Alain Crouzil, Saddam Abdulwahab (*Institut de Recherche en Informatique de Toulouse* (*IRIT*)); Grégory Jalabert, Christian Brocard, Charles-Edouard Coquillard, Laurence Abadie, Bruno Sera (*Service National de Police Scientifique* (*SNPS*) - *Laboratoire de Police Scientifique de Toulouse* (*LPS 31*)); Thierry Hartmann (*Direction de la Coopération Internationale de Sécurité*); Marjorie Le Bras (*Direction Nationale de la Police Judiciaire*)

WIDE-VR: An open-source prototype for web-based VR through adaptive streaming of 6DoF content and viewport prediction 221
May Lim (*National University of Singapore*); Abdelhak Bentaleb (*Concordia University*); Roger Zimmermann (*National University of Singapore*)

HockeyAI: A Multi-Class Ice Hockey Dataset for Object Detection **228**
Mehdi Houshmand Sarkhoosh (*OsloMet, Forzasys*); Sushant Gautam (*SimulaMet, OsloMet*); Cise Midoglu, Saeed Shafiee Sabet, Tomas Kupka (*Forzasys*); Pål Halvorsen (*SimulaMet, OsloMet, Forzasys*)

uvgVPCCenc: Practical Open-Source Encoder for Fast V-PCC Compression **235**
Louis Fréneau, Guillaume Gautier, Alexandre Mercat, Jarno Vanne (*Tampere University*)

OLED-EQ: A Dataset for Assessing Video Quality and Energy Consumption in OLED TVs Across Varying Brightness Levels **242**
Minh Nguyen, Raphael Koch, Alexander Fischer, Moustafa Ghaddar, Görkem Güclü, Martin Lasak, Robert Seeliger, Stefan Arbanowski, Stephan Steglich (*Fraunhofer FOKUS*)

HockeyRink: A Dataset for Precise Ice Hockey Rink Keypoint Mapping and Analytics **249**
Mehdi Houshmand Sarkhoosh (*Forzasys, OsloMet*); Sushant Gautam (*OsloMet, SimulaMet*); Cise Midoglu, Saeed Shafiee Sabet, Tomas Kupka (*Forzasys*); Pål Halvorsen (*SimulaMet, OsloMet, Forzasys*)

VV-DASH: A Framework for Volumetric Video DASH Streaming **256**
Hadi Heidarirad, Mea Wang (*University of Calgary*)

Nagare Media Engine: Towards Self-Adapting MPEG NBMP Multimedia Workflows **263**
Matthias Neugebauer (*University of Münster*)

A Congestion Control Test Suite for Real-Time Communication **270**
Quanwei Zhang, Zhiming Huang, Jinwei Zhao, Jianping Pan (*University of Victoria*)

HockeyOrient: A Dataset for Ice Hockey Player Orientation Classification **277**
Mehdi Houshmand Sarkhoosh (*Forzasys, OsloMet*); Sushant Gautam (*OsloMet, SimulaMet*); Cise Midoglu, Saeed Shafiee Sabet, Tomas Kupka (*Forzasys*); Pål Halvorsen (*SimulaMet, OsloMet, Forzasys*)

PCVD: A Dataset of Point Cloud Video for Dynamic Human Interaction **284**
Jie Li, Shujiao Chen (*Hefei University of Technology*); Qiyue Li (*Heifei University of Technology*); Zhi Liu (*The University of Electro-Communications*)

AMIS: An Audiovisual Dataset for Multimodal XR Research **291**
Abhinav Bhattacharya (*Audiovisual Technology Group, TU Ilmenau, Germany*); Luís Fernando de Souza Cardoso, Andy Schleising (*Virtual Worlds and Digital Games Group, TU Ilmenau, Germany*); Gareth Rendle, Adrian Kreskowski (*Virtual Reality and Visualization Research Group, Bauhaus-Universität Weimar, Germany*); Felix Immohr, Rakesh Rao Ramachandra Rao (*Audiovisual Technology Group, TU Ilmenau, Germany*); Wolfgang Broll (*Virtual Worlds and Digital Games Group, TU Ilmenau, Germany*); Alexander Raake (*Audiovisual Technology Group, TU Ilmenau, Germany*)

MazeLab: A Large-Scale Dynamic Volumetric Point Cloud Video Dataset With User Behavior Traces **298**
Jérémy Ouellette, Jashanjot Singh Sidhu, Abdelhak Bentaleb (*Concordia University*)

Danger Detection and Cloud-Based Vocal Assistance System for Visually Impaired Users Using Meta Quest 3 . . . . . **305**
Fabrizio De Fiore, John Barrett, Niall Murray, Conor Keighrey (*Technological University of the Shannon: Midlands Midwest*)

Real-time Point Cloud Transmission for Immersive Teleoperation of Autonomous Mobile Robots **311**
Nunzio Barone, Walter Brescia, Gabriele Santangelo, Antonio Pio Maggio, Ivan Cisternino, Luca De Cicco, Saverio Mascolo (*Politecnico di Bari*)

Streaming Face-Off: A Testbed Analysis of Media-over-QUIC and Low-Latency DASH **317**
Minh Nguyen, Philip Nys, Stefan Pham, Daniel Silhavy, Stefan Arbanowski, Stephan Steglich (*Fraunhofer FOKUS*)

No code XR-CAx experience creation workflows for conducting Equipment Design Reviews . . . **322**
Sahir Sharma, Conor Keighrey (*Technological University of the Shannon: Midlands Midwest*); James Lardner, Shane Gilligan (*Boston Scientific*); Niall Murray (*Technological University of the Shannon: Midlands Midwest*)

Learned Compression in Adaptive Point Cloud Streaming: Opportunities, Challenges and Limitations **328**
Michael Rudolph (*Lebiniz University Hannover*); Amr Rizk (*Leibniz University Hannover*)

Efficient and Accurate Scene Text Recognition with Cascaded-Transformers **335**
Savas Ozkan, Andrea Maracani, Mete Ozay (*Samsung Research UK*); Hyowon Kim, Sijun Cho, Eunchung Noh, Jeongwon Min, Jung Min Cho (*Samsung Electronics*)

A Multi-CDN Playground for Dash.js: Enabling Integration of CDN Switching Strategies **342**
Jashanjot Singh Sidhu, Chidambar Joshi, Abdelhak Bentaleb (*Concordia University*)

Continual Error Correction on Low-Resource Devices **349**
Kirill Paramonov, Mete Ozay (*Samsung Research UK*); Aristeidis Mystakidis, Nikolaos Tsalikidis, Dimitrios Sotos, Anastasios Drosou, Dimitrios Tzovaras (*CERTH*); Hyunjun Kim, Kiseok Chang, Sangdok Mo, Namwoong Kim, Woojong Yoo, Ji Joong Moon (*Samsung Electronics*); Umberto Michieli (*Samsung Research UK*)

3D Gaussian-based Immersive Media Streaming in Networked Extended Reality **356**
Yuang Shi (*National University of Singapore*)

Author index **361**

Organising Committee

General Chair
Herman Engelbrecht (Stellenbosch University, South Africa)

Program Coordinator
Silvia Rossi (CWI, Netherlands)

Technical Program Committee Chairs
Silvia Rossi (CWI, Netherlands)
Simon Gunkel (TNO, Netherlands)
Pan Hui (Hong Kong University of Science and Technology (Guangzhou), China)

Proceedings Chair
Yao Liu (Rutgers University**,** USA*)*

Publicity Chairs
Raphael Abreu (Federal Fluminense University, Brazil)
Willie Brink (Stellenbosch University, South Africa)

Reproducibility Chair
Jeroen van der Hooft (Ghent University, Belgium)
Nabajeet Barman (Sony Interactive Entertainment, United Kingdom*)*

Open Source Software and Dataset Chairs
Andrew Freeman (Baylor University, USA)
Amr Rizk (Leibniz University Hanover, Germany)

Demo Track Chairs
Müge Sayit (University of Essex, United Kingdom)
Rensu Theart (Stellenbosch University, South Africa)

Doctoral Symposium Chairs
Carsten Griwodz (University of Oslo, Norway)
Cheng-Hsin Hsu (National Tsing Hua University, Taiwan)

Social Media and Website Chairs
Thomas Röggla (CWI, Netherlands)
Conor Keighrey (Technological University of the Shannon, Ireland)
Anna Ferrarotti (Roma Tre University, Italy)

Inclusivity and Diversity Chair
Maria Torres Vega (KU Leuven, Belgium)
Fatuma Simba Ikuja (University of Dar es Salaam**,** Tanzania*)*

Sponsorship and Industry Engagement Chair
Michael Riegler (Simula, Norway)

Awards Chair
Wei-Tsang Ooi (National University of Singapore, Singapore)

MMVE Workshop Chair
Marta Orduna (Nokia XR Lab, Spain)

NOSSDAV Workshop
Xiaofei Wang (Tianjin University, China)
Bo Han (George Mason University, USA)

Technical Program Committees

Research Track

Abdelhak Bentaleb, Concordia University
Ahmad Alhilal, Hong Kong University of Science and Technology
Ahmed H. Zahran, University College Cork
Alan Guedes, University of Reading
Alessandro Floris, University of Cagliari
Alexander Raake, Audiovisual Technology Group - TU Ilmenau
Ali C. Begen, Ozyegin University
Amr Rizk, Leibniz University Hannover
Andrew Freeman, Baylor University
Arvind Narayanan, AT&T Labs - Research
Baochun Li, University of Toronto
Bo Chen, University of Illinois at Urbana-Champaign
Bryan Dunphy, Technological University of the Shannon
Chaminda Hewage, Cardiff Metropolitan University
Cheng-Hsin Hsu, National Tsing Hua University
Chih-Fan Hsu, Inventec Corporation
Christian Timmerer, Alpen-Adria Universität Klagenfurt
Darijo Raca, University of Sarajevo
David K.Y. Yau, Singapore University of Technology and Design
Débora C. Muchaluat-Saade, Fluminense Federal University
Deniz Mevlevioglu, Technological University of the Shannon
Emin Zerman, Mid Sweden University
Emir Halepovic, AT&T Labs - Research
Eoghan Hynes, Software Research Institute
Eun-Seok Ryu, Sungkyunkwan University (SKKU)
Eva Cheng, University of Technology Sydney
Evangelos Alexiou, Xiaomi Technology
Farzad Tashtarian, Alpen-Adria Universität Klagenfurt
Federica Battisti, Università degli Studi di Padova
Filip De Turck, Ghent University - imec
George Ghinea, Brunel University
Guan-Ming Su, Dolby Labs
Gwendal Simon, Synamedia
Haiping Wang, Bytedance
Hans-Jürgen Zepernick, Blekinge Institute of Technology
Herman Engelbrecht, Stellenbosch University
Jean Botev, University of Luxembourg
Jeroen van der Hooft, Ghent University - imec
Jesus Gutierrez, Universidad Politécnica de Madrid
Jiangchuan Liu, Simon Fraser University
Jianping Pan, University of Victoria
Keith Curtis, Technological University of the Shannon

Ketan Mayer-Patel, University of North Carolina
Kjell Brunnström, RISE Research Institutes of Sweden AB
Klara Nahrstedt, University of Illinois at Urbana-Champaign
Laura Toni, University College London
Lea Skorin-Kapov, University of Zagreb
Luca De Cicco, Politecnico di Bari
Luigi Atzori, University of Cagliari
Maha Abdallah, Sorbonne Université
Mallesham Dasari, Northeastern University
Marco Carli, Universita' degli Studi Roma TRE
Mario Montagud Climent, University of Valencia & i2CAT Foundation
Marta Orduna, Nokia XR Lab
Mea Wang, University of Calgary
Michael Seufert, University of Augsburg
Michael Zink, University of Massachusetts Amherst
Mirko Suznjevic, University of Zagreb
Mohamed Hefeeda, Simon Fraser University
Mu Mu, University of Northampton
Müge Sayit, University of Essex
Nabajeet Barman, Sony Interactive Entertainment
Nabil Sarhan, Wayne State University
Niall Murray, Technological University of the Shannon
Niklas Carlsson, Linköping University
Nikolaos Thomos, University of Essex
Ouldooz Baghban Karimi, Simon Fraser University
Ozgu Alay, University of Oslo
Pablo Cesar, Centrum Wiskunde & Informatica
Pablo Perez, Nokia XR Lab
Pan Hui, Hong Kong University of Science and Technology (Guangzhou)
Rakesh Rao Ramachandra Rao, TU Ilmenau
Ramesh Sitaraman, UMass Amherst & Akamai Tech
Raphael Abreu, Fluminense Federal University
Roberto Azevedo, Disney Research
Roger Zimmermann, National University of Singapore
Samira Afzal, Alpen-Adria-Universitat Klagenfurt
Sara Baldoni, Università degli Studi di Padova
Sheng Wei, Rutgers University
Silvia Rossi, Centrum Wiskunde & Informatica
Simon Gunkel, TNO
Simone Porcu, University of Cagliari
Songqing Chen, George Mason University
Stefano Petrangeli, Adobe
Tanja Kojic, Technische Universität Berlin
Thiago Braga, Software Research Institute
Thomas Zinner, NTNU
Venkatesh Kodukula, Amazon

Wei Tsang Ooi, National University of Singapore
Wu-chi Feng, Portland State University
Yao Liu, Rutgers University
Yusuf Sani, University College Cork
Zhisheng Yan, George Mason University

Open Source and Dataset Track

Ahmed Zahran, University College Cork
Aladine Chetouani, University Sorbonne Paris Nord
Alan Guedes, University of Reading
Alessandro Floris, University of Cagliari
Ali C. Begen, Ozyegin University
Amr Rizk, Leibniz University Hannover
Andrew Freeman, Baylor University
Bo Chen, University of Illinois at Urbana-Champaign
Bryan, Dunphy, Technological University of the Shannon
Chih-Fan Hsu, Inventec Inc.
Christian Timmerer, Alpen-Adria Universität Klagenfurt
Darijo Raca, University of Sarajevo
David Yau, Singapore University of Technology and Design
Eun-Seok Ryu, Sungkyunkwan University
Evangelos Alexiou, Xiaomi
Farzad Tashtarian, Alpen-Adria Universität Klagenfurt
Federica Battisti, University of Padova
Gregor Schiele, University of Duisburg-Essen
Hans-Juergen Zepernick, Blekinge Institute of Technology
Herman Engelbrecht, Stellenbosch University
Ionut Schiopu, Huawei Technologies Oy (Finland) Co. Ltd
Jean Botec, University of Luxembourg
Jesus Gutierrez, Universidad Politécnica de Madrid
Jiangchuan Liu, Simon Fraser University
Jianping Pan, University of Victoria
Joshua Ebenezer, Samsung Research America
Keith Curtis, Technological University of the Shannon
Ketan Mayer-Patel, University of North Carolina
Luca De Cicco, Politecnico di Bari
Mallesham Dasari, Northeastern University
Mathis Engelbart, Technical University Munich
Mea Wang, University of Calgary
Michael Rudolph, Lebiniz University Hannover
Michael Seufert, University of Augsburg
Mirko Suznjevic, University of Zagreb
Nabajeet Barman, Sony Interactive Entertainment
Niall Murray, Athlone Institute of Technology

Ouldooz Baghban Karimi, Simon Fraser University
Pablo Cesar, Centrum Wiskunde & Informatica
Rakesh Rao Ramachandra Rao, Technische Universität Ilmenau
Raphael Silva de Abreu, Federal Fulminense University
Reza Farahani, University of Klagenfurt
Samira Afzal, Alpen-Adria-Universitat Klagenfurt
Sara Baldoni, Università degli Studi di Padova
Stefano Petrangeli, Adobe
Takuya Fujihashi, Osaka University
Tanja Kojic, Technische Universität Berlin
Venkatesh Kodukula, Amazon Lab126
Wu-chi Feng, Portland State University
Xiangxu Yu, Washington University at Saint Louis
Yusuf Sani, University College Cork

Demo and Industry Track

Cornelius Hellge, Fraunhofer HHI
Helard Becerra, University College Dublin
Khaled Diab, Hewlett Packard Labs
Micael Seufert, University of Augsburg
Müge Sayit, Ege University
Rensu Theart, Stellenbosch University
Roberto Viola, Fundación Vicomtech
Saba Ahsan, Nokia Corporation
Sejin Oh, Dolby Laboratories Inc.
Sergio Cabrero Barros, Fundación Vicomtech
Silvia Rossi, Centrum Wiskunde & Informatica
Stefano Petrangeli, Adobe
Tobias Hossfeld, University of Wuerzburg
Werner Robitza, ACEQ GmbH

Doctoral Symposium

Carsten Griwodz, University of Oslo
Cheng-Hsin Hsu, National Tsing Hua University

Sponsors and Support

Platinum Support
Adobe



Silver Support
Stellenbosch University



Bronze Support
Alphawave



Technical Sponsors

ACM



SIGMM



360 Video Viewing with Virtual Reality Headsets: Connecting User Head Movements to Intentions

Mohammed Metwaly
mmetwaly@purdue.edu
Purdue University
West Lafayette, Indiana, USA

Alexander J. Quinn
alexanderjquinn@gmail.com
West Lafayette, Indiana, USA

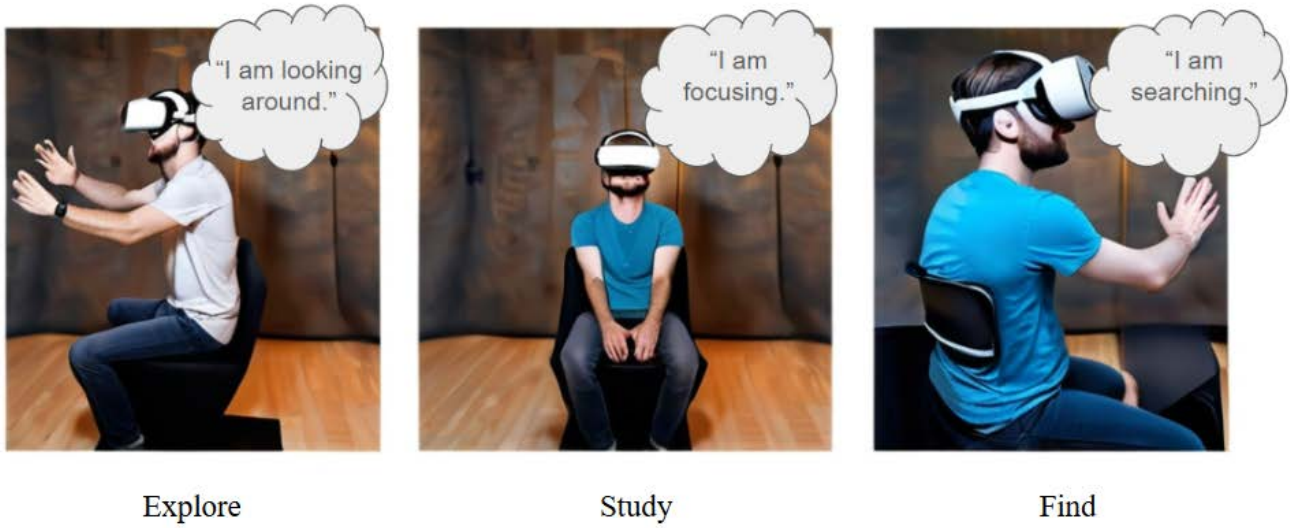


Figure 1: 360 Video Viewing with VR, Connecting User Head Movements to Intentions. Credit: openart

Abstract

Even as virtual reality devices and applications become more pervasive, our understanding of user behavior remains in its infancy. Typical studies involve some combination of post hoc summative evaluation (i.e., questionnaires after an experience is finished) and interaction log analysis (i.e., head position and clicks). Such methods do not illuminate the reasons, intentions, and causality underlying the user's actions. Our research aims to bridge this gap in the context of one specific application area: 360° video viewing. The fundamental question we ask is *why* users pan when they do. We present a study of users (n = 24) viewing 360° video and reporting their intentions each time they move their head to look at something new. By categorizing viewer behavior with detailed annotations, we established a structured framework for understanding viewer engagement. Additionally, we introduced a compiled dataset that merged pan-action data with corresponding user-reported motivations, presenting a valuable asset for subsequent research on VR viewer behavior. Through our approach, we facilitate the comprehension of human interaction within VR contexts, providing a foundational tool for future studies to incorporate viewer intentions with their movements.

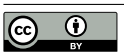
CCS Concepts

• **Human-centered computing** → **Human computer interaction (HCI)**; **Virtual reality**;

Keywords

360 degree video, Human-Computer Interaction, Virtual reality

ACM Reference Format:
Mohammed Metwaly and Alexander J. Quinn. 2025. 360 Video Viewing with Virtual Reality Headsets: Connecting User Head Movements to Intentions. In *The 16th ACM Multimedia Systems Conference (MMSys '25), March 31–April 4, 2025, Stellenbosch, South Africa.* ACM, New York, NY, USA, 11 pages. https://doi.org/10.1145/3712676.3714433



This work is licensed under a Creative Commons Attribution 4.0 International License.
MMSys '25, Stellenbosch, South Africa
© 2025 Copyright held by the owner/author(s).
ACM ISBN 979-8-4007-1467-2/25/03
https://doi.org/10.1145/3712676.3714433

1 Introduction

The growing popularity of platforms like YouTube VR and Meta Quest, which allow for easy sharing and watching of 360° videos, has greatly improved access to these immersive experiences for users. Head-mounted devices (HMDs) enable users to change their viewpoint by simply moving their heads, supporting interactive

exploration of the video environment. The recent introduction of advanced headsets, such as Oculus Quest 3 and Apple Vision Pro, has further fueled interest in 360° videos, increasing the need for their continued development [11].

At its best, 360° video can enable users to interact with the environment in ways that are fundamentally more powerful than conventional video provides. 360° videos offer a panoramic viewing experience, allowing users to navigate visual content in all directions [22]. Ranieri [19] highlights the effectiveness of 360° videos in promoting attention and improving the quality of educational experiences. However, simply replacing conventional video with 360° video does not ensure such value and may simply encumber the user experience by requiring special hardware to view content that could be consumed using a standard computer or mobile device screen with no loss of value. By understanding why people pan—an action uniquely supported by 360°videos where users purposeful head movement–we better understand the value that this medium brings, relative to conventional video. This understanding is crucial to optimizing content delivery, improving user engagement, and tailoring the immersive experience to meet the evolving expectations of the audience.

Despite significant advancements in understanding and modeling human interactions within VR environments, a notable omission remains evident in the existing literature: the explicit consideration of human intentionality behind these interactions. Previous approaches often overlook the subjective intentions that guide users' engagement with VR content, even though human intention can directly affect user movement, as Li et al. [13] claim.

Building on that, it becomes clear that the role of human emotion or intention is crucial, as it shapes user behavior and viewing patterns within VR. Multiple research studies corroborate the assertion that human emotion plays a crucial role in deciphering human behavior and viewing patterns during the consumption of 360° videos. Guimard et al. [9] demonstrate a link between self-reported emotional states and patterns in head and eye gaze movements, in addition to physiological responses. Similarly, Xue et al. [28, 29] identify a strong correlation between the emotional responses of the participants and their behavioral patterns in different videos. Furthermore, their findings suggest that users show consistent viewing behaviors when categorizing human emotions in conjunction with head movements. Zeynali et. al. [30] looks at how user head movements affect video quality-of-experience (QoE). The results underline how important it is to take human behavior into account while improving QoE in 360° video streaming. These studies collectively underscore the significant influence of emotional states on user interaction within VR settings, providing valuable information to enhance the accuracy of predictive modeling in VR applications. Therefore, this encourages a focused investigation into how and why individuals move when viewing 360° videos, aiming to identify and interpret patterns based on their intentions to enhance our comprehension of human behavior within VR settings.

In response, this study aims to investigate intentions that influence user engagement with 360° video content. By incorporating the aspect of intentionality, this research seeks to provide a more holistic understanding of user behavior. We examine how, and more importantly, why viewers move while watching 360° videos by conducting a study where viewers provide real-time reasoning behind their movement.

We address the following research questions.

(1) Can the panning intentions of users while viewing 360° videos be effectively categorized into a discrete set of types?
(2) How do head movement metrics vary across different identified intent categories?
(3) What can intentions reveal about viewer engagement and interaction with 360° content?

In our study, we investigate the motivations behind viewers' head movements while they watch 360° videos, gathering insights from participants' self-reported intentions, and applying thematic analysis. We organize these head movements into categories based on the participants' own explanations. We also assemble a dataset that combines head movements with the participants' rationale for those movements. Our introduced framework facilitates the incorporation of viewer intentions with their movements for future studies.

We identified 19 intention categories (4 intention themes) from user annotations, explained the reasoning behind each category and theme, provided an example of incorporating head movement along with intentions, and described viewer engagement with video content based on intentions. Additionally, we provide a dataset of the collected head movement coupled with the viewer's intentions.

The remainder of the paper is organized as follows: Section 2 reviews related work that has explored user interaction with 360° videos and VR environments. Section 3 describes the study design, including the participant recruitment process, study procedure, and the selection of 360° videos. Section 4 presents the data collection process for head movements and intentions, followed by an analysis of viewer behavior in Section 5. Finally, Sections 6, 7, and 8 conclude with limitations, a discussion of our findings, and potential future research directions.

2 Related Work

Prior to our study, research has been done to study user behavior while watching 360° videos. These studies introduce head and eye movement datasets along with methods for analyzing user behavior and quality of experience while viewing 360° video content. We focus on the available literature that produces subjective and/or objective datasets on human behavior and quality of experience.

Understanding head movement and emotional responses is crucial for analyzing user behavior in 360° videos. Wu et al. [23] introduced one of the first datasets to track head movements, identifying head movement as a key indicator of attention and preferences. Similarly, Corbillon et al. [6] examined how video content influences navigation patterns using head movement data from 59 participants. Another study [21] analyzed head tracking data from 45 participants to understand visual behavior in multiview content, while Xu et al. [24] extended this work by using head and eye movement data from 58 subjects to predict head movements. This paper [5] analyzes human behavior by collecting head movement data to understand cross-user similarities in viewing patterns for 360° videos. Agtzidis et al. [1] present a comprehensive eye movement dataset for applications in the evaluation of subjective experiences such

Table 1: Videos

id	genre	duration (secs.)	YouTube Link	content
1	animals	180	watch?v=88wOoYkjcEo&t=325s	Cats walking around, playing, relaxing, eating, and at times approaching the camera to interact with it
2	thrill	86	watch?v=7pD-9LUYcsU	Car accident interactions between the drivers
3	scenery	49		Park scene during spring with people and sakura flowers
4	technology	180	watch?v=i5RadTUjL1s&t=132s	Drones flying quickly around a gym
5	animals	61	watch?v=Z3FSG0tMKcU	Animal enclosure with camera encased in a glass box
6	sports	78	watch?v=Hcw70by72CY&t=13s	Children practicing soccer guided by a coach in a park
7	scenery	120	watch?v=1ZwIKfeFzIA&t=240s	Riding a motorbike around the streets of Madrid, captured from driver's point of view, with driver describing scenes and objects in view
8	music	150	watch?v=Sube5UBUDx0	Jazz band playing "Fly me to the Moon" in a little bar
9	music	77	watch?v=RIzfHM5uHuA&t=11s	University band and cheer squad performance
10	thrill	180	watch?v=O_gVu8IH6p4&t=54s	Skydiver jumping off a plane and deploying parachute
11	sports	73		Professional soccer match, in the last minute

as content quality and cognitive load. These studies collectively demonstrate the importance of head tracking in understanding user engagement and preferences.

Other research has explored visual quality and user interaction. Zhang et al. [31] introduced a database of 384 videos to explore video quality assessment, showing the impact of loss of video on subjective quality and the improvement of objective quality with the signal-to-noise ratio. This study [17] analyzes human behavior by examining head movement data, including gaze fixation, head orientation, and interaction patterns in 360° videos, to evaluate the quality of experience by understanding how users engage with the content. Hu et al. [12] studied visual attention within VR, claiming that it has often been confined to isolated, single tasks, with limited consideration of how attentional behaviors might vary between different types of activity. Xu et al. [27] assessed user engagement through head and eye movement data from 50 participants, while Fan et al. [8] conducted a study to quantify quality of experience, highlighting the connection between user behavior and their quality of experience.

Integrating physiological data with head and eye movements offers a deeper understanding of user behavior. Guimard et al. [9] presented the PEM360 dataset, combining head and eye movements with emotional evaluations and physiological data, such as heart rate and electrodermal activity, from 31 participants. The findings in [26] underscore the significance of both historical gaze paths and the content of the scene in influencing gaze direction. This comprehensive dataset enables the analysis of the interaction between content, user movement, and emotional responses. Similarly, Xue et al. [28] captured continuous emotional self-reports alongside head and eye movements and physiological responses from 32 participants, confirming the links between emotional states and viewing behavior.

Studies like Li et al. [13] focus on visual quality and behavior. The VQA-OV dataset explores the relationship between visual quality and human behavior, showing that head and eye movement patterns significantly influence subjective video quality assessments. Moreover, alignment adjustments in 360° videos can significantly alter the user experience. Dharmasiri et al. [7] proves that similar viewing patterns are observed for different video contents. Research by [2] measured the effects of alignment on user comfort and presence, finding that adjusting the field of view can affect user attention and head movement speed, ultimately affecting overall quality of experience.

The breadth of research explored in this section underscores the significance of understanding the user's intention to further explain the user behavior while watching 360° videos. It also shows the importance of incorporating human intention in predictive models; however, currently there is no concrete model or method to achieve this. To the best of our knowledge, no study provides self-reported user intentions behind head movement data while watching 360° content. Thus, pushing us to further dive into understanding how– and why– humans move while watching 360° videos.

3 Study

The purpose of our study was two-fold.

First, we sought to understand and categorize the reasons users pan (change head orientation) while viewing 360° videos, across a diverse set of content types. We hope these data may someday inform the design of 360° video viewing interfaces, as well as discussions about where the 360° video has genuine value to users, beyond the initial novelty factor.

Second, the study was designed to produce a data set of panning intentions, aligned with content files and timestamps. We aim to explore the reasons why people pan while watching 360° videos. Our goal was to categorize participants' intentions into distinct groups, making it easier to interpret their actions and understand their movement in VR environments and to acquire a data set aligned with content files and timestamps.

3.1 Phases of study

The study comprised two phases. In the first phase (n=30), participants watched 360° videos and commented verbally each time they moved their head (i.e., a pan), describing the reason for the action.

These verbal annotations were audio-recorded and transcribed, and would form the basis for our thematic analysis.

The second phase followed the same procedure, but with one addition: head movements were recorded using the HMD.

3.2 Ethics

Our study design was approved by our university's Institutional Review Board (IRB). Participants signed an informed consent form, and were compensated $10.00 for 1 hour.

3.3 Participants

Participants (total n=54) were recruited through emails to graduate student lists, and flyers posted on our university campus.

Based on responses to a pre-study questionnaire, there were 24 female and 30 male participants. Regarding age, 10 were aged 18-22, 30 were aged 23-27, 10 were aged 28-33, and one was aged 33-37; two did not answer. Regarding prior experience with VR, 36 reported some prior experience and 17 reported no prior experience; one did not answer. (See Fig. 2.)

3.4 Content selection

Eleven (11) 360° videos (Table 1) were obtained from YouTube VR, a free web site that hosts 360 video.

To ensure that the results of ours study would be generalizable to other 360 video content, we selected eleven videos spanning six genres (content categories) of 360 video found in the @360 channel (a starting point for finding 360 video content on the site). Genres included sports (2 videos), scenery (2 videos), thrill (2 videos), music videos (2 videos), animals (2 videos), and technology (1 video).

Criteria were as follows:

- **Licensed freely.** We only considered videos licensed under the Creative Commons.
- **Monoscopic.** We did not consider stereoscopic video because there was a wider variety of content in the monoscopic format, and including both monoscopic and stereoscopic video would introduce a confounding factor.
- **360° (not 180°)** field of view. Some content is actually 180°; user may rotate the field of view within a half-sphere. For consistency, we limited our search to only 360° videos.
- **Length: 45 to 180 seconds.** We chose this range of video lengths to allow participants enough time to immerse themselves in the video and engage with the content, while also allowing several videos to be viewed within the time constraints of the study.
- **Sound.** All videos have sound.

Videos were recoded using ffmpeg (open source utility) to equirectangular MP4 format. Each video was assigned a numeric ID, which we will refer to as the *video ID*. The videos were viewed using the Oculus Quest 2 headset. To ensure stream quality, headsets were connected to a dedicated wireless access point used exclusively for this study. The video content was served on a dedicated server in a data center on a university campus. [1]

[1]A supplementary table providing the codecs information for each video is provided along with the manuscript.

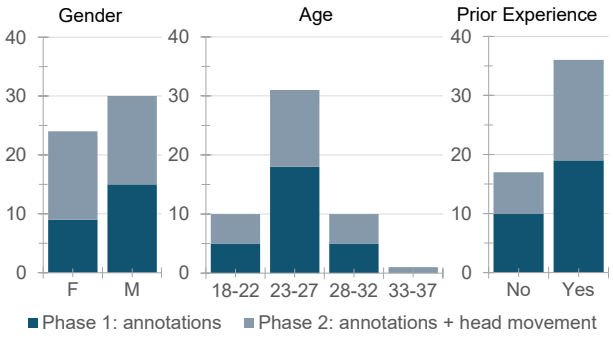


Figure 2: Participant demographics.

3.5 Study Procedure

The study procedure begins with an introduction to the HMD and the controllers, followed by the registration and instructional phases. The participants then complete a simple virtual reality tutorial designed to familiarize them with the system, ensuring that they understand the study mechanics by interacting with buttons using the controllers. Once ready, participants start the videos by selecting the "Enter VR" button, and their head movements are tracked. The entire study session lasts approximately one hour, after which participants complete a post-study questionnaire to provide feedback on their experience.

Research methods, such as post-experience questionnaires or structured interviews, were rejected because we do not believe a participant could accurately remember their reason for moving their head at a particular point.

We used an adaptation of the standard *think-aloud protocol*, in participants vocalize their reasoning in situ (i.e., while viewing the video). Studies of think-aloud protocols have found minimal impact on participants' cognitive load and task efficiency [18]. Gathering intentions as they occur is the recommended method for collecting immediate intentions during video viewing [10, 14].

Participants in our study viewed videos in a fixed sequence (i.e., video 1 to video 11). After each video, they were prompted to click 'next' to proceed. A researcher observed silently, intervening only if the participant fell silent (forgot to think aloud), in which case they were reminded with a gentle noise (ding). That procedure was introduced in the tutorial to ensure familiarity.

4 Head Movement and Intention Data Collection

Data collection mechanisms in prior work are typically platform-dependent and built on software platforms, such as Unity, which require specialized setup.

We developed a Zero-Installation viewer (ZiVO) for 360° videos that can work on a variety of head-mounted displays (HMDs) without installing any new applications. The entire data experimental apparatus—including instructions, 360° viewer, and head movement data acquisition—are implemented as a web application which is accessed from the web browser pre-installed on most HMDs (Fig. 3.

We leverage WebXR (open standard) and three.js (rendering library) to render videos and acquire head orientation data at 10 samples/second and record it in a database on the web server.

ZiVO supports real-time monitoring of the data collection by a researcher through a monitor view (page within the same web application). For example, the researcher can see when a head movement begins with an angular speed (degrees/second) exceeding a configurable threshold.

For our study (phase 2), we used the monitor view to verify that participants were continuing to provide annotations for each pan. Since the final segmentation of head orientation data into discrete panning actions requires some post-processing, for purposes of study monitoring, we configured the monitor view to display head movements exceeding 17 degrees/second, so that the researcher could remind the participant if they neglected to speak an annotation for any such head movements. That threshold (17°) was based on analysis of low resolution head movement data that was collected in phase 1 but not otherwise used. (Following phase 1, we improved the head movement data collection mechanism, and used only head movement data from phase 2 in all other analysis.)

For participants, the entire interface, including the tutorial and 360° video viewing experience, was provided via the web browser within the head-mounted display (HMD).

Textual annotations indicating the reason for each pan (purposeful head movement) were collected by the aforementioned think-aloud protocol with audio recording. Participants were instructed to verbally state a reason each time they moved their head (i.e., to view a different part of the scene).

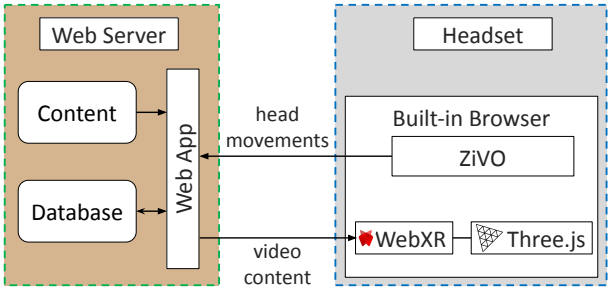


Figure 3: ZiVO Architecture

4.1 Database

Head movement data were sampled at 10 samples/second and recorded in a database on the server. The data collection flow is shown in Fig. 4. Data were stored as *interaction events*, including the video id, a view segment id (tracks pauses during viewing), head orientation (in quaternion and Euler angle formats), time (relative to video start), and the UTC time when data were collected. These data points are processed into pans, which are initiated when a participant's head movement exceeds the predefined threshold of 17 degrees. A pan continues as long as the movement remains above this threshold and concludes when the movement falls below it,

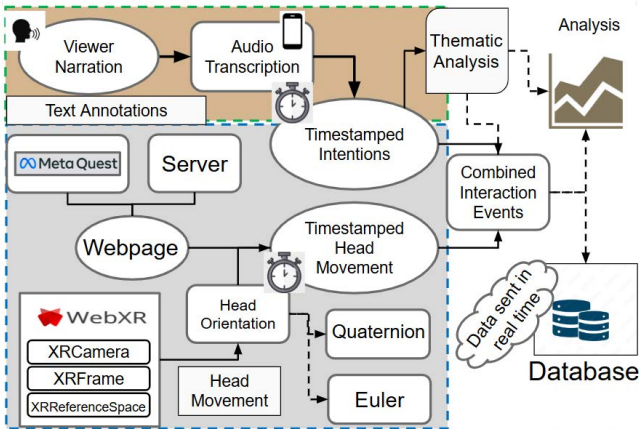


Figure 4: Data Collection Process Flow

at which point the end time (relative to video start) is recorded. Text annotations were transcribed, and timestamps were aligned with head movement data timestamps to associate head movements with annotations. This is achieved by aligning the UTC timestamps, which are recorded with each data point (millisecond precision). Each annotation includes the UTC timestamp of when it was collected, along with the corresponding open code, intention category, and intention theme. Fig. 5 shows a sample record of a pan from the database.

By providing data points that include not only head movements but also the reasoning behind those movements, researchers can gain deeper insights into the motivations driving user interactions in 360° environments. Using this database would prove useful for future studies that aim to examine user behavior, allowing for a more nuanced understanding of viewer behavior. [2]

5 Viewer Intention Analysis

Due to the brevity of the viewer's narrations, typically consisting of only 6-10 words, such as "trying to look around to get the view," NLP models are less effective in fully capturing the nuances of intention. Therefore, a qualitative approach like thematic analysis is more suitable for understanding the reasoning behind their movements. Using a thematic analysis approach, we categorize text annotations to uncover underlying patterns of intentions. This process involves several iterative stages: initial open coding, iterative open coding, axial coding (intention categorization), and finally selective coding (intention themes).

Following the structured tabular inductive and hybrid thematic analysis approach of Robinson [20], which incorporates steps from both (Braun & Clarke) [4] and Boyatzis et al. [3], our study adopts a hybrid design. It initially identified four thematic categories for participant actions and head movements: looking around, following, search, and focus. These a priori themes, taken directly from the annotations of the first 30 trial participants, were used to guide how we looked at the actions of the participants as they watched 360° videos in further studies. However, by further examining the data for the 54 participants (more than 7000 text annotations), more

[2]Data set is included as a supplementary file with this paper.

patterns and themes emerged that were not visible on the surface of the data. Combining both the semantic and latent themes we discovered during our thematic analysis, we have produced 19 intention categories (4 themes) that identify each action taken by a participant while watching 360° videos.

To ensure replicability, we aimed to achieve an inter-rater reliability score. Two researchers collaborated closely following Robinson's [20] process to maintain consistency in the categorization process. Both researchers developed a robust coding framework and followed it to validate the association of intentions with themes. The axial and selective codes of both researchers were compared and refined iteratively until an inter-rater reliability score of 72% was achieved, supporting the reproducibility of our work [15].

5.1 Breaking Down the Annotations

The primary goal of this phase is to identify the main actions of the participant or the context of their activities. Open codes, in this case, consist of an action verb (e.g., looking, watching, following) paired with an item related to the action (e.g., object, scene, sound source). When participants focus on a general area, the code 'scene' is used (e.g. 'looking back to understand scene'). For specific objects, the 'object' is used (for example, 'the parachute enters my vision' would be the 'noticed object'). Annotations without clear intentions, such as 'starting' or 'looking down,' are labeled 'NOT AN ANNOTATION' and excluded from axial coding.


```
{
  "pan_start_time_s": 23.25,
  "pan_end_time_s": 23.935,
  "rotation_data": [
     { "vid_rel_time_s": 23.25,
       "ang_displacement_deg": 4.6988 },
     { "vid_rel_time_s": 23.336,
       "ang_displacement_deg": 6.3598 },
     { "vid_rel_time_s": 23.428,
       "ang_displacement_deg": 6.7887 },
     { "vid_rel_time_s": 23.53,
       "ang_displacement_deg": 6.6534 },
     { "vid_rel_time_s": 23.65,
       "ang_displacement_deg": 6.7839 },
     { "vid_rel_time_s": 23.751,
       "ang_displacement_deg": 5.5236 },
     { "vid_rel_time_s": 23.83,
       "ang_displacement_deg": 2.8946 }
  ],
  "annotations": [
     { "participant_annotation": "make 360 check my
        left and right looks like only one person",
       "open_code": "checking surroundings",
       "intention_category": "looking around",
       "intention_theme": "explore" }
  ],
  "video_id": 10,
  "view_segment_id": "745",
  "participant_id": 86
}
```


Figure 5: Pan Data Sample in JSON

5.2 Intention Categories

Following the open coding stage, we proceeded to axial coding, where we organized the codes into categories based on actions or concepts they represented. This process resulted in the identification of 19 categories. From the open codes, we see if each open code falls under the a priori themes: 'looking around', 'follow', 'search', 'focus'. If the open code does not fall under any of these categories, we create a new category that they can fall under, and then repeat the process again until all open codes were accounted for. Intention categories reflect the main action taken by the participants. We then refined these categories by subdividing broad actions, like 'looking around', into subthemes to distinguish between different intentions, such as 'experiencing' versus 'describing' the scene. Conversely, similar actions like 'staring,' 'watching,' or 'looking at' an object, all with the intention to focus, were consolidated under the 'focus' theme. This categorization demonstrates that participant intentions can be mapped to discrete and recurring patterns. Table 2 lists the intention categories, their definitions, and example annotations. Fig. 6 shows the frequency of the most common intention category and their distribution.

5.3 Intention Themes

Subsequently, we iteratively refined our categorization process, conducting further analysis and adjustments to enhance the coherence and meaningfulness of the categories. Through this iterative approach, we finally arrived at a final set of four distinct themes. From our analysis, we found that the action *Study* is performed 1463 times, the *Explore* action is performed 1744 times, 559 instances of *Find* action were recorded, and finally 286 *Personal* actions were recorded. We can see from these frequencies that the exploration and study actions were the most frequently performed by the participants. Table 3 dives deeper into what each theme encompasses and their respective axial codes or categories.

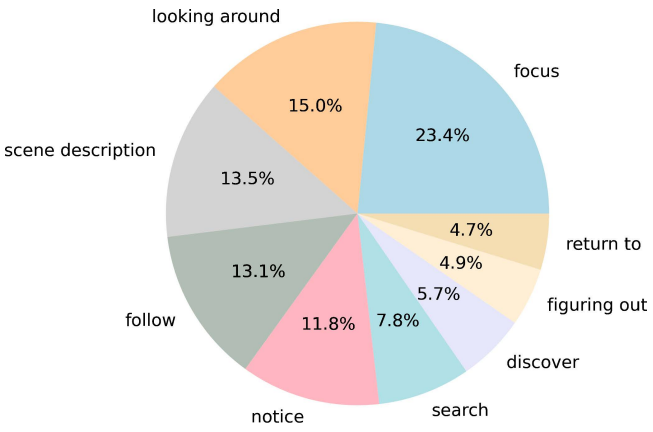


Figure 6: Intention Categories – Frequencies.

Table 2: Intention Categories

Intention Category	Definition	Example Annotation(s)
Avoid	Annotation contains the term "avoid" or expresses an attempt to stop observing a specific object or scene.	"trying not to look at the dude in front of me", "moving to move away from him"
Boredom	Participant expresses disinterest or boredom with elements of the observed scene or object.	"it doesn't seem anything too entertaining in the back', 'got bored of them so i turned"
Curiosity	Annotation contains the term "curious" or expresses curiosity or confusion about a particular scene or object.	"im looking down I was curious about what is below the camera", "what is this dog or i dont know. Im curious about it'
Discover	Participant states that they are looking in some direction or at some scene to survey it, often followed by describing the new scene.	'now im looking at the ceiling to see if there is anything fancy'
Enumerate	Participant counts or attempts to enumerate objects in the scene.	"just see how many people are there", "trying to count how many there are"
Figuring Out	Annotation includes the phrase 'Figure out' or expresses an effort to comprehend the scene or a specific object.	'trying to figure out what is going on', 'trying to figure out whats going on. looking at this weird rodent thing'
Fixing-Position	Annotations depict the participant adjusting their seating position to get more comfortable.	"might have moved because I want to find a more comfortable position on the chair"
Focus	Annotation contains terms such as 'focus', 'watch', 'stare' or 'keep looking at', or reveals the participant's sustained concentration on a specific object or scene.	"just keep looking at the drone", "I am focusing on this black cat"
Follow	Annotation contains the term "follow" or reflects an effort to adjust their visual position to maintain focus on an object.	'I am following the cat', 'following this couple with the dog'
Immersiveness	Annotations incorporate the term 'experience' or exhibit indications of the participant's immersion within the scene, such as expressions of enjoyment or deep engagement with the content.	"im looking at this way because I never experienced that', 'attention is going to seeing a bunch of things to experience them"
Instructions	Annotation shows that the participant listened to the audio of the video and followed the instructions given or adjusted the view according to where the audio points out.	"looking at the plaza to what the sound is bringing attention to", "Now look at the scooter after he brought it up"
Locating Sound	Participant listens for and attempts to locate the source of a sound.	"the piano made me look around for that", "looked behind me so I can see the audience and I can hear them"
Looking Around	Annotations encompass expressions such as "looking around," or when contextual cues suggest that the participant is surveying their immediate environment within the video.	"looking around just the environment and what's around me", "looking around the world."
Notice	Annotation contains the term "notice" or expresses participant's awareness of movement or an object within the scene, especially if the discovery occurred inadvertently or involved objects not initially within the participant's focus prior to their movement initiation.	"oh there are two people below', 'I noticed another dog and I am following it with my head"
Object Proximity	Annotation contains descriptions regarding the spatial relationship of an object in proximity to the participant, explaining their movement towards or away from it.	"looks like it hit me that's why I moved back", "leaning backwards because cats are coming towards me"
Return to	Participant intends to revisit previously observed scenes or objects.	"I will see the people again", "moving to see the people singing again"
Scene-Description	Annotations involve participants providing a descriptive account of the events unfolding within the scene, often without directing their focus toward a specific object.	"I guess it's a bunch of kids at soccer camp', 'looks like I'm in a room. they have like two doors"
Search	If annotations indicate the participant's intent to search for a particular scene or object.	'just looking for another cat to stare at', 'tried to just look for it'
Synchronization	Annotations depict the participant imitating an object's actions observed in the video or directing their gaze in accordance with the camera's orientation.	"crowd of people trying to see what they are looking at", "I am trying to figure out where they are looking"

Table 3: Intention Themes

Intention Themes	Definition	Intention Categories
Explore	The texts involve descriptions of the scene, looking around for objects of interest, being immersed or experiencing the scene, exploring through discovery, understanding the location and surroundings of the scene, stating loss of interest in video and continuing exploration, and general observations without a specific focus on following or studying one particular object.	Notice, Scene Description, Looking Around, Immersiveness, Boredom, Discover
Study	The texts involve the actions of watching, staring, admiring, or focusing on objects or scenes of interest, such as noticing particular details, following objects, or studying certain aspects.	Follow, Focus, Curiosity, Figuring Out
Find	The texts encompass activities such as searching for particular objects within the scene, pinpointing sources of sound, tallying objects, or seeking out specific interactions to gain deeper insights into the overall environment.	Search, Locating Sound, Return to, Enumerate
Personal	The text pertains to movements executed in response to objects or auditory cues within the scene. These movements encompass a variety of actions, such as evading particular scenes or objects, relocating in response to object proximity, adhering to instructions provided within the video , adjusting their seating or posture to get more comfortable, or synchronizing along with the movements of other objects present within the scene.	Avoid, Object Proximity, Synchronization, Instructions, Fixing Position

5.4 Viewer Engagement with Video Content through Intentions

Examining the intentions per video can reveal distinct patterns in participants' behaviors while watching that specific content. By identifying the most frequent intentions, we can expect that other viewers are likely to exhibit similar behaviors when watching the same video or content within a similar genre. For instance, we can see that videos that are scenery-focused, with no specific object of focus, that participants performed more of the 'Explore' movement. This aligns with our expectation that more participants will explore a video with a focus on scenery rather than objects.

Table 4 lists the videos along with the most common intention category (axial code), the most common intention theme (selective code) and a brief description of the video content. We highlight a few notable patterns in the following text. We observed that in sports-themed videos (6 and 11), participants frequently performed 'Explore' actions, specifically 'Notice' movements, indicating a tendency to pan when detecting new objects or motion. This behavior is similar to that seen in animal-themed videos (1 and 5), where the presence of multiple moving objects prompted 'Explore' actions. In 'thrilling' videos (2 and 10), participants mainly exhibited 'Study' intentions, engaging in 'Focus' actions to better understand the content. Scenic videos (3 and 6) elicited predominant 'Looking Around' intentions, supporting the notion that landscape-focused content promotes exploratory behavior. In video 4, which featured multiple objects of interest, 'Study' actions—particularly 'Follow' movements—were the most common, as participants tracked flying objects across the scene. Although ten participants reported motion sickness due to fast-moving objects in the post-study survey, 'Follow' remained the dominant intention category, suggesting that viewers remained highly engaged despite the discomfort. Videos 8 and 9, centered on a band performance, also fell under the 'Explore' theme, however, 'Focus' was the main intention category in video 8.

Table 4: Most Common Intention Category and Theme for Each Video

Video	Intention Category (Axial Code)	Intention Theme (Selective Code)
Video 1	Notice	Explore
Video 2	Focus	Study
Video 3	Looking Around	Explore
Video 4	Follow	Study
Video 5	Notice	Explore
Video 6	Notice	Explore
Video 7	Looking Around	Explore
Video 8	Focus	Explore
Video 9	Notice	Explore
Video 10	Focus	Study
Video 11	Notice	Explore

5.5 Incorporating Head Movement with Intentions

We examine how each intention category and theme is manifested through head movements to understand how metrics vary between them. By analyzing the average angular rotation and speed of pans within each category and theme, we evaluate the intensity of viewers' panning behavior and how each intention is executed.

Table 5 further expands on this analysis by presenting average angular rotations across intention categories. The 'Looking Around' category exhibits the highest average angular rotations, suggesting participants actively surveyed their surroundings while engaging

Table 5: Average Angular Rotation, Standard Deviation, Confidence Interval, and Angular Speed for Each Intention Theme.

Intention Theme	Average Angular Rotation (deg)	Average Angular Speed (deg/s)	Angular Rotation Std.	Angular Rotation CI (95%)
Explore	23.59	5.01	20.86	0.013
Find	20.56	4.92	20.39	0.021
Study	20.23	4.90	20.26	0.014
Personal	19.88	4.29	19.90	0.028

Table 6: Average Angular Rotation, Standard Deviation, Confidence Intervals, and Angular Speed Across Intention Categories.

Intention Category	Average Angular Rotation (deg)	Average Angular Speed (deg/s)	Angular Rotation Std.	Angular Rotation CI (95%)	Intention Theme
Looking Around	26.47	5.40	21.15	0.025	Explore
Scene Description	23.92	5.01	20.86	0.029	Explore
Follow	21.42	4.90	20.37	0.026	Study
Search	20.38	4.62	20.11	0.032	Find
Focus	18.08	4.28	18.80	0.020	Study
Synchronization	17.54	4.03	19.04	0.044	Personal

with exploratory content. 'Scene Description' follows closely, reflecting participants' engagement in describing scenes while exploring. Within the Study theme, the 'Follow' action demonstrates notable average angular rotation and speed, indicating a substantial level of user engagement. Meanwhile, the 'Search' action, under the Find theme, reflects moderate angular rotations, aligning with its goal of identifying specific objects or scenes. The 'Focus' and 'Synchronization' categories exhibit the lowest average angular rotations, suggesting more controlled and deliberate movements during study and personal interaction activities.

Table 6 further expands on this analysis by presenting average angular rotations across intention categories. The 'Looking Around' category exhibits the highest average angular rotations, with 'Search' and 'Follow' closely following. Notably, the Follow action within the Study category demonstrates substantial average angular rotation and speed, suggesting a significant level of user engagement. To assess whether there were statistically significant differences in head movement metrics between intention themes, ANOVA tests were conducted. The results indicated a significant difference between the 'Explore' and 'Study' themes, with participants displaying notably more dynamic head movements during exploratory actions compared to more controlled movements during study-related actions.

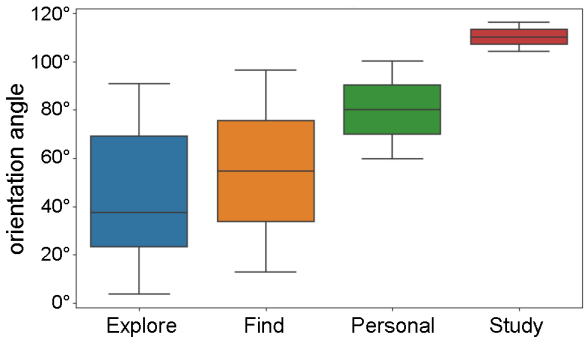


Figure 7: Distribution of Orientation (degrees) Across Intention Themes for One Participant On One Video (Video 1).

Fig. 7 provides a detailed look at the distribution of orientation angles for a single participant viewing a video. The'study' theme shows consistent orientation angles, indicating focused attention, while 'explore' and 'find' display broader ranges, signifying more dynamic head movements. Although this plot represents data from one participant watching one video (video 1), it supports the overall observed trends.

Fig. 8 presents the distribution of angular rotation across intention themes for four participants. Consistent with Tables 5 and 6, the 'explore' theme consistently shows the highest median and widest variation, suggesting dynamic head movements during exploratory actions. Conversely, the 'study' theme exhibits the lowest median and narrowest range, indicating more controlled and focused movements. Our findings underscore that each intention theme provides valuable insight into head movement metrics, such as changes in speed, periods of stillness, and changes in behavior.

This approach aligns with methodologies used in related studies [2, 16, 25], demonstrating how traditional data analysis can be enhanced by incorporating user intentions. Future research could explore alternative methods for analyzing head movements with a focus on user intentions.

6 Limitations

We acknowledge several limitations in this study and propose potential improvements. While streaming videos from a server allowed access across different HMDs, it limited video quality; storing videos locally could enhance visual fidelity and improve the accuracy of

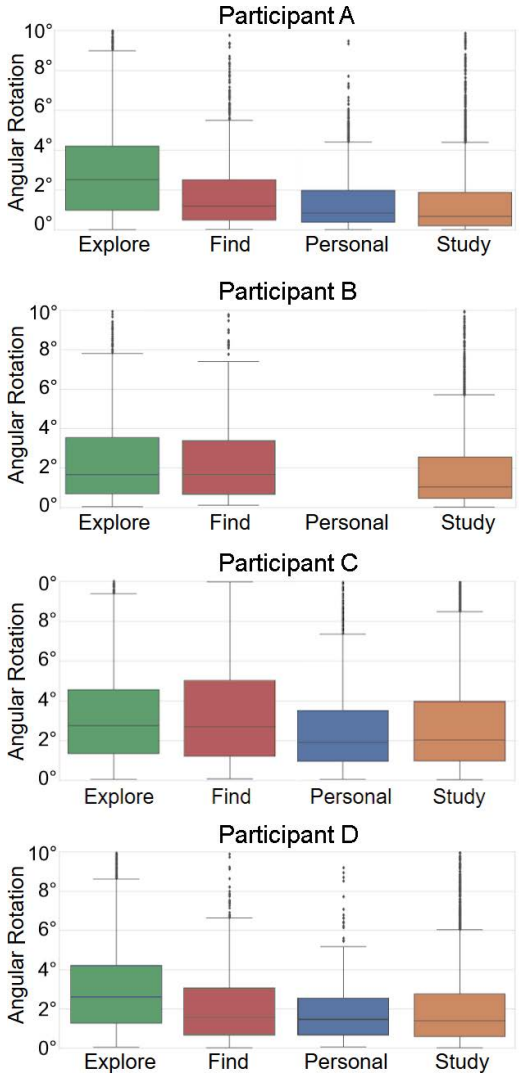


Figure 8: Box-Plot Distribution of Angular Rotation (in Deg.) Across Intention Themes for Four Participants.

self-reported intentions. Furthermore, reliance on a single HMD model may have an impact on generalizability, despite the compatibility of the system with various devices. Head movement served as the primary focus of movement analysis in this dataset, providing a detailed examination of user interactions. Future studies could integrate multimodal data, such as gaze tracking and physiological signals, to capture a more comprehensive interplay of influencing factors. Moreover, environmental factors such as headset fit and susceptibility to motion sickness may have influenced participant head movements and engagement with the videos. Future studies could better control these variables to isolate intentional behaviors more effectively. Lastly, requiring participants to express their intentions may have distracted them from audio cues, potentially affecting their responses. Integrating passive tracking methods with the think-aloud protocol could provide a more comprehensive approach by capturing uninterrupted behavior alongside real-time user insights.

7 Conclusion

Our study explored the behavior of viewers engaging with 360° videos, focusing on the nuances of panning (or purposeful head movements) and the intentions behind them.

From the thematic analysis, we identified 19 distinct intention categories or actions performed by participants while watching our set of videos. We observed consistent patterns in participant behavior both within individual videos and across different video genres. The 19 intention categories are organized into four themes, which broadly explain the reasons for movement: (1) exploring the 360° video, (2) locating objects and scenes, (3) examining objects and scenes, and (4) reacting to the video due to personal interests or physical comfort.

These findings address our first research question (whether pan intentions during 360° video viewing can be categorized). They also illuminate aspects of human behavior in video consumption.

We examined the viewer's engagement with video content through their reported intentions and explored the feasibility of connecting the viewer's movements with their underlying intentions. Variations in head movement metrics, speed and degree of rotation, were observed across different intent categories, with the explore theme displaying the highest speed and degree of rotation.

Additionally, we provide a dataset[3] that combines head movement data with participants' self-reported explanations, offering insights to improve user experience design in VR. This classification serves as an essential tool in improving our understanding of human movement within virtual spaces. By recognizing and categorizing viewers' underlying intentions, we can develop more refined models of head movement classification.

8 Future Work

Future studies could explore the application of machine learning algorithms to predict viewer intentions based on real-time head movement data. This would aid in developing adaptive video streaming systems that adjust content delivery to align with viewers' intentions, enhancing the user experience. Furthermore, insights from this study could inform adaptive streaming algorithms by predicting user focus areas, enabling more efficient bandwidth allocation and improving streaming quality in 360° video systems. Visual aids, such as heatmaps, could provide deeper behavioral insights, while larger, more diverse participant samples would help analyze demographic factors like motion sickness, gender, age, and prior video experience. Moreover, to improve the generalizability of our findings, we propose conducting similar studies with a more extensive set of videos across multiple viewing environments. Additionally, machine learning models could be employed to test whether intention categories hold across a broader dataset. The framework from this study could also guide the design of interactive VR environments, tailoring content and interaction to intention themes for improved learning and entertainment outcomes.

[3]Data set is included as a supplementary file with this manuscript.

Acknowledgments

This work was funded in part by the National Science Foundation (NSF) Award CNS 1956018. Any opinions, findings, conclusions, or recommendations expressed in this material are those of the authors and do not necessarily reflect the views of NSF.

References

[1] Ioannis Agtzidis, Mikhail Startsev, and Michael Dorr. 2019. 360-Degree Video Gaze Behaviour: A Ground-Truth Data Set and a Classification Algorithm for Eye Movements. In *Proceedings of the 27th ACM International Conference on Multimedia* (Nice, France) *(MM '19)*. Association for Computing Machinery, New York, NY, USA, 1007–1015. doi:10.1145/3343031.3350947

[2] Lucas S. Althoff, Alessandro R Silva, Marcelo Carvalho, and Mylene Q. Farias. 2024. 360Align: An Open Dataset and Software for Investigating QoE and Head Motion in 360° Videos with Alignment Edits. In *Proceedings of the 2024 ACM International Conference on Interactive Media Experiences* (Stockholm, Sweden) *(IMX '24)*. Association for Computing Machinery, New York, NY, USA, 41–55. doi:10.1145/3639701.3656311

[3] Richard E Boyatzis. 1998. *Transforming qualitative information: Thematic analysis and code development*. Sage.

[4] Victoria Braun and Victoria Clarke. 2006. Using thematic analysis in psychology. *Qualitative Research in Psychology* 3, 2 (2006), 77–101. doi:10.1191/1478088706qp063oa

[5] Niklas Carlsson and Derek Eager. 2023. Cross-User Similarities in Viewing Behavior for 360° Video and Caching Implications. *ACM Trans. Multimedia Comput. Commun. Appl.* 19, 5, Article 152 (March 2023), 24 pages. doi:10.1145/3507917

[6] Xavier Corbillon, Francesca De Simone, and Gwendal Simon. 2017. 360-Degree Video Head Movement Dataset. In *Proceedings of the 8th ACM on Multimedia Systems Conference* (Taipei, Taiwan) *(MMSys'17)*. Association for Computing Machinery, New York, NY, USA, 199–204. doi:10.1145/3083187.3083215

[7] Amaya Dharmasiri, Chamara Kattadige, Vincent Zhang, and Kanchana Thilakarathna. 2021. Viewport-Aware Dynamic 360 Video Segment Categorization. arXiv:2105.01701 [cs.MM]

[8] Ching-Ling Fan, Tse-Hou Hung, and Cheng-Hsin Hsu. 2022. Modeling the User Experience of Watching 360 Videos with Head-Mounted Displays. *MMSys '17* 18, 1 (2022). doi:10.1145/3463825

[9] Quentin Guimard, Florent Robert, Camille Bauce, Aldric Ducreux, Lucile Sassatelli, Hui-Yin Wu, Marco Winckler, and Auriane Gros. 2022. PEM360: A dataset of 360 videos with continuous Physiological measurements, subjective Emotional ratings and Motion traces. In *MMSys 2022 - 13th ACM Multimedia Systems Conference*. Athlone, Ireland. doi:10.1145/3524273.3532895

[10] Morten Hertzum and Kristin Due Holmegaard. 2013. Thinking aloud in the presence of interruptions and time constraints. *International Journal of Human-Computer Interaction* 29, 5 (2013), 351–364.

[11] Amit Hirway, Yuansong Qiao, and Niall Murray. 2024. A Quality of Experience and Visual Attention Evaluation for 360° Videos with Non-spatial and Spatial Audio. *ACM Trans. Multimedia Comput. Commun. Appl.* 20, 9, Article 271 (Aug. 2024), 20 pages. doi:10.1145/3650208

[12] Zhiming Hu, Andreas Bulling, Sheng Li, and Guoping Wang. 2023. EHTask: Recognizing User Tasks From Eye and Head Movements in Immersive Virtual Reality. *IEEE Transactions on Visualization and Computer Graphics* 29, 4 (2023), 1992–2004. doi:10.1109/TVCG.2021.3138902

[13] Chen Li, Mai Xu, Xinzhe Du, and Zulin Wang. 2018. Bridge the Gap Between VQA and Human Behavior on Omnidirectional Video: A Large-Scale Dataset and a Deep Learning Model. In *Proceedings of the 26th ACM International Conference on Multimedia* (Seoul, Republic of Korea) *(MM '18)*. Association for Computing Machinery, New York, NY, USA, 932–940. doi:10.1145/3240508.3240581

[14] Esteban Segarra Martinez, Ayesha A. Malik, and Ryan P. McMahan. 2024. CLOVR: Collecting and Logging OpenVR Data from SteamVR Applications. In *2024 IEEE Conference on Virtual Reality and 3D User Interfaces Abstracts and Workshops (VRW)*. 485–492. doi:10.1109/VRW62533.2024.00095

[15] Nora McDonald, Sarita Schoenebeck, and Andrea Forte. 2019. Reliability and inter-rater reliability in qualitative research: Norms and guidelines for CSCW and HCI practice. *Proceedings of the ACM on human-computer interaction* 3, CSCW (2019), 1–23.

[16] Afshin Taghavi Nasrabadi, Aliehsan Samiei, and Ravi Prakash. 2020. Viewport Prediction for 360 Videos: A Clustering Approach *(NOSSDAV '20)*. Association for Computing Machinery, New York, NY, USA, 34–39. doi:10.1145/3386290.3396934

[17] Anh Nguyen and Zhisheng Yan. 2023. Enhancing 360 Video Streaming through Salient Content in Head-Mounted Displays. *Sensors* 23, 8 (2023). doi:10.3390/s23084016

[18] Erica L. Olmsted-Hawala, Elizabeth D. Murphy, Sam Hawala, and Kathleen T. Ashenfelter. 2010. Think-aloud protocols: a comparison of three think-aloud protocols for use in testing data-dissemination web sites for usability. In *Proceedings of the SIGCHI Conference on Human Factors in Computing Systems* (Atlanta, Georgia, USA) *(CHI '10)*. Association for Computing Machinery, New York, NY, USA, 2381–2390. doi:10.1145/1753326.1753685

[19] M. Ranieri, D. Luzzi, and S. Cuomo. 2022. If and how do 360° videos fit into education settings? results from a scoping review of empirical research. *Journal of Computer Assisted Learning* 38 (2022), 1199–1219. Issue 5. doi:10.1111/jcal.12683

[20] Oliver C Robinson. 2022. Conducting thematic analysis on brief texts: The structured tabular approach. *Qualitative Psychology* 9, 2 (2022), 194.

[21] Tiago Soares da Costa, Maria Teresa Andrade, Paula Viana, and Nuno Castro Silva. 2023. A Dataset for User Visual Behaviour with Multi-View Video Content. In *Proceedings of the 14th ACM Multimedia Systems Conference* (Vancouver, BC, Canada) *(MMSys '23)*. Association for Computing Machinery, New York, NY, USA, 437–443. doi:10.1145/3587819.3592556

[22] Maria Grazia Violante, Enrico Vezzetti, and Pietro Piazzolla. 2019. Interactive virtual technologies in engineering education: Why not 360 videos? *International Journal on Interactive Design and Manufacturing (IJIDeM)* 13, 2 (2019), 729–742.

[23] Chenglei Wu, Zhihao Tan, Zhi Wang, and Shiqiang Yang. 2017. A Dataset for Exploring User Behaviors in VR Spherical Video Streaming. Association for Computing Machinery, New York, NY, USA. doi:10.1145/3083187.3083210

[24] Mai Xu, Yuhang Song, Jianyi Wang, Minglang Qiao, Liangyu Huo, and Zulin Wang. 2019. Predicting Head Movement in Panoramic Video: A Deep Reinforcement Learning Approach. *IEEE Transactions on Pattern Analysis and Machine Intelligence* 41, 11 (2019), 2693–2708. doi:10.1109/TPAMI.2018.2858783

[25] Ruisi Xu, Chenyu Liu, Mengqi Hu, Size Qian, Yuan Zhang, and Tao Lin. 2024. OMMS: Multiple Control based Adaptive 360° Video Streaming. In *Proceedings of the 15th ACM Multimedia Systems Conference* (Bari, Italy) *(MMSys '24)*. Association for Computing Machinery, New York, NY, USA, 429–434. doi:10.1145/3625468.3653435

[26] Yanyu Xu, Yanbing Dong, Junru Wu, Zhengzhong Sun, Zhiru Shi, Jingyi Yu, and Shenghua Gao. 2018. Gaze Prediction in Dynamic 360 Immersive Videos. In *Proceedings of the IEEE Conference on Computer Vision and Pattern Recognition (CVPR)*.

[27] Yutong Xu, Junhao Du, Jiahe Wang, Yuwei Ning, Sihan Zhou, and Yang Cao. 2024. Panonut360: A Head and Eye Tracking Dataset for Panoramic Video. In *Proceedings of the 15th ACM Multimedia Systems Conference* (Bari, Italy) *(MMSys '24)*. Association for Computing Machinery, New York, NY, USA, 319–325. doi:10.1145/3625468.3652176

[28] Tong Xue, Abdallah El Ali, Tianyi Zhang, Gangyi Ding, and Pablo Cesar. 2023. CEAP-360VR: A Continuous Physiological and Behavioral Emotion Annotation Dataset for 360 VR Videos. *IEEE Transactions on Multimedia* 25 (2023), 243–255. doi:10.1109/TMM.2021.3124080

[29] Tong Xue, Abdallah El Ali, Tianyi Zhang, Gangyi Ding, and Pablo Cesar. 2021. RCEA-360VR: Real-time, Continuous Emotion Annotation in 360 VR Videos for Collecting Precise Viewport-dependent Ground Truth Labels. In *Proceedings of the 2021 CHI Conference on Human Factors in Computing Systems* (, Yokohama, Japan,) *(CHI '21)*. Association for Computing Machinery, New York, NY, USA, Article 513, 15 pages. doi:10.1145/3411764.3445487

[30] Ali Zeynali, Mohammad H. Hajiesmaili, and Ramesh K. Sitaraman. 2024. BOLA360: Near-optimal View and Bitrate Adaptation for 360-degree Video Streaming. In *Proceedings of the 15th ACM Multimedia Systems Conference* (Bari, Italy) *(MMSys '24)*. Association for Computing Machinery, New York, NY, USA, 12–22. doi:10.1145/3625468.3647607

[31] Bo Zhang, Junzhe Zhao, Shu Yang, Yang Zhang, Jing Wang, and Zesong Fei. 2017. Subjective and objective quality assessment of panoramic videos in virtual reality environments. 163–168. doi:10.1109/ICMEW.2017.8026226

Received 20 February 2007; revised 12 March 2009; accepted 5 June 2009

Palantír: Towards Efficient Super Resolution for Ultra-high-definition Live Streaming

Xinqi Jin*
School of Software & BNRist, Tsinghua University
jinxq21@mails.tsinghua.edu.cn

Zhui Zhu*
Department of Automation & BNRist, Tsinghua University
z-zhu22@mails.tsinghua.edu.cn

Xikai Sun
Department of Automation & BNRist, Tsinghua University
sxk23@mails.tsinghua.edu.cn

Fan Dang
School of Software Engineering, Beijing Jiaotong University
dangfan@bjtu.edu.cn

Jiangchuan Liu
School of Computing Science, Simon Fraser University
jcliu@sfu.ca

Jingao Xu
Computer Science Department, Carnegie Mellon University
jingaox@andrew.cmu.edu

Kebin Liu
Global Innovation Exchange, Tsinghua University
kebinliu2021@tsinghua.edu.cn

Xinlei Chen
Shenzhen International Graduate School, Tsinghua University
chen.xinlei@sz.tsinghua.edu.cn

Yunhao Liu†
Global Innovation Exchange & Department of Automation & BNRist, Tsinghua University
yunhao@tsinghua.edu.cn

Abstract

Neural enhancement through super-resolution (SR) deep neural networks (DNNs) opens up new possibilities for ultra-high-definition (UHD) live streaming. Yet, the heavy SR DNN inference overhead leads to severe deployment challenges. To reduce the overhead, existing systems propose to apply DNN-based SR only on carefully selected anchor frames while upscaling non-anchor frames via the lightweight reusing-based SR approach. However, frame-level scheduling is coarse-grained and fails to deliver optimal efficiency. In this work, we propose Palantír, the first neural-enhanced UHD live streaming system with fine-grained patch-level scheduling.

At the core of Palantír is its SR video quality estimation strategy which guides the low-delay selection of the most beneficial anchor patches. Although existing systems propose estimation strategies for anchor frame selection, these strategies heavily rely on empirical insights that cannot be transferred to our context, making fine-grained scheduling a challenging problem that requires a fundamentally new solution. Facing the challenge, we follow the first-principles approach and derive a directed acyclic graph (DAG) model to address the problem. The model can also be generalized to various settings due to its first-principles nature. Compared to the state-of-the-art real-time frame-level scheduling strategy for live streaming, Palantír reduces the anchor size by 80.1% at most and 38.4% on average without compromising the quality gain. Furthermore, Palantír incurs a scheduling latency accounting for only 0.6-3.9% of the end-to-end latency requirement for UHD live streaming.

CCS Concepts

• **Information systems** → **Multimedia streaming**; • **Computing methodologies** → *Computer vision.*

Keywords

Video streaming, super-resolution, video codec

ACM Reference Format:
Xinqi Jin, Zhui Zhu, Xikai Sun, Fan Dang, Jiangchuan Liu, Jingao Xu, Kebin Liu, Xinlei Chen, and Yunhao Liu. 2025. Palantír: Towards Efficient Super Resolution for Ultra-high-definition Live Streaming. In *ACM Multimedia Systems Conference 2025 (MMSys '25), March 31-April 4, 2025, Stellenbosch, South Africa.* ACM, New York, NY, USA, 12 pages. https://doi.org/10.1145/3712676.3714434

1 Introduction

UHD videos such as 4K and 8K videos are expected to form a huge market worth more than $1 trillion in the following few years [17]. More and more UHD live-streaming applications [14, 16, 24, 25] are deployed to revolutionize many aspects of our society. However, the bitrates of 4K videos (as large as 45Mbps [20]) can be significantly larger than the worldwide average uplink bandwidth of mobile broadband networks (about 11Mbps [36]), which poses great deployment challenges. Although the latest codecs such as AV1 and VVC can provide a lower bitrate, they can be rather computation-intensive and typically require specific hardware encoders to achieve real-time UHD encoding [38], which are still unavailable on many commercial devices [23]. Using fixed broadband networks for the uplink is an alternative solution, but it inevitably affects mobility and prohibits many mobile applications [10–13, 37].

Recently, neural enhancement has been proposed [8, 46, 48] and deployed [15, 27, 34, 50] to improve video streaming. It can potentially boost the broad deployment of UHD live streaming by

*Co-primary authors.
†Corresponding author.
Project repository: **https://palantir-sr.github.io**



This work is licensed under a Creative Commons Attribution-NonCommercial-ShareAlike 4.0 International License.
MMSys '25, Stellenbosch, South Africa
© 2025 Copyright held by the owner/author(s).
ACM ISBN 979-8-4007-1467-2/2025/03
https://doi.org/10.1145/3712676.3714434

allowing the streaming source to stream only a low-bitrate low-resolution (LR) video over the bandwidth-limited uplink and using a super-resolution (SR) deep neural network (DNN) to upscale the LR stream to its SR counterpart later. However, as detailed in Sec. 2.1, SR DNN inference incurs heavy computation overhead and deployment challenges. Therefore, many research efforts have been aimed at optimizing the overhead. NEMO [46] and NeuroScaler [48] achieve this by categorizing frames into anchor frames and non-anchor frames: only anchor frames undergo computation-intensive DNN-based SR, while non-anchor frames are reconstructed via reusing the SR results of reference frames. Scheduling has been a central aspect of designing such systems - the most beneficial anchor frames must be carefully selected to deliver a large quality gain with a low inference overhead. Nevertheless, these systems are inherently sub-optimal due to their coarse scheduling granularity. A beneficial anchor frame may contain some existing objects that can be well reconstructed by reusing-based SR, and vice versa.

In this paper, we propose Palantír, **the first patch-level neural-enhanced UHD live streaming system** that selects the appropriate type — anchor or non-anchor — for each patch (part of a single frame). Palantír aims to meet two design goals. The first is to accurately pinpoint the most beneficial anchor patches so that a smaller overhead can be achieved without sacrificing the quality gains. The second is to minimize the scheduling latency to better support many latency-sensitive live streaming applications [1, 16, 28, 41, 44].

Limitations of existing methods (Sec. 3). A natural idea of fulfilling fine-grained scheduling is to adapt existing frame-level solutions. However, as shown in Sec. 3, **existing frame-level solutions all lack a deep understanding of how DNN-based and reusing-based SR affect the SR quality.** Their success in frame-level scheduling heavily relies on empirical insights that either require non-realtime operations or only make sense in the limited context of frame-level scheduling. Therefore, an entirely different scheduling method is required to meet our two design goals.

Palantír: a paradigm shift. (Sec. 5). Instead of using trial and error to seek another insight suitable for the fine-grained context, which may again face the problem of limited generalizability in other contexts, we **adopt a first-principles approach**. A first-principles approach involves breaking down complex problems into their most fundamental elements and building solutions from the ground up, rather than relying on empirical insights. In our case, we systematically analyze the SR process of every basic coding block to understand how coding types and anchor configurations affect the SR error (which is inversely related to the quality gains of SR). We then naturally conclude from our pioneering analysis that the process can be simulated accurately and quickly through a DAG structure at the patch-level granularity. This analytical model allows us to efficiently and accurately estimate the SR error for any given anchor patch set and lays a rigorous foundation for the design of Palantír towards the two aforementioned goals. As a final note, **the first-principles DAG structure essentially reflects the inter-frame coding strategy, which is at the core of all mainstream codecs, and thus shows unprecedented generalizability to various codecs, coding settings, and video types.**

In addition to the paradigm shift, Palantír further incorporates two novel techniques to tackle some specific challenges and ultimately meet the two design goals (**Sec. 6**). *First*, strictly following our analytical framework to construct the DAG requires information from the high-resolution (HR) video. We expect the scheduling algorithm of Palantír to be executed at the media server for easy deployment, yet the media server often does not have access to the HR video. We overcome this challenge by approximating missing HR information with the available LR data. *Second*, we find that a naive implementation of DAG-based estimation still fails the second goal. Alternatively, we utilize a universal characteristic of all mainstream codecs to introduce two parallelism mechanisms in the DAG-based computation process, which reduce the latency by hundreds of times without changing the anchor selection results.

Results. We conduct extensive evaluations to demonstrate that Palantír meets the two design goals. When compared to the naive method of applying DNN-based SR on all the frames, Palantír can **reduce the SR DNN inference overhead by 20 times (or 60 times) while preserving 54.0-82.6% (or 32.8-64.0%) of the quality gain**. When compared to the state-of-the-art real-time frame-level scheduling strategy, Palantír can **reduce the SR DNN inference overhead by 80.1% at most (and 38.4% on average) without sacrificing the video quality.** Palantír incurs a negligible scheduling latency accounting for only 0.6-3.9% of the end-to-end latency requirement.

The rest of the paper is structured as follows. Sec. 2 reviews preliminary knowledge. We detail the core limitations of existing methods in Sec. 3. A high-level overview of Palantír is outlined in Sec. 4. We introduce our first-principles approach in Sec. 5. The two practical techniques used to build Palantír are presented in Sec. 6. We evaluate Palantír in Sec. 7. The paper is concluded in Sec. 8. Discussions on future work, related work, and some technical details can be found in the Appendix.

2 Background

2.1 Primer on SR Streaming

Currently, there are two common deployment models for SR enhancement. One is to execute the SR DNN inference on the mobile streaming clients [15, 27, 34, 50], and the other is to execute the SR DNN on a cloud server to benefit many audiences for the same video [48]. Yet, SR DNNs are of high computation complexity and lead to severe deployment challenges in both deployment models. In the first model, the battery life of mobile clients can be easily drained. Consequently, Microsoft Edge VSR disables its SR feature when the device is not being charged [34]. As for the second model, the heavy inference overhead appears in the form of the high monetary cost of using cloud servers (estimated to be at least $1.690 per hour per 4K stream [48]). Lowering the overhead is essential to a broader application of SR enhancement.

2.2 Reusing-based SR

To enable low-cost SR enhancement, researchers have built the NEMO [46] system, where an SR-enhanced decoder is adopted to reduce the cost by using video temporal redundancy. In the SR decoder, video frames are categorically divided: anchor frames are upscaled via DNN-based SR, while non-anchor frames are quickly upscaled via reusing-based SR. To appreciate the intricacies of reusing, a basic understanding of video codecs is essential. Video coding predominantly encodes a block through inter-coding; it

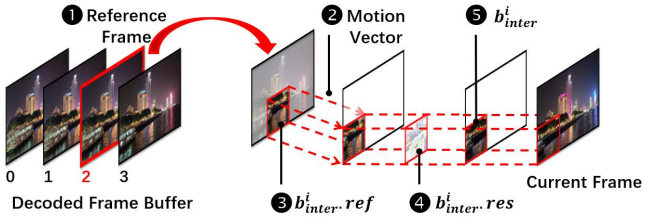


Figure 1: Video decoding pipeline.

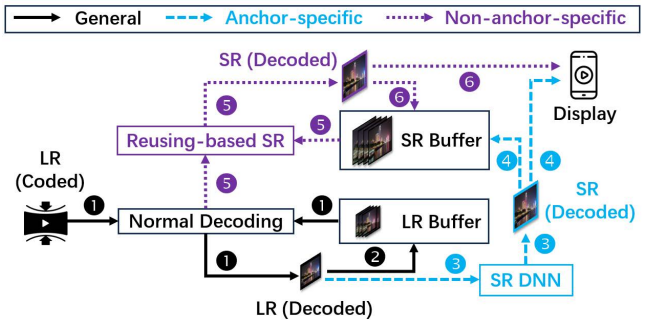


Figure 2: SR-integrated decoder overview.

identifies a similar reference block from a prior frame and only stores the subtle residual between the two blocks. A reference index is retained in the coded video to identify the frame containing the reference block or the reference frame, while a motion vector is stored to represent the potential spatial offset between the current and reference blocks. To decode an inter-coded block b^i_{inter} (*i.e.,* ❺ in Fig. 1), the decoder first parses the reference index to determine its reference frame (❶) and then parses the motion vector (❷) to determine its reference block $b^i_{inter}.ref$ (❸). The residual $b^i_{inter}.res$ (❹) is added to $b^i_{inter}.ref$ to obtain b^i_{inter}, *i.e.,*

$$b^i_{inter} = b^i_{inter}.ref + b^i_{inter}.res. \tag{1}$$

An alternative to inter-coding is intra-coding, which, while similar, encodes an intra-coded block b^i_{intra} by storing the intra-frame residual. We refer the readers to the technical specifications [21] for further details. Note that patches are not the same as encoding blocks in this paper: patches are scheduling units while blocks are encoding units.

The SR decoder in NEMO [46] is developed based on the open-source Google libvpx VP9 decoder [4]. As shown in Fig. 2, the SR decoder first decodes a frame into its LR version by referring to decoded frames in the LR buffer (❶) and can insert it into the LR buffer for future frames (❷), just as a standard decoder. Along with the LR video, a cache profile is also downloaded, each bit of which indicates whether a frame is an anchor or non-anchor. If the current frame is an anchor, the LR version will be fed into the SR DNN to obtain the SR version (❸), which may be inserted into the SR buffer for future reuse (❹). Otherwise, the SR version is obtained via reusing-based SR (❺). The reusing-based SR uses a process similar to that in Fig. 1 to decode every inter-coded block b^i_{inter} to its SR version, except that the motion vector is scaled (*e.g.,* by 4 times, when the LR and SR frames are 240p and 960p, respectively), the residual is upscaled by bilinear interpolation to match the resolution of the SR block ($b^i_{inter}.SR$), and the same reference index is used to fetch cached frames from the SR buffer rather than the LR buffer. We can formulate the process as

$$b^i_{inter}.SR = b^i_{inter}.ref.SR + interp(b^i_{inter}.res, scale), \tag{2}$$

where *scale* is the SR ratio and *interp* is the bilinear interpolation operation. We also rewrite Eq. (1) as

$$b^i_{inter}.LR = b^i_{inter}.ref.LR + b^i_{inter}.res \tag{3}$$

to distinguish between the LR and SR versions of the same block in the SR decoder. As for any intra-coded block b^i_{intra} in the non-anchor frame, it is upscaled by applying bilinear interpolation on its decoded LR version, *i.e.,*

$$b^i_{intra}.SR = interp(b^i_{intra}.LR, scale). \tag{4}$$

After upscaling every block to their SR versions, the SR version of the non-anchor frame may be inserted into the SR buffer for future reuse (❻). More details are available in [46]. Based on the open-source SR decoder in NEMO, we develop a fine-grained SR decoder, where a larger cache profile indicates the type (*i.e.,* anchor or non-anchor) of every patch, and, based on their types, patches are upscaled via either DNN-based or reusing-based SR.

3 Limitations of existing solutions

As shown in the equation (2) and (4), reusing-based SR involves only lightweight operations such as bilinear interpolation of residuals rather than heavy inference of DNNs. Despite its advantage in computation costs, bilinear interpolation may not reconstruct missing details in the LR video well, so reusing-based SR may lead to suboptimal SR qualities when compared to DNN-based SR. To trade off the SR quality and the computation overhead, existing systems like NEMO and NeuroScaler carefully select the most beneficial anchor frames. To limit the scheduling complexity, both NEMO and NeuroScaler perform independent scheduling for every LR video segment whose time duration equals the pre-defined scheduling interval. Furthermore, NEMO and NeuroScaler approximate the video qualities with estimation values to avoid the heavy measurement of video qualities and accelerate scheduling.

Palantír also bases its scheduling on quality estimations and performs independent scheduling for every LR video segment. And the design goals of Palantír are two-fold: (1) selecting the most beneficial anchor patches to further improve the quality gain without increasing the total size of all anchors; and (2) low-latency selection to support UHD live streaming. A natural way to design Palantír is to adapt existing quality estimation techniques in NEMO and NeuroScaler to the patch-level granularity. However, as revealed later in this section, **existing SR quality estimation strategies lack a thorough understanding of how DNN-based and reusing-based SR affect the quality of the resulting SR video. Alternatively, they both heavily rely on empirical insights that either incur time-consuming operations or are only valid at the frame-level granularity.** Consequently, we can't simply extend existing solutions to meet the two design goals.

Observation #1: The insight in NEMO inevitably requires a

heavy initial measurement phase and thus violates the second design goal.

The core insight in NEMO is that the quality gain of a frame is mostly determined by the most relevant anchor frame, making it reasonable to estimate the quality under a given anchor frame set by combining the actual quality measurements under relevant anchor frame sets. Specifically, NEMO first measures $FQ\left(i\middle|f|\right)$, the quality of every frame i under every single anchor frame set $|f|$. Then, NEMO estimates the quality under any anchor frame set AF as $FQ\left(i\middle|AF\right) \approx max_{f\in AF} FQ\left(i\middle|f|\right)$. Capitalized on the heuristics, NEMO requires a heavy measurement phase in nature and incurs a dramatically high latency not suitable for live streaming. The scheme can be easily extended to the patch-level granularity by firstly conducting measurements for all single anchor patch sets, but the latency of initial measurements can be further increased.

Observation #2: The linked list model in NeuroScaler is highly inaccurate and the type-based prioritization contributes the most to the success of NeuroScaler. Yet, the type-based prioritization in NeuroScaler is only valid at the frame-level granularity, while its patch-level counterpart is invalid and cannot lead us toward the first design goal.

The method in NeuroScaler [48] can support real-time scheduling with its simplified modeling of the problem. It models the super-resolution error propagation process as a linked list, where each node corresponds to a frame and each pair of temporally consecutive frames is linked. The SR error of an anchor frame is assumed to be 0, while the error of a non-anchor frame equals that of its preceding frame node plus the residual between the two frames. The quality is estimated as the inverse of the sum of the errors over all frame nodes. In addition to the linked list model, NeuroScaler further incorporates the insight that some types of frames (*i.e.,* keyframes and alternative reference frames in the VP9 codec [21]) may have a high degree of reference (*i.e.,* be directly referred to by many subsequent frames) and thus lead to a large quality gain when determined as anchors. Therefore, NeuroScaler proposes to prioritize these types of frames as anchor frames.

To summarize, two core parts of NeuroScaler are linked list-based modeling and frame type-based prioritization. We quantify the importance of the two parts by comparing NeuroScaler with two variants. The first variant only uses frame type-based prioritization, while the second variant only uses linked list-based modeling. A detailed description of the two variants is available in Appendix § A.1. For a comprehensive comparison, we also use the solution in Palantír for frame-level scheduling by setting the patch size equal to the frame size, although we will not go into details about the solution in Palantír for now. The preliminary experiment is based on the first benchmark video used in the evaluation part (Sec. 7). As shown in Fig. 3, the second variant achieves the worst SR quality (computed as the PSNR between the SR video and the original HR video). Therefore, the linked list model is highly inaccurate, and frame type-based prioritization is indispensable to NeuroScaler.

Considering the above result, it seems that adapting NeuroScaler to fine-grained scheduling requires using the fine-grained counterpart of type-based prioritization, *i.e.,* the types of small encoding

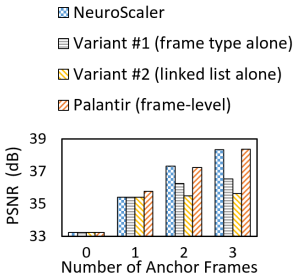


Figure 3: The SR qualities of NeuroScaler, two variants of NeuroScaler, and frame-level Palantír.

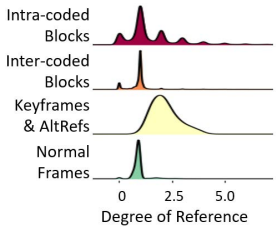


Figure 4: The distribution of degrees of reference for different coding units.

units such as blocks could implicitly imply their degrees of references and consequently their benefits as anchors. We validate such a counterpart using the 480p version of the first benchmark video. We separately measure the distribution of degrees of reference among different types of frames and blocks (including prioritized frame types, unprioritized normal frames, intra-coded blocks, and inter-coded blocks). The degree of reference for a given frame (or block) is quantitatively defined as $\frac{num_references}{resolution}$, where $num_references$ denotes the number of pixels that refer to the given frame (or block) for inter coding and $resolution$ denotes the number of pixels in the given frame (or block). As shown in Fig. 4, the degrees of references of keyframes and AltRefs are significantly greater than those of normal frames, while the distribution range of degrees of references for intra-coded blocks overlaps heavily with that for inter-coded blocks. We further use the common language effect size (CLES) [33] to quantify the results. The CLES is defined as the probability that a value randomly sampled from one distribution will be greater than that from another distribution. The CLES reaches 97.4% (close to the best case of 100%) between prioritized frame types and normal frames but is as low as 58.9% (close to the worst case of 50%) between different block types. Therefore, type-based prioritization is only helpful in the limited context of frame-level scheduling but hardly applies to patch-level scheduling.

4 Palantír Overview

Core methodology. As existing methods heavily rely on empirical insights that either require heavy operations or make sense in a limited context, we take an entirely new and first-principles approach. Our approach involves decomposing the SR video quality into the SR quality of every basic coding block and analyzing how the SR error propagates and accumulates at every basic coding block. Our first-principles reasoning suggests a DAG structure to represent the SR error propagation process. We discuss the approach later in Sec. 5.

Practical techniques. We also introduce two practical techniques so that we can construct the DAG by only using the LR video and improve the scheduling latency. We discuss them in Sec. 6.

Workflow. The workflow of Palantír is shown in Fig. 5. The streamer only uploads the LR video to the media server when it detects a limited uplink bandwidth. The server generates an SR error

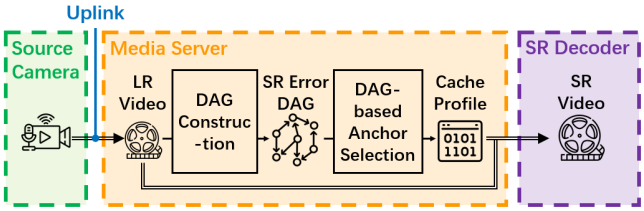


Figure 5: Palantír overview.

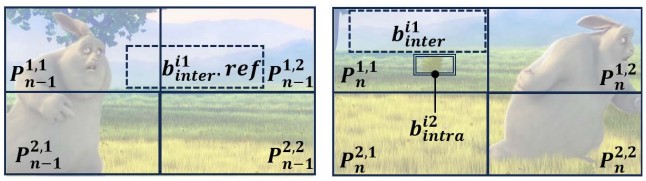


Figure 6: An example of SR error propagation.

DAG for every LR video segment whose time duration equals the pre-defined scheduling interval. The SR quality under any possible anchor patch set can be quickly estimated using the DAG, and beneficial anchor patches are greedily searched via DAG-based quality estimation. A cache profile is created, every bit of which indicates whether a patch in the LR segment is an anchor or not. In the existence of a powerful cloud server [48], both the LR segment and its corresponding cache profile are streamed to the server and then processed by the SR decoder on the server. Or, alternatively, the SR decoder can be executed in the streaming client.

5 First Principles Modeling

We take a first-principles approach to build Palantír. A first-principles approach involves breaking down complex problems into their fundamental parts and building up the solution from there. In neural-enhanced streaming, the anchor selection problem can be solved by estimating the video qualities under different anchor sets. The overall SR video quality is the average of the patch qualities, and each patch quality is fundamentally determined by the average quality of its most basic components - the blocks. In Sec. 5.1, we will follow this way of thinking to deduce our SR quality estimation strategy. In Sec. 5.2, we introduce our DAG model, which derives naturally from our analysis in Sec. 5.1 and allows us to fulfill lightweight yet reliable SR quality estimation.

5.1 Analysis of SR Error

We now separately discuss the SR errors (which are inversely related to the qualities) of non-anchor patches and anchor patches.

- **Case #1: non-anchor patches**.

  Analysis #1.1: every non-anchor patch P may consist of multiple inter-coded blocks ($b_{inter}^1, ..., b_{inter}^{n1}$) and intra-coded blocks ($b_{intra}^1, ..., b_{intra}^{n2}$). For analysis purposes, we split those blocks spanning multiple patches into multiple sub-blocks, each within a single patch. The SR error of P is:

$$\begin{aligned} &P.error \\ =&||P.SR - P.HR||_2^2 \\ =&\sum_{i=1}^{n1} ||b_{inter}^i.SR - b_{inter}^i.HR||_2^2 + \sum_{i=1}^{n2} ||b_{intra}^i.SR - b_{intra}^i.HR||_2^2 \\ =&\sum_{i=1}^{n1} b_{inter}^i.error + \sum_{i=1}^{n2} b_{intra}^i.error. \end{aligned} \tag{5}$$

  Example #1.1: as shown in Fig. 6, $P_n^{1,1}.error$ equals the sum of $b_{inter}^{i1}.error$, $b_{intra}^{i2}.error$, and the errors of many other blocks within $P_n^{1,1}$ (not marked due to the limited space).

  Conclusion #1.1: the SR error of a non-anchor patch is the sum of the SR errors of all inter-coded and intra-coded blocks within the patch.

  Analysis #1.2: According to the equation (2), we can further write the SR error of an inter-coded block b_{inter}^i as

$$\begin{aligned} &b_{inter}^i.error \\ =&||b_{inter}^i.SR - b_{inter}^i.HR||_2^2 \\ =&||b_{inter}^i.ref.SR + interp(b_{inter}^i.res, scale) - b_{inter}^i.HR||_2^2 \\ \approx&b_{inter}^i.ref.error + b_{inter}^i.res.complexity. \end{aligned} \tag{6}$$

  where

$$b_{inter}^i.ref.error = ||b_{inter}^i.ref.SR - b_{inter}^i.ref.HR||_2^2 \tag{7}$$

  and

$$\begin{aligned} &b_{inter}^i.res.complexity \\ =&||interp(interp(b_{inter}^i.HR - b_{inter}^i.ref.HR, scale^{-1}), scale) \\ &- (b_{inter}^i.HR - b_{inter}^i.ref.HR)||_2^2. \end{aligned} \tag{8}$$

  A more detailed derivation of the equation (6)-(8) is in Appendix § A.2. Here, $b_{inter}^i.res.complexity$ relates to the **texture complexity** of the HR residual (*i.e.*, $b_{inter}^i.HR - b_{inter}^i.ref.HR$), since an HR residual with more complex texture details will experience a more significant deviation after the process of downscaling and re-upsampling, *i.e.*, $interp(interp(\cdot, scale^{-1}), scale)$.

  Similarly, according to the equation (4), we have

$$b_{intra}^i.error = b_{intra}^i.complexity, \tag{9}$$

  where

$$\begin{aligned} &b_{intra}^i.complexity \\ =&||interp(interp(b_{intra}^i.HR, scale^{-1}), scale) - b_{intra}^i.HR||_2^2 \end{aligned} \tag{10}$$

  relates to the **texture complexity** of the HR content $b_{intra}^i.HR$.

  Example #1.2: for b_{inter}^{i1} in the frame F_n (see Fig. 6), its SR error equals $b_{inter}^{i1}.res.complexity + b_{inter}^{i1}.ref.error$, where $b_{inter}^{i1}.ref$ is its reference block in F_{n-1}, a reference frame of F_n; for b_{intra}^{i2}, its SR error equals $b_{intra}^{i2}.complexity$.

  Conclusion #1.2: the SR error of an inter-coded block depends on both the texture complexity of its HR residual

and the SR error of its reference block; while the SR error of an intra-coded block depends on the texture complexity of its HR content.

Analysis #1.3: combining the equation (5), (6), and (9), we have

$$P.error \approx P.TC + P.AE, \tag{11}$$

where

$$P.TC = \sum_{i=1}^{n1} b^i_{inter}.res.complexity + \sum_{i=1}^{n2} b^i_{intra}.complexity \tag{12}$$

indicates the **texture complexity** of the HR content or the HR residual, and

$$P.AE = \sum_{i=1}^{n1} b^i_{inter}.ref.error \tag{13}$$

is the **accumulated error** from depending blocks.

To reformulate (13) and simplify the modeling of patch-level SR error propagation, we make the following assumption (further discussed in Appendix § A.9):

Assumption #1.3: the per-pixel SR error within a patch is uniform - every pixel in the same patch shares exactly the same amount of error. Or, formally speaking, we assume that

$$p.error = ||p.SR - p.HR||_2^2 = \frac{P.error}{patch_size} \tag{14}$$

holds for every pixel p in some patch P.

Denoting the set of reference patches of P as P^1, ..., P^{n3} and according to the **Assumption #1.3**, we can reformulate the equation (13) as

$$\begin{aligned} P.AE &= \sum_{i=1}^{n1} b^i_{inter}.ref.error \\ &= \sum_{i=1}^{n1} \sum_{p \in b^i_{inter}.ref} p.error \\ &= \sum_{i=1}^{n3} W^i \cdot P^i.error, \end{aligned} \tag{15}$$

where the weight coefficient W^i indicates the ratio of the number of referenced pixels in P^i to the patch size.

Example #1.3: for convenience, we assume that b^{i1}_{inter} is the only inter-coded block in $P^{1,1}_n$ (see Fig. 6). The weight between $P^{1,1}_n$ and $P^{1,1}_{n-1}$ equals 0.105, as the size of the intersecting region between $b^{i1}_{inter}.ref$ and $P^{1,1}_{n-1}$ is 0.105 of the patch size; similarly, the weight between $P^{1,1}_n$ and $P^{1,2}_{n-1}$ equals 0.323. According to the equation (11) and (15), $P^{1,1}_n.error$ can be approximated as $P^{1,1}_n.TC + 0.105 \cdot P^{1,1}_{n-1}.error + 0.323 \cdot P^{1,2}_{n-1}.error$.

From the equation (11), (12), and (15) we can make the final conclusion:

Conclusion #1.3: under the Assumption #1.3, the SR error of a non-anchor patch equals the weighted sum of the SR errors of its depending patches plus the texture complexities of its inter-coded HR residuals and its intra-coded HR contents.

- **Case #2: anchor patches**. In many neural-enhanced live streaming systems [30, 48], it is common to train or fine-tune the SR DNNs based on the current video content. Furthermore, it has been demonstrated to be feasible to share the same model across similar videos to reduce the overhead of training or fine-tuning [32]. Therefore, we make the following assumption (further discussed in Appendix § A.9):

  Assumption #2: the SR errors of anchor patches are 0.

5.2 DAG Modeling

A patch P may depend on another patch P^i for inter-coding only if the frame containing P^i is a reference frame of that containing P. Since the frame-level reference relationship is directed and acyclic, the patch-level dependency is also directed and acyclic. Therefore, we propose to use a directed acyclic graph (DAG) to represent the dependency, where every node corresponds to a patch and every edge indicates an inter-coding dependency.

A static ***weight*** attribute is associated with every edge to reflect the inter-node coding dependency level (*i.e.,* the ratio of pixels in the destination patch node that refer to the source patch node for coding), corresponding to W^i in the equation (15). Three attributes are associated with each node P: the static ***P.TC*** attribute, representing the texture complexity defined in the equation (12); the ***P.is_anchor*** attribute, indicating whether the patch node is an anchor or non-anchor under a given anchor patch set; and the ***P.error*** attribute, representing the SR error. When *P.is_anchor* equals 1, *P.error* equals 0 (according to the **Assumption #2**); otherwise, *P.error* equals the weighted sum of the *error* attributes of the predecessor nodes plus $P.TC$ (according to the **Conclusion #1.3**).

Formulation of the first design goal. As introduced in Sec. 4, the first design goal is to pinpoint the anchor patches that can result in a high-quality SR video. With our DAG-based modeling, the quality can be estimated as a proxy variable $-\sum_P P.error$. Furthermore, the inference overhead is affected by the number of anchor patches (*i.e.,* $\sum_P P.is_anchor$). The first goal can then be reformulated as determining the optimal value of $P.is_anchor$ for each node P in order to maximize $-\sum_P P.error$, while ensuring that the value of $\sum_P P.is_anchor$ does not surpass some limit.

Comparison with the linked list model in NeuroScaler. We demonstrate before in Sec. 3 that our DAG model outperforms the linked list model even in the case of frame-level scheduling (by setting the DAG node size equal to the frame size). Here we give a brief discussion on the architectural improvements of the DAG model over the linked list model from two perspectives, and a more detailed discussion can be found in Appendix §A.3. *First*, the DAG model enables more accurate dependency representation. Although both the two models use edges to represent paths along which SR error propagates, our DAG model creates edges only where

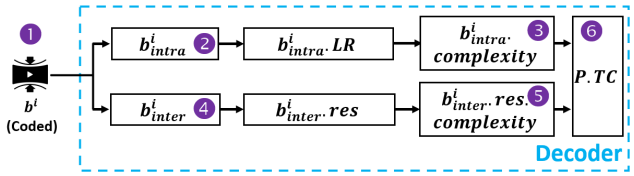


Figure 7: Determining the value of the static *TC* attribute.

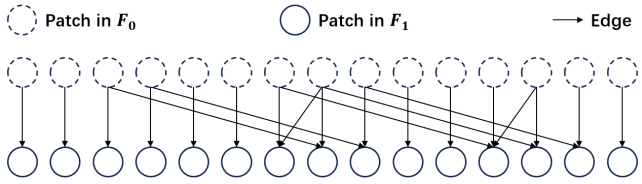


Figure 8: The part of a DAG, with 15 patches in each frame and F_0 being a reference frame of F_1.

actual coding dependencies exist, unlike the linked list model's fixed sequential connections. This is crucial for capturing the actual dependency in modern video codecs, including scenarios such as hierarchical B-picture coding structure [39, 40] and multi-reference video encoding [22]. *Second*, the DAG model adopts a dynamic ***weight*** attribute based on actual pixel reference ratios, whereas the linked list model assumes a fixed ***weight*** of 1. The adaptive attribute allows the DAG model to accurately model cases such as scene changes dominated by intra-frame coding, and multiple reference scenarios where dependency is distributed across frames.

Discussion of generalizability. We also discuss the generalizability of the DAG model in Appendix §A.4.

6 Practical Techniques

6.1 DAG Construction

At first glance of the equation (8) and (10), we need data from the HR video to compute the block-level texture complexities and then aggregate the block-level results to obtain the static $P.TC$ attribute according to (12). However, we find that using data from the HR video faces several challenges: (1) Commodity cameras can capture the HR data in the place, but they are typically not programmable [49] and thus not suitable for processing the task of DAG construction. Furthermore, video encoding and transmission alone are already computation-intensive and can cause severe problems such as device overheating [29]. Further running the scheduling task at the sender device can even make such problems more severe. (2) We choose to deploy the scheduling algorithm on the media server considering the above challenges of sender-side deployment. However, the media server only has access to the LR video due to the limited uplink bandwidth.

To enable scheduling on the media server, we propose to approximate the complexity of an inter-coded residual via

$$\begin{aligned} & b^i_{inter}.res.complexity \\ =&||interp(b^i_{inter}.res, scale) - (b^i_{inter}.HR - b^i_{inter}.ref.HR)||_2^2 \\ \approx& C||interp(interp(b^i_{inter}.res, 0.5), 2) - b^i_{inter}.res||_2^2 \end{aligned} \tag{16}$$

and approximate the complexity of an intra-coded block via

$$\begin{aligned} & b^i_{intra}.complexity \\ =&||interp(b^i_{intra}.LR, scale) - b^i_{intra}.HR||_2^2 \\ \approx& C||interp(interp(b^i_{intra}.LR, 0.5), 2) - b^i_{intra}.LR||_2^2, \end{aligned} \tag{17}$$

with C being a constant coefficient. We base our approximation on the following observation: if some content (*i.e.,* $b^i_{intra}.HR$) or some residual (*i.e.,* $b^i_{inter}.HR - b^i_{inter}.ref.HR$) is of high texture complexity, its downsampled version (*i.e.,* $b^i_{intra}.LR$ or $b^i_{inter}.res$) also tends to have high texture complexity. In other words, the texture complexity of a content or a residual is linearly related to that of its downsampled version. We validate the observation through a preliminary experiment and find that the Pearson correlation coefficient between the approximated texture complexity and the actual texture complexity is as high as 0.96.

With the above approximations, we can determine the $P.TC$ attribute by only resorting to data in the LR video. We slightly modify the decoder to fulfill the computation process. The workflow is shown in Fig. 7. While decoding a coded block b^i (❶), it computes either the complexity of its content or its residual, based on its coding type, and then adds the value to $P.TC$ (❻), where P is the patch containing the block. In the case of intra-coding (❷), $b^i_{intra}.complexity$ (❸) is computed from the decoded $b^i_{intra}.LR$, following the equation (17). In the case of inter-coding (❹), $b^i_{inter}.res.complexity$ (❺) is computed from the parsed $b^i_{inter}.res$, following the equation (16).

6.2 Parallel Selection

We employ a greedy searching algorithm to iteratively select the new anchor patch based on the estimated quality. We refer to the sequential implementation of this method as the vanilla Palantír. Specifically, the estimation processes for different anchor patch sets are executed sequentially, and the *error* attributes for different patch nodes under a given anchor patch set are also computed sequentially. As demonstrated later in our ablation experiment (see Sec. 7.4), the vanilla Palantír fails our second design goal of low-latency anchor selection. To address the issue, we propose a novel strategy to enable parallel and low-latency anchor selection in Palantír. The parallelization does not alter the selection results, so its benefits in latency do not come at the cost of compromising the energy efficiency of on-device neural enhancement or the monetary cost of cloud-based neural enhancement.

For clarity, we use an example to introduce our parallelism mechanism. As shown in Fig. 8, edges always start from some patch in F_0 and point to some patch in F_1. There exist neither edges along the opposite direction nor edges connecting patches within the same frame. We will utilize this phenomenon to optimize Palantír via both intra-set and inter-set parallelism. Note that the observed phenomenon is not accidental: edges indicate inter-frame coding references, and the coding reference relationship among frames is directed and acyclic in all mainstream codecs. Therefore, our optimization should be widely applicable.

Intra-set parallelism. Based on the above phenomenon, we can follow the frame decoding order to enumerate the DAG for quality estimation, *i.e.,* firstly compute the *error* attributes for the nodes in

F_0 and then deal with the nodes in F_1. The TC attributes of nodes in F_0 (or F_1) can be denoted as a vector TC_0 (or TC_1). Similarly, we use the notation $Error_i$ for the *error* attributes and Is_anchor_i for the *is_anchor* attributes ($i = 0, 1$). The *weight* attributes of the edges can be denoted by a sparse $Weight$ matrix. Note that the $Weight$ matrix is sparse since each patch in F_1 only refers to a limited set of patches in F_0 due to the limited range of motion vectors. For simplicity of discussion we assume that F_0 is the only reference frame of F_1, and the computation process of $Error_1$ can be formalized as $Error_1 = (TC_1 + Weight \cdot Error_0) \circ Is_anchor_1$, where $\circ$ indicates the element-wise multiplication and $Weight \cdot Error_0$ is a parallelizable sparse matrix-vector multiplication (SpMV) operation. Since Is_anchor_1 is a binary vector, we implement the element-wise multiplication by using Is_anchor_1 as a mask to parallelly set the corresponding entries in $TC_1 + Weight \cdot Error_0$ to zero. As SpMV is a common operation in many applications and has already attracted many optimization efforts [18, 31, 45], parallelized SpMV can be achieved by using a mainstream matrix-related computation package such as PyTorch.

Inter-set parallelism. Batching is widely used to improve DNN inference throughput [6] due to the effect of the data dimension on parallelism opportunities [5, 35]. Therefore, we execute the quality estimation under several searched anchor sets in parallel by adding a batch dimension to both $Error_i$ and Is_anchor_i. The same $Weight$ matrix and TC_i vectors are shared among different samples in the batch. With the inter-set parallelism, the SpMV operations are converted into the sparse matrix-matrix (SpMM) operations, which are also well studied and supported for parallel implementation.

7 Evaluation

We evaluate Palantír by answering three questions:

- Does Palantír achieve the first design goal of selecting a beneficial anchor patch set?
- Does Palantír achieve the second design goal of incurring a negligible latency overhead for UHD live streaming?
- How does each component of Palantír contribute to its overall performance?

7.1 Experimental Setup

Implementation. We develop our decoder based on the open-source SR codec in NEMO [46]. We incorporate two novel modes into the decoder. The first is to obtain the data required for graph construction (as introduced in Sec. 6.1). The second is to take both an LR video and a corresponding cache profile as input, and then upscale patches by DNN-based or reusing-based SR. About 1500 lines of code are added to the open-source codec to support the two modes, accounting for less than 0.5% of the total lines of code in the entire codec. **The source code is available at https://palantir-sr.github.io.**

Video. We download six popular 4k@30fps videos from YouTube. To demonstrate the universality of Palantír, the videos contain six distinct categories, including `makeup review`, `computer gaming`, `skit`, `shopping`, `car review`, and `unboxing`. We use FFmpeg (v3.4) [3] to transcode the HR video into the 480p (854×480) LR version in real time. We follow encoding guidelines to set the bitrate to `1800` `kbps`, the encoding speed to `5` [2], and the group of pictures (GoP) to `60` frames (*i.e.,* 2 seconds) [43]. We use the `-auto-alt-ref` option in FFmpeg to enable the alternative reference frame feature required by the anchor selection algorithm in NeuroScaler. Unless noted otherwise, we use the first five minutes of each video.

SR DNN. We adopt the DNN model of NAS [47]. We empirically set the number of residual blocks to 8 and the number of filters to 48. The DNN upscales the resolution of the LR video by 4 times. As the feasibility of online training for live streaming has been demonstrated [30], we train the DNN model for each benchmark video. When comparing the performance of different methods on the same video, the same DNN model is used for fairness.

Anchor patch size. We use a patch size of 170×160 to trade off anchor effectiveness and scheduling latency. Consequently, each LR frame consists of 15 patches.

Baselines. We use four baselines in this part. The first is the Per-frame baseline, which applies DNN-based SR on all the frames. The second is the NeuroScaler baseline, which uses the algorithm in NeuroScaler [48] to select the anchor frame set. The third is the NEMO [46] baseline. The fourth is the Key+Uniform baseline, which selects all the patches in the keyframe and equally spaced patches in the remaining frames as anchor patches.

Scheduling interval. Unless otherwise noted, we use a scheduling interval equal to the GoP (*i.e.,* 2 seconds).

Parallelism. The parallelized Palantír and the vanilla Palantír are two different implementations of the same selection method and lead to the same anchor patch set, so the results in Sec. 7.2 apply to both implementations. The latency results in Sec. 7.3 are obtained using the parallelized Palantír. Finally, the two implementations are compared in Sec. 7.4.

Hardware. We use a server with a 16-core AMD Ryzen processor and an NVIDIA A10 GPU as our media server, where graph construction and anchor selection are performed. The scheduling latency is measured on the server. Considering the deployment cost constraint in real-world applications, we only use CPU for Palantír, the NeuroScaler baseline, and the Key+Uniform baseline. However, to speed up our experiment, we further use the NVIDIA A10 GPU for the NEMO baseline, as the NEMO baseline requires numerous times of SR DNN inferences for anchor frame selection. We also use a Xiaomi 12S smartphone, which was announced in July 2022 and equipped with the Qualcomm Snapdragon 8+ Gen 1 Mobile Platform, to measure the energy efficiency when running SR DNN inference on mobile receiver devices.

7.2 Anchor Effectiveness

Quality Gain. We compute the peak-signal-to-noise-ration (PSNR) between the SR video and the original HR video to quantify the effectiveness of an anchor set. To make a fair comparison, we keep the total sizes of the anchor regions the same, *i.e.,* compare the quality gain under the m-element anchor frame set selected by some frame-level baseline with that under the $(15 \cdot m)$-element anchor patch set selected by Palantír or the Key+Uniform baseline. The only exception here is the Per-frame baseline, where all the frames are always treated as anchors and the size of the anchor

Table 1: A comparison of anchor effectiveness.

Video	Per-frame (dB)	DNN-based SR + Reusing-based SR (dB)			
		NeuroScaler	NEMO	Key+Uniform	Palantír
1	10.5	(2.2, 4.1, 5.1)	(2.8, 4.4, 5.4)	(2.1, 2.8, 3.2)	**(3.8, 5.4, 6.4)**
2	5.7	(0.5, 2.0, 2.6)	(1.7, 2.4, 3.0)	(0.5, 1.4, 2.0)	**(1.9, 2.8, 3.3)**
3	6.5	(1.1, 2.3, 3.1)	**(2.3, 3.2, 3.7)**	(1.1, 1.6, 1.9)	(2.1, 3.0, 3.5)
4	11.6	(4.4, 6.4, 7.4)	(4.7, 6.6, 7.6)	(4.4, 5.1, 5.6)	**(5.4, 7.1, 8.0)**
5	6.6	(3.8, 4.8, 5.2)	(4.2, 5.0, 5.3)	(3.8, 4.3, 4.5)	**(4.2, 5.1, 5.5)**
6	8.9	(3.0, 4.4, 5.1)	(3.5, 4.7, 5.3)	(3.0, 3.3, 3.6)	**(4.0, 5.3, 5.9)**
Average	8.3	(2.5, 4.0, 4.7)	(3.2, 4.4, 5.1)	(2.5, 3.1, 3.5)	**(3.6, 4.8, 5.4)**

Annotations: For any triple located in the last three columns, it is composed of the quality gain under $m = 1$, that under $m = 2$, and that under $m = 3$.

Table 2: A comparison of energy overhead.

Video	NeuroScaler (mAh)	Palantír (mAh)
1	(27.63, 41.33, 63.40)	**(21.21, 30.73, 41.72)**
2	(28.94, 42,27, 56,21)	**(18.06, 26.17, 40.82)**
3	(26.92, 41.48, 63.94)	**(21.07, 34.90, 47.91)**
4	(28.75, 41.71, 63.40)	**(26.08, 37.77, 47.05)**
5	(26.98, 41.18, 62.79)	**(26.37, 35.88, 45.47)**
6	(27.15, 40.64, 63.05)	**(23.98, 32.10, 40.85)**

Annotations: For any triple located in the second column, it is composed of the energy overhead under $|AF| = 1$, that under $|AF| = 2$, and that under $|AF| = 3$. As for any triple located in the last column, each of its elements is the result under some AP corresponding to the AF.

regions is thus always larger than other methods. Furthermore, we empirically limit m to not be greater than 3 since: (1) $m = 3$ can deliver quality gains that are comparable to the setting of applying DNN-based SR on all frames; (2) further increasing the value of m leads to a significant overhead.

The results are shown in Table. 1, from which we have several observations: (1) Palantír outperforms all the baselines with its ability to identify beneficial patches. (2) When compared with the NeuroScaler baseline, the state-of-the-art real-time scheduling method, Palantír boosts the quality gain of neural enhancement by 3.7 times at most and 1.4 times on average. (3) Compared to the non-realtime NEMO baseline, which requires significant time for initial measurements and does not support live streaming, Palantír still achieves a quality enhancement of up to 1.4 times in optimal conditions and 1.1 times on average. However, Palantír exhibits slightly inferior performance to NEMO in the worst-case scenario, specifically on the third benchmark video. A comprehensive analysis of this performance decline is provided in Appendix § A.5. (4) The fine-grained scheduling-based Key+Uniform baseline even falls behind the coarse-grained scheduling-based NeuroScaler baseline, so the effect of fine-grained scheduling heavily depends on the anchor selection method. (5) Palantír reduces the SR DNN inference overhead by 20 times with $m = 3$ (or 60 times with $m = 1$) while compared to the Per-frame baseline. With such a remarkable overhead reduction, Palantír still preserves 54.0-82.6% (or 32.8-64.0%) of the quality gain of the Per-frame baseline.

Energy Efficiency. We now examine how the anchor efficiency of Palantír transfers to energy efficiency when running the SR decoder on mobile devices. The detailed energy consumption measurement method is presented in Appendix § A.6.

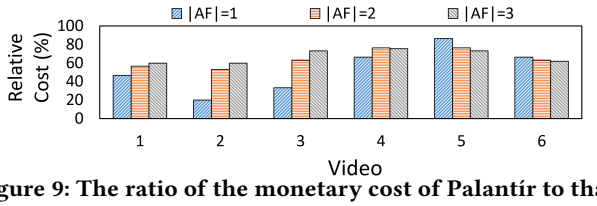


Figure 9: The ratio of the monetary cost of Palantír to that of the NeuroScaler baseline.

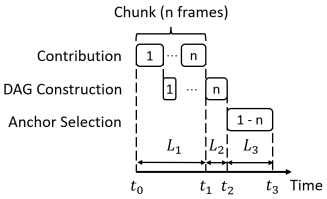


Figure 10: The timeline of Palantír. The numbers within the blocks represent the frame indices.

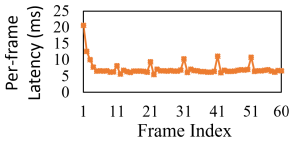


Figure 11: Per-frame latency of decoding for DAG construction during a GoP.

For each anchor frame set AF selected by the NeuroScaler baseline, we find the minimal anchor patch set AP which is selected by Palantír and achieves an equivalent or higher PSNR than AF. We compare the energy overhead under AF with that under AP. As shown in Table. 2, Palantír reduces the energy overhead over all cases. The reduction ratio is 38.1% at most and 22.4% on average.

Monetary cost reduction. We now present how Palantír reduces the monetary cost when running the SR decoder on cloud servers. We use the same method as measuring energy efficiency to find the corresponding AF for every AP. The monetary cost is estimated to be linear to the DNN computation complexity under the cache profile (AF or AP), and the keras-flops package [42] is used to measure the computation complexity. The ratio of the monetary cost incurred by Palantír to that incurred by the NeuroScaler baseline is presented in Fig. 9. Compared to NeuroScaler, Palantír reduces the monetary cost by 80.1% at most and 38.4% on average.

7.3 Scheduling Latency

End-to-end (E2E) latency is an important metric in live streaming [1, 7, 19, 41]. To ensure that the live streaming latency can be lower than the GoP, modern streaming standards such as CMAF [26] allow a chunk (which can be part of a GoP) to be immediately packaged (*i.e.,* chunked packaging [9]) and delivered (*i.e.,* chunked delivery [9]) when ready. Here we denote the chunk length as n frames and assume the scheduling interval of Palantír to be equal to n for simplicity. As shown in Fig. 10, the streamer contributes new video frames at a constant rate. Every new chunk of n frames is contributed per time duration of $L_1 = \frac{n}{frame_rate}$. In a traditional streaming pipeline without neural enhancement, the new chunk can be immediately packaged and delivered at t_1. However, two additional latency sources are presented in Palantír, *i.e.,* the DAG construction latency L_2 and the DAG-based anchor selection L_3. We evaluate whether $L_2 + L_3$ is small enough to well support latency-sensitive UHD live streaming applications.

We first examine the value of L_2. Note that we can directly feed a newly contributed frame to the decoder (working in the first mode introduced in Sec. 7.1) for DAG construction, so L_2 should be equal to the processing latency of the last frame in the scheduling interval if the decoder runs above 30fps. As shown in Fig. 11, the measured per-frame decoding latency for DAG construction is indeed always below 33 ms, so we estimate L_2 to be the average of the measured per-frame decoding latencies in Fig. 11, *i.e.,* 7.2ms.

The DAG-based anchor selection latency L_3 depends on the scheduling interval and the ratio of anchor patches. For ULL UHD live streaming applications [41] requiring an E2E latency below 200ms, we set the scheduling interval to be 66.67ms (*i.e.,* 2 frames in the 30-fps evaluation videos). As for LL live streaming applications [1, 7, 19] whose E2E latency requirements range from 2 seconds to 10 seconds, we consider five different settings (with the latency requirement being 2s, 4s, 6s, 8s, and 10s, respectively) and set the scheduling interval to be one-fifth of the latency requirement under each setting. As illustrated in Sec. 7.2, using only an anchor ratio of 5% can lead to large quality gains, so we set the ratio to be 5% for latency measurement. Under the above settings, the relationship between the overall scheduling latency $L_2 + L_3$ (with L_2 fixed to 7.2ms) and the E2E latency requirement is measured and plotted in Table. 3. We have two observations from the results: (1) Although the scheduling latency of Palantír is a little higher than that of NeuroScaler, it only accounts for a negligible portion (*i.e.,* 0.6-3.9%) of the E2E latency requirement. (2) As for the NEMO baseline, its scheduling latency is 88.2-225.6 times higher than that of Palantír and fails to support live streaming. A detailed discussion of NEMO's high latency is available in Appendix §A.7.

As a final note, the scheduling latency of Palantír can be even further optimized if necessary. We demonstrate in Appendix §A.8 that a simple modification can further reduce the latency by 3.1 times while incurring a quality degradation of no more than 0.1dB.

7.4 Ablation Study

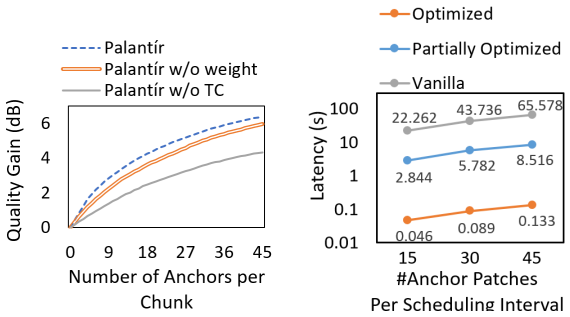
SR error DAG. The key to selecting a beneficial anchor patch set is our DAG-based modeling. We use a theoretical analysis to determine the values of the static *weight* attributes of the edges and the static *TC* attributes of patch nodes (see Sec. 5 and 6.1). To quantify the importance of setting appropriate values, we evaluate with the `makeup review` video and compare Palantír with two variants. In the first variant (Palantír w/o *weight*), the only predecessor node of the patch node $P_n^{i,j}$ (located at the i-th row and j-th column of the patch grid of the n-th frame) is $P_{n-1}^{i,j}$ and the weight of the edge connecting them equals 1. However, the *TC* attributes in the first variant are kept the same as in Palantír. Note that the first variant resembles the NeuroSclaer baseline when the patch size equals the frame resolution. In the second variant (Palantír w/o *TC*), the *weight* attributes are kept the same as in Palantír, but the *TC* attributes of all nodes are set to 1. As shown in Fig. 12(a), Palantír consistently outperforms the two variants.

Parallel Searching. We have introduced intra-set and inter-set parallelism (see Sec. 6.2) to speed up the quality estimation process in our greedy searching algorithm. We measure the DAG-based anchor selection latency (*i.e., L_3*) under three different settings: (1) the vanilla Palantír- nodes in the original DAG are processed serially for estimation; (2) the partially optimized Palantír setting - only the intra-set parallelism is enabled; (3) the optimized Palantír- both the two parallelism mechanisms are enabled. The measured DAG-based selection latencies (*i.e., L_3*) are shown in Fig. 12(b), showing that the optimized Palantír speeds up selection by 493 times at most.

Table 3: The relationship between the overall scheduling latency and the E2E latency requirement.

Setting	E2E Latency Requirement (ms)	Scheduling Latency (ms)		
		NeuroScaler	Palantír	NEMO
ULL	200	0.5	7.7	678.9
LL: Setting 1	2000	1.4	12.6	2842.2
LL: Setting 2	4000	2.0	27.0	5975.5
LL: Setting 3	6000	2.6	56.9	10509.3
LL: Setting 4	8000	5.0	92.4	16656.3
LL: Setting 5	10000	6.5	140.6	23126.7

Annotations: The presented result is the average latency over all the videos as the latencies for different videos are fairly close.



(a) Analysis on the DAG's attributes.

(b) Analysis on the parallelism mechanisms.

Figure 12: Ablation study.

8 Conclusion

In this work, we propose Palantír, the first neural-enhanced UHD live streaming system with fine-grained scheduling. Palantír seeks to improve efficiency via reasonable scheduling while minimizing the scheduling latency. Based on our first-principles reasoning, Palantír adopts DAG-based quality estimation to select a beneficial anchor patch set with low computation cost. Palantír further integrates two practical challenges to improve its feasibility and latency. The evaluation results demonstrate the superiority of Palantír.

Acknowledgments

This work is supported in part by the National Key Research and Development Program of China under grant No. 2024YFC2607400, the National Natural Science Foundation of China under grant No. 62302259, 62432008, 62472248, 62371269, and the Guangdong Innovative and Entrepreneurial Research Team Program under grant No. 2021ZT09L197.

References

[1] 2022. GB/T 28181-2022 PDF English. https://www.chinesestandard.net/PDF.aspx/GBT28181-2022, last accessed on Feb. 18, 2025.

[2] 2023. Live encoding with VP9 using FFmpeg. https://developers.google.com/media/vp9/live-encoding, last accessed on Mar. 15, 2024.

[3] 2024. FFmpeg. https://ffmpeg.org/, last accessed on Mar. 15, 2024.

[4] 2024. webm/libvpx - Git at Google. https://chromium.googlesource.com/webm/libvpx, last accessed on Mar. 15, 2024.

[5] Martín Abadi, Paul Barham, Jianmin Chen, Zhifeng Chen, Andy Davis, Jeffrey Dean, Matthieu Devin, Sanjay Ghemawat, Geoffrey Irving, Michael Isard, Manjunath Kudlur, Josh Levenberg, Rajat Monga, Sherry Moore, Derek G. Murray, Benoit Steiner, Paul Tucker, Vijay Vasudevan, Pete Warden, Martin Wicke, Yuan Yu, and Xiaoqiang Zheng. 2016. TensorFlow: A System for Large-Scale Machine Learning. In *Proceedings of the 12th USENIX Symposium on Operating Systems Design and Implementation*. USENIX Association, Savannah, GA, 265–283.

[6] Ahsan Ali, Riccardo Pinciroli, Feng Yan, and Evgenia Smirni. 2020. Batch: machine learning inference serving on serverless platforms with adaptive batching. In *Proceedings of the 2020 International Conference for High Performance Computing, Networking, Storage and Analysis*. IEEE, Article 69, 15 pages.

[7] Amazon Web Services, Inc. 2025. Video Latency in Live Streaming. https://aws.amazon.com/media/tech/video-latency-in-live-streaming, last accessed on Feb. 18, 2025.

[8] Duin Baek, Mallesham Dasari, Samir R. Das, and Jihoon Ryoo. 2021. DcSR: Practical Video Quality Enhancement Using Data-Centric Super Resolution. In *Proceedings of the 17th International Conference on Emerging Networking EXperiments and Technologies*. Association for Computing Machinery, New York, NY, USA, 336–343.

[9] Abdelhak Bentaleb, May Lim, Mehmet N. Akcay, Ali C. Begen, Sarra Hammoudi, and Roger Zimmermann. 2023. Toward One-Second Latency: Evolution of Live Media Streaming. (2023). arXiv preprint arXiv:2310.03256.

[10] Xuecheng Chen, Haoyang Wang, Yuhan Cheng, Haohao Fu, Yuxuan Liu, Fan Dang, Yunhao Liu, Jinqiang Cui, and Xinlei Chen. 2024. DDL: Empowering Delivery Drones With Large-Scale Urban Sensing Capability. *IEEE Journal of Selected Topics in Signal Processing* 18, 3 (2024), 502–515.

[11] Xuecheng Chen, Haoyang Wang, Zuxin Li, Wenbo Ding, Fan Dang, Chengye Wu, and Xinlei Chen. 2023. DeliverSense: Efficient Delivery Drone Scheduling for Crowdsensing with Deep Reinforcement Learning. In *Adjunct Proceedings of the 2022 ACM International Joint Conference on Pervasive and Ubiquitous Computing and the 2022 ACM International Symposium on Wearable Computers*. Association for Computing Machinery, New York, NY, USA, 403–408.

[12] Xuecheng Chen, Zijian Xiao, Yuhan Cheng, Chen-Chun Hsia, Haoyang Wang, Jingao Xu, Susu Xu, Fan Dang, Xiao-Ping Zhang, Yunhao Liu, and Xinlei Chen. 2024. SOScheduler: Toward Proactive and Adaptive Wildfire Suppression via Multi-UAV Collaborative Scheduling. *IEEE Internet of Things Journal* 11, 14 (2024), 24858–24871.

[13] Yuhan Cheng, Jirong Zha, Renjue Yang, Zhi Sun, Susu Xu, and Xinlei Chen. 2024. Multi-Agent Target Pursuit Using Perception Uncertainty-Aware Reinforcement Learning. In *Proceedings of the 30th Annual International Conference on Mobile Computing and Networking*. Association for Computing Machinery, New York, NY, USA, 1992–1997.

[14] Coastal Safety Group. 2021. Beach Cameras and Image Analytics——Coastal Safety Group. https://coastalsafetygroup.com.au/news/beach-cameras-and-image-analytics, last accessed on Jun. 26, 2024.

[15] CommsEase. 2022. AI Super-Resolution. https://doc.commsease.com/en/nertc/guide/zYzMjc0NTA?platform=android, last accessed on Jun. 26, 2024.

[16] Flyability. 2024. Ultimate Guide to Wind Turbine Inspection Techniques. https://www.flyability.com/blog/wind-turbine-inspection, last accessed on Jun. 26, 2024.

[17] Global Market Insights Inc. 2024. 4K Technology Market, Share & Analysis Report, 2024-2032. https://www.gminsights.com/industry-analysis/4k-technology-market, last accessed on Jun. 26, 2024.

[18] Constantino Gómez, Filippo Mantovani, Erich Focht, and Marc Casas. 2021. Efficiently running SpMV on long vector architectures. In *Proceedings of the 26th ACM Symposium on Principles and Practice of Parallel Programming*. Association for Computing Machinery, New York, NY, USA, 292–303.

[19] Google. 2025. Live streaming latency - YouTube Help. https://support.google.com/youtube/answer/7444635?hl=en, last accessed on Feb. 18, 2025.

[20] Google. 2025. YouTube recommended upload encoding settings. https://support.google.com/youtube/answer/1722171?hl=en, last accessed on Jun. 18, 2025.

[21] Adrian Grange, Peter de Rivaz, and Jonathan Hunt. 2016. VP9 Bitstream & Decoding Process Specification v0.6. https://storage.googleapis.com/downloads.webmproject.org/docs/vp9/vp9-bitstream-specification-v0.6-20160331-draft.pdf, last accessed on Mar. 15, 2024.

[22] Yu-Wen Huang, Bing-Yu Hsieh, Shao-Yi Chien, Shyh-Yih Ma, and Liang-Gee Chen. 2006. Analysis and complexity reduction of multiple reference frames motion estimation in H.264/AVC. *IEEE Transactions on Circuits and Systems for Video Technology* 16, 4 (2006), 507–522.

[23] Indrajit Ghosh. 2024. List of the Best AV1 Hardware Support in 2024 – Encode & Decode. https://www.faceofit.com/av1-hardware-support-in-2024/, last accessed on Feb. 18, 2025.

[24] International Olympic Committee. 2022. Beijing 2022 set to be the most immersive Olympic Winter Games yet. https://olympics.com/ioc/news/beijing-2022-set-to-be-the-most-immersive-olympic-winter-games-yet, last accessed on Jun. 26, 2024.

[25] International Olympic Committee. 2024. Intel unveils AI-Platform Innovation for Paris 2024. https://olympics.com/ioc/news/intel-unveils-ai-platform-innovation-for-paris-2024, last accessed on Jun. 26, 2024.

[26] ISOIEC JTC 1SC 29. 2024. ISOIEC 23000-19:2024 — Multimedia application format (MPEG-A) — Part 19: Common media application format (CMAF) for segmented media. https://www.iso.org/standard/85623.html, last accessed on Feb. 18, 2025.

[27] JDT Developer. 2022. The practice and application of video super-resolution technology. https://developer.jdcloud.com/en/article/2267, last accessed on Jun. 26, 2024.

[28] Zhuozhu Jian, Qixuan Li, Shengtao Zheng, Xueqian Wang, and Xinlei Chen. 2024. LVCP: LiDAR-Vision Tightly Coupled Collaborative Real-time Relative Positioning. arXiv:2407.10782 [cs.RO] https://arxiv.org/abs/2407.10782

[29] Soowon Kang, Hyeonwoo Choi, Sooyoung Park, Chunjong Park, Jemin Lee, Uichin Lee, and Sung-Ju Lee. 2019. Fire in Your Hands: Understanding Thermal Behavior of Smartphones. In *Proceedings of the 25th Annual International Conference on Mobile Computing and Networking*. Association for Computing Machinery, New York, NY, USA, Article 13, 16 pages.

[30] Jaehong Kim, Youngmok Jung, Hyunho Yeo, Juncheol Ye, and Dongsu Han. 2020. Neural-Enhanced Live Streaming: Improving Live Video Ingest via Online Learning. In *Proceedings of the Annual Conference of the ACM Special Interest Group on Data Communication on the Applications, Technologies, Architectures, and Protocols for Computer Communication*. Association for Computing Machinery, New York, NY, USA, 107–125.

[31] Kenli Li, Wangdong Yang, and Keqin Li. 2015. Performance Analysis and Optimization for SpMV on GPU Using Probabilistic Modeling. *IEEE Transactions on Parallel and Distributed Systems* 26, 1 (2015), 196–205.

[32] Jinyeong Lim, Juncheol Ye, Jaehong Kim, Hwijoon Lim, Hyunho Yeo, and Dongsu Han. 2023. Neural Cloud Storage: Innovative Cloud Storage Solution for Cold Video. In *Proceedings of the 15th ACM Workshop on Hot Topics in Storage and File Systems*. Association for Computing Machinery, New York, NY, USA, 1–7.

[33] K. O. McGraw and S. P. Wong. 1992. A common language effect size statistic. *Psychological Bulletin* 111, 2 (1992), 361–365.

[34] Microsoft Edge Team. 2023. Video super resolution in Microsoft Edge. https://blogs.windows.com/msedgedev/2023/03/08/video-super-resolution-in-microsoft-edge/, last accessed on Jun. 26, 2024.

[35] Graham Neubig, Yoav Goldberg, and Chris Dyer. 2017. On-the-fly Operation Batching in Dynamic Computation Graphs. In *Proceedings of the 31st International Conference on Neural Information Processing Systems*, Vol. 30. Curran Associates, Inc.

[36] Ookla, LLC. 2024. Speedtest Global Index - Internet Speed around the world. https://www.speedtest.net/global-index, last accessed on Jun. 26, 2024.

[37] Jiyuan Ren, Yanggang Xu, Zuxin Li, Chaopeng Hong, Xiao-Ping Zhang, and Xinlei Chen. 2023. Scheduling UAV Swarm with Attention-based Graph Reinforcement Learning for Ground-to-air Heterogeneous Data Communication. In *Adjunct Proceedings of the 2023 ACM International Joint Conference on Pervasive and Ubiquitous Computing & the 2023 ACM International Symposium on Wearable Computing*. Association for Computing Machinery, New York, NY, USA, 670–675.

[38] Mário Saldanha, Marcel Corrêa, Guilherme Corrêa, Daniel Palomino, Marcelo Porto, Bruno Zatt, and Luciano Agostini. 2020. An Overview of Dedicated Hardware Designs for State-of-the-Art AV1 and H.266/VVC Video Codecs. In *Proceedings of the 27th IEEE International Conference on Electronics, Circuits and Systems*. IEEE, 1–4.

[39] Heiko Schwarz, Detlev Marpe, and Thomas Wiegand. 2006. Analysis of Hierarchical B Pictures and MCTF. In *Proceedings of the 2006 IEEE International Conference on Multimedia and Expo*. IEEE, 1929–1932.

[40] Heiko Schwarz, Detlev Marpe, and Thomas Wiegand. 2007. Overview of the Scalable Video Coding Extension of the H.264/AVC Standard. *IEEE Transactions on Circuits and Systems for Video Technology* 17, 9 (2007), 1103–1120.

[41] Soliton Systems. 2025. Beyond Line of Sight Drones with Ultra Low Latency. https://www.solitonsystems.com/low-latency-video/remote-operation/beyond-line-of-sight-command-and-control-of-drones, last accessed on Feb. 18, 2025.

[42] Tokusumi. 2020. keras-flops · pypi. https://pypi.org/project/keras-flops/, last accessed on Feb. 18, 2025.

[43] Twitch. 2024. Broadcast Guidelines. https://help.twitch.tv/s/article/broadcast-guidelines?language=en_US, last accessed on Mar. 15, 2024.

[44] Haoyang Wang, Xuecheng Chen, Yuhan Cheng, Chenye Wu, Fan Dang, and Xinlei Chen. 2023. H-SwarmLoc: Efficient Scheduling for Localization of Heterogeneous MAV Swarm with Deep Reinforcement Learning. In *Proceedings of the 20th ACM Conference on Embedded Networked Sensor Systems*. Association for Computing Machinery, New York, NY, USA, 1148–1154.

[45] Guoqing Xiao, Kenli Li, Yuedan Chen, Wangquan He, Albert Y. Zomaya, and Tao Li. 2021. CASpMV: A Customized and Accelerative SpMV Framework for the Sunway TaihuLight. *IEEE Transactions on Parallel and Distributed Systems* 32, 1 (2021), 131–146.

[46] Hyunho Yeo, Chan Ju Chong, Youngmok Jung, Juncheol Ye, and Dongsu Han. 2020. NEMO: Enabling Neural-Enhanced Video Streaming on Commodity Mobile Devices. In *Proceedings of the 26th Annual International Conference on Mobile Computing and Networking*. Association for Computing Machinery, New York, NY, USA, Article 28, 14 pages.

[47] Hyunho Yeo, Youngmok Jung, Jaehong Kim, Jinwoo Shin, and Dongsu Han. 2018. Neural adaptive content-aware internet video delivery. In *Proceedings of the 13th USENIX Symposium on Operating Systems Design and Implementation*. 645–661.

[48] Hyunho Yeo, Hwijoon Lim, Jaehong Kim, Youngmok Jung, Juncheol Ye, and Dongsu Han. 2022. NeuroScaler: Neural Video Enhancement at Scale. In *Proceedings of the ACM SIGCOMM 2022 Conference*. Association for Computing Machinery, New York, NY, USA, 795–811.

[49] Mu Yuan, Lan Zhang, Xuanke You, and Xiang-Yang Li. 2023. PacketGame: Multi-Stream Packet Gating for Concurrent Video Inference at Scale. In *Proceedings of the ACM SIGCOMM 2023 Conference*. Association for Computing Machinery, New York, NY, USA, 724–737.

[50] ZEGOCLOUD. 2025. ZegoExpressEngine. https://docs.zegocloud.com/article/api?doc=express_video_sdk_API~java_android~class~ZegoExpressEngine#enable-video-super-resolution, last accessed on Feb. 18, 2025.

Spatial Visibility and Temporal Dynamics: Rethinking Field of View Prediction in Adaptive Point Cloud Video Streaming

Chen Li, Tongyu Zong, Yueyu Hu, Yong Liu, Yao Wang
New York University
USA
{chen.lee,tz1178,yh3986,yongliu,yw523}@nyu.edu

Abstract

Field-of-View (FoV) adaptive streaming significantly reduces bandwidth requirement of immersive point cloud video (PCV) by only transmitting visible points inside a viewer's FoV. The traditional approaches often focus on trajectory-based 6 degree-of-freedom (6DoF) FoV predictions. The predicted FoV is then used to calculate point visibility. Such approaches do not explicitly consider video content's impact on viewer attention, and the conversion from FoV to point visibility is often error-prone and time-consuming. We reformulate the PCV FoV prediction problem from the cell visibility perspective, allowing for precise decision-making regarding the transmission of 3D data at the cell level based on the predicted visibility distribution. We develop a novel spatial visibility and object-aware graph model (**CellSight**) that leverages the historical 3D visibility data and incorporates spatial perception, occlusion between points, and neighboring cell correlation to predict the cell visibility in the future. We focus on multi-second ahead prediction to enable the use of long pre-fetching buffers in on-demand streaming, critical for enhancing the robustness to network bandwidth fluctuations. CellSight significantly improves the long-term cell visibility prediction, reducing the prediction Mean Squared Error (MSE) loss by up to 50% compared to the state-of-the-art models when predicting 2 to 5 seconds ahead, while maintaining real-time performance (more than 30fps) for point cloud videos with over 1 million points.

CCS Concepts

• **Information systems** → **Multimedia streaming**; • **Computing methodologies** → *Machine learning approaches.*

Keywords

Point Cloud Video, Field of View (FoV) Prediction, Immersive Video Streaming, 6 Degree of Freedom (DoF), Virtual Reality (VR), GNN

ACM Reference Format:

Chen Li, Tongyu Zong, Yueyu Hu, Yong Liu, Yao Wang. 2025. Spatial Visibility and Temporal Dynamics: Rethinking Field of View Prediction in Adaptive Point Cloud Video Streaming. In *ACM Multimedia Systems Conference 2025 (MMSys '25), March 31-April 4, 2025, Stellenbosch, South Africa.* ACM, New York, NY, USA, 11 pages. https://doi.org/10.1145/3712676.3714435

Permission to make digital or hard copies of all or part of this work for personal or classroom use is granted without fee provided that copies are not made or distributed for profit or commercial advantage and that copies bear this notice and the full citation on the first page. Copyrights for components of this work owned by others than the author(s) must be honored. Abstracting with credit is permitted. To copy otherwise, or republish, to post on servers or to redistribute to lists, requires prior specific permission and/or a fee. Request permissions from permissions@acm.org.
MMSys '25, March 31-April 4, 2025, Stellenbosch, South Africa
© 2025 Copyright held by the owner/author(s). Publication rights licensed to ACM.
ACM ISBN 979-8-4007-1467-2/2025/03
https://doi.org/10.1145/3712676.3714435

1 Introduction

Augmented Reality (AR) and Virtual Reality (VR) applications are gaining popularity rapidly. Streaming high-quality immersive videos, such as 360-degree videos and point cloud videos, to viewers is one of the most critical components for the wide adoption of AR/VR. It can also be integrated with interactive methods, such as the user force-aware approach in [33], further enhancing the user experience. However, immersive videos require significantly higher bandwidth than the traditional 2D planar videos. For example, a point cloud video consisting of 1 million points per frame requires streaming bandwidth of more than 120 Mbps even with lossy compression [22], while future production of high quality point cloud can scale up to 10M points per frame. A promising solution is FoV adaptive streaming that only streams video content within a viewer's current viewport. For 360-degree video, if the viewport is 120 degree (horizontal) by 90 degree (vertical), only 1/6 video data fall into a viewport, resulting into 6-fold bandwidth reduction. For point cloud video, due to points occlusion in 3D space, not all points falling into a viewport are visible. One can save even more bandwidth by not just omitting the points outside the viewport, but also removing the hidden points within the viewport. The complete point cloud frame illustrated in Fig. 1 contains more than 1 million points. The number of points visible for the viewport given in Fig. 1(a) are less than 150k, after removing the points outside the viewport and the occluded points. In cell-based PCV streaming, a point cloud frame is divided into cells, as shown in Fig. 1(c), and each cell is independently coded and transmitted.[1] To deliver the actual view in Fig. 1(b), only cells covering those visible points in Fig. 1(c) need to be transmitted, leading to 7-fold of streaming bandwidth reduction.

In immersive video streaming, a viewer can freely change her viewpoint (X, Y, Z), as well as her view angle *(yaw, pitch, roll)*, resulting in a total of 6 Degree-of-Freedom (6DoF). Cell-based FoV adaptive streaming dynamically prefetches cells that are predicted to be visible into local streaming buffer based on the viewer's past viewport trajectory. The key challenge is to accurately predict *cell visibility* in real time so that more bandwidth can be allocated to prefetch cells with higher visibility. The visibility of a cell can be simply measured by its overlap ratio with the viewport. Considering occlusion, cell visibility should be refined by discounting occluded points. Finally, cells at different viewing distances contribute differently to the viewer's viewing experience [9, 36]. More precise cell

[1] To clarify the terminology, we use "cell" to denote the geometry-based PCV unit, as described in [9].

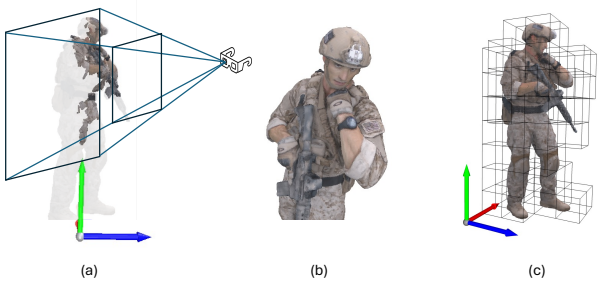


Figure 1: Illustration of PCV Point Visibility. As in Fig. 1(a), the content within the pyramid, defined by the near plane and far plane, is potentially visible in the viewport, while any content outside the pyramid will not be visible. Furthermore, some points inside the viewport are still not visible if occluded by other points. When a viewer watches the point cloud content from the viewpoint indicated in Fig. 1(a), only the highlighted part in Fig. 1(a) would be visible. The actual view for the viewer is shown in Fig. 1(b). If the point cloud is divided into 3D cells like in Fig. 1(c), only the cells covering the visible points need to be transmitted.

visibility metric should also take into account the cell-viewpoint distance. For on-demand PCV streaming, a relative long streaming buffer, e.g., 2 to 5 seconds, is preferred to provide sufficient margin for smooth video streaming and video processing. Consequently, predicting cell visibility in medium and long future horizons has become an important and challenging research problem.

Naturally, most of the existing FoV-adaptive PCV streaming studies [9, 11, 16, 20] predict point/cell visibility in two steps: 1) predict the viewer's future viewport based on her past viewport trajectory; 2) calculate visible points and cell visibility using the predicted viewport. This approach has several drawbacks: 1) trajectory-based viewport prediction does not explicitly consider the impact of video content on the viewer's attention; 2) small errors in any 6DoF viewport coordinates prediction may lead to large errors in visible points prediction; and 3) Hidden Point Removal (HPR) at high point density is time consuming. These limitations motivate us to rethink FoV prediction for PCV streaming: *Can we directly predict cell visibility based on the viewer's viewport trajectory, cell visibility history, and spatial features of objects to be viewed in PCV?* There are several advantages of this direct approach: 1) by directly predicting cell visibility based on the cell visibility history (we assume that the viewer's past viewport is fed back to the sender so that the sender can produce the visibility history data), we avoid the potential error amplification in the process of mapping 6DoF viewport to cell visibility; 2) using the spatial features of PCV objects to be viewed (for which we also have the ground-truth in on-demand streaming) as an input, we explicitly take into account the impact of PCV objects and their movements on the viewer's attention; 3) PCV object movements and viewer viewpoint movements are both continuous in time and space. As a result, point occlusion and visibility vary continuously in time and space. Motivated by these observations, we propose a TransGraph-GRU based spatial-temporal model for predicting 3D cell visibility. For the spatial aspect, a graph-based model is well-suited because if one cell lies within the FoV, its neighbors are likely to be visible as well. For the temporal aspect, an RNN-based model effectively captures dynamic dynamics [35], such as when a cell begins moving out of the FoV, making it likely to remain less visible in the near future. We opt for GRU over LSTM due to its lighter computational overhead—a critical factor when managing many cells. Prior work [10] has demonstrated the efficiency of integrating GRU with graph structures. Additionally, Transformer-based techniques [25] can effectively learn hidden spatial dependencies among neighboring cells, further enhancing visibility prediction. A well-trained spatial-temporal machine learning model can thus leverage these continuous dynamics to achieve accurate point/cell visibility prediction. Building on these insights, we make the following contributions in this paper:

- We design **CellSight**, a novel PCV FoV prediction framework that directly predicts the cell visibility in a future frame to a viewer based on the viewer's past viewport trajectory and the point cloud spatial features to be viewed.
- We construct and predict a set of cell visibility features that quantify the importance of streaming a cell by considering its overlap ratio with the viewport, the fraction of visible points after occlusion, and its angular span in the viewport determined by the viewing distance.
- We develop a spatial-temporal graph model which can capture the spatial and temporal correlations of cell visibility in consecutive PCV frames for accurate prediction.
- Through comprehensive experiments, we demonstrate that CellSight can generate real-time cell visibility predictions, and improve the long-term prediction accuracy of the state-of-the-art methods by up to 50% on the real point cloud video and viewport trajectory datasets.

2 Related Work

Immersive video streaming, whether for 360-degree (3DoF) videos or point cloud (6DoF) videos, demands strategies to reduce bandwidth usage while preserving user experience. In this section, we review prior work for 3DoF and 6DoF streaming scenarios and discuss the limitations that motivate our approach.

3DoF 360-degree Video Streaming. To mitigate the high bandwidth consumption inherent in 360-degree video streaming, many studies focus on predicting the user's Field-of-View (FoV) so that only the most relevant parts of the video are streamed at high quality. For instance, [2, 18] leverages personalized and cross-user behaviors to forecast future FoV, while [8] enhances prediction accuracy by incorporating content-based features in addition to historical FoV data. Furthermore, [19] extends these ideas by using variations of Long Short-Term Memory (LSTM) models to predict future FoV from past viewport trajectories, and even applies 2D spatial and temporal ConvLSTM models to forecast tile visibility. Although effective for 360-degree video, these methods rely on the assumption that all pixels within the 360-degree view are visible—a condition that does not hold in the more complex PCV context.

6DoF Point Cloud Video Streaming. Point cloud videos introduce additional challenges due to occlusions and vastly larger data volumes. In these scenarios, closer objects can obscure those further away, making the prediction of visibility more complicated.

Research in this area has addressed quality assessment, compression,reconstruction, segmentation and system design etc. For example, subjective studies in [23] and objective evaluations in [1, 29] have characterized the quality of point cloud videos, while [3] explores quality assessment using 3D visual saliency. To handle high data volumes, progressive coding strategies [24] and real-time 3D compression frameworks [4] have been proposed. Additional work includes human avatar reconstruction from few-shot views [32], and medical point cloud image segmentation [6]. Besides, system-level approaches for PCV streaming have been explored [5]. For example, [21] developed a mobile transcoding framework, and [12] proposed a real-time, human-centered method that segments the human body from the static background to reduce bandwidth overhead. When it comes to FoV prediction for point cloud videos, the challenges intensify due to 6DoF. [26] employing a 3DoF Bundle Adjustment to estimate 6DoF estimation. In [11], a combined LSTM and Multi-Layer Perceptron (MLP) model is used to predict user motion and head orientation for the next frame—yielding a very short prediction horizon (approximately 11 ms). The Vivo system [9] extends this horizon to 200 ms by incorporating factors like distance and occlusion, yet its trajectory-based prediction remains limited for long-term prediction. Similarly, CaV3 [20] improves prediction accuracy by integrating content features, but still suffers from the inherent limitations of trajectory-based approaches over long prediction horizons.

Our Contribution: CellSight. Unlike the existing methods that predominantly rely on trajectory-based FoV prediction, our approach, **CellSight**, leverages spatial and temporal models to capture detailed 3D cell visibility patterns. This design allows us to extend the accurate FoV prediction horizon to 5 seconds, thereby providing the streaming system with a substantially larger optimization window. In doing so, CellSight improves the robustness of adaptive streaming systems against network condition fluctuations.

3 Background and Problem Formulation

3.1 Point Cloud Video

A point cloud video (PCV) consists of a sequence of frames, each of which is a cloud of points on the surface of objects in a captured scene. Each point is described by it 3D coordinate and color. It can be rendered on 2D displays or VR goggles based on the user's view point, allowing users to explore the 3D scene from any angle and depth with 6 degrees of freedom (X, Y, Z, yaw, pitch, roll). We assume a PCV is represented by a geometry-based method and can be partitioned into 3D cells. Octree is one example that has been used in previous PCV streaming system [9]. For streaming, each PCV frame is typically partitioned into small 3D cells, which can be encoded independently at multiple quality levels [30], as shown in Fig. 1(c). For a given viewer viewport, only a small subset of cells are visible. If we know exactly which cells are visible, we just need to stream those visible cells at the highest quality allowed by the bandwidth. To deal with the unavoidable cell visibility prediction errors, we can also stream some cells with low visibility at low quality similar to the 360-degree video streaming strategy [28]. The final PCV quality perceived by a viewer and the PCV streaming overhead is largely determined by the accuracy of cell visibility prediction.

Symbol	Description
T_τ	6-DoF coordinates trajectory at time τ
$\mathcal{T}_h$	history trajectory
C	3D cell set
O^τ	graph occupancy features at time τ
F^τ	graph viewport overlap ratio feature at time τ
V^τ	graph occlusion-aware visibility features at time τ
E^τ	graph other features at time τ
A^τ	graph angular span feature at time τ
B^τ	graph visible angular span feature at time τ
G_τ	all graph raw features at time τ
S_τ	hidden states output of GRU at time τ
$\hat{H}_\tau$	hidden states output of GNN at time τ
o_i^τ	occupancy feature of cell i at time τ
f_i^τ	viewport overlap ratio feature of cell i at time τ
v_i^τ	occlusion-aware visibility feature of cell i at time τ
α_i^τ	angular span feature of cell i at time τ
β_i^τ	visible angular span feature of cell i at time τ
e_i^τ	other features of cell i at time τ

Table 1: Notation table

3.2 Problem Formulation

We assume that the client has a frame buffer of length f, so that when the client is displaying frame h, the server needs to send frame $h+f$. Our goal is to predict the cell visibility for a future frame $h+f$ from the viewer's past history viewport trajectory, the past point cloud frames and the future point cloud frames up to frame $h+f$. We denote the viewer's history viewport trajectory by

$$\mathcal{T}_h = \{T^1, ..., T^h\}$$

where $T^\tau = (x^\tau, y^\tau, z^\tau, \psi^\tau, \theta^\tau, \phi^\tau)$ is the 6-DoF coordinates at time τ, and the corresponding PCV frame sequence by $\mathcal{P}_h = \{P^1, ..., P^h\}$. Each frame P^τ is partitioned into a set C of cells, each cell contains a set of points. The visibility of a cell i at time τ is defined as the number of visible points in that cell, denoted by as v_i^τ. The cell visibility vector for frame τ is $V^\tau = \{v_i^\tau, i \in C\}$. Our goal is to predict the cell visibility for a future frame Y^{h+f} at time $\tau = h+f$ from $\mathcal{T}_h, \mathcal{P}_h, P^{h+f}$ using a learned function $\mathcal{F}$:

$$Y^{h+f} = \mathcal{F}(\mathcal{T}_h, \mathcal{P}_h, P^{h+f}),$$

where Y^{h+f} denotes cell visibility features to be defined in Sec. 4.2. In principle, all frames in $\mathcal{P}_{h+f}$ can be used for prediction. For simplicity, we use $\mathcal{P}_h$ and P^{h+f}.

4 Methodology

We propose a graph-based prediction model that exploits Spatial Visibility and Temporal Dynamics of PCV. Fig. 2 presents an overview of CellSight. We divide the space covered by the entire point cloud video into multiple equal-sized cells. For the example in Fig 3, the space is partitioned into $5 * 6 * 8$ cells and all cells form a grid-like graph. Neighboring cells in the same frame have strong visibility correlations, which can be exploited by a graph model. Each node in the graph model corresponds to a cell. Each node has its neighboring cells as neighbors in the graph. For example, a cell can have

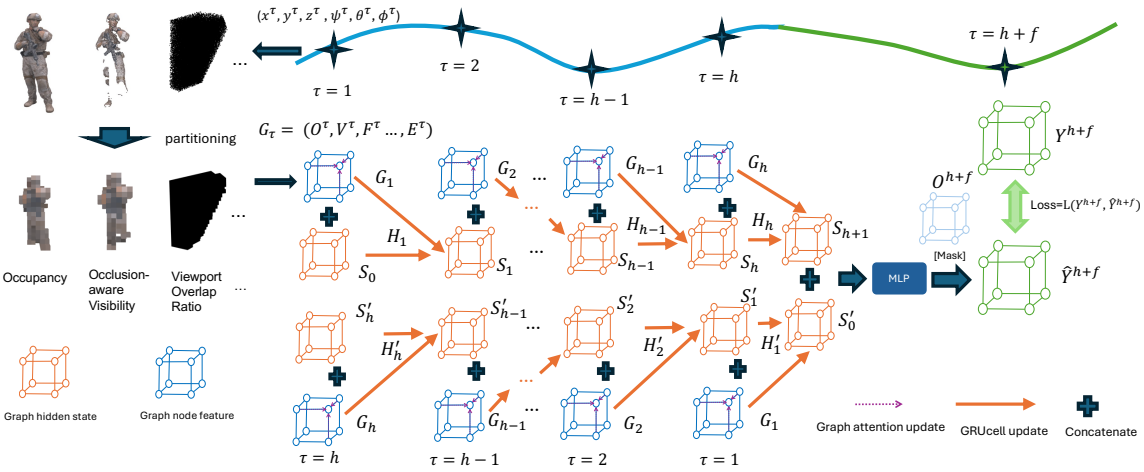


Figure 2: Overview of our cell visibility prediction system.

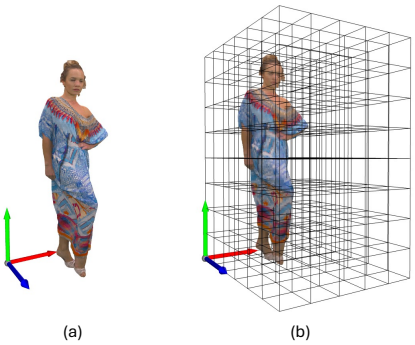


Figure 3: (a) is an example frame of a point cloud video. (b) shows how the entire 3D space is divided into 3D cells. We build a graph treating each cell as a node in the graph.

6 neighbors sharing the same side, or up to 26 neighbors sharing the same corner/edge/side. To simplify the illustration, we use a 8-node graph to represent the whole graph in Fig. 2. For each node, we have several features for visibility prediction. The curve on the top is the viewer's 6DoF viewport trajectory, illustrating one of the 6DoF coordinates $(x, y, z, \psi, \theta, \phi)$ at each frame time τ for a point cloud sequence set $\mathcal{P}^{h+f}$. For each frame, based on the 6DoF coordinates and P^τ, we can calculate, for each cell i, occupancy feature, visibility features, as well as location and distance features, which will be discussed in the following. We use $G_\tau = [O^\tau, F^\tau, V^\tau, ..., E^\tau]$ to represent the node features for each frame. We use a temporal Bidirectional GRU model and a spatial transformer-based graph model to capture patterns in viewer attention and cell visibility. GRU model captures each node's temporal pattern over time, which is encoded into the hidden state. In the graph model, each node aggregates its neighbor's information and hidden state from GRU, as the dash line shows. The bidirectional GRU can go through the history forward and backward, and have an enhanced hidden state to capture the history pattern. The details for Graph and GRU will be introduced in Section. 4.4. After we get the S_{h+1} and $S_0^{'}$ from bi-directional GRU, a shared MLP is used to predict the cell visibility for each cell at the target time stamp $\tau = h + f$. The final prediction output $\hat{Y}^{h+f}$ can be different cell visibility features designed to quantify the importance of a cell for the rendered view in different scenarios. The predicted cell visibility will be used by streaming algorithms to optimize bandwidth allocation among cells and improve streaming QoE. This proposed solution can be applied for video on demand or live streaming that can tolerate some playback lags, e.g. 1-5 second. Longer prediction accuracy of the solution will give more flexibility for the buffer-based streaming algorithms. We will introduce those features considering the cell's overlap ratio with the viewport, occlusions among points, and the viewing distance in Sec. 4.2. Before the final output, the O^{h+f} can be optionally applied as a mask to refine the predicted visibility $\hat{Y}^{h+f}$, since in the streaming system, the server has the point cloud at frame $h+f$. In the following, we will introduce how we construct features, including Cell Occupancy Feature, Cell Visibility Features, and Locality and Distance Features.

4.1 Cell Occupancy Feature

After partitioning the point cloud video into cells, we can get the number of points in each cell. Since viewer's attention can be driven by the objects in the point cloud video, the point density in a cell will affect the viewer's view interest for it and can be used as an important feature for cell visibility prediction. For example, work [14] studied the impact of video content on user FoV trajectory and observed that "users' gaze often follows the activity of the object inside the video", "for volumetric scenes with small movements, the gaze may move back and forth with irregular movement, but it generally still focuses on the target object". Clearly, if a cell is empty, it is unlikely to be viewed. We introduce the cell occupancy feature for frame τ as:

$$O^\tau = \{o_i^\tau, i \in C\}, \tag{1}$$

where o_i^τ is the number of points in cell i of frame τ.

4.2 Cell Visibility Features

We now introduce different cell visibility features that quantify the contribution of a cell to the viewing quality by considering different factors.

4.2.1 Viewport Overlap Ratio. Viewport Overlap Ratio is the overlap ratio between the current viewport and each cell as the cell-based viewport feature. To simplify the notation, we will use viewport feature in the rest of the paper. To obtain the overlap ratio between the viewport and cell i, we randomly generate R_i virtual points in cell i, and using the intrinsic and extrinsic matrices associated with the viewing camera (used for rendering according to the viewer's FoV) to calculate that. More specifically, given a set $\mathbf{P}$ of points in the world coordinates, we transform them to the viewing camera coordinates using:

$$\mathbf{P}_{\text{cam}} = \mathbf{E} \cdot \begin{bmatrix} \mathbf{P} \\ 1 \end{bmatrix},$$

where $\mathbf{E}$ is the extrinsic matrix determined by which represents the camera's orientation (yaw, pitch, roll), and a translation vector, which specifies the camera's coordinate system relative to the world coordinate system. We then project the transformed points onto the image plane using:

$$\mathbf{P}_{\text{img}} = \mathbf{I} \cdot \mathbf{P}_{\text{cam}},$$

where $\mathbf{I}$ is the intrinsic matrix, which depends on the intrinsic camera parameters including focal-length, etc. After we map all points on the camera's image, we can filter the points in actual FoV based on image dimensions and depth:

$$\text{Set of Points in Viewport} = \left\{ \mathbf{P}_{\text{img}} : \begin{vmatrix} 0 \le \mathbf{P}_{\text{img}}[0,:] < \text{width} \\ 0 \le \mathbf{P}_{\text{img}}[1,:] < \text{height} \\ d_{\text{near}} < \mathbf{P}_{\text{img}}[2,:] < d_{\text{far}} \end{vmatrix} \right\}$$

where d_{near} and d_{far} are the near and far plane as illustrated in Fig. 1. Denoting the number of points in the Set of Points in Viewport by K_i^τ at time τ in cell i, the viewport overlap ratio feature is defined as:

$$F^\tau = \left\{ f_i^\tau = \frac{K_i^\tau}{R_i}, i \in C \right\} \tag{2}$$

where R_i is the number of virtual points we randomly generated in cell i, and it's time independent.

4.2.2 Occlusion-aware Visibility . As we mentioned earlier, unlike 360-degree video, point cloud video has a unique property that requires consideration of occlusion in 3D space for any given 6DoF viewport. The number of visible points within a cell is crucial for describing occlusion. Given the 6DoF viewport coordinates and the point cloud object, we can use the HPR algorithm [15] to determine the number of visible points in a cell. However, applying HPR to high-density point clouds is time-consuming. In [9], the authors propose a cell-based occlusion estimation method. However, cell-based methods can introduce significant quantization errors. To achieve a better balance between accuracy and computational overhead, we calculate HPR on voxels instead of individual points, where each voxel covers an area of around $(1cm)^3$ cube in the 3D space. We assign the visibility of the voxel to all the points within. HPR on voxels has small accuracy loss from HPR with the original PCV, but remains more precise than the cell-based occlusion estimation. This enables real-time cell visibility calculations, achieving an average of approximately 45 fps on 8i data. After removing the hidden points, we apply intrinsic and extrinsic matrix mapping to the remaining points to exclude those points outside of the FoV. The final cell visibility feature for frame τ is then:

$$V^\tau = \left\{ v_i^\tau = \frac{Q_i^\tau}{N_i^\tau}, i \in C \right\}, \tag{3}$$

where N_i^τ is the number of points in cell i at time τ after downsampling, and Q_i^τ is the number of visible downsampled points. A further illustration for these three features is shown in Fig. 4.

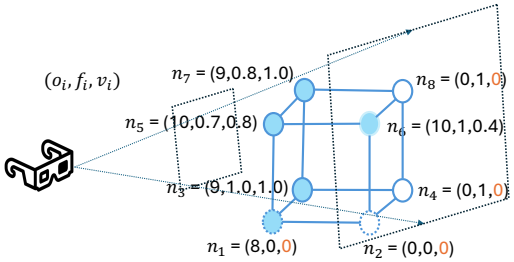


Figure 4: Given a viewer's 6DoF and the point cloud frame in 3D cells, we can get the occupancy feature, viewport overlap ratio feature and occlusion-aware visibility feature, as o_i, f_i, v_i. We have 8 nodes in total, and 5 of the nodes with color are occupied by points and have different numbers of points. For example n_6 has $o_i = 10$ points in total, the entire cell falls in the view port and hence $f_i = 1$, some points are occluded by other points and $v_i = 0.4$. For nodes without points, the occlusion-aware visibility feature is set as 0.

4.2.3 Angular Span and Visible Angular Span. Viewing distance is another important consideration for PCV streaming. An early subjective study on image viewing quality in [31] showed that the human perceived quality for an object depends on the *angular resolution*, i.e. the number of points per degree, which depends on the viewing distance and physical size of the object, as well as the image resolution. A recent subjective study on PCV in [9] also suggested that viewers' QoE changes significantly when they view a rendered point cloud object at different distances. Motivated by those studies, we model a cell's visual impact on the viewer as a function of the viewing distance. As illustrated in Fig. 5, all visible points in a cell are projected into a finite angular range within the viewer's viewport. The angular span depends on the distance of the cell to the viewpoint. A cell far away from the viewpoint contributes to only a small angular range of the viewport, and the angular resolution may saturate even at low point density [9, 36]. As a result, it has low impacts on the viewer's QoE and interest. Since we want to predict the utility of streaming each cell, we propose a new cell visibility feature to quantify the viewing distance effect. Similar to the pan-in-degree in [36], if a cell is completely inside the viewport and fully visible, as illustrated in Fig. 5, its contribution to the viewport can be quantified by the projected angular range θ,

which can be estimated as:

$$\theta = 2 \arctan\left(\frac{l}{2d}\right) \tag{4}$$

For a cell partially overlapping with the viewport, its contribution should be discounted by its overlap ratio with the viewport defined in (2):

$$A^\tau = \{\alpha_i^\tau = f_i^\tau * \theta_i^\tau, i \in C\}, \tag{5}$$

which is called *cell angular span* for cell i at time τ in the rest of the paper. Finally, if a cell is partially visible (due to occlusion), its contribution should be discounted by its visibility weight defined in (3):

$$B^\tau = \{\beta_i^\tau = v_i^\tau * \theta_i^\tau, i \in C\}, \tag{6}$$

which can be called the *cell visible angular span* for cell i at time τ.

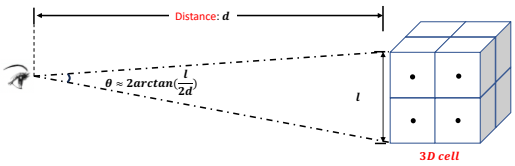


Figure 5: Inside a viewport, all points of a cell are viewed within an angular span determined by the viewing distance.

4.3 Locality and Distance Features

Other features $E^\tau = \{e_i^\tau, i \in C^\tau\}$ for cell i include the cell center coordinates and the distance from the cell center to the viewer's viewpoint. These features help the model account for changes in the viewer's position over time, similar to the way x, y, z coordinates are used in trajectory-based methods. It is important to note that the cell/node index remains fixed throughout the duration of the point cloud video.

4.4 TransGraph and GRU Model

We utilize a Transformer-based graph network [25] to build our graph model. The Transformer-based approach employs attention mechanisms [17] of transformers to capture dynamic relationships between neighbors. Given the hidden state $S_\tau = \{s_i, i \in C\}$ at time τ, we first concatenate S_τ with the raw node feature G_τ to form new feature for all cells:

$$H_\tau = \{h_i^\tau = s_i^\tau \oplus g_i^\tau, i \in C\} = S_\tau \oplus G_\tau \tag{7}$$

We then update the feature at each node using a weighted average of transformed features at its neighboring nodes, described by the following equations:

$$q_{c,i}^\tau = W_{c,q} h_i^\tau + b_{c,q} \tag{8}$$

$$k_{c,j}^\tau = W_{c,k} h_j^\tau + b_{c,k} \tag{9}$$

$$\alpha_{c,ij}^\tau = \frac{\langle q_{c,i}^\tau, k_{c,j}^\tau \rangle}{\sum_{u \in \mathcal{N}(i)} \langle q_{c,i}^\tau, k_{c,u}^\tau \rangle} \tag{10}$$

$$v_{c,j}^\tau = W_{c,v} h_j^\tau + b_{c,v} \tag{11}$$

$$\hat{h}_i^\tau = \sum_{j \in \mathcal{N}(i)} \alpha_{c,ij}^\tau v_{c,j}^\tau \tag{12}$$

where $W_{c,q}, W_{c,k}, W_{c,v}$ and $b_{c,q}, b_{c,k}, b_{c,v}$ are trainable weights and bias matrix and $\langle q, k \rangle = \exp\left(\frac{q^T k}{\sqrt{d}}\right)$ is the exponential scale dot-product function and d is the hidden size of each head [25]. We denote the above operations over all nodes collectively as

$$\hat{H}_\tau = TransGraph(H_\tau) \tag{13}$$

This is a single-layer graph model update; multiple layers can be used sequentially based on the output of the previous layer. For the GRU model, similar to the approach in [10], we employ a bi-directional GRU to capture temporal patterns. At each time step, the original hidden state s_i^τ at node i is updated to $\hat{h}_i^\tau$ using the graph operation described above, which propagates information from its neighbors in the graph. This updated hidden state will then serve as the hidden state for the next time step in the GRU model in both directions:

$$S_{t+1} = GRU_f(\hat{H}_t, G_t) \tag{14}$$

$$S'_{t-1} = GRU_r(\hat{H}'_t, G_t) \tag{15}$$

At the end of the GRU, one MLP module, which is shared by all cells, will take the combined hidden states S_{h+1} and $S_0^{'}$ of each cell to predict the visibility or viewport overlap ratio of this cell.

Since in a streaming system, when the system decides to send frame P^{h+f}, we can use occupancy feature at time $h+f$, O^{h+f}, as the mask on the output of MLP module to get the final prediction $\hat{Y}^{h+f}$. This masking operation, for example, can correct the situation when the predicted visibility is high but the actual occupancy is zero. The mask will be used for when Y^{h+f} is visibility feature V^τ and visible angular span feature B^τ.

$$\hat{Y}^{h+f} = M^{h+f} \circ \hat{Z}^{h+f} \tag{16}$$

where M^{h+f} is an indicator variable defined as:

$$m_i = \begin{cases} 1 & \text{if } o_i \geq 0 \\ 0 & \text{otherwise} \end{cases}$$

and Z^{h+f} is the output of MLP module. A binary mask is applied here because if a cell has no points, there is no need to transmit that cell. Conversely, we do not rely on the number of points within a cell, as having more points does not necessarily equate to higher visibility, which ultimately depends on the viewer's interest.

Combining all the elements, CellSight is capable of generating the cell features in real time using the historical viewport trajectory and point cloud video data. Conceptually, the graph model captures the interaction of the cell features among neighboring nodes, while the GRU model captures the temporal dynamics of the cell features. By concatenating the cell feature with the cell's hidden state as the input feature to the graph model, and using the output of the graph model as the updated hidden state of the GRU model, our overall model is able to exploit the impact of the hidden states of neighboring cells on the temporal dynamics of a chosen cell. The model's output may be cell visibility based on the streaming system or cell-based viewport feature, and the prediction accuracy will be assessed in the following section.

5 Evaluation

5.1 Dataset

We use two public point cloud video datasets with user FoV traces to train and evaluate **CellSight**. The first dataset is 8i [7], which contains four videos, `longdress`, `loot`, `redandblack`, and `soldier`, each with over one million points per frame, along with a 6DoF viewing navigation trajectory dataset [27]. Trajectories were collected from 26 users watching these videos (looped every 5 seconds) at 30 fps. Although the PCV loops every 5 seconds due to its original design, user trajectories remain continuous and can vary from loop to loop. Consequently, the only truly repeating feature is the cell occupancy feature (e.g., the number of points in each cell). All the other features reflect continuous user FoV trajectories and thus do not repeat. As a result, the looping content has minimal impact on our evaluation, since most features remain distinct and capture realistic viewing behaviors. In total, these videos amount to approximately 40k frames. We use the trajectories from the first three videos (`longdress`, `loot`, `redandblack`) for training, and the trajectory from the fourth video (`soldier`) is split into testing (first half) and validation (second half) sets. The second dataset is the Full Scene Volumetric Video Dataset (FSVVD) [13], which represents complex indoor scenes. Its accompanying trajectory dataset [14] comprises data from over 10 users watching six different point cloud videos. We selected four videos—`Chatting`, `Pulling trolley`, `News interviewing`, and `Sweeping`—and focused on the 12 users who watched all four. We excluded the other two videos because they contain dual full-scene rooms and differ significantly in scale. Similarly, we use the trajectories from the first three videos for training, while the last video (`Sweeping`) is equally divided between testing and validation. Since FSVVD is recorded at 60 fps, we downsample both the videos and the user trajectories to 30 fps to align with the 8i data. Overall, FSVVD provides over 50k frames. Notably, user FoVs in FSVVD are more dynamic than those in 8i, making prediction more challenging at longer prediction intervals.

5.2 Implementation Details

We pre-process the point cloud videos with Open3D [34] and generate cell features on a laptop with Apple M1 Pro chip, while training the graph model on an NVIDIA A100 GPU. Our graph model is implemented using PyTorch 2.3.0 and CUDA 12.2. We set the learning rate to 1e-4 and use a latent feature dimension of 128 for both the graph model and the GRU. Unless otherwise specified, a history length of 90 frames (3 seconds) is used for all methods. The model is trained for 30 epochs, with early stopping applied if the validation loss does not decrease for 5 consecutive epochs. For the 8i dataset, the entire space is partitioned into $5 \times 6 \times 8 = 240$ cells. The point cloud videos span approximately 1.8 m in height (Y-axis), with each 3D cell measuring around 0.2 m along all dimensions. For FSVVD, the space is partitioned into $7 \times 5 \times 8$ cells. The viewing camera image resolution is (1920*1080). The intrinsic matrix of camera is $\begin{bmatrix} f_x & 0 & c_x \\ 0 & f_y & c_y \\ 0 & 0 & 1 \end{bmatrix}$, where $f_x = f_y = 525$, $c_x = 1920/2$, $c_y = 1080/2$.

To deploy CellSight in a real system, we must consider two main sources of latency: feature calculation latency and model inference latency. Take the 8i dataset as an example: on Mac M1 Pro Chip, we achieved 32.5fps without occlusion-aware features and 18.1fps with occlusion-aware features. And model inference time is 0.279 seconds. The GPU memory usage is 45GB to train the model when the batch size is 32. For faster HPR, the downsampled frame has around 15k points.

5.3 Baseline Methods and Evaluation Metrics

We compare our proposed model against several trajectory-based baseline models to evaluate its performance in predicting cell visibility. The trajectory-based models first predict the future viewport 6DoF coordinates $(\hat{x}, \hat{y}, \hat{z}, \hat{\psi}, \hat{\theta}, \hat{\phi})$ based on the historical viewport trajectory. These predicted coordinates are then used to calculate cell features defined in the Methodology section. In contrast, CellSight directly predicts cell features, including cell viewport overlap ratio, angular span, occlusion-aware visibility, and visible angular span, which are ready to be directly used in point cloud streaming systems for bit-rate allocation to 3D cells.

For trajectory-based methods, the angle values of ψ, θ, and ϕ wrap around, e.g. from 2π to 0. There are different approaches to address this issue. In our approach, we convert the orientation coordinates to the sine and cosine domains, using two coordinates to represent each angle coordinate. After prediction, we convert these coordinates back to the original angle value using the arctangent function. The baselines are as follows:

- **Linear Regression (LR):** We employ linear regression to predict each 6DoF coordinate individually by using a linear combination of historical values over a fixed window. This simple model serves as a fundamental baseline and, as demonstrated in [9], can sometimes yield higher FoV prediction accuracy than MLP. To accommodate different scenarios, we implement two variants—LR90 and LR30—using history window lengths of 90 and 30 frames, respectively.
- **Truncated Linear Regression (TLR):** This method leverages the most recent monotonically increasing or decreasing segment of the history window to linearly extrapolate future values. TLR has shown good performance in short-term FoV prediction [19] and, similar to LR, is applied to predict each coordinate individually.
- **Multi-task Multilayer Perceptron (MLP):** Following [11], we utilize a feedforward neural network to capture nonlinear relationships between the input features and the future FoV. As demonstrated in [11], predicting all coordinates simultaneously using the entire history window can enhance performance. Consequently, we use an MLP model with two hidden layers, each comprising 60 neurons in a fully connected architecture, and employ the ReLU activation function. This configuration is consistent with [9, 11, 20].
- **Multi-task LSTM (LSTM):** LSTM, a type of recurrent neural network (RNN), is well-suited for sequence prediction by learning long-term dependencies in time series data. In line with [11], we implement a two-layer LSTM with 60 neurons per layer to predict future FoV coordinates from historical data, with all coordinates predicted simultaneously.

There are also some other SOTA methods, e.g. [9, 20]. For [20], it is not open-sourced and the system is hard to reproduce. But we

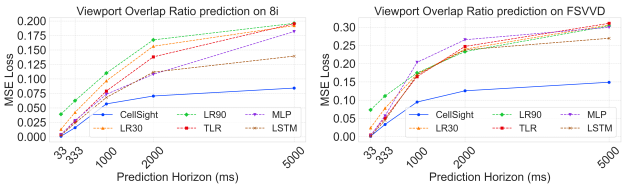
infer its performance based on its reported improvement over [9]. Authors of [9] built a point cloud streaming system considering Viewport Visibility (VV), Occlusion Visibility (OV), and Distance Visibility (DV), which are counterparts of our cell-based viewport feature, visibility feature and angular span feature. MLP and LSTM are used for FoV prediction. For the cell visible angular span prediction, we use MLP and LSTM to compare its performance with CellSight. Besides, we also use point-wise HPR to estimate the occlusion visibility to improve the its cell-based occlusion estimation.

We evaluate the performance of CellSight and the baselines by comparing the MSE across all cells for the predicted frame, including empty cells. This is because some cell may be temporarily empty when the objects move out of the space occupied by the cell. In CellSight, we will predict the visibility weight for each cell and the predicted weights can be directly used to determine the bit-rates allocated to all cells, not just cells falling into the predicted viewport. MSE is an accuracy metric for the visibility weight prediction. More accurate prediction leads to better cell bit-rate allocation for improved streaming quality. Besides, we use the R^2 score to assess the MSE relative to the variance of the ground truth variables, defined as $R^2 = 1 - \frac{\text{MSE}}{\text{GT variance}} = 1 - \frac{\sum_i \sum_f (y_{if} - \hat{y}_{if})^2}{\sum_i \sum_f (y_{if} - \bar{y})^2}$. In our experiments, we calculate residual variance and total variance over all cells and all frames. It measures the proportion of variance in the dependent variable that is predictable from the independent variables. An R^2 score of 1 indicates perfect prediction, and a ground truth mean prediction will a score of 0. It can be negative when model introduces more error than simply using the mean of the target data as the prediction. Good visual results are demonstrated when $R^2 >$ 0.43 in Fig. 11.

5.4 Prediction without Considering Occlusion: Cell Viewport Overlap Ratio and Angular Span

In this section, we will evaluate the viewport and angular span prediction. As discussed earlier, occlusion takes time to calculate, and even though we can downsample the point cloud video for HPR to achieve a real-time prediction, it is still a big overhead for a streaming system, which may be serving many clients. For some practical streaming system, sometimes it may be more appropriate to predict cell viewport overlap ratio without considering occlusion. We also notice that in 8i and FSVVD, if we only transmit the points inside the viewport (without considering occlusion), the total traffic for streaming can still be reduced to 1/2 and 1/6 of the original traffic. We apply CellSight on the cell viewport overlap ratio prediction and angular span prediction.

5.4.1 Viewport Overlap Ratio. For cell viewport overlap ratio prediction, the input features are occupancy feature, cell-based viewport feature, and other features, i.e., $G = \{O, F, E\}$. The Model will predict F. Fig. 6a illustrates the MSE for viewport overlap ratio prediction on the 8i dataset, demonstrating that CellSight consistently outperforms the others across varying prediction horizons, with substantially more gains at longer horizons. Similarly, Fig. 6b depicts the MSE for the prediction on the FSVVD dataset, where CellSight also achieves the lowest error compared to baseline methods, highlighting its robustness and accuracy.



(a) 8i Cell Viewport Overlap Ratio **(b) FSVVD Cell Viewport Overlap Ratio**

Figure 6: MSE losses for Cell Viewport Overlap Ratio

Table. 2 presents the R^2 scores for viewport overlap ratio prediction on the 8i dataset across different prediction horizons. Our proposed model consistently achieves the highest R^2 scores at all time steps. And the performance advantage of CellSight becomes more pronounced as the prediction horizon increases. Results on FSVVD dataset in Table. 3 show similar trend to 8i dataset, and CellSight outperforms the baselines, especially at longer prediction horizons. Since the user's trajectory data are more dynamic in FSVVD datset, some methods even produce negative R^2 scores at long time horizons, indicating poor performance. But CellSight still maintains positive scores, demonstrating its robustness and predictive capability even under challenging conditions.

Table 2: Viewport Overlap Ratio R² Scores on 8i

Time (ms)	LR30	LR90	TLR	MLP	LSTM	CellSight
33	0.941	0.824	0.982	0.983	0.989	**0.995**
333	0.809	0.718	0.879	0.873	0.893	**0.928**
1000	0.563	0.504	0.644	0.669	0.693	**0.743**
2000	0.294	0.246	0.378	0.514	0.497	**0.682**
5000	0.135	0.114	0.116	0.178	0.370	**0.618**

Table 3: Viewport Overlap Ratio R² Scores on FSVVD

Time (ms)	LR30	LR90	TLR	MLP	LSTM	CellSight
33	0.876	0.621	0.978	0.987	**0.995**	0.994
333	0.597	0.425	0.738	0.706	0.769	**0.828**
1000	0.112	0.100	0.153	-0.049	0.127	**0.512**
2000	-0.240	-0.195	-0.268	-0.362	-0.220	**0.355**
5000	-0.562	-0.558	-0.595	-0.539	-0.385	**0.232**

We also visualize the predicted viewport overlap ratio. We choose 2 second ahead as the prediction horizon to visualize it. In Fig. 7, the MSE loss for CellSight and LSTM are 0.05 and 0.14, respectively. Visually, our predicted viewport overlap ratio is closer to the ground-truth, especially at the edges of the viewport. LSTM prediction missed a large part of the leg, hands and gun, since its FoV coordinates were overly influenced by the historical trajectory alone. Our prediction can cover more around the edges of the viewport. In addition, CellSight exhibits smoother boundaries compared to trajectory-based models. This smoothness arises because neural networks typically filter out high-frequency signals, resulting in a naturally smoother output. When the predicted cell visibility is used to determine streaming bandwidth allocation, cells surrounding the FoV are also allocated with non-zero rates. This increases the streaming system's robustness against FoV prediction errors.

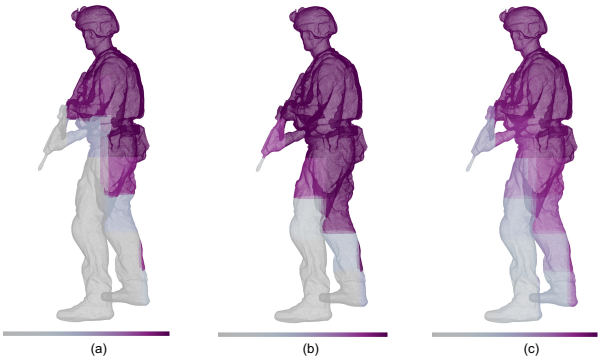


Figure 7: Cell viewport overlap ratio prediction results on the 8i dataset, where the prediction horizon is 2000ms. Visual comparison of predicted viewport using (a) LSTM model, (b) Ground Truth, and (c) CellSight. The color represents the prediction confidence, transitioning from dark (low confidence) to bright (high confidence).

Beyond using one video for testing/validation to provide a straightforward demonstration of generalization, we also conduct cross-validation by rotating the testing/validation videos among four PCVs. The cross-validation results are shown in Table. 4. CellSight has lower MSE and higher R^2 compared with LSTM, which further demonstrates the generalization of CellSight.

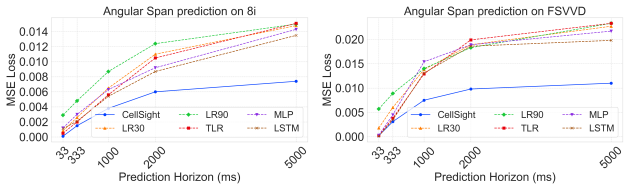
Table 4: Viewport Overlap Ratio Prediction Cross Validation on 8i

Testing	CellSight		LSTM	
	MSE↓	R^2↑	MSE↓	R^2↑
Longdress	0.0714	0.675	0.095	0.573
Redandblack	0.0610	0.720	0.100	0.545
Loot	0.0683	0.679	0.099	0.534
Soldier	0.0710	0.682	0.112	0.497
Average	**0.0679**	**0.689**	0.1015	0.5373

5.4.2 Angular Span. To predict the angular span for each cell, the history angular spans for all cells are added beyond the features of viewport overlap ratio prediction feature set, i.e., $G = \{O, F, E, A\}$. The model will predict A. The prediction MSE is reported in Fig. 8a and Fig. 8b respectively. CellSight consistently outperforms baselines, except in one case for the FSVVD dataset. In that case, LSTM is slightly better than CellSight for the very short predicted horizon of 33 ms (1 frame). R^2 score is displayed in Table 5 and Table 6.

Table 5: Angular Span R^2 Scores on 8i

Time (ms)	LR30	LR90	TLR	MLP	LSTM	CellSight
33	0.957	0.835	0.975	0.934	0.941	**0.994**
333	0.857	0.733	0.890	0.835	0.882	**0.917**
1000	0.639	0.516	0.687	0.651	0.696	**0.787**
2000	0.387	0.308	0.418	0.490	0.516	**0.668**
5000	0.174	0.162	0.156	0.199	0.246	**0.585**



(a) 8i Angular Span **(b) FSVVD Angular Span**

Figure 8: MSE losses for Angular Span

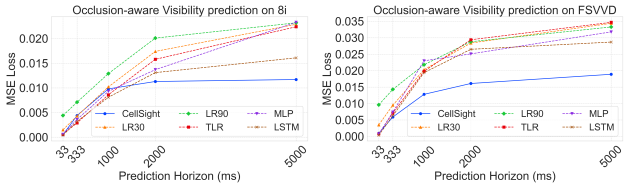
Table 6: Angular Span R^2 Scores on FSVVD

Time (ms)	LR30	LR90	TLR	MLP	LSTM	CellSight
33	0.881	0.617	0.978	0.983	**0.993**	0.988
333	0.594	0.403	0.743	0.698	0.766	**0.794**
1000	0.069	0.058	0.131	-0.033	0.117	**0.493**
2000	-0.275	-0.227	-0.334	-0.269	-0.251	**0.338**
5000	-0.524	-0.560	-0.564	-0.453	-0.329	**0.261**

5.5 Prediction considering Occlusion: Cell Occlusion-aware Visibility and Visible Angular Span

To achieve more bandwidth saving in point cloud streaming, FoV prediction can additionally consider occlusion after HPR. In this section, we evaluate the occlusion-aware visibility and visible angular span predictions.

5.5.1 Occlusion-aware Visibility. For the cell occlusion-aware visibility prediction, beyond all features used in cell viewport overlap ratio prediction, the cell occlusion-aware visibility history is added, i.e., $G = \{O, F, V, E\}$. The model will predict V. In Fig. 9a, while our proposed model demonstrates superior performance in longer prediction horizons, some baseline methods, such as LSTM, achieve slightly better results in shorter horizons due to the less dynamic nature of user behaviors in the 8i dataset. However, CellSight's long-term accuracy shows its robustness and effectiveness. For the more dynamic user FoV data FSVVD, as shown in Fig. 9b, CellSight consistently achieves the best performance across all prediction horizons. This highlights the adaptability and reliability of CellSight in handling scenarios with frequent changes in user focus, making it well-suited for dynamic FoV prediction. R^2 scores on 8i and FSVVD datasets are shown in Table 7 and Table 8. CellSight achieves the best performance for medium and long term horizon, especially on more dynamic FSVVD dataset.



(a) 8i Occlusion-aware Visibility **(b) FSVVD Occlusion-aware Visibility**

Figure 9: MSE losses for Occlusion-aware Visibility

Table 7: Occlusion-aware Visibility R^2 Scores on 8i

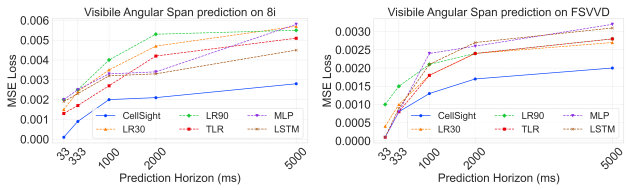
Time (ms)	LR30	LR90	TLR	MLP	LSTM	CellSight
33	0.965	0.897	0.986	0.985	**0.990**	0.987
333	0.898	0.836	**0.934**	0.916	0.926	0.896
1000	0.761	0.702	0.803	0.786	**0.814**	0.774
2000	0.595	0.537	0.636	0.684	0.699	**0.739**
5000	0.468	0.475	0.494	0.473	0.637	**0.736**

Table 8: Occlusion-aware Visibility R^2 Scores on FSVVD

Time (ms)	LR30	LR90	TLR	MLP	LSTM	CellSight
33	0.883	0.683	0.974	0.978	**0.988**	0.969
333	0.685	0.530	0.765	0.757	0.792	**0.807**
1000	0.331	0.284	0.349	0.244	0.362	**0.578**
2000	0.067	0.074	0.053	0.193	0.144	**0.480**
5000	-0.094	-0.046	-0.092	0.001	0.094	**0.402**

5.5.2 Visible Angular Span. To consider viewport, visibility and angular span all together, we evaluate CellSight when predicting the visible angular span β on the two datasets. We incorporate the cell's visible angular span feature β history along with the history of all the other features, i.e., $G = \{O, F, V, E, A, B\}$. The model will predict B. Figure 10a and Fig. 10b show that CellSight consistently achieves the best performance, particularly for longer prediction horizons and even in less dynamic user scenarios, highlighting its ability to effectively predict visible angular span. These results show that CellSight is not only accurate in combining viewport, visibility, and angular features but also robust across datasets with varying levels of user interaction dynamics. Table 9 and Table 10 present the R^2 scores for visible angular span for 8i and FSVVD.

Additionally, we visualize the visible angular span prediction on one frame, where prediction horizon is 2000ms in FSVVD data in Fig. 11, where the prediction MSE values for CellSight and LSTM are 0.0016 and 0.0026, respectively, which is close to the average MSE across all frames for both methods in Fig. 10b. The prediction by CellSight closely matches the ground-truth, whereas LSTM produces significantly different results due to error amplification in the degrees of freedom, leading to substantial discrepancies in the final visible cell angular spans.



(a) 8i Visible Angular Span **(b) FSVVD Visible Angular Span**

Figure 10: MSE losses for Visible Angular Span

6 Conclusion & Future Work

In this paper, we introduce **CellSight**, a novel approach for predicting long-term cell visibility in PCV. CellSight leverages both the spatial and temporal dynamics of PCV content and viewer behaviors to directly predict cell visibility for streaming systems. It outperforms state-of-the-art methods in terms of accuracy and robustness, particularly at long prediction horizons (beyond 2 seconds). By integrating Transformer-based Graph Neural Networks with recurrent neural networks, CellSight efficiently captures the complex interactions between PCV content and viewer interests, as well as the correlations among neighboring cells. Unlike the trajectory-based FoV prediction methods, which often overlook the full spatial context, our method yields more accurate and stable predictions for long-term 6-DoF FoV prediction.

For future work, we plan to utilize the predicted cell visibility metrics to guide cell-level bandwidth allocation in on-demand PCV streaming. Furthermore, while the current approach relies on a centralized architecture—where all computations are performed on the streaming server—we intend to explore distributed computing strategies to reduce server load and enhance system scalability. We have made the code and technical report publicly available[2] to support further research and development in this area.

Table 9: Visible Angular Span R^2 Scores on 8i

Time (ms)	LR30	LR90	TLR	MLP	LSTM	CellSight
33	0.873	0.827	0.888	0.826	0.829	**0.990**
333	0.788	0.777	0.855	0.779	0.794	**0.924**
1000	0.687	0.650	0.766	0.692	0.722	**0.828**
2000	0.587	0.550	0.639	0.707	0.717	**0.822**
5000	0.504	0.530	0.562	0.503	0.614	**0.760**

Table 10: Visible Angular Span R^2 Scores on FSVVD

Time (ms)	LR30	LR90	TLR	MLP	LSTM	CellSight
33	0.871	0.649	0.961	0.965	**0.974**	0.955
333	0.671	0.484	**0.741**	0.728	0.713	0.728
1000	0.388	0.297	0.400	0.175	0.286	**0.544**
2000	0.189	0.210	0.206	0.136	0.111	**0.436**
5000	0.124	0.095	0.087	-0.041	-0.008	**0.364**

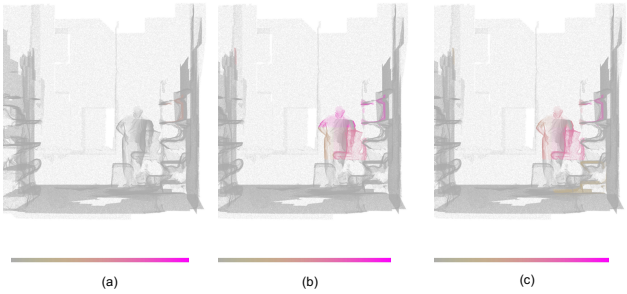


Figure 11: Visible angular span prediction for each cell on one frame of FSVVD dataset, where the prediction horizon is 2000ms. (a) LSTM model, (b) Ground Truth, and (c) CellSight. The color transition from gray to red corresponds to visible angular span from small to large.

7 Acknowledgments

This work was supported in part by the National Science Foundation (NSF) grant 2312839.

[2]https://github.com/chenli1996/CellSight

References

[1] Evangelos Alexiou, Yana Nehmé, Emin Zerman, Irene Viola, Guillaume Lavoué, Ali Ak, Aljosa Smolic, Patrick Le Callet, and Pablo Cesar. 2023. Subjective and objective quality assessment for volumetric video. In *Immersive Video Technologies*. Elsevier, 501–552.

[2] Yixuan Ban, Lan Xie, Zhimin Xu, Xinggong Zhang, Zongming Guo, and Yue Wang. 2018. Cub360: Exploiting cross-users behaviors for viewport prediction in 360 video adaptive streaming. In *2018 IEEE International Conference on Multimedia and Expo (ICME)*. IEEE, 1–6.

[3] Salima Bourbia, Ayoub Karine, Aladine Chetouani, Mohammed El Hassouni, and Maher Jridi. 2024. Blind point cloud quality assessment via 3D visual saliency and point-based neural network. In *The 13th International Conference on Image Processing Theory, Tools and Applications*.

[4] Ruopeng Chen, Mengbai Xiao, Dongxiao Yu, Guanghui Zhang, and Yao Liu. 2023. patchVVC: A real-time compression framework for streaming volumetric videos. In *Proceedings of the 14th Conference on ACM Multimedia Systems*. 119–129.

[5] Jaeyeol Choi, Jong-Beom Jeong, Soonbin Lee, and Eun-Seok Ryu. 2022. Overview of the Volumetric Video Capturing System for Immersive Media. In *2022 13th International Conference on Information and Communication Technology Convergence (ICTC)*. IEEE, 574–577.

[6] Yuxin Du, Fan Bai, Tiejun Huang, and Bo Zhao. 2023. Segvol: Universal and interactive volumetric medical image segmentation. *arXiv preprint arXiv:2311.13385* (2023).

[7] Eugene d'Eon, Bob Harrison, Taos Myers, and Philip A Chou. 2017. 8i voxelized full bodies-a voxelized point cloud dataset. *ISO/IEC JTC1/SC29 Joint WG11/WG1 (MPEG/JPEG) input document WG11M40059/WG1M74006* 7, 8 (2017), 11.

[8] Ching-Ling Fan, Jean Lee, Wen-Chih Lo, Chun-Ying Huang, Kuan-Ta Chen, and Cheng-Hsin Hsu. 2017. Fixation prediction for 360 video streaming in head-mounted virtual reality. In *Proceedings of the 27th workshop on network and operating systems support for digital audio and video*. 67–72.

[9] Bo Han, Yu Liu, and Feng Qian. 2020. ViVo: Visibility-aware mobile volumetric video streaming. In *Proceedings of the 26th annual international conference on mobile computing and networking*. 1–13.

[10] Hangtao He, Linyu Su, and Kejiang Ye. 2023. Graphgru: A graph neural network model for resource prediction in microservice cluster. In *2022 IEEE 28th International Conference on Parallel and Distributed Systems (ICPADS)*. IEEE, 499–506.

[11] Xueshi Hou and Sujit Dey. 2020. Motion Prediction and Pre-Rendering at the Edge to Enable Ultra-Low Latency Mobile 6DoF Experiences. IEEE Open Journal of the Communications Society 1 (2020), 1674–1690.

[12] Kaiyuan Hu, Yongting Chen, Kaiying Han, Junhua Liu, Haowen Yang, Yili Jin, Boyan Li, and Fangxin Wang. 2023. LiveVV: Human-Centered Live Volumetric Video Streaming System. *arXiv preprint arXiv:2310.08205* (2023).

[13] Kaiyuan Hu, Yili Jin, Haowen Yang, Junhua Liu, and Fangxin Wang. 2023. FSVVD: A dataset of full scene volumetric video. In *Proceedings of the 14th Conference on ACM Multimedia Systems*. 410–415.

[14] Kaiyuan Hu, Haowen Yang, Yili Jin, Junhua Liu, Yongting Chen, Miao Zhang, and Fangxin Wang. 2023. Understanding user behavior in volumetric video watching: Dataset, analysis and prediction. In *Proceedings of the 31st ACM International Conference on Multimedia*. 1108–1116.

[15] Sagi Katz, Ayellet Tal, and Ronen Basri. 2007. Direct visibility of point sets. In *ACM SIGGRAPH 2007 papers*. 24–es.

[16] Muhammad Jalal Khan, Abdelhak Bentaleb, and Saad Harous. 2021. Can accurate future bandwidth prediction improve volumetric video streaming experience?. In *2021 International Wireless Communications and Mobile Computing (IWCMC)*. IEEE, 1041–1047.

[17] Chen Li, Xiaoyu Wang, Tongyu Zong, Houwei Cao, and Yong Liu. 2023. Predictive edge caching through deep mining of sequential patterns in user content retrievals. *Computer Networks* 233 (2023), 109866.

[18] Chen Li, Tingwei Ye, Tongyu Zong, Liyang Sun, Houwei Cao, and Yong Liu. 2023. Coffee: Cost-Effective Edge Caching for 360 Degree Live Video Streaming. *arXiv preprint arXiv:2312.13470* (2023).

[19] Chenge Li, Weixi Zhang, Yong Liu, and Yao Wang. 2019. Very long term field of view prediction for 360-degree video streaming. In *2019 IEEE Conference on Multimedia Information Processing and Retrieval (MIPR)*. IEEE, 297–302.

[20] Junhua Liu, Boxiang Zhu, Fangxin Wang, Yili Jin, Wenyi Zhang, Zihan Xu, and Shuguang Cui. 2023. Cav3: Cache-assisted viewport adaptive volumetric video streaming. In *2023 IEEE Conference Virtual Reality and 3D User Interfaces (VR)*. IEEE, 173–183.

[21] Yu Liu, Bo Han, Feng Qian, Arvind Narayanan, and Zhi-Li Zhang. 2022. Vues: Practical mobile volumetric video streaming through multiview transcoding. In *Proceedings of the 28th Annual International Conference on Mobile Computing And Networking*. 514–527.

[22] Ohji Nakagami, Sebastien Lasserre, Sugio Toshiyasu, and Marius Preda. 2023. White paper on G-PCC. In *ISO/IEC JTC 1/SC 29/AG 03 N0111*. https://www.mpeg.org/wp-content/uploads/mpeg_meetings/142_Antalya/w22804.zip

[23] Minh Nguyen, Shivi Vats, Sam Van Damme, Jeroen Van Der Hooft, Maria Torres Vega, Tim Wauters, Christian Timmerer, and Hermann Hellwagner. 2023. Impact of quality and distance on the perception of point clouds in mixed reality. In *2023 15th International Conference on Quality of Multimedia Experience (QoMEX)*. IEEE, 87–90.

[24] Michael Rudolph, Aron Riemenschneider, and Amr Rizk. 2024. Progressive Coding for Deep Learning based Point Cloud Attribute Compression. In *Proceedings of the 16th International Workshop on Immersive Mixed and Virtual Environment Systems*. 78–84.

[25] Yunsheng Shi, Zhengjie Huang, Shikun Feng, Hui Zhong, Wenjin Wang, and Yu Sun. 2020. Masked label prediction: Unified message passing model for semi-supervised classification. *arXiv preprint arXiv:2009.03509* (2020).

[26] Han Song, Zhongche Qu, Zhi Zhang, Zihan Ye, and Cong Liu. 2024. Advancements in translation accuracy for stereo visual-inertial initialization. In *2024 9th Asia-Pacific Conference on Intelligent Robot Systems (ACIRS)*. IEEE, 210–215.

[27] Shishir Subramanyam, Irene Viola, Alan Hanjalic, and Pablo Cesar. 2020. User centered adaptive streaming of dynamic point clouds with low complexity tiling. In *Proceedings of the 28th ACM international conference on multimedia*. 3669–3677.

[28] Liyang Sun, Fanyi Duanmu, Yong Liu, Yao Wang, Yinghua Ye, Hang Shi, and David Dai. 2019. A two-tier system for on-demand streaming of 360 degree video over dynamic networks. *IEEE Journal on Emerging and Selected Topics in Circuits and Systems* 9, 1 (2019), 43–57.

[29] Jeroen van der Hooft, Maria Torres Vega, Christian Timmerer, Ali C Begen, Filip De Turck, and Raimund Schatz. 2020. Objective and subjective QoE evaluation for adaptive point cloud streaming. In *2020 twelfth international conference on quality of multimedia experience (QoMEX)*. IEEE, 1–6.

[30] Lisha Wang, Chenglin Li, Wenrui Dai, Junni Zou, and Hongkai Xiong. 2021. QoE-driven and tile-based adaptive streaming for point clouds. In *ICASSP 2021-2021 IEEE International Conference on Acoustics, Speech and Signal Processing (ICASSP)*. IEEE, 1930–1934.

[31] Joyce HDM Westerink and Jacques AJ Roufs. 1989. Subjective image quality as a function of viewing distance, resolution, and picture size. *SMPTE journal* 98, 2 (1989), 113–119.

[32] Xihe Yang, Xingyu Chen, Daiheng Gao, Shaohui Wang, Xiaoguang Han, and Baoyuan Wang. 2024. Have-fun: Human avatar reconstruction from few-shot unconstrained images. In *Proceedings of the IEEE/CVF Conference on Computer Vision and Pattern Recognition*. 742–752.

[33] Yunxiang Zhang, Benjamin Liang, Boyuan Chen, Paul M Torrens, S Farokh Atashzar, Dahua Lin, and Qi Sun. 2022. Force-aware interface via electromyography for natural VR/AR interaction. *ACM Transactions on Graphics (TOG)* 41, 6 (2022), 1–18.

[34] Qian-Yi Zhou, Jaesik Park, and Vladlen Koltun. 2018. Open3D: A Modern Library for 3D Data Processing. *arXiv:1801.09847* (2018).

[35] Tongyu Zong, Chen Li, Yuanyuan Lei, Guangyu Li, Houwei Cao, and Yong Liu. 2022. Cocktail edge caching: Ride dynamic trends of content popularity with ensemble learning. *IEEE/ACM Transactions on Networking* 31, 1 (2022), 208–219.

[36] Tongyu Zong, Yixiang Mao, Chen Li, Yong Liu, and Yao Wang. 2023. Progressive Frame Patching for FoV-based Point Cloud Video Streaming. *arXiv preprint arXiv:2303.08336* (2023).

StreamWise: An Intelligent Content Steering for DASH

Chidambar Joshi*
Concordia University, Canada
chidambar.joshi@mail.concordia.ca

Jashanjot Singh Sidhu*
Concordia University, Canada
jashanjot.sidhu@mail.concordia.ca

Abdelhak Bentaleb
Concordia University, Canada
abdelhak.bentaleb@concordia.ca

Abstract

Multi-CDN strategies have become increasingly important in enhancing Quality of Experience (QoE) for adaptive video streaming. The recent development of the content steering standard (ETSI TS 103 998) aims to facilitate real-time decision-making about the best-performing CDN by gathering statistics from both players and CDNs. However, this task presents significant challenges. Existing solutions for CDN selection rely on heuristics, which often fail to adapt to diverse network conditions and suffer from issues such as prolonged CDN switching delays and/or complex implementation requirements, resulting in poor QoE. To address these limitations, we present StreamWise—a learning-based solution for real-time CDN selection that works effectively for both on-demand and live adaptive video streaming. StreamWise implements on the content steering standard, leveraging a deep reinforcement learning (DRL) framework to learn and predict in real-time the optimal CDN selection policy by continuously interacting with the environment. Our solution adapts in real-time to network conditions, content characteristics, and viewer requirements ensuring the delivery of high-quality content to users. We evaluate StreamWise through extensive trace-driven emulation-based experiments, demonstrating its superior performance compared to conventional CDN selection strategies. Our results demonstrate a significant improvement in VMAF by ∼8.5% with ∼1.5x higher average bitrate and QoE improvement of ∼48% with minimal rebuffering events for video-on-demand streaming and an improvement in VMAF by ∼7%, with ∼1.2x higher average bitrate and a QoE improvement of ∼22% while substantially reducing the rebuffering events by ∼10× for live streaming.

CCS Concepts

• **Information systems → Multimedia streaming**.

Keywords

Multi-CDN, Real-time CDN Selection, QoE, DASH, DRL.

ACM Reference Format:
Chidambar Joshi, Jashanjot Singh Sidhu, and Abdelhak Bentaleb. 2025. StreamWise: An Intelligent Content Steering for DASH. In *ACM Multimedia Systems Conference 2025 (MMSys '25), March 31-April 4, 2025, Stellenbosch, South Africa.* ACM, New York, NY, USA, 11 pages. https://doi.org/10.1145/3712676.3714436

*The first two authors contributed equally to this work.

Permission to make digital or hard copies of all or part of this work for personal or classroom use is granted without fee provided that copies are not made or distributed for profit or commercial advantage and that copies bear this notice and the full citation on the first page. Copyrights for components of this work owned by others than the author(s) must be honored. Abstracting with credit is permitted. To copy otherwise, or republish, to post on servers or to redistribute to lists, requires prior specific permission and/or a fee. Request permissions from permissions@acm.org.
MMSys '25, Stellenbosch, South Africa
© 2025 Copyright held by the owner/author(s). Publication rights licensed to ACM.
ACM ISBN 979-8-4007-1467-2/25/03
https://doi.org/10.1145/3712676.3714436

1 Introduction

HTTP Adaptive Streaming (HAS) [7] now represents a substantial portion of global internet traffic, a trend expected to accelerate in the coming years [40]. As HAS traffic continues to grow rapidly, viewers' expectations for quality of experience (QoE) have risen significantly, with increasing demand for high video quality, minimal rebuffering, and seamless playback. To meet these viewers' expectations, content providers are continuously striving to optimize video delivery workflow, ensuring a superior viewing experience while keeping delivery costs manageable.

To ensure optimal delivery of content to end users, most providers rely on Content Delivery Networks (CDNs) [52] to distribute video streams across different geographic locations. CDNs help minimize latency, enhance scalability, and reduce delivery costs. However, relying on a single CDN presents significant risks [6, 17, 36], such as outages or performance bottlenecks, which can lead to service disruptions and severely impact QoE. For instance, during high-traffic events or CDN failures, a single CDN may struggle to deliver consistent performance across different regions or networks, making it difficult to guarantee smooth playback.

To mitigate these challenges, large content providers increasingly employ *multi-CDN* architectures [6, 24, 36, 45]. This approach combines the strengths of various CDNs, allowing for intelligent traffic routing and dynamic load balancing across multiple vendors, thereby improving performance and reducing the likelihood of failure, especially when single CDNs face outage issues [28], which can lead to reduced Quality of Service. As a result, multi-CDN setups have become crucial for maintaining high QoE [6, 36]. In particular, *CDN selection* (also called *CDN switching*) strategies have gained traction, allowing providers to switch dynamically between CDNs midstream to ensure the best possible streaming experience based on real-time statistics from the system, *i.e.*, the video players and CDNs. This strategy has been standardized under the concept of *content steering* (ETSI TS 103 998) [1, 36], where CDN selection is automated and continuously optimized to match network conditions and viewer demand. Despite the advantages of multi-CDN, existing methods [6, 8, 26, 36] predominantly rely on heuristic-based approaches for CDN switching. These heuristics often struggle to adapt effectively to diverse and rapidly changing network conditions, and viewer demands, which can lead to sub-optimal decisions and a degradation in QoE. For example, strategies [6, 8, 26, 36] based on DNS-based switching or manifest rewriting may suffer from delays, inflexibility, and errors while client-side switching can be difficult to implement and manage at scale. Therefore, developing a good multi-CDN strategy is not a trivial task.

1.1 Problem and Motivation

To confirm the limitations of a single CDN, we conducted experiments for both video-on-demand (VoD) and low-latency live streaming (LLL) scenarios. The test setup consisted of three components:

CDN-side, traffic shaper, and client-side, all deployed in Docker containers and connected via Docker Compose networks to regulate traffic flow. A traffic shaper (tc Netem) was used to emulate various network conditions based on real-world traces (refer to Figure 5). On the client-side, a DASH client (goDASH [33]) was used with the Merina [23] learning-based adaptive bitrate (ABR) algorithm [7]. On the CDN-side, we utilized four CDNs. For the traffic shaper, we randomly selected five network traces with varying bandwidth fluctuation patterns from existing datasets. Each CDN connection runs a different trace: CDN1 (3G Norway) [3], CDN2 (F4G5G, described in Section 4.2), CDN3 (Twitch) [49], and CDN4 (Cascade) [20], while the client-side connection uses a Netflix 5G trace [32]. The video and player parameters for each scenario are outlined in Table 2. We compared the performance of the single-CDN approach to a multi-CDN setup (represented by `StreamWise`), evaluating the following metrics: video quality (Video Multi-Method Assessment Fusion (VMAF)) [34] and rebuffering duration ratio (RD). The results are highlighted in Figure 1.

The results demonstrate the advantages of a multi-CDN strategy, like `StreamWise`, over single-CDN solutions in enhancing viewer QoE for both LLL and VoD. In LLL scenarios, (Figure 1; left), `StreamWise` shows a remarkable improvement in both VMAF and RD, outperforming CDN1 by ~7%, CDN4 by ~3%, and slightly surpassing CDN2 and CDN3. Our multi-CDN strategy outperforms other competitors by significantly reducing rebuffering by ~20% on average, ensuring smoother playback. In VoD scenarios (Figure 1; right), the multi-CDN strategy continues to dominate, achieving VMAF improvements of 27%, 6%, 9%, and 11%, compared to CDN1 through CDN4, respectively.

1.2 `StreamWise`: A DRL Multi-CDN Strategy

To overcome the limitations of single-CDN and heuristic-based multi-CDN strategies, we propose `StreamWise`, a DRL-based solution for content steering that dynamically selects the best-performing CDN to maximize viewer QoE. `StreamWise` leverages an Actor-Critic neural network (NN) architecture [5] combined with the Proximal Policy Optimization (PPO) algorithm [41] to learn optimal CDN selection policies. The RMSProp optimizer is used to update network parameters efficiently, improving convergence during training. In real-time, the NN processes vectors of aggregated round-trip time (RTT), selected bitrates, and rebuffering events from all clients and outputs the optimal CDN for each client, ensuring QoE is maximized. To facilitate seamless cooperation between clients, CDNs, and steering server without adding unnecessary complexity, `StreamWise` leverages the emerging Common Media Client Data (CMCD) [12] and Common Media Server Data (CMSD) [13] standards. CMCD defines a standardized set of data collected by the client, which is transmitted alongside HTTP requests as query parameters or header extensions to the steering server running `StreamWise`. On the CDN-side, CMSD enables the steering server to collect CDN logs through HTTP response headers, allowing for efficient, real-time communication and decision-making without introducing additional overhead.

We evaluate the performance of `StreamWise` against several multi-CDN strategies by running real-world trace-driven experiments. Our results demonstrate that `StreamWise` significantly improves QoE in both VoD and LLL scenarios. Specifically, it achieves

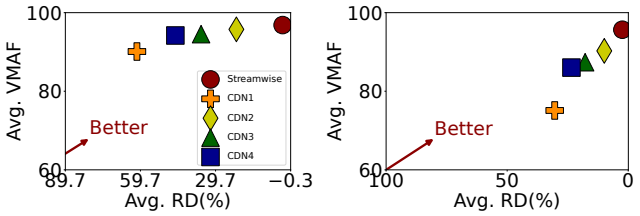


Figure 1: Single vs. multi-CDN analysis (LLL: left, VoD: right).

a VMAF improvement of ~(7%–8.5%), increases average bitrate by ~(1.2x–1.5x), and enhances overall viewer QoE by ~(22%–48%). Additionally, it eliminates rebuffering events in VoD streaming and reduces rebuffering by ~10× in LLL scenarios. These results underscore the broad effectiveness of `StreamWise` in boosting viewer QoE performance across diverse network conditions. The contributions of this paper are three-fold.

(1) We propose a novel, learning-based (DRL) CDN selection strategy, referred to as `StreamWise`. Deployed at the content steering server, `StreamWise` makes real-time CDN selection decisions aimed at maximizing QoE, using only three key inputs from each client: RTT, selected bitrate, and rebuffering duration. The `StreamWise` integration demonstration, along with its implementation, is publicly available at [Demo] and [Code].
(2) We extend the Vegvisir framework [19] to implement a comprehensive end-to-end streaming platform, comprising three key components: CDN-side, traffic shaper, and client-side. All components are deployed in Docker containers and interconnected via Docker Compose networks to regulate traffic flow. The traffic shaper (tc Netem) emulates various network conditions using real-world traces to connect clients and CDNs. On the client-side, we use goDASH clients with multiple ABR algorithms, while on the CDN-side, we deploy four CDNs, each operating under its own distinct network conditions. Content steering including `StreamWise` is integrated at the traffic shaper, with CMCD/CMSD standards facilitating the collection of client and CDN statistics for the steering server.
(3) We conduct extensive experiments covering a wide range of ABR schemes, network traces, video content types (with varying encoding parameters), and streaming scenarios (VoD and LLL). The experimental results demonstrate that `StreamWise` consistently outperforms competing multi-CDN strategies, achieving significant improvements across QoE and its metrics.

The rest of the paper is organized as follows: Section 2 reviews related work in multi-CDN, Section 3 describes the `StreamWise` solution, followed by performance evaluation in Section 4, and discussions in Section 5. Finally, Section 6 concludes the paper.

2 Related Work

The increasing demand for high QoE streaming has necessitated the development of multi-CDN strategies [6, 24, 36], where video content is distributed across multiple CDNs to ensure reliability, optimize performance, and mitigate failures. Multi-CDN architectures [10, 36] dynamically switch between CDNs to enhance viewer experience, balance load, and recover from network issues, but the methods for implementing these strategies vary in complexity and effectiveness. A couple of strategies have been proposed for CDN switching which are presented as follows:

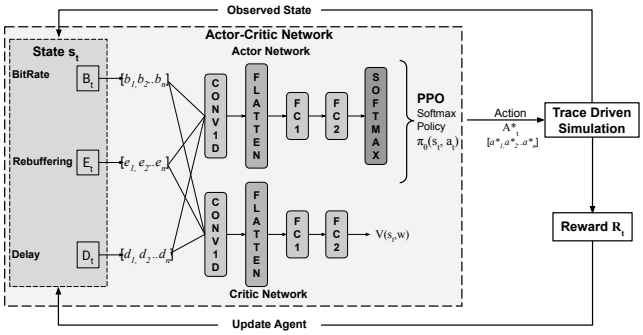


Figure 2: StreamWise DRL architecture.

DNS-based switching [43] is the simplest and involves using a constant source video URL while switching the CDN via DNS resolution. Though easy to implement, DNS-based approaches suffer from high switch delays, ranging from 300s to several minutes in case of CDN failures [47], which severely impacts viewer QoE.

An alternative approach is *manifest rewriting* [42], which enables midstream switching by altering the content manifest during the video streaming session. This method allows for faster switching and avoids the cascade effect caused by session resets during CDN failures. However, manifest rewriting may introduce errors, and the switching process is not entirely seamless as it takes time for the server to detect and react to a failed CDN.

Server-side switching [37, 45] simplifies the process by centralizing the decision-making on the server, allowing the operator greater control. This method involves the server collecting and analyzing data from multiple clients before making CDN switching decisions. While relatively simple to deploy, it is prone to added delays during page loading and does not always consider the specific conditions of individual clients, relying instead on aggregated data.

Client-side switching [6, 8, 18], on the other hand, relies on the client collecting local, real-time performance metrics (*e.g.*, buffer level, throughput, latency) to make CDN switching decisions. This approach offers seamless midstream switching and more accurate QoE data since it reflects the unique network conditions experienced by each viewer. However, it is more complex to implement at scale, particularly for platforms with proprietary players, and the algorithms required for client-side switching demand careful planning and implementation.

Recent advancements in HAS standards have addressed the above-mentioned challenges. The introduction of *content steering* [1, 36, 45] in the latest DASH and HLS specifications standardizes CDN switching behavior across players, simplifying multi-CDN implementation. This reduces integration complexity and ensures consistent switching behavior, enabling providers to effectively leverage multiple CDNs. Importantly, this standard is forward-compatible, maintaining reliability amid future streaming updates. Current CDN switching strategies at the steering server often rely on basic load-balancing heuristics [8, 25], such as round-robin or least connections, which can struggle with varying network conditions, leading to sub-optimal performance and low QoE. In contrast, StreamWise dynamically adjusts to real-time network fluctuations and player performance, enhancing the efficiency and reliability of CDN switching.

3 StreamWise Solution

Unlike traditional CDN selection heuristics, StreamWise leverages a DRL approach to dynamically choose the optimal CDN for each client by analyzing logs from both servers (*i.e.*, CDNs) and clients. It uses an Actor-Critic (A2C) NN [5] combined with Proximal Policy Optimization (PPO) [41] to enhance decision-making. PPO improves training stability by limiting abrupt policy changes through a clipped objective function, ensuring more reliable and consistent learning outcomes. The overall A2C NN architecture of StreamWise is illustrated in Figure 2.

3.1 Problem Formulation

At each time step t, StreamWise determines the best performing CDN for each client. It considers the subsequent state s_{t+1} based on the current state s_t. A mathematical formulation of the problem can be expressed as follows:

$$\begin{cases} \text{find CDN}_t^{\star}(\pi), \forall\, t \in \{1..., k\}, \\ \arg\max_{\pi} r_t(\pi), \end{cases} \tag{1}$$

where CDN* is the optimal CDN based on current network statistics (players + CDNs). π is an RL policy that helps to decide the best CDN, and k is the duration of a streaming session. The problem (1) is described as a *single-agent* decision problem aimed at finding the best performing CDN at each time t. This setup aligns with the formulation of an Markov Decision Process (MDP), where the agent's decisions are based on its current state and the available observations, aiming to maximize long-term rewards. The MDP process is a 4-tuple (S, A, P, R), where each element is defined as:

- $S = \{s_1, \ldots, s_k\}$ is a set of finite and continuous agent states.
- $A = \{a_1, \ldots, a_k\}$ is a set of finite and discrete actions.
- $P = S \times S \times A \rightarrow [0, 1]$ is the state transition probability function $P(s_{t+1} \mid s_t, a_t)$ from the state s_t to state s_{t+1} when action a_t is taken.
- $R = \{r_1, \ldots, r_k\}$ is the set of expected immediate rewards, which depends on the state and action taken by the agent at every time step t. $R(s_t, a_t, s_{t+1}) = r_t$ is the step reward.

At each time step $t \in \{1, \ldots, k\}$, the agent observes the environment through a state vector $\vec{s_t}$. Based on these observations, the agent selects the optimal CDN by choosing the one with the highest probability, as determined by the softmax layer over all available CDNs. The StreamWise DRL solves the MDP problem using a single agent A2C NN with clipped PPO, generalized advantage function (GAE) [5], and RMSProp optimizer for policy $\pi^{\star}$.

3.2 StreamWise offline Training

Typically, companies do not make their CDN logs publicly accessible, and no official dataset is available for CDN selection. As a result, we utilized Pensieve's [27] experience trajectory data for training. ABR algorithms optimize video quality by using metrics such as bandwidth, buffer level, and latency. As these metrics are directly co-related with the optimal CDN selection, we used them in the design of StreamWise. Specifically, for the training of StreamWise, we selected the most relevant input metrics as listed below:

(1) `Selected Bitrate (B)` refers to the selection of ABR which is the amount of data processed per unit of time in a video stream, typically measured in Mbps.
(2) `Rebuffering Events (E)` occurs when the video player pauses playback to load more data due to buffer depletion, typically measured in seconds.
(3) `Round-Trip Time (D)` is the time required for a data packet to travel from the client to the CDN and back again, typically measured in milliseconds.

During training, we iteratively provided the model with four consecutive experience entries from the trajectory. Each set of four entries was treated as separate CDN samples, and the `StreamWise` model was tuned to select the optimal CDN based on these inputs.

▹**Input State:** At each time step t, the agent receives a state vector $\vec{s_t}$ consisting of three key state metrics in the form of a feature matrix, which includes data from all clients connected to the CDNs. These state metrics are tuple $< \vec{B}_t, \vec{E}_t, \vec{D}_t >$. As illustrated in Figure 2, each feature forms a vector for $t \in \{1, \ldots, n\}$, where n represents the number of clients from which logs are collected. To prevent potential bias in the input states, the three state metrics are normalized across all clients and time steps.

▹**Action Space:** The action space A is defined as the optimal CDN for the next state s_{t+1} for every client. For each client and time step t, the `StreamWise` policy $\pi^\star$ maps the state vector $\vec{s_t}$ to a discrete action space A and selects $a_t \in A$.

▹**Reward:** To evaluate the action a_t at each time step t, a corresponding reward r_t is computed, which guides the agent toward selecting the optimal policy π^*. For `StreamWise`, we design a clear and effective reward function,

$$r_t = w_1 \times b_t - w_2 \times e_t - w_3 \times d_t, \tag{2}$$

where at each time t, b_t represents the selected bitrate, e_t represents the rebuffering duration and d_t represents the delay (RTT). We employed Pensieve$^+$ (refer to Section 4.2) on a dataset of 2K video sequences [15] to fine-tune the reward function weights w_1, w_2, and w_3. This mitigates potential bias that could arise from fine-tuning the same reward weights across the same dataset. Our results reveal that the optimal values for the weights are $w_1 = 1$, $w_2 = 5$, and $w_3 = 0.5$. Note that the state values passed to the reward function are normalized beforehand.

▹**`StreamWise` NN Architecture:** The `StreamWise` A2C NN architecture is shown in Figure 2 and it consists of two networks, which are the actor and the critic. Each network uses three fully connected (FC) layers. Each FC layer contains 64 neurons and utilizes the ReLU activation function. The output of the last FC layer in the actor network uses a `softmax` activation function, providing an output in the range $[0, 1]$. In the critic network, $\vec{s_t}$ and a_t are concatenated in the first FC layer for each time step t. The final output is derived from the last FC layer. The actor and critic get trained and optimized with the RMSProp optimizer.

▹**Policy Gradient and Training Algorithm:** The fundamental goal of `StreamWise` is to choose the best performing CDN for every video segment. At every segment download t, the A2C NN of `StreamWise` maps the state $\vec{s_t}$ to the best preforming CDN, *i.e.*, the CDN with lowest RTT, which delivers the highest bitrate with zero or minimal rebuffering, which results in the best advantage function Λ that is expressed as follows:

$$\Lambda_t = \sum_{i=t}^{T_{\pi_\theta}} r_t + \gamma \times \delta_t + \gamma \times \lambda \times \Lambda_{t+1}, \tag{3}$$

$$a_t = \arg\max_a \mathbb{E}[\Lambda_t(\alpha_t, a)],$$

where T_{π_θ} represents the batch size used for updating the gradient policy π_θ. The discount factor, denoted as γ, lies within the range $[0, 1]$, and δ_t refers to the temporal difference (TD) error at time step t, which is calculated as $V_{s_{t+1}} - V_{s_t}$. Here, V_{s_t} represents the value function as computed by the critic network. The parameter λ corresponds to the GAE, and θ denotes the policy parameter. Additionally, $\pi_\theta(\alpha_t, a_t)$ is the probability of action a_t being taken in state s_t under the policy π_θ.

TD error provides immediate feedback on the deviation between predicted and actual rewards. The observed values aid in value function updates, while the GAE enhances advantage estimation by incorporating both short-term and long-term effects of actions. This leads to more informed policy updates and improved learning efficiency. After the `StreamWise` A2C NN has collected an experience trajectory, the PPO algorithm uses mini-batches to compute a clipped surrogate objective function for gradient updates to ensure stable and efficient policy learning. The clipped surrogate objective function can be expressed as follows:

$$L^{clip}(\theta) = -\mathbb{E}_t \left[\min \left(\mathfrak{R}_t(\theta) \cdot \hat{\Lambda}_t,\ \text{clip}(\mathfrak{R}_t(\theta), 1-\epsilon, 1+\epsilon) \cdot \hat{\Lambda}_t \right) \right], \tag{4}$$

where, $L^{clip}(\theta)$ represents the clipped surrogate objective function, which is used to define the policy loss. The ratio $\mathfrak{R}_t(\theta)$ is the comparison between the probability of the selected action under the new policy and its probability under the old policy. Additionally, $\hat{\Lambda}_t$ is the estimated advantage of the action at time step t, and ϵ is the clipping parameter that controls the size of the policy update.

The negative sign in the equation indicates that we aim to minimize the clipped surrogate objective to improve the policy with small, stable updates towards better performance. We also compute the value function loss as follows:

$$L^{value}(\theta) = \frac{1}{2}\mathbb{E}_t \left[(V(s_t;\theta) - G_t)^2 \right] \tag{5}$$

where $V(s_t;\theta)$ is the value function predicted by the critic network for state s_t with θ and G_t are the observed return from state s_t.

Additionally, we compute the entropy loss to maintain a certain level of randomness in the agent's actions, which can lead to better exploration of the environment. This can be expressed as follows:

$$L^{ent}(\theta) = \mathbb{E}_t \left[\sum_a \pi(a|s_t;\theta) \log \pi(a|s_t;\theta) \right] \tag{6}$$

where $\pi(a|s_t;\theta)$ is the probability of selecting action a given state s_t and parameters θ. Then, we compute the gradient of the total policy loss L^{total} with respect to the policy parameters θ as follows:

$$\nabla_\theta L^{total}(\theta) = \nabla_\theta L^{clip}(\theta) + \nabla_\theta L^{value}(\theta) + \nabla_\theta L^{ent}(\theta) \tag{7}$$

Finally, The gradient update step is computed for each epoch, resulting in the optimal policy $\pi^\star$, which is computed as follows:

$$\theta \leftarrow \theta - \omega \nabla_\theta L^{total}(\theta) \tag{8}$$

where ω is the learning rate for the respective A2C NN.

Table 1: StreamWise training/testing parameters.

	Parameter	Symbol	Value
A2C	Discount Factor	γ	0.95
	Advantage estimator	λ	0.9
	Actor & Critic learning rates	$\omega, \bar{\omega}$	2 x 10^{-4}, 1 x 10^{-3}
PPO	Clip parameter	ϵ	0.21
	Entropy Coefficient	δ	1.9
	PPO policy update interval, steps	T_θ, n_s	1000, 9
NN	Batch size	B_s	64
	Exploration size, number of episodes	θ, n_e	10, 30
	Number of epochs	itr	90000

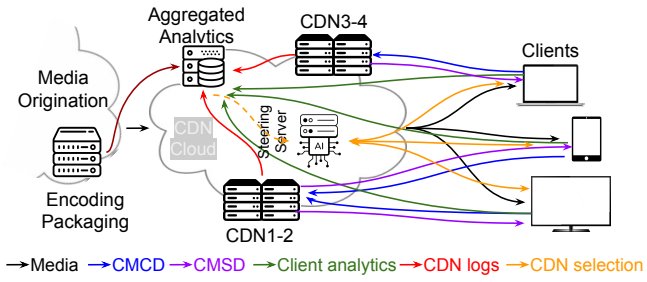


Figure 3: The end-to-end streaming architecture.

4 Performance Evaluation

4.1 StreamWise Implementation

4.1.1 Choice of StreamWise *Parameters.* StreamWise was trained on Pensieve's experience trajectories, which were diversified across a range of network conditions. Given that the platforms including network traces and video contents for training and testing were distinct, all traces in [27] were utilized for training purposes only. Training parameters are crucial for optimizing any NN architecture. Therefore, the hyper-parameter tuning for the StreamWise model was conducted using Bayesian optimization [16] techniques. The optimal set of parameters for StreamWise, as observed during this process, are summarized in Table 1.

4.1.2 Offline Training. StreamWise 's NN architecture was implemented in Pytorch 3.10.12+cu121 with Python 3.10.12. The training was performed in a continuous trace-driven fashion as shown in Figure 2.

4.1.3 Online Testing. To evaluate the StreamWise model on real-world data in an online setting, we utilize the Vegvisir framework [19]. Vegvisir provides a robust and flexible environment for testing by creating a fully integrated ecosystem where servers, clients, and traffic shapers are each deployed within isolated Docker containers. We extend the Vegvisir framework to support the following:

(1) Two setups: (*i*) multiple CDNs connected to a single client (denoted as MS-SC) through a single shaper in a many-to-one mapping, and (*ii*) multiple CDNs connected to multiple clients (denoted as MS-MC) in either a one-to-one or a one-to-many mapping with a single shaper as the link bottleneck.
(2) CMCD/CMSD to facilitate the collection of client and CDN statistics for the steering server.
(3) Content steering with StreamWise at the shaper side.

The end-to-end streaming architecture is depicted in Figure 3. To inform the client of the availability of multiple CDNs, several base URLs are included in the manifest (MPD) file, which the client retrieves from the origin server. A snapshot of this MPD file is

```
<BaseURL serviceLocation="0">https://<CDN1>/
    MOI/</BaseURL>
<BaseURL serviceLocation="1">https://<CDN2>/
    MOI/</BaseURL>
<BaseURL serviceLocation="2">https://<CDN3>/
    MOI/</BaseURL>
<BaseURL serviceLocation="3">https://<CDN4>/
    MOI/</BaseURL>
<ServiceDescription id="0">
</ServiceDescription>
```

Figure 4: MPD snapshot for multi-CDN.

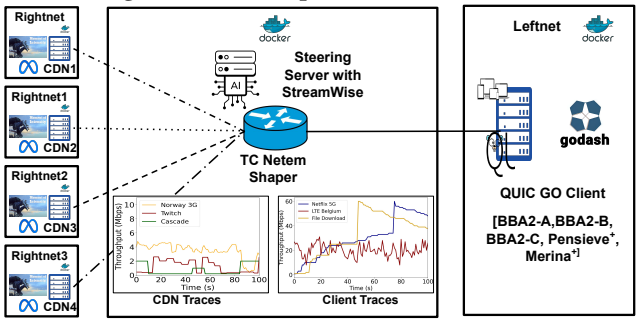


Figure 5: StreamWise's evaluation setup (online testing).

provided in Figure 4, highlighting how the client is made aware of the diverse CDN options. Within the MPD, the CDN servers CDN1, CDN2, CDN3, and CDN4 are available for the client to request video segments. The content steering server continuously gathers real-time statistics from both the clients (via CMCD) and the CDNs (via CMSD), which are then fed to StreamWise to make data-driven decisions on the best performing CDN. The CDN selection is then communicated to the client through the HTTP response, ensuring that subsequent segment requests are made to the optimal CDN.

To track the state of each CDN in real-time, we utilize the MPD file (8 kB), which is cached at each CDN and can be fetched with negligible latency (~ 1 ms). Each client periodically requests the MPD file from all available CDNs to monitor the RTT and throughput. RTT is directly measured during the MPD fetch, while throughput is interpolated based on the RTT and the average 4K segment size (8 MB). Further implementation details of StreamWise can be found at [44].In LLL mode, each client fetches the MPD every 200 ms, ensuring frequent updates of RTT and throughput metrics. The RTT and throughput values are smoothed using the average of the last 10 measurements from segment requests, helping to reduce the impact of transient variations and measurement errors. For VoD mode, the MPD fetch interval is extended to every 500 ms, reflecting the larger segment sizes and the longer intervals between requests typical in VoD mode. These intervals have been empirically chosen to balance the trade-off between maintaining accuracy in RTT and throughput estimates and minimizing the overhead introduced by frequent requests. Based on these values, a reliable estimate of stall is made using the current buffer level of each client.

To evaluate the performance of StreamWise's DRL model, we conduct comprehensive experiments using diverse network traces on both the CDN and client sides, varying buffer levels, and employing different ABR algorithms at the client. We analyze StreamWise's effectiveness across two key streaming scenarios: VoD and LLL. The overall architecture used for testing is visualized in Figure 5. The CDN, the client and the shaper are all available in separate docker

networks and form an integrated end-to-end streaming system. The client, using the goDASH [33] player, downloads segments from the four available CDN servers. Based on the collected statistics, `StreamWise`'s model dynamically selects the optimal CDN and informs the client in real-time which CDN to use for the next segment. We emulated a typical HAS video streaming session using multiple videos under various network conditions. The client-side employs the QUIC-go implementation [31], running the goDASH player in headless mode, which simplifies the extraction of performance metrics. This setup avoids the added complexity of using a web-based dash.js [14] player. On the CDN side, we used the mvfst QUIC implementation [29]. For network emulation, we used tc-Netem with multiple real-world traces, as shown in Figure 5. In the MS-SC setup, we employed one client and four CDNs, whereas in the MS-MC setup, we used ten clients and four CDNs.

4.1.4 Challenges.

1. The ITU-P.1203 [39] implementation is limited to 1080p content and the extended ITU-P.1204 implementation for 4K content is in its initial stages, cannot parse frame by frame, and is only suitable for short videos with 10 seconds maximum [9, 46].

2. We initially considered extending state of the art load-balancers like Nginx [35] and HAProxy [30] to function as content-steering servers for CDNs. Despite supporting standalone QUIC operation, they are unable to proxy QUIC traffic, defaulting to TCP instead. This limitation adds complexities, hindering fair comparisons with other solutions.

3. We excluded VMAF and PSNR in our 8K ablation study due to the absence of an official 8K VMAF model. Notably, using the 4K VMAF model for 8K content yielded unreliable results.

4.2 Methodology and Evaluation Setup

4.2.1 Video Sample and Parameters. The *Moment of Intensity* (MOI) video content [48] was used during `StreamWise` 's online testing, with varying segment durations, and encoded in AVC format with 10 bitrate levels ranging between 288p@0.5Mbps—2160p@40Mbps (4K), and maximum playback buffer capacities. `StreamWise` 's performance was evaluated across two key scenarios: VoD and LLL, each with multiple sub-scenarios, the details of which can be found in Table 2.

4.2.2 Network Traces. For online testing of `StreamWise`, we used seven network traces: the first four—Norway 3G [38], F4G5G, Twitch [49], and Cascade [20]—with average bandwidths (sampling intervals) of 3.5 Mbps (1s), 137 Mbps (1s), 1.4 Mbps (5s), and 0.8 Mbps (10–20s), respectively, at the CDN side; and the last three—Netflix 5G [32], LTE Belgium [50], and File Download (FD) [32]—with average bandwidths (sampling intervals) of 23.9 Mbps (1s), 18.9 Mbps (1s), and 21.9 Mbps (1s), respectively, at the client side. The F4G5G trace, constructed from real-world traces [32], mimics 5G/4G switching behavior. Figure 5 illustrates the traces, except F4G5G (137Mbps and ±67Mbps), omitted for clarity due to its higher throughput.

4.2.3 ABR Algorithms. We used BBA2-XLDouble (denoted as BBA2-C), a buffer-based [21], and two learning-based ABR algorithms, namely, Pensieve$^+$ [27] and Merina$^+$ [23]. BBA2-C incorporates an additional stall prevention mechanism to proactively reduce playback stalls. Pensieve$^+$ and Merina$^+$ are enhanced versions of the original Pensieve [27] and Merina [23] algorithms with the addition of BBA2-C's rebuffering prevention logic. Initially, both ABRs were designed to select from 6 bitrate representations. However, since our video samples contain 10 bitrate levels, we re-trained them with more epochs to accommodate this broad range.

Table 2: Experiment scenarios.

Scen.	VMAF	FPS	Bitrate (Mbps)	Seg. Dur.	Max. Buff. (s)
A: LLL	37-96	60	0.5 - 40	2	12
B: VoD	37-96	60	0.5 - 40	4	60

4.2.4 CDN Switching Strategies. We compared `StreamWise` against existing well-known heuristic-based CDN switching strategies:

- *Random*: It randomly selects a CDN from the available options to fetch each video segment.
- *Deterministic-1*: It allocates video segments to CDNs by a fixed ratio: 50% to CDN1, 30% to CDN2, 15% to CDN3, and 5% to CDN4.
- *Deterministic-2*: It allocates video segments to CDNs with a different fixed ratio, with 50% served by CDN1, 20% each by CDN2 and CDN3, and the rest of 10% by CDN4.
- *Round-Robin (RR)*: It sequentially selects each CDN in turn to ensure an even and fair distribution of segments across all CDNs.
- *Least Connections (LC)*: It directs traffic to the CDNs with the least open connections at the time to balance the load efficiently.

Leveraging the same real-time CDN state tracking as `StreamWise`, we also implemented two dynamic heuristic solutions:

- *DH1*: It selects the best CDN with the highest throughput.
- *DH2*: It selects the best CDN with the least RTT.

4.2.5 Performance Metrics. We evaluated `StreamWise` using four widely used performance metrics: (1) VMAF [34], (2) PSNR [4], (3) the number of segments downloaded in ultra-HD quality (nSeg-UHD), and (4) the Yin QoE Model [51]. The Yin QoE model incorporates various factors, including selected bitrate, rebuffering duration (RD), startup delay, and bitrate switches, to compute a normalized QoE score ranging from 1 (poor) to 5 (excellent), providing a comprehensive assessment of the viewer's experience.

4.2.6 Evaluation Setup and Scenarios. Our online testing setup, shown in Figure 5, used a physical machine running Ubuntu 22.04.3 LTS, featuring an NVIDIA GA104 (GeForce RTX 3060) GPU, a 24-core CPU, and 64GB of RAM. We deployed the clients, traffic shaper, and servers in separate Docker containers connected via two Docker Compose networks to manage traffic effectively. All trace-driven emulations were performed twice, each lasting approximately 120 seconds, with results presented as the average of these runs.

4.3 Results and Analysis

We observed consistent results across all scenarios. Here, are the key findings for scenarios **A** and **B** (refer to Table 2).

4.3.1 MS-SC Setup Results. The results for this setup are highlighted in Tables 3 and 4 for LLL and VoD scenarios, respectively.

▹**Scenario A.** Table 3 illustrates the comparison of `StreamWise` against its competitors in terms of several QoE metrics for various

Table 3: Average results of the QoE and its metrics for different network traces for LLL scenario A and content MOI (2s). ↑: higher is better (green), ↓: lower is better (green), lowest performance (red).

			Avg. Perceptual Quality (VMAF)	Avg. Perceptual Quality (PSNR)	Avg. Selected Bitrate (Mbps)	Average RD (%)	Average Yin	Average nSeg-UHD (%)
LTE Belgium	Pensieve+	**StreamWise**	**94.14** ±1.86↑	**50.82** ±3.99↑	**30.17** ±6.06↑	0.09 ±0.88	5.00↑	**86.27**↑
		Random	89.60 ±1.16	49.72 ±3.09	23.50 ±6.69	7.21 ±16.53	5.00↑	61.54
		Deterministic-1	89.63 ±1.15	49.67 ±2.84	25.19 ±4.30	1.80 ±2.78	5.00↑	65.91
		Deterministic-2	69.39 ±0.55	45.40 ±2.94	10.38 ±9.80	**0.00**↓	1.00	50.00
		Round-Robin	92.74 ±0.96	50.66 ±3.53	27.69 ±7.50	8.90 ±19.22	5.00↑	61.54
		Least connections	88.70 ±1.15	49.51 ± 2.84	23.93 ±6.00	6.80 ±23.08	1.00	49.36
		DH1	90.84 ±1.36	50.13 ±7.82	27.22 ±5.60	4.40 ±0.37	5.00↑	78.57
		DH2	93.07 ±2.32	50.74 ±4.43	29.42 ±6.24	3.00 ±1.36	5.00↑	82.00
	Merina+	**StreamWise**	**96.84** ±1.64↑	**51.43** ±2.22↑	**36.00** ±6.50↑	**2.87** ±4.32↓	5.00	**97.67**↑
		Random	92.13 ±0.82	50.75 ±3.79	29.00 ±10.44	4.82 ±5.34	5.00	91.67
		Deterministic-1	93.26 ±0.87	50.75 ±3.27	29.21 ±10.37	3.71 ±4.10	5.00	91.89
		Deterministic-2	90.43 ±0.65	51.40 ±3.09	27.11 ±7.23	12.54 ±20.25	5.00	90.97
		Round-Robin	94.43 ±0.65	51.01 ±3.09	31.41 ±9.47	13.18 ±18.41	5.00	93.94
		Least connections	94.43 ±0.65	51.01 ±3.09	31.41 ±9.47	12.61 ±18.35	5.00	93.94
		DH1	78.54 ±0.96	47.96 ±4.76	12.00 ±9.88	7.74 ±4.23	5.00	63.16
		DH2	92.85 ±0.89	48.42 ±2.86	33.27 ±6.32	3.13 ±3.69	5.00	92.87
	BBA2-C	**StreamWise**	**86.5** ±1.68↑	**48.94** ±2.85↑	**13.39** ±5.44↑	0.00	5.00	-
		Random	81.94 ±1.14	47.09 ±2.17	6.41 ±2.94	0.00	5.00	-
		Deterministic-1	74.36 ±2.06	45.71 ±4.02	4.32 ±2.18	0.00	5.00	-
		Deterministic-2	81.44 ±4.19	46.90 ±2.58	8.17 ±3.83	0.00	5.00	-
		Round-Robin	80.47 ±1.28	46.76 ±2.71	6.58 ±2.60	0.00	5.00	-
		Least connections	80.18 ±0.95	46.51 ±2.61	5.89 ±2.01	0.00	5.00	-
		DH1	83.77 ±1.35	47.55 ±2.70	9.32 ±4.77	0.00	5.00	-
		DH2	80.67 ±2.14	46.76 ±4.27	5.51 ±2.41	0.00	5.00	-
FD	Pensieve+	**StreamWise**	**94.74** ±1.02↑	**51.04** ±3.03↑	**31.33** ±5.71↑	**0.18** ±0.86↓	5.00	**90.16**↑
		Random	78.67 ±0.64	46.78 ±2.60	6.15 ±4.33	13.16 ±18.73	5.00	-
		Deterministic-1	88.70 ±1.15	49.51 ±2.84	23.93 ±6.00	6.80 ±23.08	5.00	49.36
		Deterministic-2	82.84 ±0.43	47.30 ±1.79	8.71 ±3.50	19.42 ±27.14	5.00	45.18
		Round-Robin	89.46 ±1.09	50.56 ±3.73	27.50 ±7.50	9.84 ±14.69	5.00	52.78
		Least connections	82.57 ±1.00	48.45 ±2.68	21.10 ±8.10	22.30 ±28.26	5.00	42.50
		DH1	93.71 ±0.50	50.78 ±1.92	29.44 ±5.19	4.60 ±2.33	5.00	80.65
		DH2	90.71 ±1.72	49.97 ±2.32	28.83 ±5.80	3.60 ±2.29	5.00	64.00
	Merina+	**StreamWise**	**96.84** ±1.64↑	**51.43** ±2.22↑	**36.00** ±6.50↑	**1.73** ±6.25↑	5.00	**97.67**↑
		Random	93.43 ±2.94	49.50 ±3.09	30.90 ±9.80	16.03 ±25.89	5.00	92.97
		Deterministic-1	94.98 ±0.96	51.01 ±4.76	32.25 ±9.00	7.39 ±13.38	5.00	93.30
		Deterministic-2	90.43 ±0.65	51.40 ±3.09	27.11 ±7.20	12.54 ±20.25	5.00	90.97
		Round-Robin	94.89 ±0.87	50.99 ±3.27	32.10 ±9.10	10.51 ±19.69	5.00	94.59
		Least connections	89.29 ±0.89	50.08 ±2.86	9.09 ±6.65	14.84 ±20.75	5.00	85.11
		DH1	79.47 ±1.39	48.16 ±2.81	12.73 ±10.15	8.06 ±3.31	5.00	65.00
		DH2	92.48 ±0.64	48.37 ±4.28	32.22 ±7.15	2.68 ±6.18	5.00	92.06
	BBA2-C	**StreamWise**	**89.46** ±1.64↑	**49.11** ±3.16↑	**13.34** ±7.8↑	0.00	5.00	-
		Random	80.39 ±1.08	46.86 ±2.74	7.14 ±2.82	0.00	5.00	-
		Deterministic-1	79.05 ±2.51	47.12 ±6.43	8.46 ±6.29	0.00	5.00	-
		Deterministic-2	75.98 ±2.06	46.00 ±4.02	4.85 ±2.44	0.00	5.00	-
		Round-Robin	80.67 ±2.14	46.76 ±4.27	5.51 ±2.41	0.00	5.00	-
		Least connections	80.18 ±0.95	46.51 ±2.61	5.89 ±2.00	0.00	5.00	-
		DH1	85.83 ±1.55	48.24 ±2.66	11.35 ±6.10	0.00	5.00	-
		DH2	85.66 ±1.82	48.08 ±8.6	11.56 ±6.12	0.00	5.00	-

network traces and ABR combinations for LLL scenario. For the LTE Belgium trace with Pensieve+ as the ABR algorithm, StreamWise outperforms its competitors on average in terms of VMAF (*col*.1) by ~7% with significantly less rebuffering (*col*.4). Additionally, with StreamWise, the clients download ~22% more segments in UHD (*col*.6), achieve the best PSNR (*col*.2) and Yin (*col*.5) scores with minimal rebuffering (*col*.4). This can be attributed to the ability of StreamWise to dynamically adjust to the existing network conditions and select the optimal CDN for serving content, which improved the average bitrate (*col*.3) selected by the clients by ~1.5x as compared with other CDN switching strategies, which in turn contributes to higher QoE (*col*.5). Random, Deterministic 1, Round-Robin and Least connections perform comparably and although they only trail behind StreamWise slightly in terms of VMAF, they suffer from significantly more rebuffering events and interruptions, which ultimately degrade the user QoE. Deterministic 2 is the only strategy to provide a smooth streaming experience but it lags behind substantially in terms of VMAF by 3× Just Noticeable Difference (JND) [11] threshold (6 units in VMAF). DH1 and DH2 provide comparable VMAF scores to StreamWise, but they incur ~400% more RD compared to StreamWise. This significant increase in RD can be attributed to their reliance on a single metric, which prevents them from effectively balancing the trade-offs between RTT, throughput, and stalls. For the LTE Belgium trace, the results with Merina+ closely mirror that of Pensive+ as ABR, while few exceptions arise. StreamWise continues to offer the highest VMAF and PSNR scores with the least RD. Deterministic 2 performs comparably to StreamWise in terms of VMAF but suffers from major rebuffering events (~ 600% more) compared to StreamWise. Despite relying on throughput, DH1 lags behind in terms of VMAF by 3× the JND threshold compared to StreamWise. Merina+ is more aggressive than Pensieve+, and DH1's sole reliance on throughput results in more ultra-HD segment downloads initially. However, this approach triggers the stall predictor rule more frequently, which ultimately leads to subsequent segment requests in the lowest quality, resulting in lower VMAF scores. With BBA2-C as ABR, the results resemble that with Pensieve+. BBA2-C, being a buffer-based ABR, is severely impacted by the low buffer sizes in LLL scenario and is unable to download the highest quality segments. However, with the enhanced design of StreamWise, it successfully downloads segments one quality level beyond full HD, contributing to its high VMAF scores, by an average of 1× the JND threshold in VMAF.

The results for FD trace resemble that of the LTE Belgium trace, which shows collective VMAF improvement of ~9% and ~1.5× in average selected bitrate. Figure 6 (left) shows results for the Netflix 5G trace, closely mirroring those of the LTE Belgium and FD traces, highlighting the clear dominance of StreamWise.

Overall, StreamWise achieves ~8% better VMAF and a ~1.5x improvement in average selected bitrate compared to it's competitors. These results demonstrate the superior CDN selection logic of StreamWise and underscore its adaptability to varying network conditions and ABR algorithms, allowing it to select the optimal CDN for enhanced QoE.

▹**Scenario B.** Table 4 highlights the performance of StreamWise compared to its competitors across various QoE metrics in both the Netflix 5G and LTE Belgium traces, tested under different ABR algorithms in the VoD scenario. For the Netflix 5G trace with Pensieve$^+$, StreamWise achieved a substantial ~8.5% improvement in VMAF on average in comparison to other competitors, along with zero rebuffering. Deterministic 1 and 2, Round-Robin and Least Connections exhibited similar performance. Random and DH1 lagged behind in terms fo VMAF by 2× and 3× respectively. This underscores the unreliability of DH1 as a CDN selections strategy as relying on throughout alone can lead to sub-optimal results. Notably, DH2 performs similar to StreamWise. The results with Merina$^+$ and BBA2-C as ABR align closely to the results observed with Pensieve+, with StreamWise at the forefront of all CDN selection strategies with highest VMAF and PSNR scores with minimal rebuffering. The results for LTE Belgium trace mirrored Netflix 5G performance, with an altogether ~7.5% VMAF improvement and ~1.5× higher average bitrate. Notably, with BBA2-C, the number of segments downloaded in ultra-HD quality surpasses that of Netflix 5G, as LTE Belgium provides higher throughput during the first 40 seconds. This highlights StreamWise's superior design, which facilitates the download of the highest ultra-HD segments. For brevity, results for the FD trace are shown in Figure 6 (right) and align closely with Netflix 5G and LTE Belgium, further validating StreamWise's superiority. Overall, StreamWise consistently outperformed competitors, achieving an average 8% VMAF improvement and ~1.5× better average bitrate across traces and ABR algorithms.

Across both scenarios, StreamWise consistently outperformed its competitors in terms of VMAF and RD, delivering a superior streaming experience. Although DH1 and DH2 demonstrate better performance compared to static CDN selection strategies, their improvements are unreliable. Both suffer from significant RD in LLL mode, and DH1 struggles to maintain high VMAF in multiple scenarios. These limitations underscore the efficiency of StreamWise 's DRL design, which adeptly balances the complex trade-offs between RTT, throughput, and stalls.

4.3.2 MS-MC Setup Results. The results for this setup are highlighted in Tables 5 and 6 for LLL and VoD scenarios, respectively.

▹**Scenario A.** Table 5 showcases the remarkable dominance of StreamWise over its competitors across key QoE metrics in both the LTE Belgium and FD traces, using Merina$^+$ as the ABR algorithm. In the LTE Belgium trace, StreamWise outperformed other CDN switching strategies on average by ~4% improvement in VMAF and delivering marginally higher average bitrate, accompanied by the highest PSNR, Yin score, and minimal rebuffering. The static

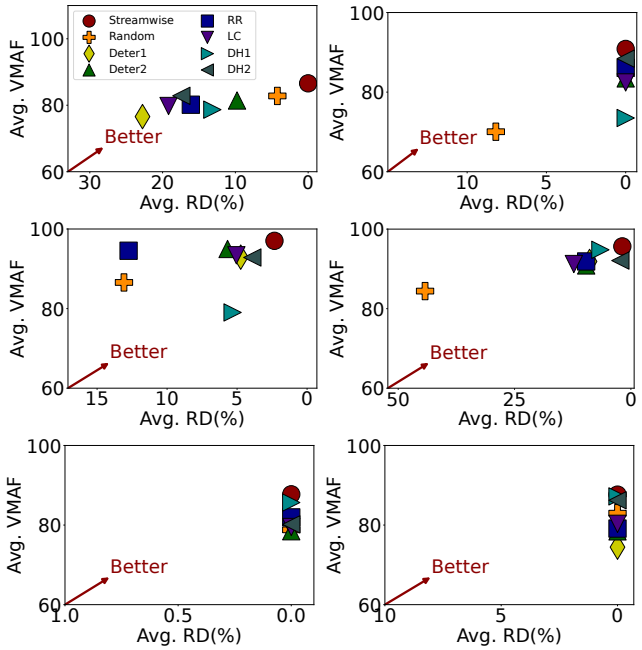


Figure 6: VMAF vs RD (%) analysis for ABR: Pensieve$^+$ (top), Merina$^+$ (center), and BBA2-C (bottom) in LLL mode for Netflix 5G trace (left) and in VoD mode for FD trace(right).

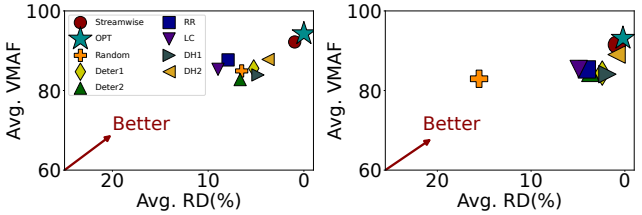


Figure 7: Optimal V/s others: LLL (left) and VoD (right).

switching strategies along with DH1 demonstrated comparable performance, albeit with significantly more rebuffering. DH2 lags behind in VMAF by 1× the JND threshold in comparison to VMAF and suffers from 3× more RD as well. The FD trace mirrored these results, with StreamWise demonstrating ~5% VMAF improvement and a ~1.3× increase in average bitrate. Overall, StreamWise consistently delivered prominent results, achieving an average VMAF improvement of ~4% with 5× less RD, demonstrating its superiority in scaled setups (10 clients) by delivering optimized QoE for VoD scenarios. These results also highlight the inherent limitations of the DH2 strategy when scaling for multiple clients, as it relies solely on RTT for metric selection.

▹**Scenario B.** Table 6 highlights the outstanding performance of StreamWise compared to its competitors across various QoE metrics in the Netflix 5G and LTE Belgium traces, using Merina$^+$ as the ABR algorithm. For the Netflix trace, StreamWise demonstrated an averaged ~5.5% improvement in VMAF as compared to its competitors, delivering highest PSNR, Yin, average bitrate, with negligible rebuffering. The DH1 although performed comparably, suffered ~6x stall as compared to StreamWise. DH2 along the static heuristics on average lags behind in JND by 1× with significantly higher stalls. The LTE Belgium trace closely mimic these results, with StreamWise delivering a ~5.5% improvement in VMAF and

Table 4: Average results of the QoE and its metrics for different network traces for VoD scenario and content MOI (4s). ↑: higher is better (green), lowest performance (red).

			Avg. Perceptual Quality (VMAF)	Avg. Perceptual Quality (PSNR)	Avg. Selected Bitrate (Mbps)	Average RD (%)	Average Yin	Average nSeg-UHD (%)
Netflix 5G	Pensieve+	**StreamWise**	**92.13** ±0.80 ↑	49.46 ±2.40	12.44 ±7.01	0.00	5.00	47.62
		Random	78.23 ±1.68	47.02 ±6.41	6.05 ±5.43	0.00	5.00	10.71
		Deterministic-1	89.38 ±0.72	49.28 ±2.51	10.61 ±6.37	0.00	5.00	35.00
		Deterministic-2	87.38 ±0.64	49.02 ±3.82	10.16 ±6.39	0.00	5.00	38.46
		Round-Robin	88.59 ±0.72	49.32 ±2.51	11.26 ±5.67	0.00	5.00	30.00
		Least connections	86.22 ±0.79	48.22 ±4.32	9.86 ±6.60	0.00	5.00	35.90
		DH1	73.49 ±1.28	46.83 ±6.28	8.45±7.40	0.00	5.00	32.14
		DH2	91.66 ±1.28	**50.08** ±6.28 ↑	**18.46** ±5.51 ↑	0.00	5.00	**51.14** ↑
	Merina+	**StreamWise**	**95.66** ±2.22 ↑	**51.24** ±5.09 ↑	**34.14** ±7.88 ↑	**2.42**±0.74 ↓	5.00	**96.15** ↑
		Random	88.80 ±2.27	49.20 ±3.72	28.60 ±10.80	12.20±21.36	5.00	80.26
		Deterministic-1	92.20 ±1.24	50.62 ±4.12	28.17 ±10.59	4.80±2.22	5.00	90.48
		Deterministic-2	91.87 ±1.32	50.52 ±3.33	27.76 ±10.69	9.61±4.32	5.00	90.00
		Round-Robin	92.20 ±1.24	50.62 ±4.12	28.17 ±10.59	10.20±4.72	5.00	90.48
		Least connections	92.20 ±1.24	50.62 ±4.12	28.17 ±10.59	9.95±4.45	5.00	90.48
		DH1	94.80 ±0.91	51.00 ±3.72	32.33 ±8.89	5.16±2.56	5.00	94.44
		DH2	92.19 ±1.32	49.05 ±3.33	30.89 ±8.60	2.72±0.88	5.00	93.00
	BBA2-C	**StreamWise**	**89.83** ±1.68 ↑	**49.11** ±4.39 ↑	**11.24** ±5.55 ↑	0.00	5.00	**18.92** ↑
		Random	83.50 ±1.06	48.09 ±3.84	7.81 ±5.69	0.00	5.00	3.51
		Deterministic-1	75.83 ±3.72	46.18 ±5.87	5.19 ±3.86	0.00	5.00	1.72
		Deterministic-2	74.74 ±6.08	45.49 ±3.66	4.24 ±2.29	0.00	5.00	3.64
		Round-Robin	77.89 ±0.00	46.01 ±0.00	3.91 ±1.90	0.00	5.00	3.51
		Least connections	77.96 ±1.93	46.12 ±3.68	4.37 ±2.23	0.00	5.00	-
		DH1	87.26 ±1.39	48.49 ±3.08	9.01 ±4.79	0.00	5.00	7.27
		DH2	87.30 ±1.83	48.44 ±3.39	8.94 ±4.36	0.00	5.00	-
LTE Belgium	Pensieve+	**StreamWise**	**89.11** ±0.72↑	**49.21** ±2.51↑	**17.28** ±6.00↑	**0.00** ↓	**5.00** ↑	**60.71**↑
		Random	84.58 ±9.07	47.94 ±6.49	11.83 ± 5.84	**0.00** ↓	**5.00** ↑	34.32
		Deterministic-1	81.87 ±2.45	47.71 ±4.19	6.90 ±5.26	**0.00** ↓	**5.00** ↑	16.25
		Deterministic-2	81.62 ±1.68	47.94 ±4.39	7.87 ±6.04	**0.00** ↓	**5.00** ↑	18.11
		Round-Robin	82.87 ±0.89	48.17 ±3.95	13.20 ±7.71	4.44±1.88	1.00	10.42
		Least connections	88.93 ±1.11	48.79 ±4.23	9.81 ±4.98	10.26±4.74	1.00	22.78
		DH1	64.89 ±1.8	45.19 ±6.67	6.48 ±5.48	**0.00** ↓	**5.00** ↑	25.00
		DH2	84.13 ±0.86	47.43 ±2.52	8.60 ±6.45	**0.00** ↓	**5.00** ↑	36.84
	Merina+	**StreamWise**	**95.40** ±1.95 ↑	**51.16** ±3.40 ↑	**33.59** ↑ ±8.22	3.39±1.18	**5.00** ↑	**95.65** ↑
		Random	85.17 ±0.46	48.68 ±1.98	15.39 ±8.90	31.29±8.96	1.00	74.12
		Deterministic-1	94.54 ±0.65	51.00 ±4.19	32.01 ±9.05	8.06±4.12	**5.00** ↑	94.12
		Deterministic-2	93.68 ±1.21	50.66 ±8.23	30.83 ±9.56	13.95±21.85	**5.00** ↑	92.86
		Round-Robin	91.22 ±0.91	50.41±3.72	26.85 ±10.90	12.26±20.42	**5.00** ↑	88.89
		Least connections	91.59 ±0.93	50.45 ±4.37	27.33 ±10.80	11.84±19.76	**5.00** ↑	89.47
		DH1	94.80 ±0.91	51.00 ±3.72	32.33 ±8.89	5.07±1.56	**5.00** ↑	94.44
		DH2	91.49 ±1.32	48.60 ±3.33	30.46 ±8.60	**2.97**±1.04 ↓	**5.00** ↑	92.60
	BBA2-C	**StreamWise**	**87.30** ±1.83 ↑	**48.44** ±3.39 ↑	**8.94** ±4.36 ↑	0.00	5.00	**43.33** ↑
		Random	82.41 ±1.83	47.72 ±3.39	7.04 ±5.09	0.00	5.00	3.33
		Deterministic-1	75.15 ±3.72	45.72 ±5.87	4.25 ±2.40	0.00	5.00	25.49
		Deterministic-2	77.69 ±7.30	46.04 ±7.56	4.23 ±2.42	0.00	5.00	6.78
		Round-Robin	77.89 ±0.00	46.01 ±0.00	3.91 ±1.90	0.00	5.00	-
		Least connections	77.96 ±1.93	46.12 ±3.68	4.37 ±2.23	0.00	5.00	-
		DH1	86.23 ±1.81	48.43 ±3.55	8.82 ±5.61	0.00	5.00	23.64
		DH2	83.50 ±1.06	48.09 ±3.84	7.81 ±5.69	0.00	5.00	21.43

a ∼1.5× increase in average bitrate along with highest download of segments in ultra-high definition, coupled with an average 6× reduction in rebuffering compared to its competitors, signifying its superiority in the scaled setups in low-latency-live scenarios. The results highlight that DH-based solutions achieve high VMAF but at the cost of increased rebuffering.

4.3.3 Summary. The primary goal of any solution is to closely approximate the theoretical optimum, denoted as OPT. OPT represents the ideal achievable outcome under the given constraints (dynamic programming based), serving as a benchmark for evaluating performance. Figure 7 illustrates the VMAF versus RD analysis, averaged across all traces and ABRs, for both LLL and VoD modes. StreamWise exhibits exceptional performance in both scenarios, achieving results within ∼2% and ∼3% of OPT for LLL and VoD modes, respectively. This underscores StreamWise's ability to make intelligent, real-time decisions, enabling highly precise and efficient CDN selection while operating near theoretical optimality.

4.3.4 CDN Switching Freuency. Across all trace and ABR combinations, we observed a CDN switching frequency of ∼18% in LLL scenario and ∼21% in VoD scenario. The switching rates for both scenarios are depicted in Figure 8. These results highlight the dynamic nature of StreamWise CDN selection, as frequent switches indicate the system's ability to adapt and choose the optimal CDN under varying network conditions.

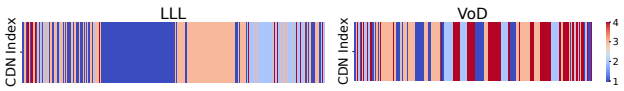


Figure 8: Average CDN switching frequency.

5 Ablation Study and Discussions

5.1 CDN with different Congestion Control

CDN servers connected to back-end servers using different congestion control (CC) algorithms can significantly affect client QoE [2]. To address this, the content-steering server must select the optimal CDN, regardless of the CC algorithm. We tested StreamWise in the MS-SC setup (Figure 5) using Netflix 5G trace at the client-side with various ABR algorithms. The servers employed CC algorithms including BBR, BBR2, Copa2, and Cubic. For brevity, results with Merina+ as the ABR are shown. As illustrated in Figure 9, StreamWise outperformed competitors by ∼3% in LLL mode and ∼10% in VoD mode in terms of VMAF.

Table 5: Average results of the QoE and its metrics for different network traces and content MOI (2s). ↑: higher is better (green), ↓: lower is better (green), lowest performance (red), and scenario: LLL.

		Avg. Perceptual Quality (VMAF)	Avg. Perceptual Quality (PSNR)	Avg. Selected Bitrate (Mbps)	Average RD (%)	Average Yin	Average nSeg-UHD (%)
LTE Belgium	**StreamWise**	**96.57** ±0.89 ↑	**51.35** ±2.54 ↑	**35.32** ↑ ±7.08	**2.91**±4.38 ↓	5.00	**97.14** ↑
	Merina+ Random	93.45 ±1.06	50.70 ±4.90	30.15 ±9.54	19.11±26.94	5.00	92.26
	Merina+ Deterministic-1	94.48 ±1.06	50.87 ±3.14	31.77 ±8.68	17.34±24.32	5.00	93.66
	Merina+ Deterministic-2	92.20 ±1.78	50.42 ±6.15	28.08 ±10.29	15.82±22.94	5.00	90.09
	Merina+ Round-Robin	94.05 ±0.84	50.84 ±5.30	30.85 ±9.70	20.64±18.06	5.00	93.39
	Merina+ Least connections	94.86 ±1.05	51.00 ±6.49	32.29 ±8.89	16.56±17.14	5.00	94.59
	Merina+ DH1	94.93 ±1.86	49.44 ±3.99	33.45 ±6.17	12.87±12.68	5.00	94.00
	Merina+ DH2	89.76 ±0.81	48.15 ±5.40	25.16 ±11.19	8.87±4.54	5.00	86.67
FD	**StreamWise**	**96.95** ±0.84 ↑	**51.52** ±4.32 ↑	**36.39** ±6.17 ↑	**2.18**±3.98 ↓	5.00	**97.93** ↑
	Merina+ Random	92.24 ±1.28	50.54 ±5.14	28.80 ±9.64	11.32±12.12	5.00	90.47
	Merina+ Deterministic-1	94.58 ±1.26	50.85 ±3.23	31.95 ±8.60	10.82±11.74	5.00	93.85
	Merina+ Deterministic-2	93.27 ±1.15	50.85 ±6.00	29.67 ±9.88	9.06±10.24	5.00	92.02
	Merina+ Round-Robin	93.45 ±0.87	50.69 ±4.10	29.85 ±10.00	20.51±17.97	5.00	92.26
	Merina+ Least connections	94.00 ±0.83	50.83 ±3.59	30.72 ±9.48	15.53±16.32	5.00	93.01
	Merina+ DH1	91.99 ±1.26	49.60 ±2.88	28.65 ±5.98	8.40±9.82	5.00	92.15
	Merina+ DH2	88.44 ±1.32	48.84 ±3.33	23.62 ±11.44	9.74±10.02	5.00	85.00

Table 6: Average results of the QoE and its metrics for different network traces and content MOI (4s). ↑: higher is better (green), ↓: lower is better (green), lowest performance (red), and scenario: VoD.

		Avg. Perceptual Quality (VMAF)	Avg. Perceptual Quality (PSNR)	Avg. Selected Bitrate (Mbps)	Average RD (%)	Average Yin	Average nSeg-UHD (%)
Netflix 5G	**StreamWise**	**96.07** ±1.73↑	**51.42** ±4.07↑	**34.89** ±7.39↑	**2.23** ±5.04↓	5.00	**96.79**↑
	Merina+ Random	91.70 ±1.46	50.29 ±3.92	27.76 ±10.60	22.40 ±18.16	5.00	89.45
	Merina+ Deterministic-1	88.99 ±1.88	49.91 ±3.55	25.18 ±10.40	16.37 ±17.12	5.00	85.55
	Merina+ Deterministic-2	93.62 ±1.23	50.84 ±10.25	30.51 ±94.00	12.66 ±12.58	5.00	91.64
	Merina+ Round-Robin	91.53 ±1.01	50.47 ±3.95	27.29 ±10.79	17.34 ±17.60	5.00	89.41
	Merina+ Least connections	90.96 ±1.16	50.34 ±3.62	26.64 ±10.88	16.14 ±14.75	5.00	88.62
	Merina+ DH1	95.75 ±1.18	51.18 ±4.99	34.17 ±7.84	12.93 ±5.66	5.00	93.37
	Merina+ DH2	86.02 ±1.01	49.49 ±3.51	21.16 ±11.40	8.52 ±8.97	5.00	81.21
LTE Belgium	**StreamWise**	**95.97** ±0.83↑	**51.26** ±3.31↑	**34.57** ±7.60↑	**2.98**↓ ±5.05	5.00	**96.53**↑
	Merina+ Random	91.85 ±1.63	50.36 ±4.38	28.33 ±10.25	18.75 ±21.84	5.00	90.12
	Merina+ Deterministic-1	89.71 ±1.34	49.97 ±5.84	25.69 ±10.70	23.76 ±22.65	5.00	86.63
	Merina+ Deterministic-2	92.52 ± 1.17	50.53 ±5.43	29.51 ±9.51	21.58 ±21.99	5.00	90.84
	Merina+ Round-Robin	90.84 ±1.09	50.38 ±6.92	26.42 ±10.97	21.53 ±20.65	5.00	88.33
	Merina+ Least connections	90.65 ±0.99	50.34 ±7.11	26.48 ±10.74	20.51 ±20.78	5.00	88.14
	Merina+ DH1	95.64 ±2.07	51.22 ±4.91	34.07 ±7.92	7.95 ±6.09	5.00	96.09
	Merina+ DH2	85.62 ±1.73	49.21 ±3.49	20.73 ±11.43	3.93 ±27.25	5.00	80.00

Table 7: 8K streaming. StreamWise vs. others.

Method	Avg. Bitrate (Mbps)	RD (%)
StreamWise	**52.17** ↑	1.32
DH1	33.67	3.86
DH2	33.21	3.86

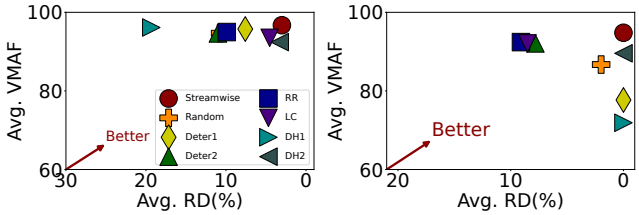


Figure 9: Different CC analysis. LLL (left) and VoD (right).

5.2 Videos with 8K Resolution

To evaluate StreamWise's scalability to 8K resolutions, we extended the experiments from Section 4 using 8K videos under Netflix 5G traces with Merina+ as the ABR in the LLL scenario. Experiments were conducted with the Atomic Hearts video [22] (120 fps) and focused on dynamic heuristics, as static strategies were excluded due to their poor performance with 8K content. As shown in Table 7, StreamWise achieved ~56% higher average bitrate with minimal rebuffering, outperforming its competitors.

5.3 StreamWise Potential Extension

Although StreamWise demonstrates substantial improvements in the QoE during video streaming sessions, there remains potential for enhancement. For large-scale systems, StreamWise can be expanded into a multi-agent framework, where multiple agents are assigned to different clients located at various regions, while a central agent functions as a load-balancer for CDN allocation. Additionally, a threshold-based switching logic could be incorporated into the StreamWise architecture, enabling the central agent to avoid CDN switching when the QoE improvements are negligible. We acknowledge the importance of addressing these limitations in the current design and aim to explore them in future work.

6 Conclusions

In this paper, we presented StreamWise, a DRL-based content steering solution that optimizes CDN selection in DASH. By integrating minimal input metrics–RTT, bitrate, and rebuffering duration–StreamWise consistently outperforms state-of-the-art multi-CDN strategies. Extensive experiments in both Video-on-Demand (VoD) and low-latency live (LLL) streaming scenarios exhibited notable QoE improvements, with up to 8.5% higher VMAF, 1.5x higher average bitrate, and 48% better QoE in VoD, alongside a ~10× reduction in rebuffering for live streaming. These results demonstrate StreamWise's effectiveness in enhancing viewer QoE, paving the way for intelligent CDN selection in future streaming systems.

Acknowledgments. This research is funded by the Natural Sciences & Engineering Research Council (NSERC)-Discovery Grant RGPIN-2023-04744 and NSERC Alliance (ALLRP 586434-23)-Mitacs Accelerate (IT36792) grant.

References

[1] ETSI TS 103 998. 2024. ETSI TS 103 998 V1.1.1:DASH-IF: Content Steering for DASH. https://cdn.standards.iteh.ai/samples/67988/93d192067f4d434c9103d1b2317eb4a9/ETSI-TS-103-998-V1-1-1-2024-01-.pdf.

[2] Soheil Abbasloo. 2023. Internet Congestion Control Benchmarking. *arXiv preprint arXiv:2307.10054* (2023).

[3] Akamai. 2024. cell-emulation-util GitHub. https://github.com/akamai/cellemulation-util Accessed: 2024-09-20.

[4] T Alexander. 2014. Visual-PSNR measure of image quality Journal of Visual Communication and Image Representation. (2014).

[5] Kai Arulkumaran, Marc Peter Deisenroth, Miles Brundage, and Anil Anthony Bharath. 2017. Deep reinforcement learning: A brief survey. *IEEE Signal Processing Magazine* 34, 6 (2017), 26–38.

[6] Abdelhak Bentaleb, Reza Farahani, Farzad Tashtarian, Hermann Hellwagner, and Roger Zimmermann. 2023. Which CDN to Download From? A Client and Server Strategies. In *Proceedings of the 2nd Mile-High Video Conference*. 135–136.

[7] Abdelhak Bentaleb, Bayan Taani, Ali C Begen, Christian Timmerer, and Roger Zimmermann. 2018. A survey on bitrate adaptation schemes for streaming media over HTTP. *IEEE Communications Surveys & Tutorials* 21, 1 (2018), 562–585.

[8] Abdelhak Bentaleb, Praveen Kumar Yadav, Wei Tsang Ooi, and Roger Zimmermann. 2020. DQ-DASH: A queuing theory approach to distributed adaptive video streaming. *ACM Transactions on Multimedia Computing, Communications, and Applications (TOMM)* 16, 1 (2020), 1–24.

[9] c3m3gyanesh. 2023. Working with Longer duration video. https://github.com/Telecommunication-Telemedia-Assessment/bitstream_mode3_p1204_3/issues/31. Accessed on Oct. 20, 2024.

[10] Fangfei Chen, Ramesh K Sitaraman, and Marcelo Torres. 2015. End-user mapping: Next generation request routing for content delivery. *ACM SIGCOMM Computer Communication Review* 45, 4 (2015), 167–181.

[11] Zhenzhong Chen and Hongyi Liu. 2014. JND modeling: Approaches and applications. In *2014 19th International Conference on Digital Signal Processing*. 827–830. https://doi.org/10.1109/ICDSP.2014.6900782

[12] Consumer Technology Association. 2020. CTA-5004: Web Application Video Ecosystem–Common Media Client Data.

[13] Consumer Technology Association. 2022. CTA-5006: Web Application Video Ecosystem–Common Media Server Data.

[14] DASH-IF. 2024. DASH Reference Client. https://reference.dashif.org/dash.js/. Accessed on Oct. 20, 2024.

[15] Dataset. 2024. Video Dataset. https://ftp.itec.aau.at/datasets/DASHDataset2014/. Accessed on Sept. 26, 2024.

[16] Peter I Frazier. 2018. A tutorial on Bayesian optimization. *arXiv preprint arXiv:1807.02811* (2018).

[17] Aditya Ganjam, Faisal Siddiqui, Jibin Zhan, Xi Liu, Ion Stoica, Junchen Jiang, Vyas Sekar, and Hui Zhang. 2015. C3:{internet-scale} control plane for video quality optimization. In *NSDI*. 131–144.

[18] Palak Goenka, Kyriakos Zarifis, Arpit Gupta, and Matt Calder. 2022. Towards client-side active measurements without application control. *ACM SIGCOMM Computer Communication Review* 52, 1 (2022), 20–27.

[19] Joris Herbots, Mike Vandersanden, Peter Quax, and Wim Lamotte. 2023. Vegvisir: A testing framework for HTTP/3 media streaming. In *Proceedings of the 14th Conference on ACM Multimedia Systems*. 403–409.

[20] Joris Herbots, Arno Verstraete, Maarten Wijnants, Peter Quax, and Wim Lamotte. 2023. Cross that boundary: Investigating the feasibility of cross-layer information sharing for enhancing ABR decision logic over QUIC. In *Proceedings of the 33rd Workshop on Network and Operating System Support for Digital Audio and Video*. 50–57.

[21] Te-Yuan Huang, Ramesh Johari, Nick McKeown, Matthew Trunnell, and Mark Watson. 2014. A buffer-based approach to rate adaptation: Evidence from a large video streaming service. In *Proceedings of the 2014 ACM conference on SIGCOMM*. 187–198.

[22] Joumxyzptlk. 2024. Atomic Hearts RTX Demo. https://joumxyzptlk.de/atomic_hearts_rtx_demo.html

[23] Nuowen Kan, Yuankun Jiang, Chenglin Li, Wenrui Dai, Junni Zou, and Hongkai Xiong. 2022. Improving generalization for neural adaptive video streaming via meta reinforcement learning. In *Proceedings of the 30th ACM International Conference on Multimedia*. 3006–3016.

[24] Burak Kara and Gwendal Simon. 2024. Power Efficient Multi-CDN Communication over Content Steering Server. In *Proceedings of the 15th ACM Multimedia Systems Conference*. 478–484.

[25] Chen Ma and Yuhong Chi. 2022. Evaluation test and improvement of load balancing algorithms of nginx. *IEEE Access* 10 (2022), 14311–14324.

[26] Liang Ma. 2024. Multi-Video CDN system: Availability, Performance and Cost characteristics. In *Proceedings of the 3rd Mile-High Video Conference*. 118–119.

[27] Hongzi Mao, Ravi Netravali, and Mohammad Alizadeh. 2017. Neural adaptive video streaming with pensieve. In *ACM SIGCOMM*. 197–210.

[28] Angelique Medina. 2024. The Akamai DNS Outage and the Case for CDN Redundancy. *ThousandEyes Blog* (2024). https://www.thousandeyes.com/blog/akamai-edge-dns-outage-analysis Accessed: 2024-11-22.

[29] mvfst. 2024. mvfst QUIC Implementation. https://github.com/facebook/mvfst. Accessed on Sept. 26, 2024.

[30] Luthfan Hadi Pramono, Robby Cokro Buwono, and Yanuar Galih Waskito. 2018. Round-robin algorithm in HAProxy and Nginx load balancing performance evaluation: a review. In *2018 international seminar on research of information technology and intelligent systems (ISRITI)*. IEEE, 367–372.

[31] QUIC-Go. 2024. QUIC-go Implementation. https://github.com/quic-go/quic-go. Accessed on Sept. 26, 2024.

[32] Darijo Raca, Dylan Leahy, Cormac J Sreenan, and Jason J Quinlan. 2020. Beyond throughput, the next generation: A 5G dataset with channel and context metrics. In *Proceedings of the 11th ACM multimedia systems conference*. 303–308.

[33] Darijo Raca, Maëlle Manifacier, and Jason J. Quinlan. 2020. goDASH — GO Accelerated HAS Framework for Rapid Prototyping. In *2020 Twelfth International Conference on Quality of Multimedia Experience (QoMEX)*. 1–4. https://doi.org/10.1109/QoMEX48832.2020.9123103

[34] Reza Rassool. 2017. VMAF reproducibility: Validating a perceptual practical video quality metric. In *2017 IEEE international symposium on broadband multimedia systems and broadcasting (BMSB)*. IEEE, 1–2.

[35] Will Reese. 2008. Nginx: the high-performance web server and reverse proxy. *Linux Journal* 2008, 173 (2008), 2.

[36] Yuriy Reznik, Guillem Cabrera, Daniel Silhavy, Stefan Pham, Alex Giladi, Alex Balk, Ali C Begen, and Will Law. 2024. Content Steering: a Standard for Multi-CDN Streaming. In *Proceedings of the 3rd Mile-High Video Conference*. 128–128.

[37] Yuriy Reznik, Adam Waldron, and Guillem Cabrera. 2023. Simplifying multi-CDN delivery with HLS/DASH content steering. In *SMPTE 2023 Media Technology Summit*. SMPTE, 12–15.

[38] Haakon Riiser, Paul Vigmostad, Carsten Griwodz, and Pål Halvorsen. 2013. Commute path bandwidth traces from 3G networks: Analysis and applications. In *Proceedings of the 4th ACM Multimedia Systems Conference*. 114–118.

[39] Werner Robitza, Steve Göring, Alexander Raake, David Lindegren, Gunnar Heikkilä, Jörgen Gustafsson, Peter List, Bernhard Feiten, Ulf Wüstenhagen, Marie-Neige Garcia, et al. 2018. HTTP adaptive streaming QoE estimation with ITU-T rec. P. 1203: Open databases and software. In *Proceedings of the 9th ACM Multimedia Systems Conference*. 466–471.

[40] Sandvine. 2024. The Global Internet Phenomena Report, White Paper. https://www.sandvine.com/global-internet-phenomena-report-2024. *Technical Report, Sandvine* (January 2024). Accessed on Oct. 20, 2024.

[41] John Schulman, Filip Wolski, Prafulla Dhariwal, Alec Radford, and Oleg Klimov. 2017. Proximal policy optimization algorithms. *arXiv preprint arXiv:1707.06347* (2017).

[42] Robert Seeliger, Daniel Silhavy, and Stefan Arbanowski. 2017. Dynamic ad-insertion and content orchestration workflows through manifest manipulation in HLS and MPEG-DASH. In *2017 IEEE Conference on Communications and Network Security (CNS)*. IEEE, 450–455.

[43] Anees Shaikh, Renu Tewari, and Mukesh Agrawal. 2001. On the effectiveness of DNS-based server selection. In *Proceedings IEEE INFOCOM 2001. Conference on Computer Communications. Twentieth Annual Joint Conference of the IEEE Computer and Communications Society (Cat. No. 01CH37213)*, Vol. 3. IEEE, 1801–1810.

[44] J. S. Sidhu, C. Joshi, and A. Bentaleb. 2025. A Multi-CDN Playground for Dash.js: Enabling Integration of CDN Switching Strategies. In *Proceedings of the 16th ACM Multimedia Systems Conference (appearing online soon)*. ACM.

[45] Daniel Silhavy, Will Law, Stefan Pham, Ali C Begen, Alex Giladi, and Alex Balk. 2023. Dynamic CDN switching-DASH-IF content steering in dash. js. In *Proceedings of the 2nd Mile-High Video Conference*. 130–131.

[46] slhck. 2020. Memory issues. https://github.com/Telecommunication-Telemedia-Assessment/bitstream_mode3_p1204_3/issues/16.

[47] Streaming Video Technology Alliance (SVTA). 2023. Investigating Approaches to Multi-CDN Delivery. https://www.svta.org/2023/01/03/investigating-approaches-to-multi-cdn-delivery/ Accessed: 2024-11-22.

[48] Babak Taraghi, Hadi Amirpour, and Christian Timmerer. 2022. Multi-codec ultra high definition 8K MPEG-DASH dataset. In *ACM MM*.

[49] Twitch. 2020. ACM MMSys 2020 grand challenge. https://github.com/Wimnet/twitch_challenge. In *ACM MMSys*.

[50] Jeroen Van Der Hooft, Stefano Petrangeli, Tim Wauters, Rafael Huysegems, Patrice Rondao Alface, Tom Bostoen, and Filip De Turck. 2016. HTTP/2-based adaptive streaming of HEVC video over 4G/LTE networks. *IEEE Communications Letters* 20, 11 (2016), 2177–2180.

[51] Xiaoqi Yin, Abhishek Jindal, Vyas Sekar, and Bruno Sinopoli. 2015. A control-theoretic approach for dynamic adaptive video streaming over HTTP. In *Proceedings of the 2015 ACM Conference on Special Interest Group on Data Communication*. 325–338.

[52] Behrouz Zolfaghari, Gautam Srivastava, Swapnoneel Roy, Hamid R Nemati, Fatemeh Afghah, Takeshi Koshiba, Abolfazl Razi, Khodakhast Bibak, Pinaki Mitra, and Brijesh Kumar Rai. 2020. Content delivery networks: State of the art, trends, and future roadmap. *ACM Computing Surveys (CSUR)* 53, 2 (2020), 1–34.



SGSS: Streaming 6-DoF Navigation of Gaussian Splat Scenes

Mufeng Zhu
Rutgers University
Piscataway, NJ, USA
mz526@rutgers.edu

Mingju Liu
University of Maryland, College Park
College Park, MD, USA
mliu9867@umd.edu

Cunxi Yu
University of Maryland, College Park
College Park, MD, USA
cunxiyu@umd.edu

Cheng-Hsin Hsu
National Tsing Hua University
Hsin-Chu, Taiwan
chsu@cs.nthu.edu.tw

Yao Liu
Rutgers University
Piscataway, NJ, USA
yao.liu@rutgers.edu

Abstract

3D Gaussian Splatting (3DGS) is an emerging approach for training and representing real-world 3D scenes. Due to its photorealistic novel view synthesis and fast rendering speed (e.g., over 100 FPS), it has the potential to transform how scenes that can be explored in 6 degrees-of-freedom (6-DoF) are represented. However, a limiting factor of 3DGS is its large size, which requires high network bandwidth for streaming reconstructed real-world 3D scenes.

In this paper, we propose *SGSS* for optimizing the streaming transmission of 3DGS scenes during 6-DoF navigation. Since not all Gaussians in the full scene are needed for rendering a user's view, *SGSS* uses view-adaptive streaming, enabled by optimized spatial partitioning of the scene, for achieving network transmission savings. For each spatial partition, *SGSS* uses an importance-based Gaussians pre-sorting scheme to enhance the initial view quality and reduce the user-perceived scene loading time. We further design a client-side view-adaptive streaming algorithm that features lightweight visibility checking, prioritized streaming, incremental processing, and stream pausing/resuming schemes. We implement *SGSS* with JavaScript and WebGL2. Evaluation results show that the quality of views rendered with *SGSS* streaming is consistently higher than or on par with state-of-the-art approaches. Furthermore, the view-adaptive streaming of *SGSS* can result in high savings in network transmission without impacting the view quality.

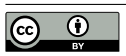
CCS Concepts

• **Information systems** → **Multimedia streaming**; • **Computing methodologies** → **Point-based models**; • **Human-centered computing** → **Mixed / augmented reality** .

Keywords

3D Gaussian Splatting, View-Adaptive Streaming, Visual Quality

ACM Reference Format:
Mufeng Zhu, Mingju Liu, Cunxi Yu, Cheng-Hsin Hsu, and Yao Liu. 2025. *SGSS*: Streaming 6-DoF Navigation of Gaussian Splat Scenes. In *ACM Multimedia Systems Conference 2025 (MMSys '25), March 31-April 4, 2025, Stellenbosch, South Africa.* ACM, New York, NY, USA, 11 pages. https://doi.org/10.1145/3712676.3714437



This work is licensed under a Creative Commons Attribution 4.0 International License.
MMSys '25, Stellenbosch, South Africa
© 2025 Copyright held by the owner/author(s).
ACM ISBN 979-8-4007-1467-2/25/03
https://doi.org/10.1145/3712676.3714437

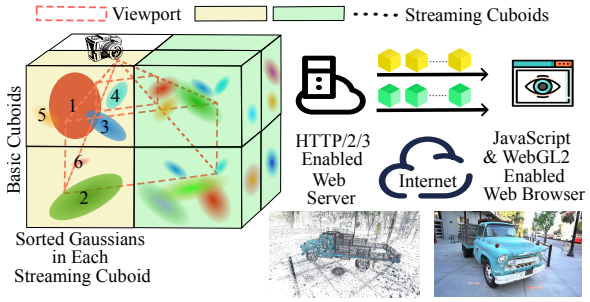


Figure 1: Illustration of our proposed SGSS streaming system.

1 Introduction

In recent years, both immersive applications and VR/AR devices such as Apple Vision Pro have gained increased popularity among users. This has resulted in an increasing demand for recording, reconstruction, and streaming of real-world 3D scenes for providing immersive experiences that can be navigated in 6 degrees-of-freedom (6-DoF) to users. Traditional representations of 3D scenes include point clouds and triangular meshes. To capture real-world 3D scenes, raw point clouds can be obtained by fusing data recorded sets of cameras and/or other sensors such as LiDAR sensors [1, 2]. Due to noises inherent in data produced by the sensors [3] and the size of rendered points, point clouds can suffer from inaccurate geometry and "holes" in rendered views [4]. Reconstructing accurate triangular meshes from point clouds is also challenging [5].

Recent advances in 3D reconstruction and novel-view synthesis have seen the emergence of learned representations such as NeRF [6] and 3D Gaussian Splatting (3DGS) [7], and much effort has been devoted to follow-up works of these representations, e.g., [8–11]. Both NeRF and 3DGS represent 3D scenes via parameters trained via gradient descent, given input images of the scene recorded from different perspectives. While NeRF and its variants can achieve photorealistic novel view synthesis, their main issue is with the high computation overhead associated with volumetric view rendering. As a result, it is presently infeasible to achieve high-quality view rendering with high frames-per-second (FPS). Compared to NeRF, 3DGS can achieve over 100 FPS rendering speed, an order of magnitude faster than NeRF-like approaches [12].

3DGS is a point-based representation. It uses anisotropic 3D Gaussians to represent the geometry of the 3D scenes. Unlike point clouds that may suffer from "holes" in rendered views, each 3D Gaussian is projected to a 2D splat (e.g., a circular/elliptical disc) during rendering, which can occupy a larger extent than a single

point's projection. 3DGS has been shown to provide photorealistic view synthesis with real-time rendering speed. Its main drawback, however, is the large representation size. Each 3D Gaussian occupies over 200 bytes of space for storing its attributes such as position, scale, rotation, and opacity. It also uses 48 spherical harmonics (SH) coefficients for representing view-dependent color. To produce a good representation of a scene, millions of Gaussians may be needed, resulting in significant storage requirements. For example, the trained 3DGS representation for the `garden` scene [11] contains over 5.8 million Gaussians and is 1.4 GBytes.

To address the significant representation size of 3DGS, approaches have been proposed to reduce the per-Gaussian size by only retaining the DC components of the SH coefficients while removing the rest (45) coefficients. This can yield 180 bytes savings out of 248 bytes (72%) of each Gaussian [13]. Even so, due to the large number of Gaussians in a scene, the total scene size is still large, e.g., hundreds of MBytes. Recent research also proposed to reduce the number of Gaussians without significantly impacting the view quality. For example, Fan et al. propose to prune Gaussians based on a global significance score calculated during training [10].

As an immersive media representation, 3DGS includes all the information needed for viewing the scene from any position in any orientation (thereby supporting 6-DoF navigation). However, at a time, only part of the scene is visible in the user's viewport. Downloading the full scene while only part of the data will be needed inevitably results in wasted data. Furthermore, after requesting the scene, the user expects to see a high-quality view of the scene as soon as possible. So Gaussians within the viewport should be transmitted first before others. To the best of our knowledge, however, no existing work has considered optimizing the transmission of Gaussians needed for rendering the user's view. Some solutions require the full 3DGS scene to be downloaded (a long time) before views are rendered to the users [14]. Others support progressive downloading and rendering of Gaussians, allowing the user to view the scene before it is fully downloaded [13, 15]. Nonetheless, both types of solutions end up downloading all Gaussians of the scene.

In this paper, we propose *SGSS* – **S**treaming 6-DoF navigation of **G**aussian **S**plat **S**cenes – for optimizing streaming of 3DGS scenes. Figure 1 provides a high-level overview of the streaming system.

Optimized Spatial Partitioning for View-Adaptive Streaming. *First*, since not all Gaussians are needed for rendering a view of the scene, we propose view-adaptive streaming of Gaussians, which can save network transmission. Due to the vast number (millions) of Gaussians within a 3DGS scene, it is infeasible to use individual Gaussians as the transmission scheduling unit. Instead, we propose to spatially partition the 3DGS scene into non-overlapping cuboids and use them to support view-adaptive streaming. To minimize the expected network downloading under a view distribution pattern while taking into account the overhead associated with using small cuboids, we formulate an optimization problem to find the best way to spatially partition the scene into cuboids.

Importance-Based Gaussians Pre-Sorting. *Second*, we propose a ranking scheme for pre-sorting Gaussians within each cuboid to enhance the initial view quality and reduce the user-perceived scene loading time. The need to reduce the user-perceived scene loading time is motivated by the "largest contentful paint" (LCP) metric for measuring how quickly the main content of a webpage is downloaded and becomes visible to the user [16]. Based on the observation that Gaussians with higher opacity and larger scale have a greater impact on the rendered view, our ranking scheme prioritizes the streaming and rendering of these important Gaussians, allowing meaningful views to be rendered faster.

***SGSS* Streaming Algorithm.** *Third*, we design a view-adaptive streaming algorithm for the client-side. It includes a lightweight cuboid visibility test for checking which cuboids are inside the viewport. Since all visible cuboids are streamed in a multiplexed connection in modern HTTP protocols, we design a practical scheme for assigning priority weights to streams, based on each visible stream cuboid's density of Gaussian ellipsoids, contribution to the viewport, and distance from the camera origin. The streaming algorithm also supports incremental response (i.e., Gaussians) downloading and processing, fully realizing the benefits of pre-sorting Gaussians inside cuboids for faster initial visual quality improvements.

We implemented *SGSS* in JavaScript with WebGL2 and conducted extensive experiments for evaluation. Results show that views rendered with *SGSS* streaming are consistently of visual quality higher or on par with state-of-the-art approaches. Furthermore, the view-adaptive transmission of *SGSS* can result in 32.9% savings on average in network transmission. In summary, this paper makes the following contributions:

- We propose optimized spatial partitioning of the 3DGS scene into cuboids to support the viewport-adaptive transmission.
- We further propose importance-based Gaussians pre-sorting within cuboids for enhancing the initial view quality.
- We design a viewport-adaptive streaming algorithm for 6-DoF navigation of 3DGS scenes and implement our proposed solution in JavaScript with WebGL2.
- Extensive experiment results show that *SGSS* can improve the visual quality during the initial streaming phase and save network bandwidth compared to state-of-the-art approaches.

2 Background and Related Works

Representation of each 3D Gaussian. A 3DGS scene is represented by a set of 3D Gaussians with their parameters optimized through a training procedure via backpropagation. Each 3D Gaussian is represented using information about its position (i.e., mean) $\boldsymbol{\mu} = (\mu_x, \mu_y, \mu_z) \in \mathbb{R}^3$, a covariance matrix $\Sigma \in \mathbb{R}^{3\times3}$, color $\mathbf{c}$, and opacity $o \in \mathbb{R}$. To facilitate optimization of parameters using gradient descent, the covariance matrix Σ is equivalently represented as the scale and rotation of an ellipsoid [7]. That is, $\Sigma = RSS^TR^T$, where $S \in \mathbb{R}^{3\times3}$ is a diagonal matrix, representing the scaling along each of the 3 dimensions; and $R \in \mathbb{R}^{3\times3}$ is a 3D rotation matrix. S is stored as $\mathbf{s} = (s_x, s_y, s_z) \in \mathbb{R}^3$, $S = \text{diag}(\mathbf{s})$; and the rotation matrix R is stored as quaternion $\mathbf{q} \in \mathbb{R}^4$. 3DGS further uses spherical harmonics (SH) for representing view-dependent color. For a single color channel, with k degrees of freedom, $(k+1)^2$ SH coefficients are needed. Given that 3 degrees of view direction freedom is required, a total of $16 \times 3 = 48$ SH coefficients are needed for R/G/B color channels. Overall, each 3D Gaussian needs to store 59 floating point numbers (236 Bytes) for representing the position $\boldsymbol{\mu} \in \mathbb{R}^3$, scale $\mathbf{s} \in \mathbb{R}^3$, rotation quaternion $\mathbf{q} \in \mathbb{R}^4$, opacity $o \in \mathbb{R}$, and view-dependent color $\mathbf{c} \in \mathbb{R}^{3\times(3+1)^2}$. 3D Gaussians are stored in a `.ply` file, with each line of the file containing attributes of a 3D Gaussian.

3D Gaussian "splatting". During rendering, 3D Gaussians are projected to 2D splats, such as circular or elliptical discs. This effectively avoids the appearance of "holes" in the rendered views, an issue with the traditional point-based representation like point clouds. The projection is done by $\Sigma' = JW\Sigma W^T J^T$, where $\Sigma' \in \mathbb{R}^{2\times 2}$ is the 2D covariance matrix, W is the view transformation matrix, and J is the affine transform approximation of the perspective projection transformation [7, 17, 18].

Pixel rendering. Given a pixel i, a list of overlapping Gaussians can be obtained, transformed to the camera space, and sorted based on their depth. With $\mathcal{N}$ sorted Gaussians, the color of pixel i can be obtained via alpha blending:

$$C_i = \sum_{n\leq \mathcal{N}} \Big(c_n \cdot \alpha_n \cdot \prod_{m<n} (1-\alpha_m)\Big),$$

where α_n is computed using the 2D covariance matrix Σ' and opacity parameter o_n [17].

Compression of 3DGS representations. Existing literature in point cloud compression has explored efficient data structures such as octrees (e.g., G-PCC [19, 20]) and kd-trees (e.g., Draco [21]) for compressing the geometry information (i.e., point positions). For compressing attributes (e.g., color) of point clouds, region-adaptive hierarchical transform (RAHT) [22], quantization, and entropy coding can be used. For dynamic scenes (i.e., point cloud video), video-based point cloud compression (V-PCC) leverages existing 2D video codec [19]. Since 3DGS is a point-based representation, it can also apply point cloud compression for compressing the geometry (i.e., positions of Gaussians) and other attributes, e.g., [23]. Since the emergence of 3DGS, several 3DGS compression schemes have been proposed. For example, Compact3DGS [24] proposes to i) reduce the number of Gaussians by identifying non-essential Gaussians with a masking strategy; and ii) compress the attributes by using a grid-based neural field and vector quantization. LightGaussian [10] proposes Gaussian pruning based on a global significance score obtained during training and an SH distillation approach for reducing the attribute size. Compressed3DGS [25] uses vector quantization [26, 27] for compression. It proposes sensitivity-aware vector clustering for creating a compact codebook. While compression is orthogonal to the tasks of our paper, we note that these compression solutions can be integrated with our proposed approaches.

Transporting 3DGS over the network. `gsplat` [14] is one of the earliest demo of 3DGS web viewers. However, it does not support streaming of 3DGS, and users must wait for the full representation to be downloaded to view the scene. `GaussianSplats3D` [15] is another 3DGS web viewer based on JavaScript and WebGL. It pre-sorts 3D Gaussians in the scene based on their distance from the world origin (i.e., $(0, 0, 0)$ in the world coordinate system). That is, pre-sorting based on the L2 norm of $\boldsymbol{\mu}$, $||\boldsymbol{\mu}||$. In this way, Gaussians are downloaded from the world origin going outward. However, this scheme works well only if the viewport is centered at the world origin. `splat` [13] also supports streaming, and it pre-sorts Gaussians based on a metric considering both the scale and opacity. This ensures that Gaussians with large volumes and high opacity are downloaded first during streaming. Despite the pre-sorting, these approaches download all Gaussians in the full scene regardless of the user's viewport, wasting network bandwidth. Unlike these approaches, in this paper, we propose spatial partitioning of the 3DGS scene to enable view-adaptive streaming, saving network bandwidth while maintaining or improving the quality of rendered views. ViVo [28] is a state-of-the-art point cloud streaming approach. It can save streaming bandwidth by adjusting the point density of each spatial cell based on its visibility in the viewport. However, it does not work well for 3DGS streaming due to the special characteristics of 3D Gaussians (Section 5.5).

3 Design of *SGSS*

We design *SGSS* to both enhance the visual quality during scene streaming and achieve network bandwidth savings. The core of *SGSS* is a view-adaptive streaming solution. Minimizing network bandwidth requires only Gaussians within the viewport to be downloaded. However, individually checking the visibility of millions of Gaussians and scheduling transmission is a daunting task. On the other hand, transmitting the full scene without spatial partitioning can result in the least visibility testing and scheduling overhead but can lead to a significant amount of unused data to be downloaded. To balance these two conflicting objectives, we propose **optimized spatial partitioning** of the 3DGS scene in Section 3.1.

Since 3DGS is a representation with parameters of Gaussians trained via a gradient descent approach, each Gaussian can contribute differently to the final view, depending on its scale and opacity, and Gaussians are not uniformly distributed in the scene. This is different from traditional point clouds. Based on these unique characteristics of 3D Gaussians, *SGSS* uses an **importance score** metric (Section 3.2) to measure the importance of each Gaussian within the streaming cuboid and pre-sorts the Gaussians accordingly. This allows the most important Gaussians to be streamed first. *SGSS* further implements an **efficient viewport-adaptive streaming algorithm** for the client-side (Section 3.3). It uses a lightweight scheme visibility checking of cuboids. It then prioritizes the streaming of cuboids that contribute more to the rendering result, based on a carefully-designed priority weight metric. In addition, *SGSS* leverages modern HTTP protocol capabilities and supports incremental cuboid downloading and processing, as well as pausing and resuming unfinished downloads for enhancing the initial view quality and quality improvement.

3.1 Optimized Spatial Partitioning of the Scene

We first generate an Axes Aligned Bounding Box (AABB) for the full 3DGS scene, based on the extent of the Gaussians' positions in the world coordinate system. We then partition it into **basic cuboids** with the same size. The total number of basic cuboids in a scene depends on the shape of the scene's AABB, ensuring that each basic cuboid approximates a cube. We define a **streaming cuboid** as a partition of the scene that can be scheduled for streaming transmission. It is constructed by one or more basic cuboids. Suppose that along the three dimensions (i.e., width, height, depth) of the AABB, we divide the scene into i, j, and k partitions, respectively. Then the total number of basic cuboids is $i \cdot j \cdot k$, and the total number of possible, valid, streaming cuboids can be obtained using binomial coefficients as:

$$num_valid_streaming_cuboids = \binom{i+1}{2} \cdot \binom{j+1}{2} \cdot \binom{k+1}{2}. \quad (1)$$

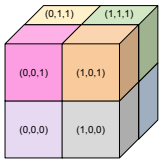


Figure 2: This figure shows a simple example where an Axes Aligned Bounding Box (AABB) of a full 3DGS scene is partitioned into 2x2x2 basic cuboids. With this partitioning, a total of 27 possible cuboids can be created (shown in Figure 3).

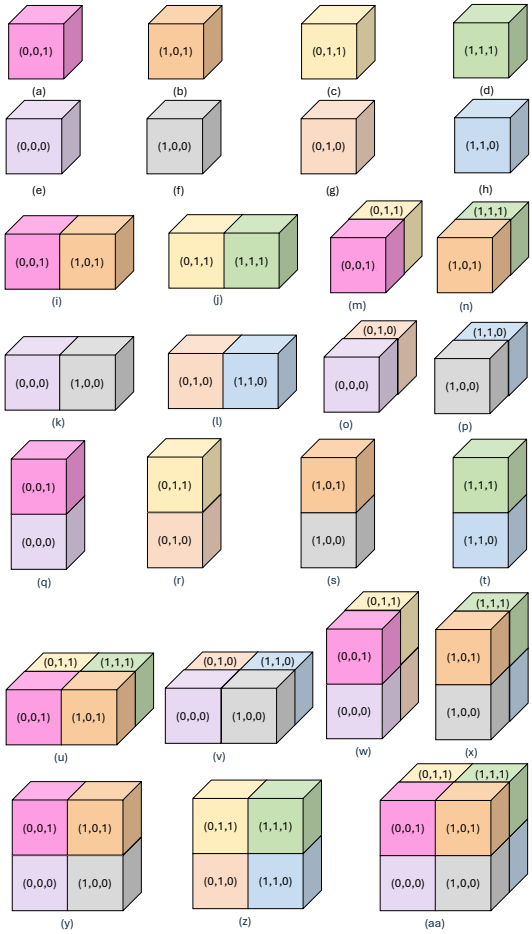


Figure 3: For an example scene with 2x2x2 basic cuboids, a total of 27 possible cuboids can be created. In general, given $i \times j \times k$ basic cuboids, the number of valid streaming cuboids can be obtained via Equation 1.

Figure 2 shows a simple example where a 3DGS scene is partitioned into 8 (2x2x2) basic cuboids. In this case, 27 possible streaming cuboids (labeled as `(a)`, `(b)`, ..., `(z)`, and `(aa)` in Figure 3) can be constructed from the 8 basic cuboids. To simplify the process of checking whether a cuboid is within the viewport (Section 3.3), we require that the constructed streaming cuboid be a regular cuboid. For example, a combination of basic cuboids $(0, 0, 0)$, $(1, 0, 0)$, $(0, 0, 1)$, and $(1, 0, 1)$ is valid. On the other hand, a combination of the following three basic cuboids $(0, 0, 0)$, $(1, 0, 0)$, and $(0, 0, 1)$ is invalid.

To formulate an optimization problem to determine the best spatial partitioning of the scene, we use a binary vector $\overrightarrow{x}$ to represent the solution of selected streaming cuboids. Given a 2x2x2 example shown in Figure 2, $\overrightarrow{x}$ would be a binary vector with a length of 27 that shows the presence of each streaming cuboid in the solution. For ease of presentation, we use the label of the basic cuboids along with a subscript to indicate the binary value. For example, in $[a_1,b_0,c_1,d_0,...,l_0,m_1,n_0...z_0,aa_0]$, a_1 represents streaming cuboid a is present in the solution, b_0 indicates streaming cuboid b is not present in the solution, and the solution only includes 3 streaming cuboids, i.e., a, c, and m.

Position	a	b	c	d	e	f	g	h	i	j	k	l	m	n	o	p	q	r	s	t	u	v	w	x	y	z	aa
(0,0,0)	0	0	0	0	1	0	0	0	0	0	1	0	0	0	1	0	1	0	0	0	0	1	1	0	1	0	1
(0,1,0)	0	0	0	0	0	0	1	0	0	0	0	1	0	0	1	0	0	1	0	0	0	1	1	0	0	1	1
(0,0,1)	1	0	0	0	0	0	0	0	1	0	0	0	0	1	0	0	0	1	0	0	0	1	0	1	0	1	0	1
(0,1,1)	0	0	1	0	0	0	0	0	0	1	0	0	1	0	0	0	0	1	0	0	1	0	1	0	0	0	1	1
(1,0,0)	0	0	0	0	0	1	0	0	0	0	1	0	0	0	0	1	0	0	0	1	0	0	1	0	1	0	1	1
(1,1,0)	0	0	0	0	0	0	0	1	0	0	0	1	0	0	0	1	0	0	0	1	0	1	0	1	0	1	0	1
(1,0,1)	0	1	0	0	0	0	0	0	1	0	0	0	0	1	0	0	0	0	1	0	1	0	0	1	1	0	1
(1,1,1)	0	0	0	1	0	0	0	0	0	1	0	0	0	1	0	0	0	0	0	1	1	0	0	1	0	1	1

Figure 4: This table shows an example binary matrix A when a scene is partitioned into 2x2x2 basic cuboids. This matrix encodes how each of the 27 possible valid streaming cuboids includes different sets of these 8 basic cuboids.

A valid solution $\overrightarrow{x}$ should include streaming cuboids that cover all the basic cuboids while ensuring no basic cuboids are covered more than once. For example, $[...,o_1,p_1,...,u_1,...]$ is a valid solution including a streaming cuboid u constructed with 2x2 basic cuboids, two streaming cuboids constructed with 1x2 basic cuboids. On the other hand, $[...,m_1,...,o_1,p_1,u_1,...]$ is invalid, since basic cuboids $(0, 0, 1)$ and $(0, 1, 1)$ are covered by two different streaming cuboids (m and u). $[...,m_1,...,o_1,p_1,...]$ is also invalid, since basic cuboids $(1, 0, 1)$ and $(1, 1, 1)$ are not included in any of the three selected streaming cuboids (i.e., m, o, and p).

We also build a vector $\overrightarrow{C_{num}}$ to represent the number of Gaussians inside each streaming cuboid. It has the same length as binary vector $\overrightarrow{x}$.

Our optimization objective aims to minimize a cost function given a distribution of views. This distribution of views can be obtained from a dataset of prior users' views of the same scene. For each view v, a set of basic cuboids is needed to render the view. We thus use d_v, a binary vector, to represent all possible streaming cuboids that cover the basic cuboids required by view v. In the simple 2x2x2 basic cuboids example, given a viewport where two basic cuboids $(0, 0, 1)$ and $(1, 0, 1)$ are visible, then d_v is $[a_1, b_1, ..., i_1, ..., m_1, n_1, ..., q_1, s_1, ..., u_1, ..., w_1, x_1, y_1,..., aa_1]$. Given a valid solution $\overrightarrow{x}$, the total number of Gaussians inside streaming cuboids selected in $\overrightarrow{x}$ is:

$$(\overrightarrow{x})^\top \mathrm{diag}(d_v)\overrightarrow{C_{num}}. \tag{2}$$

However, the actual number of Gaussians observed by view v is typically smaller than the total in all the selected streaming cuboids. Thus, if the size of a streaming cuboid is too large, it will include Gaussians outside the viewport (i.e., invisible), resulting in downloading bandwidth waste. Therefore, we use C_v to represent the actual number of Gaussians visible to v, and calculate the ratio $\overrightarrow{R_{waste}}$ between the total number of Gaussians inside all selected streaming cuboids and the actual number of visible Gaussians:

$$R_{waste}^{(v)} = \frac{p_v \cdot \overrightarrow{x}^\top \mathrm{diag}(d_v)\overrightarrow{C_{num}}}{C_v}, \tag{3}$$

where p_v represents the viewing probability of v, and $\sum_{p_{v\in \mathbf{v}}} = 1$.

The smaller R_{waste}, the fewer wasted Gaussians. However, if we simply minimize R_{waste}, the optimal solution will be all the basic cuboids, meaning the largest number of streaming cuboids. With the increasing amount of streaming cuboids, the amount of time needed for checking if each streaming cuboid is within the user's viewport also increases, i.e., the "cuboid visibility test" against the view frustum. For a valid solution $\overrightarrow{x}$, the total times for visibility test C_d is calculated as:

$$C_d = \overrightarrow{C_{num'}}^{\top} \cdot \overrightarrow{x}, \tag{4}$$

where $\overrightarrow{C_{num'}}$ is a binary version of the original $\overrightarrow{C_{num}}$ by keeping all the zero elements and replacing all the non-zero elements with 1, which can filter the empty cuboids in the solution. The larger the total times for the visibility test, the longer time we need to wait before downloading the required cuboids. Therefore, we need to minimize the total visibility test times, while in the meantime, maintaining the smallest number of wasted Gaussians. Here we use a new vector $\overrightarrow{x_{base}}$ to represent a solution of $\overrightarrow{x}$ when all the basic cuboids are selected. Then the maximum visibility test times are:

$$C_{dmax} = \overrightarrow{C_{num'}}^{\top} \cdot \overrightarrow{x_{base}}. \tag{5}$$

Similar to R_{waste}, we introduce R_{vtest} to represent the ratio between the visibility test times of the current solution and the maximum visibility test time:

$$R_{vtest} = \frac{C_d}{C_{dmax}}. \tag{6}$$

Finally, we build a binary matrix A to encode the constraints of the optimization. Each row in A represents how a specific basic cuboid is included in the possible set of streaming cuboids, and each column encodes which basic cuboids are included in a specific streaming cuboid. For the simple 2x2x2 basic cuboids example, we show the content of matrix A (an 8x27 matrix) in Figure 4. For instance, the value at row $(1, 0, 0)$ and column s is 1, since streaming cuboid s includes basic cuboid $(1, 0, 0)$.

Our ILP formulation for this problem is defined as follows:

$$\text{minimize: } R_{vtest} + \sum_{v\in \mathbf{v}} R^{(v)}_{waste}$$
$$\text{subject to: } A\overrightarrow{x} = \overrightarrow{\mathbf{1}}.$$

Here, $\overrightarrow{\mathbf{1}}$ is a vector of 1's with the same size as the binary solution vector of streaming cuboids $\overrightarrow{x}$. The solution vector $\overrightarrow{x}$ determines a set of streaming cuboids for downloading that can lead to a minimized total number of wasted Gaussians and detection time of all the streaming cuboids.

3.2 Importance Sorting in Each Streaming Cuboid

In addition to spatial partitioning, *SGSS* also leverages features of modern HTTP protocols, i.e., HTTP/2 [29] and HTTP/3 [30], for improving scene loading quality for the users. These features include stream multiplexing where multiple requests/responses share the same transport-layer connection. When combined with responses that can be "incrementally" processed (e.g., chunks of a JPEG image or progressive JPEG), it can reduce the user-perceived webpage loading time. To fully take advantage of these features, *SGSS* presorts Gaussians in each streaming cuboid based on an **importance score**. That is, during scene streaming, requests/responses for multiple streaming cuboids within the viewport are multiplexed in the same connection. This ensures that each visible streaming cuboid contributes to the initial rendered view. The most important Gaussians in each streaming cuboid in the viewport are streamed and rendered first, enhancing both the initial viewport quality and the speed of quality improvement.

Importance score. Unlike traditional point cloud where points are typically rendered as spheres or cubes of the same size, in 3DGS, each Gaussian can have different scale and opacity attributes. The size of the Gaussian "splat" on the rendered view can vary based on its scale, and its contribution to the final pixel color can be different depending on its opacity. A Gaussian with a very small scale and low opacity (i.e., nearly transparent) will be barely visible compared to a Gaussian with a larger scale and full opacity. This means by streaming and processing Gaussians that have a greater impact on the rendering first, both the initial quality and the speed of quality improvement can be enhanced. This motivated us to pre-sort Gaussians within each streaming cuboid based on an importance score calculated based on the scale and opacity of a Gaussian. We define the **importance score** (`IS`) as

$$IS^{(i)} = o^{(i)} \cdot \left(\frac{s_x^{(i)} \cdot s_y^{(i)} \cdot s_z^{(i)}}{S_{\text{max_}n}} \right)^{\beta}, \tag{7}$$

where $o^{(i)}$ represents the opacity of the i^{th} 3D Gaussian, and $s_x^{(i)}, s_y^{(i)}, s_z^{(i)}$ represent its scale. $S_{\text{max_}n}$ represents the n^{th} percentile scale product value (i.e., $s_x \cdot s_y \cdot s_z$) of all 3D Gaussians in the scene. $n = 90$ and $\beta = 0.1$. This importance score was inspired by the global significance score proposed by Fan et al. [10]. We show in Section 5.2 that pre-sorting Gaussians based on the defined importance score can effectively improve view quality during the initial scene loading.

3.3 *SGSS* Streaming Algorithm

During streaming, the client first obtains information about all the streaming cuboids for the 3DGS scene that is available at the server in a `.json` file. *SGSS* supports view-adaptive streaming of 3DGS scenes by first performing cuboid visibility tests to determine a list of visible streaming cuboids for downloading. It designs a novel scheme for assigning priority to the streaming cuboids based on their Gaussian density, contribution to the viewport, and distance from the camera origin. It also fully supports incremental cuboid downloading and processing, allowing downloaded cuboids to be rendered promptly. A simplified version of the *SGSS* streaming algorithm is shown in Algorithm 1.

Cuboid Visibility Test. Viewport-aware streaming requires the streaming client to check if a streaming cuboid is inside the user's viewport, i.e., if the cuboid intersects with the view frustum. Using simple collision detection algorithms for AABB or OBB (Oriented Bounding Boxes) detection [31] is not accurate enough and can lead to false positives, resulting in unnecessary data downloading. Instead, we use the Separating Axis Theorem [32] to detect whether a streaming cuboid collides with the current user's view frustum. Since the view frustum's near plane is usually set to a very small

Algorithm 1 SGSS Streaming Algorithm

```
Initialize downloading manager
if the viewport has changed then
    ▷ Obtain a list of visible cuboids for the current view
    cuboid_list ← VISIBILITY_TEST(all_cuboids, viewport)
    ▷ Calculate priority weight
    for cuboid in cuboid_list do
        calculate priority weight P based on Equations 9 and 10
    ▷ Streaming of cuboids
    for cuboid in cuboid_list do
        if cuboid is not yet fully downloaded then
            range ← DOWNLOADING_MANAGER(cuboid)
            if cuboid is not being downloaded then
                request cuboid with priority and range
            else if priority ≠ current_priority then
                update request cuboid with new priority/range
    for cuboids that are downloading but no longer visible do
        pause the downloading
```

value – much smaller than the shape of a basic cuboid, we simplify the view frustum into a pyramid shape. For each streaming cuboid, we first convert all eight vertices' coordinates from world coordinates to camera coordinates, using the following equation

$$[x_c, y_c, z_c] = (R_c)^T * ([x_w, y_w, z_w] - [c_x, c_y, c_z]), \tag{8}$$

where $[x_w,y_w,z_w]$ represents a vertex's position in the world coordinate system, R_c is the camera's rotation matrix, and $[c_x,c_y,c_z]$ is the camera's position in the world coordinate system.

We then project the cuboid to three planes, i.e., xy, xz, and yz planes of the camera's coordinate system, respectively, resulting in three polygons. We also project the view pyramid to the same three planes, resulting in two triangles and one rectangle. To decide if a cuboid is inside the view pyramid, we check if two polygons intersect using point-in-polygon detection and edge intersection detection. If the projection of the cuboid intersects with the projection of the view pyramid on all three planes, the cuboid is visible.

Prioritized Streaming. *SGSS* uses modern HTTP protocols for downloading all visible streaming cuboids in HTTP/2/3 streams. With HTTP/2/3 stream multiplexing, the downloading of streaming cuboids share the same transport-layer connection. While this allows important Gaussians in all streaming cuboids within the viewport frustum to be downloaded and displayed, it can inevitably lead to inefficient use of network bandwidth during the initial starting phase of the streaming. For example, not all visible streaming cuboids contribute equally to the overall rendering quality. Some visible streaming cuboids may be either not completely visible in the viewport or far away from the viewport. Among these streaming cuboids, some with a high density of Gaussians may use a high percentage of available network bandwidth while contributing little to the increase of visual quality.

To address the above challenge, we define a priority weight for each streaming cuboid given a viewport. We use this priority weight with the Extensible Prioritization Scheme for HTTP [33]. This weight considers both the density of Gaussians inside and the cuboid's contribution to the rendered viewport. The priority weight PW of a streaming cuboid c given a viewport v is calculated as:

$$PW_v^{(c)} = \frac{\rho^{(c)} * V_v^{(c)}}{z_v^{(c)} + \varepsilon}. \tag{9}$$

Here, $\rho^{(c)}$ represents the density of Gaussian ellipsoids of inside streaming cuboid c, $V_v^{(c)}$ is the visibility factor, which is the calculated as the ratio of pixels in the rendered viewport v that are occupied by streaming cuboid c, and $z_v^{(c)}$ represents the z-distance between the center of streaming cuboid c and viewport v. We set a very small value ε to avoid the distance to be 0. Since PW can range from thousands to values smaller than 1, we apply a logarithmic function to compress the range to approximately -10 to 10. The final priority weight P for a streaming cuboid is

$$P_v^{(c)} = log_{10}(PW_v^{(c)}). \tag{10}$$

We then map the priority P to the priority values supported by the `Fetch Priority API` [34]. For example,

$$\texttt{fetchpriority} = \begin{cases} \text{high} & P \geq 1 \\ \text{low} & P < 1 \end{cases}$$

In addition, due to the high computational cost of pixel-level visibility factor calculations, we set the `fetchpriority` to *low* for cuboids with a density lower than 10, regardless of their visibility factor and distance.

Supporting Incremental Cuboids Downloading/Processing. Gaussians received in each HTTP/2/3 stream are incrementally parsed, loaded to the GPU memory, and rendered. These streams, however, may not be aware of each Gaussian's memory boundary during their transport over the connection. That is, while information on each Gaussian is saved in a certain order, e.g., position, scale, color, and rotation (as adopted by *SGSS*), the actual data delivery may break the per-Gaussian boundary, leading to incomplete Gaussian information being downloaded and incrementally parsed. For example, a data chunk might contain only Y bytes of a Gaussian of X bytes, where $Y < X$. When a new data chunk for a different cuboid arrives, its position and attributes could be mistaken for the missing bytes from the previous Gaussian. This can result in misaligned information of Gaussians being sent to the WebGL vertex shader [35], rendering incorrect views. To address this issue, we design a **breakpoint handler** that keeps track of the exact byte offset where data for each cuboid was left incomplete. When a new data chunk for the same cuboid arrives, the handler checks how many bytes were missing from the last incomplete data chunk of this cuboid and fills in the missed value. It then calculates how many Gaussians are in the remaining bytes of the current data chunk and allocates memory for them. With the breakpoint handler, *SGSS* fully supports incremental cuboid downloading and processing while ensuring correct view rendering.

Stream Pause and Resumption. When the viewport changes, the visible streaming cuboids and the visibility factor of each streaming cuboid may change as well. Therefore, we use a downloading manager to record how many bytes have been downloaded for each streaming cuboid. For the cuboids to become invisible, we abort the current HTTP stream, ceasing transmission of the invisible cuboid. If the cuboid later becomes visible again, we will send a new HTTP range request with its `Range` header field set to bytes

recorded by the downloading manager to resume the downloading. For streaming cuboids with priority changes, we will first abort the current downloading and then send a new request with updated priority and `Range` field in the header.

4 Implementation

We implemented *SGSS* with JavaScript and WebGL2 within a fork of the `splat` [13] repository. In addition to the sorting worker designed in `splat`, we implemented two additional web workers to support our viewport-adaptive streaming[1]. A **visibility and priority** web worker is used for testing the visibility of streaming cuboids and calculating the priority of each visible streaming cuboid. It sends the information about visible streaming cuboids to the **streaming** web worker, which sends requests to the server and receives the Gaussians in HTTP responses. We create a simple HTTP server using `H2O` [36], which supports HTTP/2/3 protocols.

Each pre-trained 3DGS scene is pre-processed offline by a Python script. Given an existing distribution of user views for a scene, it generates spatially-partitioned cuboids based on ILP results. For solving the ILP, we used Gurobi Optimization [37]. For each streaming cuboid, we pre-sort the Gaussians according to their importance score ranking. For each Gaussian, we store its information with its original float32 type in the following order: position (12 bytes), scale (12 bytes), SH coefficients, opacity (4 bytes), and rotation (16 bytes). We then store all the streaming cuboids in the HTTP server. We also store the spatial information of each streaming cuboid, including the maximum and minimum spatial coordinates and the density of Gaussian in a `.json` file, which will be shared with the client at the beginning of the streaming session. We have implemented WebGL vertex shaders that can support rendering Gaussians with spherical harmonics of varying degrees (SH0, SH1, SH2, and SH3). These SH coefficients are converted to RGB color inside the vertex shader. We only present SH0 results in the paper due to space.

5 Performance Evaluation

5.1 Setup

3DGS scenes. We use 12 pre-trained 3DGS scenes from the original 3DGS paper [7] in our experiments. We did not include the `counter` scene as this scene is the smallest among all, and the full scene can be downloaded in a very short amount of time. The 12 scenes are from the MipNeRF360 dataset [11], the Tanks&Temples dataset [38], and the Deep Blending dataset [39]. These datasets contain a variety of scene settings, including indoor, outdoor, and natural environments.

Baseline approaches. We use six baseline approaches in our experiments, summarized in Table 1. We first compare *SGSS* with three state-of-the-art third-party web-viewers implemented based on WebGL. **FD** refers to `gsplat` [14] that waits for full scene downloading before view rendering. **DWO** refers to `Gaussian-Splat3D` [15], which pre-sorts 3D Gaussians based on their distance to the origin $(0, 0, 0)$ of the world coordinate system. It supports 3DGS scene streaming, but visible content is expanded from the world origin as Gaussians are downloaded. **IM** refers to `splat` [13], which pre-sorts all 3D Gaussians based on their scale and opacity values. 3D Gaussians with large scale and opacity values are streamed first. None of the above baseline approaches are view-adaptive.

[1]The repository of this project is at https://github.com/symmru/SGSS.

Table 1: Baseline approaches used in the evaluation.

FD [14]	**F**ull scene **D**ownloading before rendering
DWO [15]	Rendering while streaming; Gaussians are pre-sorted based on **D**istance from **W**orld **O**rigin
IM [13]	Rendering while streaming; Gaussians are pre-sorted based on an **I**mportance **M**etric
ViVo [28]	Bandwidth-efficient point cloud streaming through visibility-based point density adjustment
Ours-`{w/o OSP}`	*SGSS* but without optimized spatial partitioning, i.e., with basic cuboids only
Ours-`{w/o IS}`	*SGSS* but without importance sorting

We also compare our method with state-of-the-art viewport-adaptive volumetric scene streaming method, **ViVo** [28]. To support bandwidth-efficient streaming, ViVo streams fixed-size cells (a concept very similar to basic cuboids in our design) and adjusts the point density of each cell based on its visibility. To evaluate the effectiveness of our optimized spatial partitioning (OSP), we design a baseline approach **Ours**-`{w/o OSP}` where only basic cuboids are used for scene streaming. We also compare with **Ours**-`{w/o IS}`, where Gaussians inside the streaming cuboids are not sorted.

Two sets of viewing traces. For our experiments, we generate two sets of user viewing traces derived from the camera poses of images in the raw dataset of the 12 3DGS scenes. The first set of viewing traces simulates an ***orbital shot***, emulating the user behavior over 80 seconds. During this period, the user navigates through 40 camera poses from the scene's training dataset, one every 2 seconds. Between two adjacent camera poses, we uniformly interpolate 119 frames, enabling smooth 60 FPS rendering. These 40 camera poses allow the user to explore the full scene. With this viewing trace, we aim to demonstrate how visual quality evolves over time as more Gaussians are downloaded and the user's viewport moves. We also design a ***dolly shot*** for evaluation of total downloaded data. This shot only navigates a portion of the scene. To do so, we use the first two training camera poses for the scene and uniformly interpolate 1,199 frames between them. The user's viewport moves slowly and smoothly from one camera pose to the other in 20 seconds.

For each experiment, we replay the generated viewing trace and record the rendering result in WebGL when the viewport is located at a training camera pose. We use the original training images as ground truth for calculating the visual quality of rendered views.

Evaluation metrics. For visual quality evaluation, we use three metrics: peak signal-to-noise ratio (**PSNR**), structural similarity (**SSIM**) [40], and perceptual video quality metric **VMAF** [41, 42] for comparing visual quality of views rendered to the user and ground truth camera-recorded images. We also record the **total downloaded data** amount (e.g., in MBytes) to show the effectiveness of our approach in reducing wasted network transmission.

Network bandwidth settings. To evaluate how our proposed *SGSS* approach and baseline approaches perform under different available network bandwidth, we choose three different bandwidth settings in our experiments: 100 Mbit/s, 150 Mbit/s, and 200 Mbit/s. We use the `tc` [43] tool to emulate different available bandwidth between the server and the 3DGS viewing client and set the maximum client-server network latency to 20ms round-trip. For ***orbital shot***

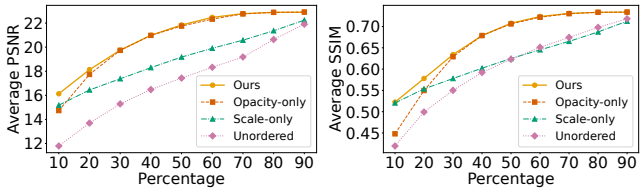


Figure 5: Comparison of different pre-sorting criteria.

traces, we evaluate the visual quality of all four methods under three bandwidth settings. For ***dolly shot traces***, focusing on the bandwidth savings, we only evaluate the performance under 200 Mbit/s. This ensures that all visible content in the scenes can be fully downloaded within 20 seconds of the viewing traces.

5.2 Pre-Sorting Criteria Results

To show the effectiveness of our importance sorting, we compare it with three other sorting criteria: `scale-only`, `opacity-only`, and `unordered`. `scale-only` ranks Gaussians solely based on their scales, `opacity-only` ranks Gaussians based on their opacity only, while `unordered` preserves the original order of Gaussians as they are trained. We evaluate 12 distinct 3DGS scenes. For each criterion, we pre-sort all Gaussians in the full `.ply` file according to the respective criterion. We then save the first 10%, 20%, ..., 90% of the sorted Gaussians and use them for rendering two different views in WebGL. These two views are also the same as the first two camera poses used to generate ***orbital shot traces***, which can be used to show the visual quality of initial views during streaming. We calculate the average visual quality results across scenes for each percentage, using the original camera images as the ground truth.

Figure 5 shows the average PSNR and SSIM values from 12 scenes. Our pre-sorting criteria demonstrate both higher initial visual quality and faster improvement in both metrics and significantly outperform non-sorting methods, demonstrating the necessity of this step. By incorporating both opacity and scale features in our sorting criteria, we achieve better results compared to approaches that consider these features independently. These findings indicate that the visual quality will increase the fastest with our pre-sorting criteria.

5.3 Spatial Partitioning Results

Table 2 shows the optimized spatial partitioning results from integer linear programming (ILP). For large scenes, we partition the 3DGS scenes into more basic cuboids. `Truck` is a scene with both small AABB size and file size, we therefore partition it into 10x10x3 basic cuboids in total. We are unable to find an existing dataset of users exploring these 12 3DGS scenes in 6-DoF. So we use the camera poses from training images of the original datasets as distribution of views, with each camera pose sharing the same view probability to set up the ILP. For each scene, the number of camera poses varies from 125 to 311. In Table 2, *Non-Empty* basic cuboids are cuboids with at least one 3D Gaussian. However, there exist many non-empty cuboids with only a few hundreds of Gaussians or even only one Gaussian. *Low-Density* represents the cuboids with a Gaussian density (as described in Equation 9) of less than 1. As discussed in Section 3.1, each of these cuboids takes the same time for the view frustum "visibility test". Our formulated optimization problem aims to reduce the number of selected streaming cuboids. Table 2 indicates that the ILP effectively works for our purpose. We can reduce the number of selected streaming cuboids to roughly 4% to 40% of non-empty basic cuboids.

Table 2: Spatial partitioning results.

Scene	# of Basic Cuboids			# of Solved/Selected	
	Total	Non-Empty	Low-Density	Streaming Cuboids	Low-Density
bicycle	2000	243	134	32	5
bonsai	1000	243	87	42	0
drjohnson	1000	285	79	112	3
flowers	1000	144	75	19	3
garden	2000	269	71	32	1
kitchen	2000	267	98	16	2
playroom	1000	431	94	139	2
room	1000	109	47	23	1
stump	2000	329	221	55	14
train	2000	237	177	9	4
treehill	2000	186	148	22	6
truck	300	108	52	16	2

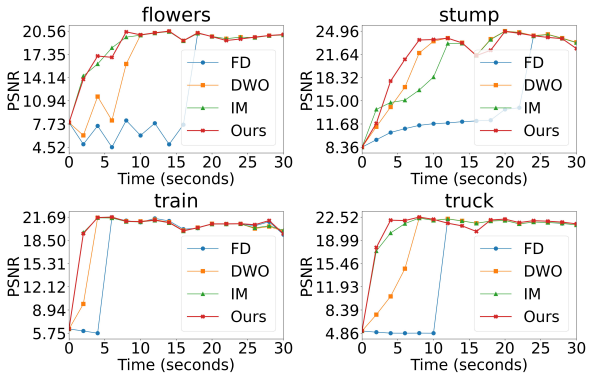
5.4 *SGSS* vs. Non-Viewport-Adaptive

We first compare *SGSS* with three state-of-the-art 3D Gaussian web-viewers, namely, **FD**, **DWO**, and **IM**.

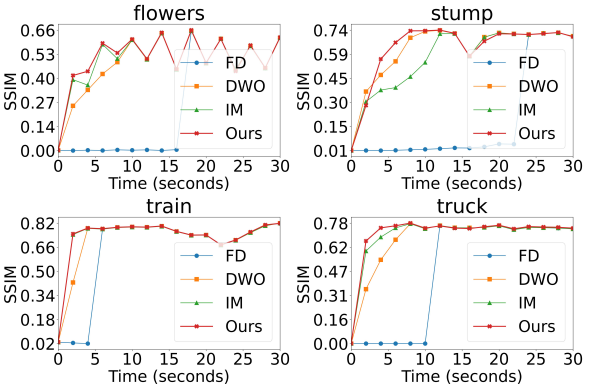
Visual quality results. None of the three baseline approaches support viewport-adaptive streaming. Even with our smallest bandwidth setting, they can finish full scene downloading within the first 30 seconds. We thus only show the results in the first 30 seconds. Benefiting from spatial partitioning, *SGSS* only downloads the streaming cuboids inside the viewport during the first 30 seconds. The total downloaded bytes are thus barely affected by the bandwidth change. Therefore, we present the comparison results under the smallest bandwidth setting, 100 Mbit/s.

Figure 6 shows the PSNR and SSIM results of three baseline approaches and *SGSS*. Compared to **FD** and **DWO**, our method achieves significantly better initial visual quality. **FD** takes a long time to download the scene. For large scenes like `stump` and `flowers`, it takes more than 20 seconds to achieve a reasonable visual quality. **DWO** performs better when the initial viewport looks at the scene's world origin. However, when the initial viewport is not looking at the world origin in the first 2 seconds, **DWO** leaves a huge blank in the viewport. Compared to **DWO**, our method downloads 3D Gaussians inside the user's viewport first, yielding better visual quality, shown in Figure 7. Besides, the density of Gaussians around world origin also affects the performance of **DWO**. In `truck`, the density of Gaussians close to the world origin is very high, therefore, it takes a long time for **DWO** to download these Gaussians, leaving the part close to viewport edges blank.

SGSS also yields better or similar initial visual quality results compared to **IM**. This is because both methods stream the 3D Gaussians with higher opacity and scale first. Different from our method, **IM** pre-sorts Gaussians over the full scenes, while we sort the Gaussians inside each streaming cuboid. If 3D Gaussians with higher importance metrics are concentrated in the background of the initial viewport, **IM** can potentially perform better than us in terms of the objective quality metrics in the first few seconds. This is because *SGSS* prioritizes streaming of cuboids closer to the viewport. However, if the Gaussians with higher importance are distributed uniformly around the scene, such as `stump`, the initial quality and



(a) PSNR (dB)↑ comparisons



(b) SSIM↑ comparisons

Figure 6: Visual quality comparison: *SGSS* vs. non-viewport-adaptive approaches under 100 Mbit/s and *orbital shot traces*.



(a) View rendered with the DWO approach. PSNR = 10.71 dB

(b) View rendered with our SGSS approach. PSNR = 30.66 dB

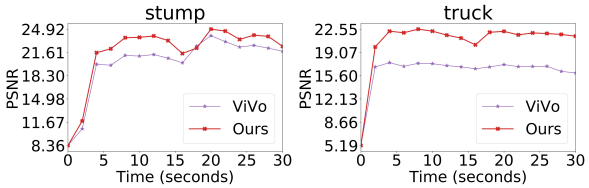
Figure 7: Side-by-side comparison: rendering result of drjohnson at the 6th second under 100 Mbit/s bandwidth.

the growth rate of visual quality will be severely affected. Compared to **IM**, our streaming method only focuses on the Gaussians inside the viewport, limiting the bandwidth wasted due to transmissions of Gaussians outside the viewport with higher significance scores.

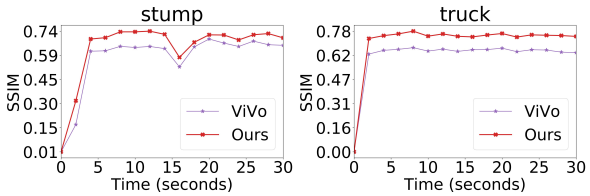
Bandwidth savings results. Table 3 shows the total bandwidth consumption in 20 seconds under 200 Mbit/s bandwidth based on the ***dolly shot traces***. With limited navigation of the viewport, only part of the scene can be viewed in 20 seconds. **FD**, **DWO**, and **IM** all download full scenes. However, our method only downloads the Gaussians in visible streaming cuboids. Notably, our method can save up to 71.47% bandwidth while streaming the drjohnson scene. Sometimes 3D Gaussians may be concentrated in one streaming cuboid, such as the table in the garden scene, and have a higher chance of being viewed, we have to download them completely.

Table 3: Total downloaded data under 200 Mbit/s and *dolly shot traces*.

Scene	DWO/IM/FD	*SGSS* (Ours)
bicycle	343.4 MB	**223.8 MB (-34.82%)**
bonsai	69.7 MB	**36.2 MB (-48.06%)**
drjohnson	190.7 MB	**54.4 MB (-71.47%)**
flowers	203.6 MB	**162.5 MB (-20.19%)**
garden	326.7 MB	**241.6 MB (-26.05%)**
kitchen	103.7 MB	**90.1 MB (-13.11%)**
playroom	142.6 MB	**59.2 MB (-58.49%)**
room	89.2 MB	**77.1 MB (-13.57%)**
stump	277.9 MB	**152.4 MB (-45.16%)**
train	57.5 MB	**53.3 MB (-7.30%)**
treehill	211.9 MB	**179.0 MB (-15.53%)**
truck	142.3 MB	**83.6 MB (-41.25%)**



(a) PSNR (dB)↑ comparison with *ViVo*



(b) SSIM↑ comparison with with *ViVo*

Figure 8: Visual quality comparison: *SGSS* vs. *ViVo* under 200 Mbit/s and *orbital shot traces*.

Therefore, the amount of saved bandwidth may vary. Nevertheless, on average, *SGSS* can achieve 32.9% network bandwidth savings.

5.5 *SGSS* vs. *ViVo*

Figure 8 shows that the visual quality of *SGSS* is significantly better than *ViVo*. This is because *ViVo* decreases the density of the cells occluded by others and far away from the view origin. However, due to the special geometry features of a 3DGS scene, Gaussians are not uniformly distributed across the cell. This results in the "fake occlusion" problem, where a cell is occluded by other cells while the Gaussians inside are not. As a result, due to density control for further-away cells used in *ViVo*, not all Gaussians required for view rendering may be downloaded. While this allows *ViVo* to use less network bandwidth compared to *SGSS*, the visual quality of *ViVo* suffers significantly. In contrast, benefiting from optimized spatial-partitioning, *SGSS* will download all the streaming cuboids which are visible in the viewport and decrease the wasted bandwidth. Figure 9 shows an example comparison. In the highlight of figures, due to the "fake occlusion" issue, required Gaussians in the background are not completely downloaded.

5.6 *SGSS* vs. *SGSS-w/o OSP* vs. *SGSS-w/o IS*

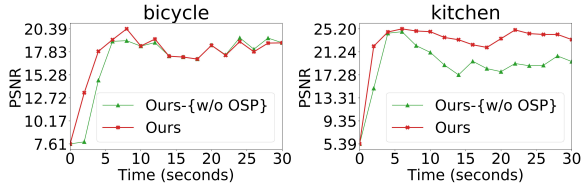
To show how *SGSS* can benefit from optimized spatial partitioning (OSP) and importance sorting (IS), we conduct experiments streaming basic cuboids only (i.e., *w/o OSP*) and streaming solved cuboids



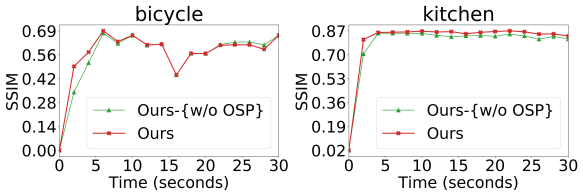
(a) View rendered with the ViVo approach. PSNR = 17.48 dB

(b) View rendered with our SGSS approach. PSNR = 22.21 dB

Figure 9: Side-by-side comparison: rendering result of garden at the 22th second under 200 Mbit/s bandwidth.

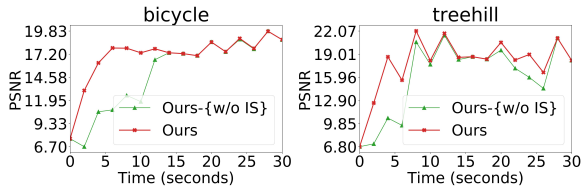


(a) PSNR (dB)↑ comparison with *SGSS-w/o OSP*

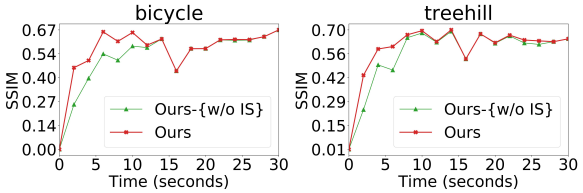


(b) SSIM↑ comparison with *SGSS-w/o OSP*

Figure 10: Visual quality comparison: *SGSS* vs. *SGSS-w/o OSP* under 200 Mbit/s and *orbital shot traces*.



(a) PSNR (dB)↑ comparison with *SGSS-w/o IS*



(b) SSIM↑ comparison with *SGSS-w/o IS*

Figure 11: Visual quality comparison: *SGSS* vs. *SGSS-w/o IS* under 100 Mbit/s and *orbital shot traces*.

without sorting Gaussians inside (i.e., *w/o IS*), both with their respective priority weights. The visual quality results are shown in Figures 10 and 11. For comparison with *SGSS-w/o OSP*, since basic cuboids with low density can benefit more from higher bandwidth, we show results under 200 Mbit/s bandwidth setting. Figure 10 shows that *SGSS* with OSP can achieve a more rapid increase at the beginning. Since OSP will merge some basic cuboids, more spatial cuboids will be assigned high priority, which promotes rapid growth in visual quality. Figure 11 shows that with importance sorting, *SGSS* allows the visual quality to improve faster than *SGSS-w/o IS*. This is

Table 4: VMAF results for all scenes across all baseline approaches under 100 Mbit/s bandwidth and *orbital shot traces*.

Scene	FD	DWO	IM	ViVo	Ours-{w/o IS}	Ours-{w/o OSP}	Ours
bicycle	27.46	84.14	88.19	53.29	91.47	92.08	**94.33**
bonsai	82.22	92.55	94.63	50.75	92.79	94.93	**96.04**
drjohnson	54.68	85.60	94.41	77.68	93.15	92.45	**94.63**
flowers	52.61	86.66	90.16	56.21	89.31	89.30	**91.86**
garden	24.58	80.86	86.87	60.16	87.58	83.63	**89.82**
kitchen	74.57	91.70	94.06	67.73	93.71	90.95	**95.34**
playroom	64.08	88.77	**95.15**	89.41	86.70	86.08	89.35
room	76.59	92.04	**95.20**	66.34	93.45	92.60	94.13
stump	39.61	90.13	77.79	75.09	90.91	90.85	**91.84**
train	83.91	92.72	94.62	79.74	94.78	93.83	**96.43**
treehill	47.60	81.72	91.76	72.18	86.52	91.77	**92.54**
truck	65.17	87.81	91.59	58.99	88.05	91.33	**94.47**

because Gaussians within a same streaming cuboid that contribute more to the rendering results are streamed and displayed earlier. We can also see that the visual qualities of the three methods are nearly the same after 10 seconds. This is because the camera in the ***orbital shot*** dataset moves slowly, causing the number of newly visible cuboids and the total Gaussians to be downloaded to remain similar, leading to almost the same visual quality.

5.7 Perceptual Video Quality

We report the perceptual visual quality results of all scenes under 100 Mbits/s bandwidth and ***orbital shot traces***, using the VMAF metric in Table 4. Since we do not have raw camera video footage of these scenes, we use a video rendered using all Gaussians (i.e., without streaming) in each scene as reference for computing the VMAF scores. *SGSS* consistently achieves high VMAF scores compared to baseline approaches. It performs the best in 10 out of 12 scenes, demonstrating the effectiveness of *SGSS*. As discussed earlier, when Gaussians with higher importance are concentrated in the initial viewport, **IM** produces results similar to those of *SGSS*. Moreover, since **IM** is a non-viewport-adaptive approach, it can achieve good visual quality performance once the entire scene is fully downloaded, as no additional Gaussians need to be downloaded for the later viewports. In contrast, *SGSS* may need to download new visible streaming cuboids. This has caused **IM** to perform slightly better than *SGSS* in VMAF results in two scenes.

6 Conclusion

3D Gaussian Splatting has the potential to transform real-world 3D scene representations. In this paper, we made a first effort to optimize the streaming transmission of 3D Gaussians to improve initial viewing quality while saving network bandwidth. We designed *SGSS* for view-adaptive streaming of optimized spatial partitions of the full 3DGS scene. Additionally, we implemented a pre-sorting scheme to quickly enhance initial visual quality. Furthermore, we developed an efficient streaming strategy that is compatible with modern HTTP protocols. Experiment results show that our approach is effective in achieving savings in network transmission without impacting the quality of views.

Acknowledgments

We appreciate constructive comments from anonymous referees. This work is partially supported by NSF under grants CNS-2200042 and CNS-2200048.

References

[1] Rafael Pagés, Konstantinos Amplianitis, David Monaghan, Jan Ondřej, and Aljosa Smolić. Affordable content creation for free-viewpoint video and vr/ar applications. *Journal of Visual Communication and Image Representation*, 53:192–201, 2018.

[2] Iro Armeni, Sasha Sax, Amir R Zamir, and Silvio Savarese. Joint 2d-3d-semantic data for indoor scene understanding. *arXiv preprint arXiv:1702.01105*, 2017.

[3] Branislav Jenco. Virtual lidar error models in point cloud compression. Master's thesis, 2022.

[4] Mufeng Zhu, Yuan-Chun Sun, Na Li, Jin Zhou, Songqing Chen, Cheng-Hsin Hsu, and Yao Liu. Dynamic 6-dof volumetric video generation: Software toolkit and dataset. In *2024 IEEE 26th International Workshop on Multimedia Signal Processing (MMSP)*, pages 1–6. IEEE, 2024.

[5] Raphael Sulzer, Renaud Marlet, Bruno Vallet, and Loic Landrieu. A survey and benchmark of automatic surface reconstruction from point clouds. *arXiv preprint arXiv:2301.13656*, 2023.

[6] Ben Mildenhall, Pratul P. Srinivasan, Matthew Tancik, Jonathan T. Barron, Ravi Ramamoorthi, and Ren Ng. Nerf: Representing scenes as neural radiance fields for view synthesis. In *ECCV*, 2020.

[7] Bernhard Kerbl, Georgios Kopanas, Thomas Leimkühler, and George Drettakis. 3d gaussian splatting for real-time radiance field rendering. *ACM Transactions on Graphics*, 42(4):1–14, 2023.

[8] Anpei Chen, Zexiang Xu, Andreas Geiger, Jingyi Yu, and Hao Su. Tensorf: Tensorial radiance fields. In *European Conference on Computer Vision*, pages 333–350. Springer, 2022.

[9] Sara Fridovich-Keil, Giacomo Meanti, Frederik Rahbæk Warburg, Benjamin Recht, and Angjoo Kanazawa. K-planes: Explicit radiance fields in space, time, and appearance. In *Proceedings of the IEEE/CVF Conference on Computer Vision and Pattern Recognition*, pages 12479–12488, 2023.

[10] Zhiwen Fan, Kevin Wang, Kairun Wen, Zehao Zhu, Dejia Xu, and Zhangyang Wang. Lightgaussian: Unbounded 3d gaussian compression with 15x reduction and 200+ fps. *arXiv preprint arXiv:2311.17245*, 2023.

[11] Jonathan T Barron, Ben Mildenhall, Dor Verbin, Pratul P Srinivasan, and Peter Hedman. Mip-nerf 360: Unbounded anti-aliased neural radiance fields. In *Proceedings of the IEEE/CVF Conference on Computer Vision and Pattern Recognition*, pages 5470–5479, 2022.

[12] Maximum Wilder-Smith, Vaishakh Patil, and Marco Hutter. Radiance fields for robotic teleoperation, 2024.

[13] Github: antimatter15/splat. https://github.com/antimatter15/splat.

[14] gsplat — 3D Gaussian Splatting WebGL viewer . https://gsplat.tech/.

[15] Github: mkkellogg/GaussianSplats3D. https://github.com/mkkellogg/GaussianSplats3D.

[16] Defining the Core Web Vitals metrics thresholds. https://web.dev/articles/defining-core-web-vitals-thresholds.

[17] Vickie Ye and Angjoo Kanazawa. Mathematical supplement for the `gsplat` library. *arXiv preprint arXiv:2312.02121*, 2023.

[18] Matthias Zwicker, Hanspeter Pfister, Jeroen Van Baar, and Markus Gross. Ewa splatting. *IEEE Transactions on Visualization and Computer Graphics*, 8(3):223–238, 2002.

[19] Sebastian Schwarz, Marius Preda, Vittorio Baroncini, Madhukar Budagavi, Pablo Cesar, Philip A Chou, Robert A Cohen, Maja Krivokuća, Sébastien Lasserre, Zhu Li, et al. Emerging mpeg standards for point cloud compression. *IEEE Journal on Emerging and Selected Topics in Circuits and Systems*, 9(1):133–148, 2018.

[20] D Graziosi, O Nakagami, S Kuma, A Zaghetto, T Suzuki, and A Tabatabai. An overview of ongoing point cloud compression standardization activities: Video-based (v-pcc) and geometry-based (g-pcc). *APSIPA Transactions on Signal and Information Processing*, 9:e13, 2020.

[21] Draco 3D. https://google.github.io/draco/.

[22] Ricardo L De Queiroz and Philip A Chou. Compression of 3d point clouds using a region-adaptive hierarchical transform. *IEEE Transactions on Image Processing*, 25(8):3947–3956, 2016.

[23] Yuan-Chun Sun, Yuang Shi, Wei Tsang Ooi, Chun-Ying Huang, and Cheng-Hsin Hsu. Multi-frame bitrate allocation of dynamic 3d gaussian splatting streaming over dynamic networks. In *Proceedings of the 2024 SIGCOMM Workshop on Emerging Multimedia Systems*, pages 1–7, 2024.

[24] Joo Chan Lee, Daniel Rho, Xiangyu Sun, Jong Hwan Ko, and Eunbyung Park. Compact 3d gaussian representation for radiance field. *arXiv preprint arXiv:2311.13681*, 2023.

[25] Simon Niedermayr, Josef Stumpfegger, and Rüdiger Westermann. Compressed 3d gaussian splatting for accelerated novel view synthesis. *arXiv preprint arXiv:2401.02436*, 2023.

[26] Towaki Takikawa, Alex Evans, Jonathan Tremblay, Thomas Müller, Morgan McGuire, Alec Jacobson, and Sanja Fidler. Variable bitrate neural fields. In *ACM SIGGRAPH 2022 Conference Proceedings*, pages 1–9, 2022.

[27] Lingzhi Li, Zhen Shen, Zhongshu Wang, Li Shen, and Liefeng Bo. Compressing volumetric radiance fields to 1 mb. In *Proceedings of the IEEE/CVF Conference on Computer Vision and Pattern Recognition*, pages 4222–4231, 2023.

[28] Bo Han, Yu Liu, and Feng Qian. Vivo: Visibility-aware mobile volumetric video streaming. In *Proceedings of the 26th annual international conference on mobile computing and networking*, pages 1–13, 2020.

[29] Martin Thomson and Cory Benfield. HTTP/2. RFC 9113, June 2022.

[30] Mike Bishop. HTTP/3. RFC 9114, June 2022.

[31] Christer Ericson. *Real-time collision detection*. Crc Press, 2004.

[32] Johnny Huynh. Separating axis theorem for oriented bounding boxes. *URL: jkh. me/files/tutorials/Separating% 20Axis% 20Theorem% 20for% 20Oriented% 20Bounding% 20Boxes. pdf*, 2009.

[33] Kazuho Oku and Lucas Pardue. Extensible Prioritization Scheme for HTTP. RFC 9218, June 2022.

[34] Optimize resource loading with the Fetch Priority API . https://web.dev/articles/fetch-priority.

[35] WebGL Shaders and GLSL . https://webglfundamentals.org/webgl/lessons/webgl-shaders-and-glsl.html.

[36] H2O - an optimized HTTP server with support for HTTP/1.x, HTTP/2 and HTTP/3. https://github.com/h2o/h2o/?tab=readme-ov-file/.

[37] Gurobi Optimization. https://www.gurobi.com/.

[38] Arno Knapitsch, Jaesik Park, Qian-Yi Zhou, and Vladlen Koltun. Tanks and temples: Benchmarking large-scale scene reconstruction. *ACM Transactions on Graphics (ToG)*, 36(4):1–13, 2017.

[39] Peter Hedman, Julien Philip, True Price, Jan-Michael Frahm, George Drettakis, and Gabriel Brostow. Deep blending for free-viewpoint image-based rendering. *ACM Transactions on Graphics (ToG)*, 37(6):1–15, 2018.

[40] Zhou Wang, Alan C Bovik, Hamid R Sheikh, and Eero P Simoncelli. Image quality assessment: from error visibility to structural similarity. *IEEE transactions on image processing*, 13(4):600–612, 2004.

[41] VMAF: The Journey Continues. https://medium.com/netflix-techblog/vmaf-the-journey-continues-44b51ee9ed12.

[42] VMAF - Video Multi-Method Assessment Fusion. https://github.com/Netflix/vmaf.

[43] tc(8) — Linux manual page . https://man7.org/linux/man-pages/man8/tc.8.html.

LL-Sparse: Low-Latency 6-DoF Field of View Prediction

Jérémy Ouellette
Concordia University
Montréal, Canada
o_jerem@live.concordia.ca

Abdelhak Bentaleb
Concordia University
Montréal, Canada
abdelhak.bentaleb@concordia.ca

ABSTRACT

Field of view (FoV) prediction is crucial for optimizing 6-DoF dynamic point cloud-based volumetric video (PCV) streaming. By accurately predicting which tiles fall within the viewer's region of interest, FoV prediction enables adaptive bitrate (ABR) algorithms to allocate higher bitrates to likely viewed tiles while assigning lower bitrates to less critical areas. This improves bandwidth efficiency and enhances the quality of experience (QoE) by aligning bitrate allocation with the viewer's focus. However, current 6-DoF salience-aware FoV prediction models face challenges related to high latency, computational costs, and a lack of complex datasets with detailed FoV traces, hindering the development of more effective real-time predictors. To address these challenges, we propose the `LL-Sparse` family, a suite of three solutions for direct tile salience score prediction: `LL-Adapter`, an extension of HMD-trajectory-based (HTB) models, such as GRUs, tailored for tile scoring; `LL-PointNet`, which integrates a GRU with PointNet to enhance salience-aware prediction; and `LL-SparseConv`, a scalable variant of `LL-PointNet` that employs sparse convolution in place of PointNet, serving as a proof of concept. These models strike a balance between practical performance and theoretical advancements in tile salience prediction. Furthermore, we introduce the `MazeLab` dataset, a novel, large-scale dynamic point cloud dataset that mimics real-world PCV scenarios to effectively benchmark FoV prediction models. Experimental results highlight the `LL-Sparse` family's exceptional scalability, reduced latency, and enhanced accuracy, establishing it as a promising solution for efficient real-time volumetric media applications.

CCS CONCEPTS

• **Information systems** → **Multimedia streaming**.

KEYWORDS

Low Latency Point Cloud Streaming, FoV Prediction, Sparse CNN.

ACM Reference Format:
Jérémy Ouellette and Abdelhak Bentaleb. 2025. `LL-Sparse`: Low-Latency 6-DoF Field of View Prediction. In *The 16th ACM Multimedia Systems Conference (MMSys '25), March 31–April 4, 2025, Stellenbosch, South Africa.* ACM, New York, NY, USA, 11 pages. https://doi.org/10.1145/3712676.3714438

Permission to make digital or hard copies of all or part of this work for personal or classroom use is granted without fee provided that copies are not made or distributed for profit or commercial advantage and that copies bear this notice and the full citation on the first page. Copyrights for components of this work owned by others than the author(s) must be honored. Abstracting with credit is permitted. To copy otherwise, or republish, to post on servers or to redistribute to lists, requires prior specific permission and/or a fee. Request permissions from permissions@acm.org.
MMSys '25, March 31–April 4, 2025, Stellenbosch, South Africa
© 2025 Copyright held by the owner/author(s). Publication rights licensed to ACM.
ACM ISBN 979-8-4007-1467-2/25/03
https://doi.org/10.1145/3712676.3714438

1 INTRODUCTION

Multimedia streaming has evolved dramatically, progressing from traditional 2D videos to immersive 360-degree content, and now to advanced volumetric videos [28]. Volumetric videos are often created using point clouds [10, 28, 37] captured by depth cameras, such as those in Microsoft Azure Kinects or the latest iPhones, producing detailed 3D maps of scenes [10]. This technology allows users to experience playback from any angle and move freely in six degrees of freedom (6-DoF) along the *X*, *Y*, and *Z* axes, as well as rotate in *pitch*, *yaw*, and *roll*, using a Head-Mounted Display (HMD). While it offers unprecedented immersion, it also comes with significant challenges. Achieving smooth, realistic surfaces requires vast numbers of points, resulting in substantial data sizes that can overwhelm network capacity. This tension between immersive quality and network limitations presents a critical obstacle to the advancement of multimedia streaming technology, demanding innovative solutions for efficient data management and transmission.

To address these challenges, tile-based HTTP adaptive streaming [2, 28] has emerged as an efficient solution to deliver point cloud-based volumetric videos (PCVs). This solution involves the spatial segmentation of video chunks into distinct, non-overlapping tiles. The system then uses HTTP adaptive streaming (HAS) [2] to intelligently pre-fetch only those tiles that fall within the viewer's predicted field of view (FoV) [29] at high quality compared to those outside the FoV. Therefore, tile-based streaming effectively reduces bandwidth usage while enhancing the viewer's QoE. Despite its significance, no existing 6-DoF FoV prediction model currently achieves a balance of *low latency*, *high scalability*, and *acceptable accuracy*. A key challenge hindering the development of such models is the absence of comprehensive public datasets that capture both complex dynamic point cloud environments and corresponding FoV HMD traces. These datasets are crucial for training and evaluating neural network-based FoV prediction models, and their scarcity has slowed progress in enhancing PCV streaming systems.

This paper introduces the `LL-Sparse` family of models, a set of efficient FoV prediction solutions tailored for tile-based PCV streaming. The family includes three models: `LL-Adapter`, which extends HMD-trajectory-based (HTB) models like GRUs [25] for direct tile salience prediction; `LL-PointNet`, a hybrid model combining GRU-based temporal processing with PointNet [22] for salience-aware (SA) prediction; and `LL-SparseConv`, a variant of `LL-PointNet` in which PointNet is replaced by a Sparse Convolutional Neural Network (SCNN) [15] to improve scalability. Together, these models aim to deliver accurate real-time FoV predictions, improving both streaming performance and viewer's QoE. Additionally, we present `MazeLab`, a dynamic point cloud dataset that serves as a challenging benchmark for evaluating and comparing FoV prediction models. The main contributions of this paper are three-fold:

(1) **LL-Sparse Family.** We introduce three novel variants of models for accurate FoV prediction. (*i*) LL-Adapter: a model that extends HTB-based models, such as GRUs, to directly predict salience scores for tiles. (*ii*) LL-PointNet: a model that combines a GRU with PointNet to enhance SA prediction, effectively integrating spatial awareness with sequential modeling. (*iii*) LL-SparseConv: a variant of LL-PointNet that replaces PointNet with a SCNN to improve scalability and efficiency, presented as a proof of concept.
(2) **Novel MazeLab Dataset.** We introduce the MazeLab dataset, a large-scale dynamic point cloud dataset that includes complex environments and detailed FoV traces, providing a robust benchmark for evaluating FoV prediction models.
(3) **Model Benchmarking.** We conduct a comprehensive benchmarking of the LL-Sparse models, where the LL-Adapter is applied to the most popular HTB state-of-the-art FoV prediction models to assess their scalability, latency, and accuracy.

This paper is organized as follows: we provide background on FoV prediction and SCNN techniques in Section 2. We then introduce the MazeLab dataset in Section 3 and present the problem formulation in Section 4. The design of the LL-Sparse family is detailed in Section 5, with the evaluation and results discussed in Section 6. Finally, we conclude the paper in Section 7.

2 BACKGROUND AND RELATED WORK

2.1 FoV Prediction

FoV prediction, also known as viewport prediction, is essential for optimizing adaptive streaming pipelines in immersive video experiences. Whether dealing with 360° videos or volumetric videos, HMDs only render a portion of any given video frame at a time. This means that the bandwidth allocated to rendering parts of the video outside the FoV does not enhance the viewer's experience, as these areas are not visible. To address this inefficiency, it is crucial to allocate bitrates more effectively by prioritizing higher bitrates for the visible portions of the video. However, predicting which parts of the video will be visible is not an easy task. Fortunately, historical data, including past HMD positions, viewing angles, and frames, can be utilized to predict the FoV at the next time step. This prediction involves considering a specific timeframe, often referred to as the *prediction horizon*, which represents the duration over which the model anticipates changes in the viewport. The two main strategies for accomplishing this prediction task include HTB prediction [19, 25, 36] and SA prediction [8, 16, 30].

2.1.1 HTB Prediction. HTB prediction [19, 25, 36] involves anticipating the future position and orientation of a viewer's HMD or viewpoint by extrapolating from its historical trajectory often represented as a sequence of sets like this: {x, y, z, yaw (ψ), pitch (θ), roll (ϕ)}. One of the earliest approaches to FoV prediction in PCV streaming was based on linear regression [24]. Later, LSTM-based [19] and GRU-based [25] models were proposed. These models can learn temporal dependencies in the data, making them suitable for predicting the viewer's future FoV based on their past and present FoVs. However, these models often fail to handle abrupt or unpredictable changes, potentially leading to inaccurate predictions. Further, these models disregard the point cloud data of each frame, failing to exploit a critical source of information. Despite these limitations, HTB prediction remains valuable as it often requires minimal computation compared to SA prediction, making it fast while still bringing a fair accuracy [16].

2.1.2 SA Prediction. SA prediction [8, 16, 30] plays a major role in improving FoV prediction as it provides a mechanism to discern regions that capture the viewer's attention. Point clouds feature extraction [11] can be accomplished either directly through computations on their points [1, 22, 23] or by applying transformations such as voxelization [18, 31] first to benefit from grid-friendly neural network such as CNNs. On one side, direct processing entails strategies such as geometry-based heuristics, exemplified by Fast Point Feature Histograms (FPFH) [11], and models like PointNet [22] and PointNet++ [23]. While heuristics like FPFH offer speed, their expressiveness is limited. Conversely, models like PointNet and PointNet++ are designed for more powerful point processing but suffer from severe slowness and VRAM usage, heavily reducing their usefulness in PCV streaming that demands low latency. On the other side, traditional 3D CNNs [18, 31] encounter challenges in point cloud processing due to the voxelization process, which generates numerous empty voxels, leading to excessive computations. Recent advancements, particularly in sparse convolution libraries like NVIDIA's Minkowski Engine, have addressed these challenges by enabling the design of more efficient architectures such as MinkowskiNet42 [5] and FastPointTransformer [21]. These models have demonstrated a 2x improvement in latency over PointNet while achieving much higher accuracy. However, despite the recent promises of sparse convolution, to the best of our knowledge, we are the first to use this technology for FoV prediction.

2.2 Sparse Convolution

Understanding sparse convolution [15] begins with voxelization, a process that discretizes a point cloud into a structured 3D grid of small volumetric units, or voxels, representing spatial regions. While voxelization facilitates processing by structuring the data, higher resolutions lead to sparsity, with most voxels being empty. Traditional CNNs struggle with this sparsity, wasting computation on empty voxels and resulting in inefficiency, especially in PCVs where empty voxels dominate. Sparse convolution effectively addresses this challenge by focusing only on non-zero (active) voxels, avoiding unnecessary computations. Below, we outline the sparse convolution process [35] and the design choices for LL-SparseConv.

(a) Voxelization: The point cloud is voxelized into a discrete 3D grid, where each voxel aggregates the feature vectors of points within it, structuring the data for convolution.
(b) Sparse Tensor Generation: Sparse tensors, often based on the Coordinate list (COO) format [5], store only non-zero voxel coordinates and their features, optimizing memory usage by avoiding dense representations.
(c) Kernel Map Generation: The kernel map specifies the coordinates of active voxels, their output locations, and kernel offsets, efficiently guiding sparse convolution operations across the sparse tensor.
(d) Gather-GEMM-Scatter (GMaS) Pipeline: Sparse convolution engines employ the GMaS approach [35] for efficiency, consisting of the following operations: (*i*) Gather: Input features are

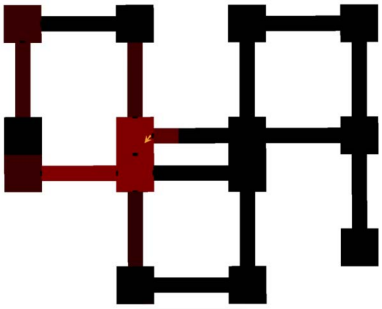


Figure 1: Bottom-up view of a MazeLab dataset sample. The participant is represented as an orange arrow, and the salient regions are highlighted in red.

fetched, buffered, and split into tiles for parallel processing; (*ii*) GEMM: Tiles undergo parallel matrix multiplications to maximize GPU performance; and (*iii*) Scatter: Results are aggregated and placed at their corresponding sparse tensor coordinates.

This pipeline reduces redundancy and optimizes GPU parallelism for scalable sparse convolution.

2.2.1 Sparse Convolution Libraries and Convolution Types. Four major sparse convolution libraries have been introduced: Minkowski Engine [5], SpConv [6], TorchSparse++ [27], and Minuet [35]. We selected SpConv for its high performance, versatility, and seamless integration with PyTorch. SpConv supports two convolution types: (*i*) Sparse Convolution, which activates kernel positions overlapping with non-zero voxels for broader coverage and higher feature extraction potential, and (*ii*) Submanifold Convolution, which restricts activation to kernels centered on active voxels, preserving sparsity and reducing computation. For our work, we chose Sparse Convolution due to its superior ability to extract features.

3 OUR DATASET (MAZELAB)

While there are many point cloud datasets available [13, 33, 34], only a few provide the necessary 6-DoF FoV traces, such as user location and gaze direction. Datasets like the Watching User Behavior dataset [12] and the 6DoF VR User Navigation dataset [26] (based on the 8iVFBv2 dataset [13]) were collected in simplistic environments characterized by a limited XYZ range and a small number of objects. These constraints present significant limitations when it comes to training and benchmarking FoV prediction models.

The existing datasets often fail to capture the complexities of real-world scenarios, where occlusion is more common and the environment is more varied. This lack of complexity means that models trained on these datasets may struggle to generalize to more challenging environments. Additionally, the limited XYZ range reduces the variability in the viewer's FoV, making prediction tasks easier but less representative of real-world conditions. Moreover, such datasets do not effectively showcase the benefits of tiling strategies, as most or all tiles can be visible simultaneously, reducing the need for accurate FoV prediction and bandwidth optimization.

To address these limitations, we created a novel dynamic point cloud dataset, referred to as MazeLab[1] [20], generated from a custom-built maze within Unreal Engine 5 (UE5) [7] specifically for 6-DoF FoV prediction. The maze features interconnected rooms, doors, hallways, bridges, and lights, covering a range of ~ 120x150x10 UE5 meters[2], mimicking real-world PCV scenarios. The MazeLab dataset, which contains more than half a trillion points saved across all participant-experiment pairs, is summarized in Table 1, and depicted in Figure 1. This environment promotes occlusion-aware predictions of participants' FoV in the maze, as structures like walls often obstruct visibility. Each data point includes eleven features: the XYZ coordinates, material RGB values, the RGB color and intensity of the nearest light source, and a flag indicating whether the object is rendered in the current frame. Due to its large size, only a limited subset of tiles is visible at any given moment, which contrasts with smaller datasets where most or all tiles may be visible simultaneously. This characteristic presents effective predictors with a valuable opportunity to enhance the QoE by intelligently directing rendering resources toward the most relevant areas.

Table 1: Summary of The MazeLab Dataset

Metric	Value	Description
Maze Size	120x150x10m	Dimensions in UE5 meters.
Experiments	15	90-second scenarios capturing diverse conditions.
Participants/Exp.	~14	Total of 209 participant-experiment pairs.
PLY Frame Rate	10 fps	900 frames per participant-experiment pair.
HMD Position Rate	60 Hz	5400 samples per participant-experiment pair.
Points/Frame	~3 million	XYZ, material RGB, nearest light RGBI, and render flag.

FoV traces were collected with the help of 15 participants. These participants sat on a chair with wheels and wore a Meta Quest 2 headset characterized by a FoV of 97° horizontally and 93° vertically [4]. The joystick on either of the two controllers could be pushed to move the player forward or in any other direction (forward, backward, and diagonals). The speed of movement was non-linear, with acceleration increasing as the joystick was pushed further. While the chair remained stationary, it could be rotated in place to achieve in-game rotations. This setup was necessary due to the large size of the maze, as no physical room would be large enough to accommodate real-world movements without the use of an omnidirectional treadmill.

Each participant took part in 15 experiments of 90 seconds each, where some were repeated multiple times with different starting points and paths. In all experiments, participants were instructed to look for visual cues such as material/lighting colors that stand out, geometry (closed/opened doors), blinking patterns, and object movements since they indicate a path to follow. In the event they reached the end of the path before the 90-second timer expired, they were told to continue following the path in reverse rather than remaining stationary. The experiments can be described as follows:

- **[4 variants]** All doors are open, allowing participants to explore the maze freely without any restrictions.
- **[4 variants]** Certain doors are kept closed, guiding participants along a single, predefined path through the maze.

[1] MazeLab dataset cloud drive link here.

[2] In UE5, the default unit of measurement is cm, where 1 UE5 unit equals 1 cm.

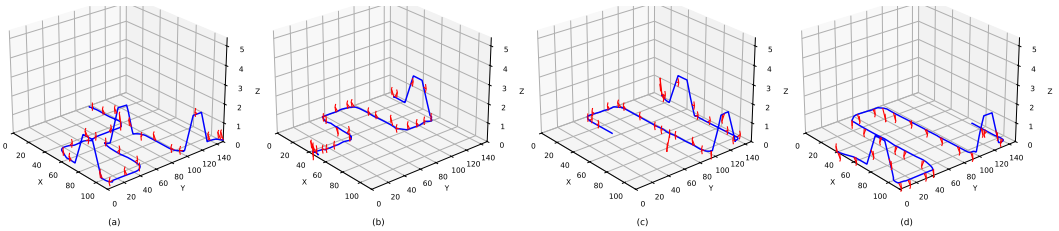


Figure 2: Paths of different participants in the maze during the first 45 seconds under varying experimental settings. Each image shows the participant's path in blue, with red arrows indicating the viewing direction at regular intervals.

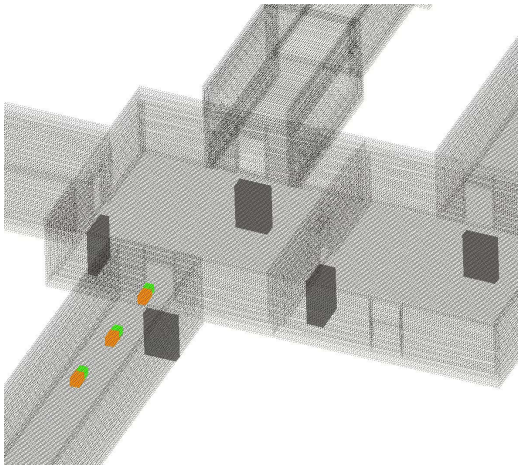


Figure 3: A participant moving through the point cloud maze, entering a room with all doors open.

- **[2 variants]** All doors are open, with a path marked by either red or blue lights while the remaining lights are uncolored.
- **[2 variants]** All doors are open, with a path marked by either red or blue lights while the rest of the maze is completely dark.
- **[1 variant]** Static PLY objects from the 8iVFBv2 dataset (*Loot*, *Longdress*, *Soldier*, and *Redandblack*) are hidden throughout the maze. Participants are encouraged to try to find all four of them.
- **[1 variant]** A fast-moving, rapidly flashing sphere that alternates between red and blue guides participants into the maze.
- **[1 variant]** Similar to the previous experiment, but the sphere leads to a closed room containing the 8iVFBv2 PLY objects.

During data collection, the in-game player's position (XYZ), HMD rotations (both Euler angles and quaternions), and the visibility state of each UE5 actor (*e.g.*, walls, doors) were recorded 60 times a second (total of 5400 times for a 90-second recording). From these logs, the paths and behaviors of participants can be accurately recreated, as depicted in Figure 2. The maze itself was recorded as a point cloud every 100 ms to save on space and processing resources. We determined that this was sufficient since dynamic objects, including the player, moved at a speed that did not require higher frame rates to effectively capture their movements. Figure 3 illustrates both the movement of a participant over time and the density of the point cloud. Three different instances of the participant are shown, spaced 24 frames apart. The orange rectangular prism represents the body, and the green prism represents the head, resembling the tip of an arrow to indicate movement direction.

4 PROBLEM FORMULATION

FoV prediction models aim to minimize the loss function that evaluates the alignment between predicted and ground truth visibility scores for each tile. In this work, we propose a custom Soft Intersection over Union (SIoU) loss function defined as follows:

$$\mathcal{L}(\mathrm{gt}, \mathrm{pd}) = 1 - \frac{\sum_{i=1}^{N} \min(\mathrm{gt}_i, \mathrm{pd}_i) + \epsilon}{\sum_{i=1}^{N} \mathrm{gt}_i + \sum_{i=1}^{N} \mathrm{pd}_i - \sum_{i=1}^{N} \min(\mathrm{gt}_i, \mathrm{pd}_i) + \epsilon},$$

where gt_i and pd_i are the ground truth and predicted visibility scores for tile i, and N is the total number of tiles. The constant ϵ ensures numerical stability. The loss approaches zero as the overlap between predicted and ground truth visibility scores increases, indicating better alignment. Conversely, a higher loss value occurs when the scores do not align well, with the maximum loss approaching one. A key feature of SIoU is its ability to penalize prediction spread. As the spread increases, the union (denominator) grows, raising the loss and encouraging predictions that closely align with the ground truth. To further enhance FoV prediction, models must also address the following key characteristics:

① **Feature Utilization**: Leverage point cloud geometry and features for better cue utilization.
② **Temporal Awareness**: Incorporate historical viewport behavior and environmental context.
③ **Fast Inference**: Ensure real-time performance with under 200 ms of glass-to-glass latency.
④ **Scalability**: Support large XYZ ranges and number of points.

The `LL-Sparse` family of models is designed to leverage these qualities, with each model offering unique advantages based on its specific trade-offs. `LL-Adapter` excels in temporal awareness ②, fast inference ③, and scalability ④. `LL-PointNet` builds upon this by adding feature utilization ① and scaling decently up to point clouds of 40K points, though at the cost of higher latency. `LL-SparseConv`, on the other hand, integrates all four characteristics making it the most theoretically well-rounded model.

5 DESIGN OF THE LL-SPARSE FAMILY

This section presents the design of the `LL-Sparse`[3] family of models. Additionally, we introduce a novel feature augmentation technique designed to enhance the salience analysis capabilities of `LL-PointNet` and `LL-SparseConv`.

5.1 LL-Adapter

This section describes how the `LL-Adapter` leverages FoV predictions from HTB models to assign a salience score to each point

[3]GitHub Repository: https://github.com/Thehappydeveloper/LL-Sparse

in the point cloud, and then groups these points into tiles, each with its salience score. The adapter is designed to be compatible with any HTB predictor that forecasts the next FoV in the form $\{x, y, z, q_w, q_x, q_y, q_z\}$, where the first three elements represent the HMD's position in 3D Cartesian coordinates, and the subsequent four elements represent its orientation as a quaternion.

5.1.1 Point Salience Scoring from FoV Prediction. To compute the salience score for each point in the point cloud, we consider three main factors: (*i*) whether the point is within the predicted FoV, (*ii*) how close it is to the center of the FoV, and (*iii*) how far it is from the predicted HMD position. Algorithm 1 offers an overview of this process and is described as follows:
(*i*) First, the quaternion output $\{q_w, q_x, q_y, q_z\}$ from the HTB model is converted into a direction vector using the participant's most recent position $\{x, y, z\}$, as detailed in Algorithm 2. This direction vector represents the predicted gaze of the participant and is used to construct a dynamic coordinate system for computing the vertical and horizontal angles between every point's position vector and the participant's gaze. The adaptive coordinate system defined by the direction vector ensures that the FoV stays aligned with the participant's gaze. In this system, the direction vector defines the x-axis, the z-axis is created by projecting the global z-axis onto the plane orthogonal to the direction vector, and the y-axis is derived as perpendicular to both the direction and z-axes. We then compute the vertical and horizontal angles for each point. The vertical angle is the angle between the direction vector and the plane where the y-axis is perpendicular. The horizontal angle is the angle between the direction vector and the plane where the z-axis is perpendicular. If the point lies within both the horizontal and vertical FoV boundaries, it is considered visible and assigned a score of 1; otherwise, it is assigned a score of 0.
(*ii*) Next, the proximity to the FoV center is determined by the dot product of each point's position vector and the direction vector. A higher dot product indicates the point is closer to the center of the FoV, whereas a lower value suggests it is closer to the edges.
(*iii*) Finally, the distance from each point to the predicted HMD position is calculated, normalized by the maximum distance, and scaled using a cosine function to smoothly reduce the score as the point moves farther from the FoV origin.

The final score for each point is the product of the three factors (i, ii, iii) computed above. This combined score ensures that points near the center of the FoV, visible within the FoV, and closer to the HMD position are assigned higher salience scores, while points on the periphery or farther away receive lower scores. The score is appended to each point as a new feature.

5.1.2 Generating Salience Scored Tiles. After computing salience scores for all points in the point cloud, `LL-Adapter` groups these points into salience-scored tiles based on spatial location using Algorithm 3. A tensor is initialized to match the tiling scheme, and each point is assigned to its corresponding tile, creating a structured representation of tiles containing points with salience scores. The final tile score is calculated as the maximum salience among its points, providing robustness against the number of points per tile (unlike summing) and effectively capturing concentrated attention in small areas (unlike averaging).

Once the tile scores are computed, they are first normalized using z-scores. To refine the score distribution, we apply a *softmax function* with a temperature of value $T = 2$ which reduces the sharpness of the score distribution, preventing extreme dominance by a few tiles with high scores. This adjustment spreads the scores more evenly across all tiles, ensuring that even lower-scoring tiles retain some weight in the final representation. The softmax function is defined as:

$$\sigma(x_i) = \frac{e^{x_i/T}}{\sum_j e^{x_j/T}}$$

where $T = 2$ is the temperature, x_i is the z-score of the i-th tile, and the sum in the denominator is taken over all tiles j.

Finally, the normalized scores are scaled by a factor of 1000. This scaling step allows the model to distribute 1000 salience points across all tiles, making the results easier to interpret and more meaningful for human understanding.

Algorithm 1 Check Point Visibility.

Require: P **(points)**, pos **(player position)**, dir **(direction vector)**, fov_h **(Horizontal FOV in °)**, fov_h **(Vertical FOV in °)**
Normalize $dir \leftarrow \frac{dir}{\|dir\|}$
Compute $y_{new}, z_{new} \leftarrow$ compute_axes(dir)
Compute $to_points \leftarrow P - pos$
Compute $dists \leftarrow \|to_points\|$
Normalize $norm_points \leftarrow \frac{to_points}{dists+1e-6}$
Compute $dot_prod \leftarrow norm_points \cdot dir$
Compute $ang_v \leftarrow$ compute_angle(P, dir, y_{new}, pos)
Compute $ang_h \leftarrow$ compute_angle(P, dir, z_{new}, pos)
Determine $visible_h \leftarrow ang_h \leq \frac{fov_h}{2}$
Determine $visible_v \leftarrow ang_v \leq \frac{fov_v}{2}$
Compute $visible \leftarrow visible_h \wedge visible_v$
Compute $dist_weights \leftarrow \cos\left(\frac{dists}{\max(dists)} \cdot \frac{\pi}{2}\right)$
Compute $scores \leftarrow visible \cdot dist_weights \cdot dot_prod$
return $|scores|$

Algorithm 2 Quaternion to Direction Vector.

Require: $q = (x, y, z, w)$ **(Quaternion components)**
Compute Rotation matrix R:
$$R \leftarrow \begin{bmatrix} 1-2(y^2+z^2) & 2(xy-zw) & 2(xz+yw) \\ 2(xy+zw) & 1-2(x^2+z^2) & 2(yz-xw) \\ 2(xz-yw) & 2(yz+xw) & 1-2(x^2+y^2) \end{bmatrix}$$
Define Forward vector:
$$f \leftarrow \begin{bmatrix} 1 \\ 0 \\ 0 \end{bmatrix}$$
Compute Direction vector $d \leftarrow R \cdot f$
return d **(Direction vector)**

Algorithm 3 Tile Point Cloud into Salience-Scored Grid.

Require: P **(points)**, min_xyz, max_xyz, $grid_shape$
Compute $tile_size \leftarrow \frac{max_xyz - min_xyz}{grid_shape}$
Compute $indices \leftarrow \left\lfloor \frac{P - min_xyz}{tile_size} \right\rfloor$
Initialize $tiles \leftarrow -\infty$ with shape $grid_shape$
For each point p in P:
  Compute $idx \leftarrow indices[p]$
  Update $tiles[idx] \leftarrow \max(tiles[idx], p_{salience})$
Replace any $-\infty$ in $tiles$ with 0 *# Handle empty tiles*
return $tiles$

5.2 LL-PointNet

This section outlines the implementation details of LL-PointNet, an extension of the LL-Adapter architecture. LL-PointNet improves point feature utilization, providing a more robust solution to 6-DoF FoV prediction.

5.3 PointNet Integration

LL-PointNet integrates the PointNet architecture into its pipeline, replacing the tiling step from the LL-Adapter (as described in Algorithm 3). This allows the model to process point clouds with an additional salience feature for each point, providing feedback on the past HMD FoV trajectory. By leveraging PointNet, LL-PointNet improves point feature utilization, which was previously limited in the LL-Adapter.

LL-PointNet uses a GRU with 128 neurons and 2 layers for the HTB prediction task, performing well across various prediction horizons, as shown in the evaluation section 6. To optimize performance, the trajectory history length is adjusted based on the prediction horizon. Experimentation found that providing a context 3x longer than the prediction horizon yields the best results. We fixed the number of samples at 15 and adjusted the vector time step so that the history sequence spans this longer time range. Since the GRU can learn non-linear relationships, the time distance between history samples does not need to match the prediction horizon exactly. Both LL-PointNet and LL-SparseConv can swap the HTB predictor during runtime, enabling independent training and insertion into the pipeline at any point.

An overview of the PointNet model configuration is provided in Table 2. The model outputs a tensor of 180 values, derived from the tiling scheme of [12 x 15 x 1] used in the evaluation section. These 180 values are reshaped to match the target tiling dimensions, and the tensor is then normalized using the same z-score normalization and softmax function as in the LL-Adapter.

Although PointNet was originally designed for fixed-sized point clouds, it is adapted in this implementation to handle variable-sized point clouds by upsampling or downsampling them to 40,000 points before processing. For more information on how PointNet works, we refer the reader to the original paper [22].

5.4 LL-SparseConv

LL-SparseConv, illustrated in Figure 4, is an advanced variant of LL-PointNet that offers significant scalability improvements in both VRAM usage and latency. These benefits, as demonstrated in the evaluation section 6, are achieved by replacing PointNet with a SCNN while maintaining an equivalent number of learnable parameters (~3.5 million). However, it is important to note that in its current form, LL-SparseConv is primarily intended to explore the theoretical limits of the field. Practical challenges remain, particularly in achieving stable model convergence, which future research must address.

Despite these challenges, LL-SparseConv strictly adheres to the four core design principles (①–④) established for robust FoV prediction models. The following two sections provide an in-depth exploration of the SCNN module and the Tiling Scheme Adapter (TSA) module, as shown in Figure 4.

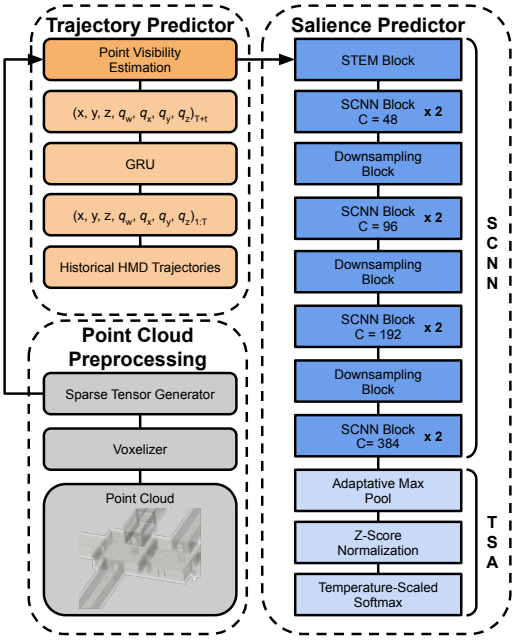


Figure 4: Architecture of LL-SparseConv.

5.4.1 SCNN Module. The SCNN module draws inspiration from the CNN modernization guidelines of ConvNeXt V1 [17] and V2 [32], which have demonstrated the ability to surpass transformer models in 2D computer vision tasks. The module, as illustrated in Figure 4, consists of three primary building blocks: (1) the stem block, (2) the SCNN blocks, and (3) the downsampling blocks.

(1) Stem Block: The SCNN begins with a *Stem Block* that downsamples the input point cloud using sparse 3D convolution with a kernel size and stride of four. Then it incorporates a *Global Response Normalization (GRN)* layer, to normalize channel activations by computing the L2 norm of each channel, then scales the input features based on their relative importance using learnable parameters. Notably, this enhances feature contrast and ensures that the model focuses on the most relevant features.

(2) SCNN Blocks: The network's core consists of *SCNN Blocks* arranged in groups of two, leading to a configuration of {2, 2, 2, 2}. Each block begins with a sparse 3D convolution using a kernel size of 3, which differs from the ConvNeXt implementation's kernel size of 7; this change helps conserve VRAM, reduce latency, and lower computational costs due to the higher resource demands of 3D convolutions. Following this, there is a sparse GRN layer that normalizes feature activations. A pointwise (1x1) convolution then expands the feature count by a factor of four, which is processed through a *Sparse GELU* activation function to introduce non-linearity. The block concludes with another pointwise convolution that reduces the feature size back to the original dimensions. To enhance learning, residual connections are integrated between the start and end of each block, preserving spatial information and stabilizing training.

(3) Downsampling Blocks: It reduces the spatial dimensions by half while doubling the number of feature channels, boosting the network's ability to capture larger receptive fields. Downsampling

Table 2: Layer Descriptions for the PointNet Forward Pass

Stage	Operation	Output Dimensions
Input Alignment	Apply initial transformation network (T-Net) to align input features, followed by matrix multiplication	Batch Size × 12 × 40,000
Initial Feature Extraction	Perform two 1D convolutions with batch normalization and ReLU activations	Batch Size × 64 × 40,000
Feature Alignment	Apply secondary transformation network (T-Net)to enhance feature alignment	Batch Size × 64 × 40,000
Advanced Feature Extraction	Execute three 1D convolutions with batch normalization and ReLU activations	Batch Size × 1024 × 40,000
Global Feature Aggregation	Perform max pooling across all points to obtain global feature vectors	Batch Size × 1024
Final Output Layers	Utilize two fully connected layers with ReLU activations to generate final output	Batch Size × 180

blocks are positioned after each SCNN block pairs, except for the last one. These blocks follow a channel progression of {48, 96, 192, 384}, enabling the network to learn increasingly complex representations.

5.4.2 TSA Module. The output shape of the SCNN module is influenced by the XYZ range of its input, as downsampling blocks reduce spatial dimensions by a constant factor of 2. This results in a variable output size, which is addressed by our Sparse Adaptive Max Pooling layer. This layer dynamically adjusts the tensor's spatial dimensions to match the tiling scheme, ensuring consistent output despite minor variations in input shape.

The adaptive pooling mechanism provides flexibility by adjusting kernel size and stride based on the input and desired output sizes, allowing the model to handle minor spatial shape differences. However, it is crucial to balance the number of downsampling layers in proportion to the input size. While this flexibility allows for the processing of a wide range of input shapes, excessively large inputs can demand too much VRAM during max pooling, while too many downsampling layers can overly reduce the spatial dimensions, making the output incompatible with the tiling scheme. Thus, a balanced approach is essential to efficiently use resources while ensuring compatibility with the tiling scheme.

After pooling, the sparse tensor output is converted into a dense tensor, and the 384 channels generated per tile through convolution are averaged to produce a single value per tile. The normalization process follows the same procedure as in `LL-PointNet` and `LL-Adapter`, ensuring consistency across the `LL-Sparse` family.

5.5 Point Cloud Feature Enrichment

To enhance temporal awareness in `LL-PointNet/SparseConv` without the computational cost of 4D convolution, we incorporate temporal cues as additional point features, by storing per-object visibility as a sequence of boolean values indicating whether each object is in the FoV. This allows us to track the duration an object stays within the FoV and compute a temporal salience score. The scoring function, Algorithm 4, increases rapidly for objects initially in the FoV and more slowly as they remain. Static objects have a maximum score of 30, while dynamic objects can reach 60. This is accomplished by removing the bias of 30 in the scoring function, ensuring that dynamic objects receive higher scores than static ones when in the FoV. The function is calibrated to reward continuous presence in the FoV, with 60% of the score achieved within 1 second, 90% by 2 seconds, and the full score at 3 seconds. After 3 seconds, the score decays with a forget rate of 14/15 over 0.5 seconds. This aggressive forget rate aligns with the observation that participants in the maze typically move forward and rarely backward, thus prioritizing reactiveness over conservativeness. If per-object sequences are not feasible, the approach can be adapted to per-tile information with a single scoring function.

Algorithm 4 Calculate Object Salience Score.

Require: vis_seq **(visibility sequence)**, $forget_rate$, $threshold$
Initialize $CTC \leftarrow 0$ **(Continuous True Count)**
Initialize $score \leftarrow 0$
Define polynomial coefficients (fine-tuned values):
- $a_3 \leftarrow 2.31481481 \times 10^{-6}$
- $a_2 \leftarrow -1.66666667 \times 10^{-3}$
- $a_1 \leftarrow 0.391666667$
- $a_0 \leftarrow 30$

for i, vis **in** $enumerate(vis_seq)$ **do**
  if vis = True **then**
    $CTC \leftarrow CTC + 1$
    $score \leftarrow a_3 \cdot CTC^3 + a_2 \cdot CTC^2 + a_1 \cdot CTC + a_0$
    $score \leftarrow \min(score, threshold)$
  else
    $CTC \leftarrow 0$
    if $i > 0$ **then**
      $score \leftarrow vis_seq[i-1] \cdot forget_rate$
    else
      $score \leftarrow 0$
    end if
  end if
  $vis_seq[i] \leftarrow score$
end for
return vis_seq

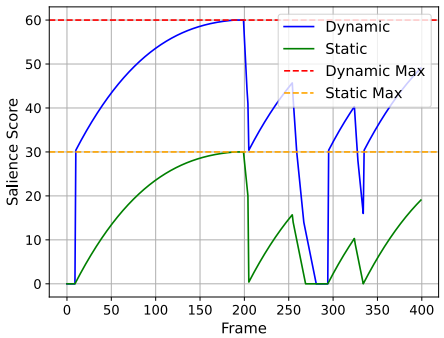


Figure 5: Effect of Algorithm 4 on Point Salience.

6 EVALUATION AND RESULTS

This section provides a detailed evaluation of our models on the `MazeLab` dataset. We benchmark the performance of six HTB predictors, each equipped with the `LL-Adapter`, and compare these results against the `LL-PointNet` and `LL-SparseConv` models.

6.1 HTB FoV Prediction Models

We begin by presenting an overview of the HTB models selected for benchmarking, chosen for their popularity and demonstrated effectiveness in recent years:

- **NAIVE**: This model predicts future states as identical to the last observed state, serving as a simple baseline for comparison.
- **LINEAR** [24]: A linear model is fitted for each feature, providing a straightforward method for capturing linear trends.

- **LSTM** [19]: Employs memory cells and gating mechanisms to capture long-term dependencies effectively while maintaining moderate computational demands.
- **GRU** [25]: An RNN with fewer gates than LSTM, capturing temporal dependencies with lower computational cost.
- **MAMBA** [9]: This approach simplifies Transformers by incorporating a Selective-State-Spaces (SSM) block, which reduces complexity and improves inference speed, achieving linear scaling with sequence length. Additionally, it features hardware-aware parallelism for enhanced efficiency.
- **JAMBA** [14]: A combination of the transformer architecture with MAMBA's SSM. This model provides global dependency capture with efficient processing to enhance performance.

6.2 Experimental Setup

6.2.1 Dataset. The training and evaluation utilized the `MazeLab` dataset, which encompasses 15 experiments and 15 participants. To balance robustness and efficiency, the dataset was partitioned into 92.4% (approximately 171K samples) for training and 7.6% (approximately 13K samples) for validation. Samples were carefully selected to ensure comprehensive coverage of experiments and participants. Unlike smaller datasets, which often require a larger proportion for validation to maintain representativeness, the size and diversity of the `MazeLab` dataset allowed strong coverage with a relatively smaller validation split. Each sample comprises two point clouds: one for inference and another for generating ground truth tile scores, with each point cloud containing roughly 3 million points. This configuration ensures both practical benchmarking and representativeness, with the validation process taking approximately 3 hours to complete for all models in the `LL-Sparse` family on an A100 GPU.

6.2.2 HTB Predictors. The `LL-Adapter` adapter is designed specifically for pre-trained HTB models. As a result, GRU, LSTM, MAMBA, and JAMBA models were trained without the `LL-Adapter`, focusing on predicting the next HMD position and orientation (represented as quaternions) rather than tile scores. Their performance during training and evaluation was measured using Mean Squared Error (MSE) [3], which quantifies the accuracy of predicted positions and angles. Once trained, these models were evaluated with the `LL-Adapter` using the SIoU metric, consistent with the approach used for other SA models. Although the LINEAR and NAIVE models require no training, they were evaluated in the same manner—first as standalone models and then with the adapter. All HTB models were provided with input sequences three times the length of the prediction horizon, consisting of 15 evenly spaced data points.

6.2.3 SA Predictors. All modes `LL-Adapter`, `LL-PointNet`, and `LL-SparseConv` utilized the same voxelization pipeline. The point cloud was voxelized with a voxel size of $3 \times 3 \times 3$, reducing the original set of approximately 3 million points to around 415,000 points. After voxelization, a tiling scheme was applied using a grid configuration of $(x = 12, y = 15, z = 1)$, where each tile spanned approximately 100 UE5 meters in the X and Y dimensions and extended fully along the Z-axis. The only variation among the models was in the handling of voxelized points. `LL-PointNet` required downsampling to 40,000 points via random sampling to meet its fixed point input requirement, whereas `LL-Adapter` and `LL-SparseConv` processed the full voxelized set, benefiting from their higher scalability. While `LL-Adapter` itself did not require training, it was evaluated in the same manner as `LL-PointNet` and `LL-SparseConv`, using the SIoU metric. The training configurations for these models (HTB and SA) are detailed in Table 3.

Table 3: Hyperparameter Settings for Model Training

Parameter	HTB	SA
Batch Size	256	5
Initial LR	0.001	0.001
Minimum LR	0.0001	0.0001
LR Scheduler	Cosine Annealing	Cosine Annealing
Optimizer	ADAM	ADAM
Epochs	25	2 (~115 hours)
Early Stopping	3 epochs	None
Hidden Size	128 (two layers)	N/A
Loss Function	MSE	SIoU

6.3 Standalone HTB Models Evaluation

The evaluation results for various HTB models, measured in terms of their MSE across prediction horizons ranging from 0.5 to 5 seconds, are shown in Figure 7. The GRU and LSTM models demonstrate similar performance trends, maintaining relatively low MSE values across all horizons. MAMBA and JAMBA also perform well but exhibit a more pronounced increase in error as the prediction horizon lengthens. In contrast, the NAIVE and LINEAR models consistently show the highest MSE values, highlighting their significantly lower performance. These models also experience a steep rise in error with increasing prediction horizons. Among the evaluated models, GRU and LSTM stand out as the most reliable for accurate FoV predictions, particularly over extended prediction horizons. Given the GRU's slightly lower latency compared to the LSTM, we integrated it into both `LL-PointNet` and `LL-SparseConv`. However, we acknowledge that either model could have been used interchangeably due to their virtually equivalent performance.

6.4 Tile Salience Score Prediction Benchmark

6.4.1 Results Overview. This section evaluates the SIoU loss and scalability of six HTB predictors equipped with the `LL-Adapter`, comparing their performance against the `LL-PointNet` and `LL-SparseConv` models. Due to the significant training time required for these models, the analysis focuses on a prediction horizon of 0.5 seconds, aligned with the benchmark configuration described earlier. Figure 6 summarizes the comparative performance of the `LL-Sparse` family across four key metrics: average validation loss (left), standard deviation of validation loss (middle left), latency (middle right), and VRAM usage (right). While the figure presents latency and VRAM usage as ratios relative to the number of points processed by each model, Table 4 provides the corresponding absolute values for a more detailed comparison.

The results demonstrate that `LL-Adapter` achieves an excellent balance between low loss, low latency, and low VRAM usage. Among the models, `LL-Adapter` delivers the best performance in

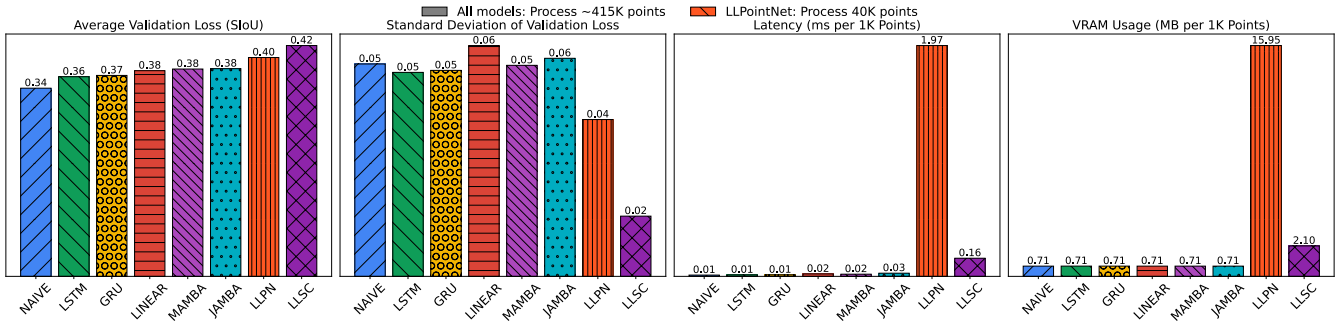


Figure 6: Comparison of loss, latency, and VRAM usage across different FoV prediction models with 0.5 second horizon.

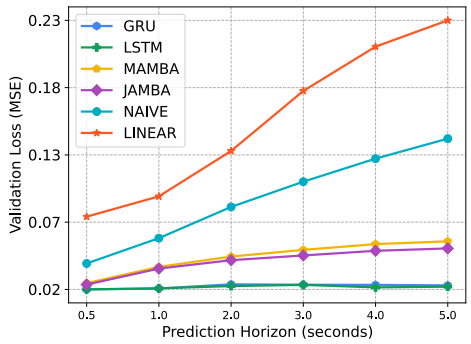


Figure 7: Validation loss at different prediction horizons.

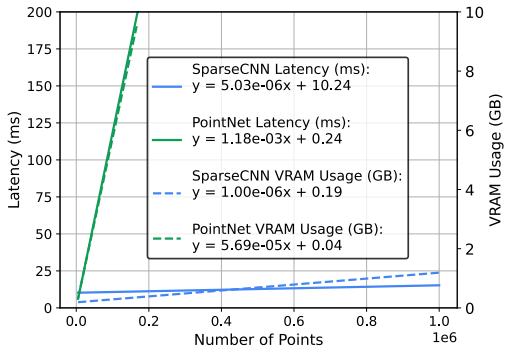


Figure 8: Latency and VRAM Usage: SCNN vs. PointNet.

Table 4: Latency and VRAM Usage for Different Models

Model	Latency (ms)	VRAM Usage (MB)
Naive	3.39	292.56
Linear	5.65	293.62
GRU	3.61	293.34
LSTM	3.99	294.55
MAMBA	5.15	292.47
JAMBA	6.89	292.93
LL-PointNet	67.13	590.23
LL-SparseConv	60.87	821.71

terms of loss, with the NAIVE model showing the lowest loss, followed closely by the LSTM and GRU. In contrast, LL-PointNet (LLPN) achieves a loss comparable to LL-Adapter but incurs significantly higher latency and VRAM usage. This increase is expected due to LL-PointNet's point-wise feature extraction, which LL-Adapter does not require. On the other hand, LL-SparseConv (LLSC) exhibits much lower latency and VRAM usage ratios compared to LL-PointNet, despite using a comparable number of learnable parameters. However, its convergence challenges are evident in its relatively higher loss, reflecting its current limitations. All models in the LL-Sparse family utilized less than 1 GB of VRAM and maintained latency below 75 ms on our A100 GPU, with LL-Adapter-based models performing significantly better in both metrics. This efficiency highlights their suitability for scenarios where computational resources and latency are tightly constrained, reinforcing their practical usability in demanding real-time applications.

6.4.2 Detailed Scalability Evaluation. To further illustrate the scaling properties of both LL-PointNet and LL-SparseConv, Figure 8 shows the latency and VRAM usage of their main feature extraction components as a function of input size. The figure is truncated at 200 ms to focus on low-latency predictions, with both models demonstrating linear scalability within this limit. As shown in the graph, the SCNN scales significantly better than PointNet, processing up to a million points with around 1GB of VRAM and under 25 ms of latency, while PointNet struggles to process even 200K points within the 200 ms threshold.

6.4.3 Caveats and Considerations. Model performance depends heavily on the specific context and the conditions of the task at hand. Despite the inclusion of key salience cues in the MazeLab dataset, such as open/closed doors, light paths, and a guiding sphere, the maze's repetitive structure and long corridors still provide a clear advantage to trajectory-based models. Therefore, we should be cautious with loss metrics alone, as they do not necessarily indicate which model is superior. Visualizing prediction behavior offers a more comprehensive understanding, as shown in the next section.

6.5 Prediction Behavior Analysis

In this part of the analysis, we evaluate the behavior of the LL-Adapter with the six HTB models and LL-PointNet, focusing on how they allocate salience scores to different areas of the maze. In its current form, LL-SparseConv assigns an equal score to all tiles, meaning that no differentiation is made between regions, and therefore, we will not analyze it further. We begin by presenting the target point cloud, followed by its representation as ground truth tile scores, and then the model predictions, enabling a direct comparison. This evaluation highlights each model's approach to interpreting visual attention, handling occlusions, and distributing scores across tiles.

6.5.1 Point Cloud Visualization. Figure 1 illustrates the processed target point cloud from a bottom-up perspective, emphasizing regions with higher salience scores (indicated in red) as calculated by Algorithm 4. Since the participant is hidden by the structure, we draw an orange arrow over the point cloud to indicate the participant's position and orientation for easier visualization. The target point cloud represents the point cloud at the specific prediction horizon where we aim to forecast the FoV. The salience score of each point, as defined by Algorithm 4, allows us to control how conservative the ground truth should be, particularly by adjusting the weighting of regions that were in the FoV previously but have not been recently visible. A clear correlation can be observed between the salience score (represented by redness) of a region and the participant's viewing direction. However, due to the behavior of Algorithm 4, exceptions can occur, such as residual salience in areas previously within the FoV, like the tile behind the participant, which remains red due to earlier exposure.

6.5.2 Ground Truth Representation. The next step involves converting the point cloud into tiles using Algorithm 3. The results, displayed in the first image of Figure 9, show a tiling scheme of [12, 15, 1], where the tile scores closely correspond to the salience of the underlying target point cloud. The participant's position aligns with the tile at $(x = 5, y = 6)$, which has the highest score, as Algorithm 3 assigns each tile the salience score of its most salient point. Tiles at $(x = 4, y = 0)$ and $(x = 5, y = 0)$ are also highly salient, as the participant had partial visibility of the back wall at the end of the corridor, which is more salient than the corridor itself due to the door frame hiding most of it. This area appears red in the point cloud but is not visible in the bottom-up perspective of Figure 1, where only the floor is prominently visible.

6.5.3 Model Predictions. Finally Figure 9 illustrates the models' prediction behaviors once applied to the sequence responsible for forecasting the target point cloud tile salience scores. The results indicate that most models equipped with `LL-Adapter` closely predicted the participant's trajectory. The NAIVE and GRU models perfectly identified the participant's XYZ position at tile $(x = 5, y = 6)$. In contrast, the other HTB models accurately predicted the FoV direction but slightly overestimated the movement speed, placing the participant further along the trajectory while remaining close to the true location. Despite this, these models were heavily affected by occlusion, assigning high salience scores to tiles hidden from the participant by walls. This behavior is expected from models that do not use point features, which would otherwise help provide context about the surrounding environment. The cosine function, however, reduces the impact of occlusion by lowering the salience of tiles further from the participant. Additionally, all `LL-Adapter`-based models only attribute salience to tiles they predict will be in the FoV, assigning no score to others. While efficient, incorrect predictions risk QoE drops if no safeguards are in place. In contrast, `LL-PointNet`, which incorporates the GRU, demonstrates a distinct behavior by assigning high salience to the entire column of rooms and corridors aligned with the participant's position, as well as some salience to the rooms behind (influenced by past visibility). This approach ensures that, regardless of the participant's movement—whether they look left, right, or backtrack—some quality is maintained. This is made possible by the per-point scores from Algorithm 4, which `LL-PointNet` effectively leverages. Furthermore, all non-empty tiles receive at least a small salience score. This conservative strategy is unavailable to `LL-Adapter`-based models, which by design cannot leverage information stored as point features.

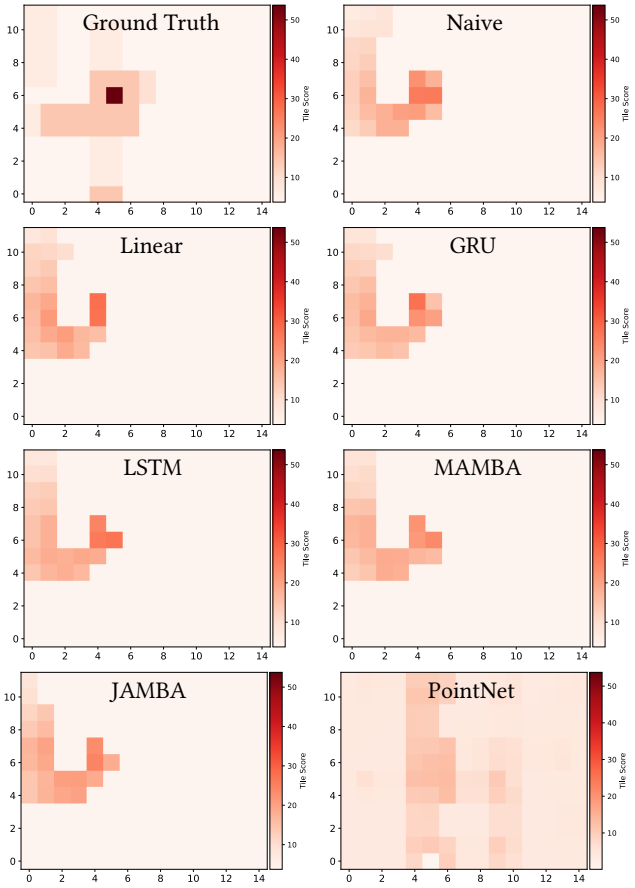


Figure 9: Predicted vs. ground truth tile score distribution.

7 CONCLUSION

This paper presents the `LL-Sparse` family of models for efficient low-latency 6-DoF FoV prediction in PCV streaming. The family consists of three variants—`LL-Adapter`, `LL-PointNet`, and `LL-SparseConv`—each capable of forecasting tile-based salience scores and offering distinct advantages depending on their specific trade-offs. `LL-SparseConv`, in particular, serves as a proof of concept, offering a new direction for research in this area. Additionally, we introduce the `MazeLab` dataset, a large-scale dynamic point cloud benchmark designed for evaluating FoV prediction models. Experimental results highlight the `LL-Sparse` family's superior scalability, low latency, and high accuracy, making it a promising solution for real-time volumetric media applications.

Acknowledgments. This research is funded by the Natural Sciences & Engineering Research Council (NSERC)-Discovery Grant RGPIN-2023-04744 and NSERC Alliance (ALLRP 586434-23)-Mitacs Accelerate (IT36792) grant.

REFERENCES

[1] Yizhak Ben-Shabat, Michael Lindenbaum, and Anath Fischer. 2017. 3D Point Cloud Classification and Segmentation using 3D Modified Fisher Vector Representation for Convolutional Neural Networks. arXiv:1711.08241 [cs.CV] https://arxiv.org/abs/1711.08241

[2] Abdelhak Bentaleb, Bayan Taani, Ali C. Begen, Christian Timmerer, and Roger Zimmermann. 2019. A Survey on Bitrate Adaptation Schemes for Streaming Media Over HTTP. *IEEE Communications Surveys Tutorials* 21, 1 (2019), 562–585. https://doi.org/10.1109/COMST.2018.2862938

[3] Alexei Botchkarev. 2019. A New Typology Design of Performance Metrics to Measure Errors in Machine Learning Regression Algorithms. *Interdisciplinary Journal of Information, Knowledge, and Management* 14 (2019), 045–076. https://doi.org/10.28945/4184

[4] Rory Brown. [n. d.]. Oculus Quest 2: Full Specification. https://vr-compare.com/headset/oculusquest2 VRcompare.

[5] Christopher Choy, JunYoung Gwak, and Silvio Savarese. 2019. 4D Spatio-Temporal ConvNets: Minkowski Convolutional Neural Networks. arXiv:1904.08755 [cs.CV] https://arxiv.org/abs/1904.08755

[6] Spconv Contributors. 2022. Spconv: Spatially Sparse Convolution Library. https://github.com/traveller59/spconv

[7] Epic Games. 2025. Unreal Engine. https://www.unrealengine.com/en-US/home Accessed: 2025-02-19.

[8] Xianglong Feng, Zeyang Bao, and Sheng Wei. 2019. Exploring CNN-Based Viewport Prediction for Live Virtual Reality Streaming. In *2019 IEEE International Conference on Artificial Intelligence and Virtual Reality (AIVR)*. 183–1833. https://doi.org/10.1109/AIVR46125.2019.00038

[9] Albert Gu and Tri Dao. 2024. Mamba: Linear-Time Sequence Modeling with Selective State Spaces. arXiv:2312.00752 [cs.LG] https://arxiv.org/abs/2312.00752

[10] Yongjie Guan, Xueyu Hou, Nan Wu, Bo Han, and Tao Han. 2023. MetaStream: Live Volumetric Content Capture, Creation, Delivery, and Rendering in Real Time. In *Proceedings of the 29th Annual International Conference on Mobile Computing and Networking* (Madrid, Spain) *(ACM MobiCom '23)*. Association for Computing Machinery, New York, NY, USA, Article 29, 15 pages. https://doi.org/10.1145/3570361.3592530

[11] Xian-Feng Han, Jesse S. Jin, Juan Xie, Mingjie Wang, and Wei Jiang. 2018. A comprehensive review of 3D point cloud descriptors. *ArXiv* abs/1802.02297 (2018). https://api.semanticscholar.org/CorpusID:3631450

[12] Kaiyuan Hu, Haowen Yang, Yili Jin, Junhua Liu, Yongting Chen, Miao Zhang, and Fangxin Wang. 2023. Understanding User Behavior in Volumetric Video Watching: Dataset, Analysis and Prediction. In *Proceedings of the 31st ACM International Conference on Multimedia* (Ottawa ON, Canada) *(MM '23)*. Association for Computing Machinery, New York, NY, USA, 1108–1116. https://doi.org/10.1145/3581783.3613810

[13] Maja Krivokuća, Philip A. Chou, and Patrick Savill. 2018. 8i Voxelized Surface Light Field (8iVSLF) Dataset. ISO/IEC JTC1/SC29 WG11 (MPEG) input document m42914. https://mpeg-pcc.org/index.php/pcc-content-database/8i-voxelized-surface-light-field-8ivslf-dataset/

[14] Opher Lieber, Barak Lenz, Hofit Bata, Gal Cohen, Jhonathan Osin, Itay Dalmedigos, Erez Safahi, Shaked Meirom, Yonatan Belinkov, Shai Shalev-Shwartz, Omri Abend, Raz Alon, Tomer Asida, Amir Bergman, Roman Glozman, Michael Gokhman, Avashalom Manevich, Nir Ratner, Noam Rozen, Erez Shwartz, Mor Zusman, and Yoav Shoham. 2024. Jamba: A Hybrid Transformer-Mamba Language Model. arXiv:2403.19887 [cs.CL] https://arxiv.org/abs/2403.19887

[15] Baoyuan Liu, Min Wang, Hassan Foroosh, Marshall Tappen, and Marianna Penksy. 2015. Sparse Convolutional Neural Networks. In *2015 IEEE Conference on Computer Vision and Pattern Recognition (CVPR)*. 806–814. https://doi.org/10.1109/CVPR.2015.7298681

[16] Junhua Liu, Boxiang Zhu, Fangxin Wang, Yili Jin, Wenyi Zhang, Zihan Xu, and Shuguang Cui. 2023. CaV3: Cache-assisted Viewport Adaptive Volumetric Video Streaming. In *2023 IEEE Conference Virtual Reality and 3D User Interfaces (VR)*. 173–183. https://doi.org/10.1109/VR55154.2023.00033

[17] Zhuang Liu, Hanzi Mao, Chao-Yuan Wu, Christoph Feichtenhofer, Trevor Darrell, and Saining Xie. 2022. A ConvNet for the 2020s. arXiv:2201.03545 [cs.CV] https://arxiv.org/abs/2201.03545

[18] Zhijian Liu, Haotian Tang, Yujun Lin, and Song Han. 2019. Point-Voxel CNN for Efficient 3D Deep Learning. arXiv:1907.03739 [cs.CV] https://arxiv.org/abs/1907.03739

[19] Gioacchino Manfredi, Vito Andrea Racanelli, Luca De Cicco, and Saverio Mascolo. 2023. LSTM-based Viewport Prediction for Immersive Video Systems. In *2023 21st Mediterranean Communication and Computer Networking Conference (MedComNet)*. 49–52. https://doi.org/10.1109/MedComNet58619.2023.10168847

[20] Jérémy Ouellette, Jashanjot Singh Sidhu, and Abdelhak Bentaleb. 2025. MazeLab: A Large-Scale Dynamic Volumetric Point Cloud Video Dataset With User Behavior Traces. In *Proceedings of the ACM Multimedia Systems Conference 2025 (MMSys '25)* (Stellenbosch, South Africa). ACM, New York, NY, USA, 7. https://doi.org/10.1145/3712676.3718345

[21] Chunghyun Park, Yoonwoo Jeong, Minsu Cho, and Jaesik Park. 2022. Fast Point Transformer. In *2022 IEEE/CVF Conference on Computer Vision and Pattern Recognition (CVPR)*. 16928–16937. https://doi.org/10.1109/CVPR52688.2022.01644

[22] Charles R. Qi, Hao Su, Kaichun Mo, and Leonidas J. Guibas. 2017. PointNet: Deep Learning on Point Sets for 3D Classification and Segmentation. arXiv:1612.00593 [cs.CV] https://arxiv.org/abs/1612.00593

[23] Charles R. Qi, Li Yi, Hao Su, and Leonidas J. Guibas. 2017. PointNet++: Deep Hierarchical Feature Learning on Point Sets in a Metric Space. arXiv:1706.02413 [cs.CV] https://arxiv.org/abs/1706.02413

[24] Feng Qian, Lusheng Ji, Bo Han, and Vijay Gopalakrishnan. 2016. Optimizing 360 video delivery over cellular networks. In *Proceedings of the 5th Workshop on All Things Cellular: Operations, Applications and Challenges* (New York City, New York) *(ATC '16)*. Association for Computing Machinery, New York, NY, USA, 1–6. https://doi.org/10.1145/2980055.2980056

[25] Mehdi Setayesh and Vincent W.S. Wong. 2023. A Content-based Viewport Prediction Framework for 360° Video Using Personalized Federated Learning and Fusion Techniques. In *2023 IEEE International Conference on Multimedia and Expo (ICME)*. 654–659. https://doi.org/10.1109/ICME55011.2023.00118

[26] Shishir Subramanyam, Irene Viola, Alan Hanjalic, and Pablo Cesar. 2020. User Centered Adaptive Streaming of Dynamic Point Clouds with Low Complexity Tiling. In *Proceedings of the 28th ACM International Conference on Multimedia* (Seattle, WA, USA) *(MM '20)*. Association for Computing Machinery, New York, NY, USA, 3669–3677. https://doi.org/10.1145/3394171.3413535

[27] Haotian Tang, Shang Yang, Zhijian Liu, Ke Hong, Zhongming Yu, Xiuyu Li, Guohao Dai, Yu Wang, and Song Han. 2023. TorchSparse++: Efficient Point Cloud Engine. In *2023 IEEE/CVF Conference on Computer Vision and Pattern Recognition Workshops (CVPRW)*. 202–209. https://doi.org/10.1109/CVPRW59228.2023.00025

[28] Jeroen van der Hooft, Hadi Amirpour, Maria Torres Vega, Yago Sanchez, Raimund Schatz, Thomas Schierl, and Christian Timmerer. 2023. A Tutorial on Immersive Video Delivery: From Omnidirectional Video to Holography. *IEEE Communications Surveys Tutorials* 25, 2 (2023), 1336–1375. https://doi.org/10.1109/COMST.2023.3263252

[29] Irene Viola and Pablo Cesar. 2022. Volumetric video streaming: Current approaches and implementations. arXiv:2209.01982 [cs.MM] https://arxiv.org/abs/2209.01982

[30] Shibo Wang, Shusen Yang, Hairong Su, Cong Zhao, Chenren Xu, Feng Qian, Nanbin Wang, and Zongben Xu. 2024. Robust Saliency-Driven Quality Adaptation for Mobile 360-Degree Video Streaming. *IEEE Transactions on Mobile Computing* 23, 2 (2024), 1312–1329. https://doi.org/10.1109/TMC.2023.3235103

[31] Zongji Wang and Feng Lu. 2018. VoxSegNet: Volumetric CNNs for Semantic Part Segmentation of 3D Shapes. arXiv:1809.00226 [cs.CV] https://arxiv.org/abs/1809.00226

[32] Sanghyun Woo, Shoubhik Debnath, Ronghang Hu, Xinlei Chen, Zhuang Liu, In So Kweon, and Saining Xie. 2023. ConvNeXt V2: Co-designing and Scaling ConvNets with Masked Autoencoders. arXiv:2301.00808 [cs.CV] https://arxiv.org/abs/2301.00808

[33] Zhirong Wu, Shuran Song, Aditya Khosla, Fisher Yu, Linguang Zhang, Xiaoou Tang, and Jianxiong Xiao. 2015. 3D ShapeNets: A Deep Representation for Volumetric Shapes. arXiv:1406.5670 [cs.CV] https://arxiv.org/abs/1406.5670

[34] Qiangeng Xu, Xudong Sun, Cho-Ying Wu, Panqu Wang, and Ulrich Neumann. 2021. Grid-GCN for Fast and Scalable Point Cloud Learning. arXiv:1912.02984 [cs.CV] https://arxiv.org/abs/1912.02984

[35] Jiacheng Yang, Christina Giannoula, Jun Wu, Mostafa Elhoushi, James Gleeson, and Gennady Pekhimenko. 2023. Minuet: Accelerating 3D Sparse Convolutions on GPUs. arXiv:2401.06145 [cs.DC] https://arxiv.org/abs/2401.06145

[36] Qin Yang, Junni Zou, Kexin Tang, Chenglin Li, and Hongkai Xiong. 2019. Single and Sequential Viewports Prediction for 360-Degree Video Streaming. In *2019 IEEE International Symposium on Circuits and Systems (ISCAS)*. 1–5. https://doi.org/10.1109/ISCAS.2019.8702654

[37] Tongyu Zong, Yixiang Mao, Chen Li, Yong Liu, and Yao Wang. 2024. Progressive Frame Patching for FoV-based Point Cloud Video Streaming. arXiv:2303.08336 [cs.MM] https://arxiv.org/abs/2303.08336



Decoupling Video Upscaling from Rendering for Cloud Gaming

Deniz Ugur
School of Computing Science
Simon Fraser University
Burnaby, BC, Canada

Ihah Amer
Advanced Micro Devices, Inc.
Markham, ON, Canada

Mohamed Hefeeda
School of Computing Science
Simon Fraser University
Burnaby, BC, Canada

ABSTRACT

Many recent video games require powerful hardware to render them. To reduce such high hardware requirements, upscalers have been proposed in the literature and industry. Upscalers save computing resources by first rendering games at lower resolutions and frame rates and then upscaling them to improve players' quality of experience. Current upscalers, however, are tightly coupled with the rendering logic of video games, which requires updating the source code of each game for every upscaler. This increases the development cost and limits the use of upscalers. The tight coupling also stifles the deployment of upscalers in cloud gaming platforms to reduce the required computing resources. We propose decoupling upscalers from game renderers, which allows utilizing various upscalers with games without changing their source code. It also accelerates deploying upscalers in cloud gaming. Decoupling upscalers from renderers is, however, challenging because of the diversity of upscalers, their dependency on information at different rendering stages, and the strict timing requirements of video games. We present an efficient solution that addresses these challenges. We implement the proposed solution and demonstrate its effectiveness with two popular upscalers. We also develop a cloud gaming system in the emerging Media-over-QUIC (MoQ) protocol and implement the proposed approach with it. Our experiments show the potential savings in computing resources while meeting the strict timing constraints of video games.

CCS CONCEPTS

• **Information systems** → *Multimedia information systems.*

KEYWORDS

Super Resolution, Video Games, Cloud Gaming

ACM Reference Format:
Deniz Ugur, Ihah Amer, and Mohamed Hefeeda. 2025. Decoupling Video Upscaling from Rendering for Cloud Gaming. In *The 16th ACM Multimedia Systems Conference (MMSys '25), March 31–April 4, 2025, Stellenbosch, South Africa.* ACM, New York, NY, USA, 11 pages. https://doi.org/10.1145/3712676.3714439

Permission to make digital or hard copies of all or part of this work for personal or classroom use is granted without fee provided that copies are not made or distributed for profit or commercial advantage and that copies bear this notice and the full citation on the first page. Copyrights for components of this work owned by others than the author(s) must be honored. Abstracting with credit is permitted. To copy otherwise, or republish, to post on servers or to redistribute to lists, requires prior specific permission and/or a fee. Request permissions from permissions@acm.org.
MMSys '25, March 31–April 4, 2025, Stellenbosch, South Africa
© 2025 Copyright held by the owner/author(s). Publication rights licensed to ACM
ACM ISBN 979-8-4007-1467-2/25/03…$15.00
https://doi.org/10.1145/3712676.3714439

1 INTRODUCTION

Recent video games strive to offer a rich and engaging experience to players. This requires video games to have high frame rates to support fast motions, complex lighting and shadowing effects to increase realism, and high resolutions to represent detailed textures and graphical elements. All of these features, however, require extensive computing resources. For example, the popular Star Wars Outlaws game supports up to 144 frames per second (fps) and 4K resolution, which requires a GPU with 10-24 GB of memory [23]. This kind of GPU costs hundreds of dollars and requires powerful/expensive gaming consoles or workstations, which may not be available to many players, limiting the potential reach of the game. To partially address this problem, *upscalers* have been proposed in the literature, e.g., RenderSR [11], ExtraSS [32], Mob-FGSR, [34], and Neural Supersampling [33], and developed by industry, including AMD FidelityFX Super Resolution (FSR) [7], NVIDIA Deep Learning Super Sampling (DLSS) [10], and Intel Xe Super Sampling (XeSS) [9]. Upscalers enable running video games with less computing resources. This is done by first rendering the game at lower resolutions and/or frame rates and then upscaling them across the spatial and/or temporal domains.

Upscalers, however, are quite different in their software designs and algorithms. For example, some upscalers use computational image processing methods, while others use deep learning models. Moreover, the computational methods and deep learning models vary across versions of the same upscaler. The myriad of upscalers, with multiple versions each, pose a significant challenge for game developers, as they need to customize the source code of their games for each upscaler and often versions of the same upscaler. This not only increases the development time and cost but also practically limits the possible number of upscalers that can be supported.

One of the critical problems behind increasing the development time/cost of utilizing upscalers and limiting their wide deployment is the *tight coupling* of the upscalers with game renderers. That is, the upscaler code currently must be integrated within the game code itself. Although common upscalers from the industry provide instructions and libraries to facilitate the integration process with various game development and rendering frameworks, the process has to be repeated for every single upscaler and often for the different versions of the same upscaler. In addition, the tight coupling of upscalers and game renderers complicates the deployment of upscalers in cloud gaming environments, which are gaining popularity but require extensive computing and bandwidth resources.

To address this problem, we propose decoupling upscalers from game renderers, which allows them to be developed separately with minimal dependence on each other. It also allows various upscalers to be easily integrated with video games without significant changes in the games' source codes, accelerating the wide adoption

of upscalers in stand-alone and cloud gaming environments. Realizing this decoupling, however, is challenging because of the diversity of upscalers, their dependency on information at different stages of the rendering process, and the strict timing requirements of video games. We address these challenges and present mechanisms to achieve this decoupling efficiently. In particular, the contributions of this paper are:

- We propose the idea of decoupling upscalers from game renderers, providing flexibility and cost-effectiveness.
- We analyze common upscalers in practice and identify the graphics resources and camera parameters required to enable upscalers and renderers to run independently.
- We design efficient synchronization schemes to support distributing upscalers and renderers on different GPUs and machines.
- We implement the proposed decoupled approach in an open-source game rendering engine to demonstrate its practicality.
- We implement an end-to-end cloud gaming system and show the potential resource saving achievable by upscalers. Our results also show that the decoupled upscaling approach does not increase latency and provides high-quality streams in real time.

2 BACKGROUND AND RELATED WORK

2.1 Background

Video Games. The logic, structure, and graphics of video games are written using frameworks such as Unity [29], Unreal [13], and O3DE [12], which are called rendering engines. Such frameworks offer high-level constructs for developing games. These constructs abstract and utilize APIs from common 3D graphics libraries such as Microsoft DirectX, OpenGL, and Vulkan.

Game Rendering. When a player issues a command, e.g., pressing a button, the game engine determines what needs to happen, e.g., firing a weapon. It then updates the game state, such as object positions, animations, and lighting. Most modern video games, e.g., Cyberpunk 2077, Call of Duty, and Battlefield, use *ray tracing* [24] to render realistic scenes. In ray tracing, light rays are cast into the scene, where each pixel corresponds to a ray traveling through the scene. Rays are then intersected with the scene objects to determine their appearance, which also depends on the material properties of objects, secondary reflections, and shadows. Rendering is *computationally expensive* and typically done on GPUs.

We provide a high-level illustration of rendering in Figure 1. The Initialization step pre-loads objects, textures, and other assets that will be used in rendering each frame. This is done at the start of the game and when significant changes occur, e.g., selecting a different game level. After initialization, multiple steps are performed for each frame, constituting the *rendering loop*. The Scene Setup step collects player actions and game decisions to create the scene, including objects, cameras, and lights. Culling techniques [21] determine which objects will be in the camera's view and need to be rendered. Then, the Geometry Processing step transforms the 3D models into a form that can be rendered on the GPU. This includes applying transformations like rotation, scaling, and translations. It also applies materials and textures to objects. The Lighting and Shading step determines how light interacts with various surfaces, accounting for multiple light sources, reflections, and subtle effects like ambient occlusion. Then, Shadow Mapping determines which

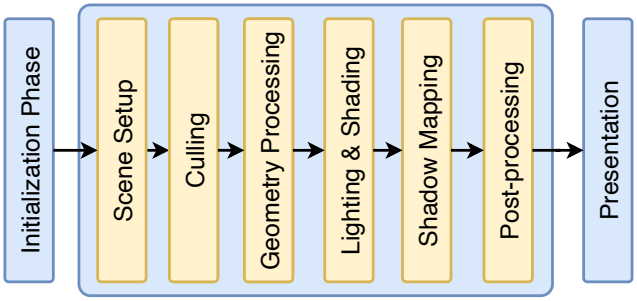


Figure 1: Main rendering steps in video games (simplified).

areas are shadowed from the perspective of each light source. Finally, post-processing effects, e.g., color correction and anti-aliasing, are applied, and the frame is presented on the display.

Spatial and Temporal Upscaling. To save computational resources, rendering can be done on a subset of pixels. Then, an *upscaling* method is used to estimate the remaining pixels. This is called spatial or resolution upscaling. Estimating pixels typically requires less computational resources than rendering them. To save even more resources, the game can be rendered at a lower frame rate. Upscaling can then be used to interpolate additional frames between the rendered ones. This is called temporal upscaling or frame generation. Example commercial upscalers include FSR [7] and DLSS [10]; both provide spatial and temporal upscaling.

Limitations of the Current Approach. In the current approach for utilizing upscalers in video games, developers need to tightly integrate and customize their code for each upscaler and even for different versions of the upscalers. As mentioned before, this process increases development costs and limits the number of upscalers that can be implemented. In addition, the tight integration of upscalers with renderers prevents upscalers from being utilized to reduce the required computational resources in cloud gaming. This is because current upscalers are designed for end users: they assume direct access to various renderers' data structures and display buffers.

Cloud Gaming. Players in cloud gaming run thin clients on virtually any device. The computationally expensive rendering process runs on cloud servers. The thin client captures the player's actions and sends them to a cloud server. The server runs the game logic, renders scenes, encodes frames using a video encoder, and streams them to the client. The client decodes and plays the received frames using any of the widely available video decoders. The entire processing pipeline must be completed within a strict deadline (around 30 msec, depending on the game), which poses a major challenge for any optimization method designed for cloud gaming.

2.2 Related Work

Upscalers. Upscalers are used to reduce the required hardware to play recent video games, which have rich graphics and fast motions [1]. There are several upscalers developed by the industry, including DLSS [10], FSR [7], and XeSS [9]. Each upscaler has its guidelines for integrating with video games. This requires developers to integrate each upscaler separately, increasing development time and cost.

In this paper, we do not present new upscalers. Instead, we propose a method to decouple renderers from upscalers.

Cloud Gaming Optimizations. Multiple works strive to address the challenges of cloud gaming, including reducing latency, bandwidth, and hardware requirements [14]. Latency is critical as even minor delays can severely impact the gaming experience, especially in fast-paced games where responsiveness is critical [5]. Bandwidth requirements of cloud gaming pose a challenge due to the high data transmission demands needed to stream high-quality graphics and maintain a smooth experience [14]. On the hardware side, several studies highlight the complexity of providing sufficient GPU resources to support diverse gaming workloads across different devices [15, 25, 27].

The proposed decoupling approach enables cloud gaming platforms to utilize upscalers to reduce computational resources.

Distributed Rendering. Some prior approaches focus on reducing the computational complexity of game renderers by modularizing and distributing them over multiple computing nodes [17, 19, 22, 28, 35]. For example, the authors of [17, 28, 35] propose distributing rendering tasks to edge servers, which can deliver lower latency to end users. The approach in [3] utilizes edge servers to support cloud gaming servers when they cannot meet the required quality. The work in [28] reduces bandwidth costs on the backbone network by sending lower-resolution game streams to edge servers, which are then upscaled using a game-specific model. Further, multiple works, e.g., [2, 19, 22], propose dividing the rendering workload between the gaming server and client.

This body of research is orthogonal to our work and can be combined with it to achieve further resource savings.

3 PROPOSED SOLUTION

3.1 Overview and Challenges

Overview. In contrast to the current approach of implementing upscalers within the game code, we propose decoupling the upscalers from renderers as illustrated in Figure 2. That is, the game renderer and upscaler run as independent processes, either on the same machine or different machines. Further, the upscaler and renderer can run on different GPUs within the same machine. This is enabled by identifying the critical graphics elements and structures required by upscalers and efficiently sharing them across distributed processes.

The current approach requires changing the source code of each game and customizing it for each upscaler version separately, limiting its scalability. Whereas the proposed approach does not require changing the source code of the game. It only makes minor modifications to the rendering framework and adds thin wrappers around various upscalers without changing their source codes. This enables upscalers to be easily and transparently used with many games.

Challenges. Realizing the proposed decoupling approach faces multiple challenges. First, upscalers are quite diverse in their internal designs and requirements. Some, e.g., DLSS and XeSS, are implemented using deep learning models, while others, e.g., FSR, are implemented using computational image processing methods. Computational upscalers tend to require accessing more graphical structures than deep learning ones, as they execute detailed mathematical equations on various aspects of the pixels and frames. Deep learning upscalers rely on the mapping power of neural networks.

In addition, most upscalers are designed assuming tight integration with the renderer, which allows them to access the needed

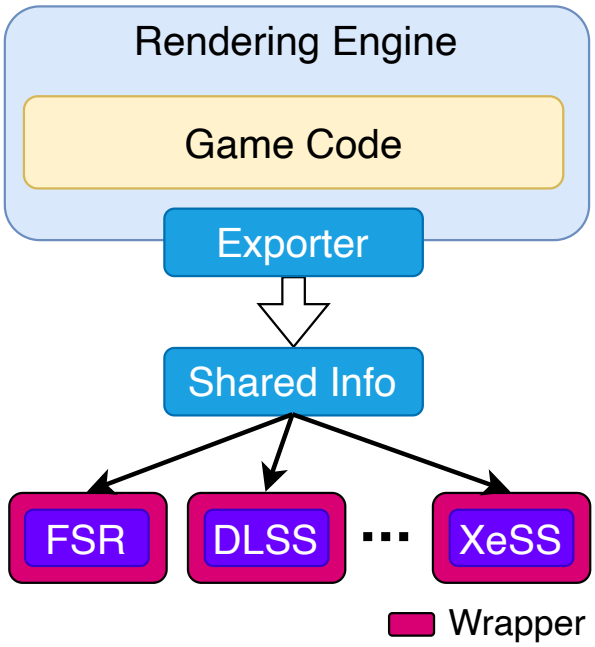


Figure 2: Overview of the proposed approach of decoupling upscalers and renderers.

structures and information at different stages of the rendering process, not necessarily at the end of the pipeline. For example, FSR may require accessing color textures pre- and post-translucency passes [8]. In §3.2, we analyze the most commonly used upscalers in practice, FSR and DLSS, and identify the general requirements to support various upscalers.

The second main challenge stems from running the upscaler and renderer as independent processes instead of one as in the current approach. This is because they run at different speeds. Thus, they need synchronization to access the shared data structures. However, current video games can render up to 144 fps at 4K, generating large amounts of data and imposing high computational demands that must be performed by strict deadlines. Further, the upscaler and renderer processes can run on different GPUs or even different machines, making it harder to meet the deadlines over the network. In §3.4, we present an efficient solution using ring buffers and *fences*. Fences are synchronization primitives for sharing resources within GPUs as well as between GPUs and CPUs, and they support remote direct memory access (RDMA). Fences or their equivalent are available in common 3D graphics libraries, including DirectX, OpenGL, Vulkan, and Metal.

Finally, rendering engines and upscalers have large and complex code bases. Thus, changes to them must be clearly defined and minimized for the proposed approach to be practical. In §3.3, we present a simple wrapper around upscalers that does not change their code. We also identify and localize the required minor modifications to rendering engines.

Benefits of Decoupling Upscalers and Renderers. The proposed approach offers multiple advantages to game developers, cloud providers, and players, as summarized below.

Benefits for Game Developers: The proposed approach does not require integrating upscalers inside the game code, which reduces the development time. Upscalers can be used transparently and on different machines, increasing the game's availability to players.

Benefits for Cloud Gaming Providers: Decoupling upscalers from renderers allows utilizing upscalers in cloud gaming, which is currently not possible. Upscalers save substantial computing resources, as discussed in §2.1. Decoupling also allows running upscalers and renderers on different machines, i.e., offloading upscalers to other machines. Given the different computational requirements of renderers and upscalers, this enables the optimization of their performance separately by customizing the machines that run each. Further, upscalers are themselves quite diverse regarding the ideal platforms to run each. For example, DLSS runs better on NVIDIA GPUs because it relies on native NVIDIA APIs, while FSR may yield similar performance on various GPUs as it uses standard computational image processing functions. The proposed decoupling approach allows large cloud gaming providers to deploy multiple upscalers on different machines.

Finally, the proposed approach enables cloud providers to dynamically manage the computing resources allocated to different gaming sessions. This can be achieved by controlling the spatial and temporal scaling factors of the upscaler, which can easily be accessible since the upscaler is running as an independent process.

Benefits for Players: Players benefit from the ability to deploy various upscalers that best suit their hardware and the games they are playing, improving their gaming experience.

3.2 Identifying Upscalers Requirements

Simple upscaling methods, e.g., bilinear and bicubic interpolation, only require a color input to function. The results from these methods significantly lack quality compared to methods like FSR and DLSS. The underlying reason is that the upscaler lacks the context around each pixel and, therefore, must apply generic upscaling for each pixel. Modern upscaling methods require additional textures from the rendering engine to understand the scene in greater detail.

We analyzed the most popular upscalers currently deployed: AMD FSR [7], NVIDIA DLSS [10], and Intel XeSS [9]. We inspected their code integration guidelines and steps. We also experimented by implementing them with different games. Based on our analysis, we define three categories of resources for decoupling upscalers from renderers: (i) required resources, (ii) optional resources, and (iii) camera parameters.

The required resources are essential for the operation of upscalers. They determine which regions of the scene need to be upscaled and how. We identified three common resources needed by all upscalers: Depth Map, Motion Vectors, and Final Color. Optional resources assist in improving the quality produced by upscalers but are not necessary for the operation. For example, the UI Color and Alpha resource helps ensure that UI elements like health bars, menus, and text do not become blurry during the upscaling process. Optional resources vary across upscalers. Thus, we identified the optional resources for each upscaler. Then, we export the union of all optional resources to support a wide variety of upscalers. Based on our analysis, the most common optional resources are: Final Color Subrect, Hudless, UI Color and Alpha, Bidirectional

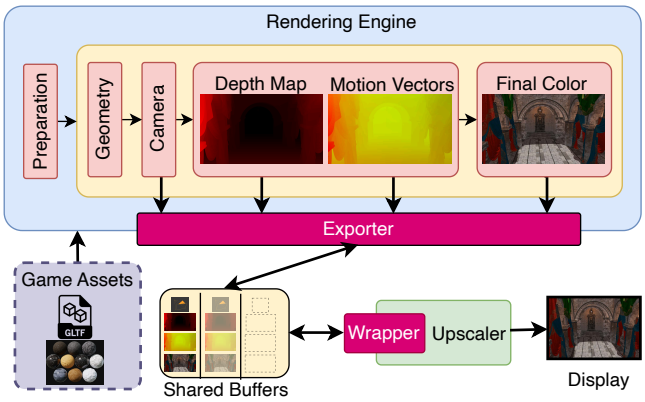


Figure 3: Collecting graphics resources and camera parameters at different rendering stages.

Distortion Field, T&C Mask, Reactive Mask, and Opaque Color. All of the required and optional resources are handled by the GPU.

The third category of resources for decoupling upscalers from renderers is the camera parameters. There are two types of parameters: intrinsic and extrinsic. Both are used in game rendering to accurately capture and display 3D scenes. Intrinsic parameters include: Focal Length (determines the zoom level of the camera), Principal Point (where the optical axis intersects the image plane), Skew Coefficient (describes the angle between the x and y pixel axes), and Lens Distortion Coefficients (corrects distortions caused by the camera lens). Extrinsic parameters define the position and orientation of the camera in the 3D world, and they include: Rotation Matrix (specifies the camera's orientation) and Translation Vector (specifies the camera's position). Camera parameters are essential for upscalers. For example, the Focal Length and Field of View parameters help the upscaler understand the depth and perspective and the scene. Similarly, the extrinsic position and orientation parameters ensure that the upscaled frames align with the game's world and have consistent motions.

In addition, camera parameters are essential for upscalers to utilize *sub-pixel jittering* in collecting more information and correctly upscaling frames. Sub-pixel jittering means slightly shifting the camera's position by a fraction of a pixel, capturing more detail in each frame. Sub-pixel jittering allows upscalers to sample different points within a pixel across multiple frames, giving it more information to upscale lower-resolution frames to higher quality. The intrinsic camera parameters, e.g., focal length, principal point, and lens distortion coefficients, affect how sub-pixel jittering is applied and how frames are upscaled. Similarly, the extrinsic parameters define the camera's position and orientation in the world, and sub-pixel jittering requires these parameters to ensure that the small movements are accurately captured and aligned.

Camera parameters are managed by the CPU, unlike the other two categories of resources, which are stored on the GPU.

3.3 Modifying Renderers and Upscalers

We propose a general approach for decoupling upscalers from renderers, illustrated in Figure 3. We identify all required resources and make them available to different upscalers. This approach allows

Algorithm 1 Renderer Process

```
1:  InitializeResources()
2:  while RendererRunning() do
3:      WaitOnFence(ResourceEvicted, bufferIndex)
4:      ProcessUserInputs()
5:      for all render module rm do
6:          GenerateCommandList(rm)
7:          if RequiredForUpscaling(rm) then
8:              CopyTextureRegion(texture, buffer)
9:      ExecuteCommandList()
10:     SignalFence(ResourceReady, bufferIndex)
```

Algorithm 2 Wrapper Process

```
1:  InitializeResources()
2:  while WrapperRunning() do
3:      WaitOnFence(ResourceReady, bufferIndex)
4:      for all bufferTexture in buffer do
5:          texture = GetReference(bufferTexture)
6:      EvaluateUpscaler()
7:      ExecuteCommandList()
8:      PresentFrame()
9:      SignalFence(ResourceEvicted, bufferIndex)
```

switching upscalers on demand and easily utilizing future upscalers. As discussed above, the required and optional graphics resources are stored in the GPU memory, whereas the camera parameters are maintained in the main CPU memory. Thus, we create two shared buffers: one in the GPU and another in the CPU. In addition, our approach allows running multiple upscalers independent of the game renderer. A small wrapping code enables the upscalers to transparently access the needed graphics resources and camera parameters. The wrapping code also handles synchronization of the shared buffers with renderers, even if both run on different machines, as explained in §3.4.

Implementation in Rendering Engines. We first note that game rendering engines, such as Unity and Unreal Engine, provide all the required textures in their own way. Internally, the process needed to generate these textures is largely the same. Existing upscaler plugins for these engines modify the rendering pipeline to generate these textures or obtain a reference to them if they are already being used by the developer. These textures are fundamental building blocks of any rendering engine, and game engine developers ensure they are easily accessible.

A typical rendering pipeline consists of multiple render modules, each responsible for a specific aspect of the rendering process. For example, geometry processing transforms 3D scene data into screen-space coordinates, while lighting modules compute how light interacts with objects, and texturing modules apply image textures to surfaces. Afterward, post-processing modules may add visual effects such as bloom or depth of field.

Each render module generates *commands* that describe actions like drawing objects, applying shaders, or switching textures. These commands are collected into a master command list, which is then submitted to the GPU for execution. Since this process is performed for every frame and these steps may occasionally take longer than the duration of a single frame, the renderer uses a **swap chain** to avoid blocking the main thread. A swap chain is a sequence of render targets (buffers) that are used in rotation. While the GPU executes commands to render the current frame to one buffer (called the back buffer), the renderer processes the next frame in another buffer. When it is time to present the next frame, the back buffer and front buffer are swapped, displaying the newly rendered frame while the next one is prepared.

We summarize the proposed rendering engine modifications to support decoupling renderers and upscalers in Algorithm 1. Mainly, we added new commands to copy various resources at different rendering stages (lines 7–8) and synchronize shared memory buffers (lines 3 and 10). Everything else in the rendering engine is not changed. Conventional upscaling methods are tightly integrated with the game's rendering pipeline, allowing them to reference the texture directly rather than copying it. However, if a texture is only valid at a particular stage (i.e., it may be released before being presented), it is copied to an intermediary texture to preserve its state. Upscalers need this process to reference all necessary resources to upscale the frame. When the upscaler is decoupled, we must always preserve the texture's state, so we copy the textures whenever the renderer finishes using them.

Additionally, when a frame completes execution on the GPU, it signals the shared resource buffer to indicate that the frame data is ready for upscaling. This is shown in Algorithm 1 at line 10 for the renderer and in Algorithm 2 at line 3 for the wrapper. Both processes use fences to synchronize their state and determine when to act on a buffer. This is crucial because the wrapper acquires references to individual textures in the buffer slot as if they were its own, allowing the upscaler to use them as usual. The synchronization process is explained in §3.4.

Wrappers for Upscalers. To support decoupling upscalers from renderers, we have developed a wrapper that emulates the actual game environment. The wrapper supplies the necessary resources and textures to upscalers, just as the game would if the upscalers were tightly integrated. We summarize the wrapper design in Algorithm 2. It waits until resources are ready in the shared resource ring buffer. Upon receiving the signal, the wrapper continues to acquire references to textures within the buffer. Then wrapper calls the upscaler (line 6), which generates the necessary commands to upscale the frame on the GPU. These commands are appended to the main command list before sending it to GPU (line 7). Once the commands are executed, the upscaled frame is presented on the display (line 8). Finally, the wrapper evicts the buffer slot that was used by the frame by signaling the synchronization fence.

TransparentSR Library. We abstracted the creation and management of shared resources into an open-source library called TransparentSR [30]. This library simplifies the integration of detached upscaling into custom rendering engines. Our repository provides examples demonstrating how this library can be used effectively. For instance, the renderer can generate a low-resolution 3D scene and leverage the `TSROps::TransferToSharedBuffer` method to transfer the necessary resources to the upscaler. Additionally, the library offers a utility function, `tsr_map_udta`, which facilitates

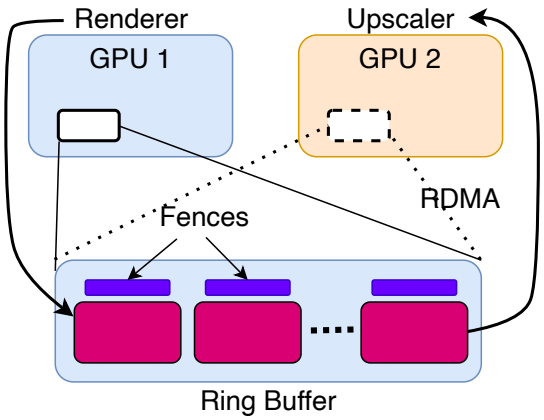


Figure 4: Synchronization of shared resources between renderer and upscaler deployed on different machines using fences and RDMA (remote direct memory access).

seamless management of user data (e.g., camera parameters) across frames, ensuring consistency and ease of use.

3.4 Synchronizing Shared Buffers

The proposed approach runs the game renderer and upscaler as independent processes, which provides flexibility and ease of utilizing various upscalers. However, both processes execute at different speeds and can be allocated on different machines. Thus, efficiently sharing the required graphics resources and camera parameters among them is critical for ensuring the quality and timeliness of rendered frames.

We propose a general approach that allows renderers and upscalers to run on: (i) the same GPU, (ii) different GPUs within the same machine, and (iii) GPUs running on distributed machines. Figure 4 provides a high-level illustration of the proposed approach. Specifically, we create a shared buffer on the renderer machine. Then, we virtually map this buffer to the upscaler machine to avoid duplicating the resources. The upscaler process can access the shared buffer using DMA (Direct Memory Access) if it resides on the same machine as the renderer. If the upscaler is on a different GPU or machine, it uses RDMA (Remote DMA). Current technologies, such as AMD Radeon Open Compute (ROCm), NVIDIA GPUDirect, and Intel oneAPI, support RDMA. Distributing renderers and upscalers to different machine is particularly beneficial for cloud gaming platforms, where the renderer may run on GPUs optimized for ray tracing and graphics operations and upscalers may be deployed on GPUs with tensor cores optimized for neural network operations and/or computational upsampling.

To enable this generality of deploying renderers and upscalers, we propose synchronizing access to the shared buffer through fences. Fences are synchronization primitives designed for inter-CPU and inter-GPU communications. They are supported in widely-used 3D graphics libraries, such as DirectX, OpenGL, Vulkan, and Metal. Fences also function in distributed environments and support RDMA. As described in our implementation (§4), we create fences and share their handlers between renderers and upscalers.

We design the shared buffer as a ring buffer with a fixed size of three slots. Having more slots wastes memory and could introduce undesirable delay between when the frames are rendered and when they are displayed (i.e., after the upscaler finishes processing them). Video gaming is an interactive application, and players expect to see the effect of their actions on the display within a strict deadline. On the other hand, having less than three slots in the ring buffer can lead to wasting computing resources, as one of the processes may have to wait to access the shared buffer: The renderer may have to wait for a free slot to write the output frame in, and the upscaler may wait until for a frame to be available to upscale it.

We note that in most cases, the upscaling process will run faster than the renderer. Thus, at least two slots in the shared buffer are needed so that while the renderer is writing frame data into one, the upscaler is reading from the other. The third slot in our design serves as a relief buffer. If upscaling takes longer than expected due to complex operations or OS-level scheduling, for example, the renderer can continue without stalling.

Implementation wise, we place fences at the end of the command list to signal when the resources are ready for upscaling. When the command list is executed on the GPU, the data will be copied to the next available slot in the shared ring buffer. Simultaneously, we copy the camera data to the shared buffer region located in the system memory. When the fence is signaled, our wrapper immediately upscales the frame using the frame data generated by the renderer. Once the upscaling is complete, the upscaled frame is presented, and the shared buffer region is signaled to release the frame data.

3.5 Overheads and Practical Considerations

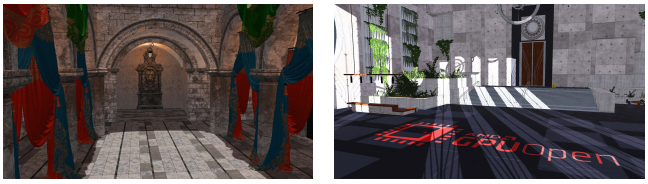
The proposed decoupling solution requires minor modifications to the rendering engine, and it imposes small overheads. The modifications can be easily added to existing rendering engines. Importantly, these modifications are applied once to the rendering engine rather than to each individual game. This significantly reduces the development effort and streamlines the integration process across multiple games.

Regarding overheads, the primary considerations are the memory and time associated with copying resources. In our experiments, copying resources to the shared ring buffer adds approximately 10.6 MB for the entire frame data, a small amount for modern systems. Also, it takes, on average, 0.16 ms for the exporter to copy the needed resources to the shared buffer. This is insignificant, considering the typical rendering time per frame is 33.3 ms (for a frame rate of 30 fps). Even for a high frame rate of 100 fps (i.e., rendering time of 10 ms), the added overhead time is 1.6%.

We believe these are small overheads considering the potential gains of easily and transparently integrating various upscalers with many video games.

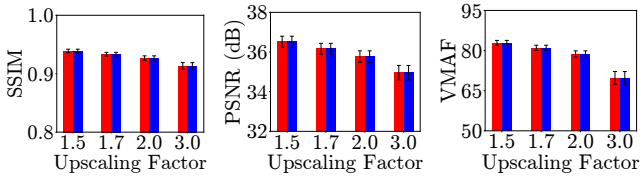
4 EVALUATION

First, we describe our proof-of-concept implementation in an open-source rendering platform. Then, we demonstrate the effectiveness of the proposed decoupled approach with two common upscalers (FSR and DLSS) with different upscaling configurations and game scenes. Then, we present our implementation in a cloud gaming



(a) Sponza (b) Brutalism

Figure 5: Game scenes used in our experiments. They are open-source samples with AMD FSR upscaler [8].



(a) SSIM (b) PSNR (c) VMAF

Figure 6: Comparison of the quality achieved by the decoupled (Red) and current (Blue) approaches. Results for DLSS with no frame generation processing Sponza game.

platform using Media-over-QUIC, and we conduct end-to-end analysis showing the feasibility of utilizing upscalers to save computing resources in cloud gaming.

4.1 Prototype Implementation

We implement our proposed solution in the open-source rendering engine Cauldron [6], which is used to showcase various rendering technologies, including FSR [8]. To modify this engine to support our decoupled approach, we had to understand how it prepares the rendering command list, updates the camera matrices, and allows us to access the graphics resources. We summarize our modifications in the following.

Modifying the Renderer. The renderer modifications primarily involved ensuring that textures and related upscaling data are consistently copied for each frame at the appropriate times in the rendering pipeline. To achieve this, we issue copy commands once the renderer finishes that particular piece of data. For GPU-bound data, we append the copy command to the command list when a resource is not modified further. The necessary operation is similar for CPU-bound data; we copy it when the renderer has completely finished with that frame. Before copying textures to their destination, we initialize the shared ring buffers.

To achieve the synchronization described in §3.4, we create two shared ring buffers, one on the GPU and one on the CPU. Additionally, we establish shared fences to facilitate synchronization between the GPU and CPU. This requires creating textures with the `D3D12_HEAP_FLAG_SHARED` flag and fences with the `D3D12_FENCE_FLAG_SHARED` flag. These flags allow the renderer or upscaler processes to obtain a handle to the resources using DirectX 12 device methods `OpenSharedHandleByName` and `OpenSharedHandle`. Each buffer stores the textures associated with a frame in a contiguous memory region, and both processes access the buffer using offsets. When both processes are active, they wait for an available slot in the ring buffer to operate on. The renderer requires an empty slot to write to, while the upscaler needs a slot with all the frame data written and ready for upscaling. Each process signals the other about the status of a particular slot, allowing coordination for further work. By keeping both processes synchronized and aware of the shared ring buffer's status, we effectively eliminate any race conditions that may occur in multi-process designs.

Wrappers for FSR and DLSS. Wrappers create a renderer-like environment for the upscalers. They supply the frame data in the same manner as a typical renderer would. From the upscalers' perspective, there is no difference in the input data compared to when they are tightly integrated with the actual renderer. When the wrapper is signaled that a slot on the shared ring buffer is available, it copies the textures into its own address space, along with the camera properties stored in the CPU-side shared ring buffer. This information is continuously supplied to the upscalers, allowing them to track the temporal data as needed. The wrapper itself does not alter the upscalers' implementation; it simply enables them to function by redirecting the necessary data from the actual renderer.

4.2 Experimental Setup

Upscalers and Game Scenes. We consider two popular upscalers: FSR and DLSS. FSR uses computational image processing methods and is open source. DLSS uses deep learning models and is partially open source. Both support various configurations, including frame generation and multiple spatial upscaling ratios.

We utilize two game scenes that are open source and diverse, as shown in Figure 5. We summarize all parameters used in the experiments in the following.

- **Upscalers:** None, FSR 3, DLSS 3
- **Modes:** Decoupled, Conventional
- **Frame Rate (FPS):** 20, 30
- **Frame Generation:** Enabled, Disabled
- **Upscaling Ratios:** 1.5, 1.7, 2.0, 3.0
- **Scenes:** Sponza, Brutalism

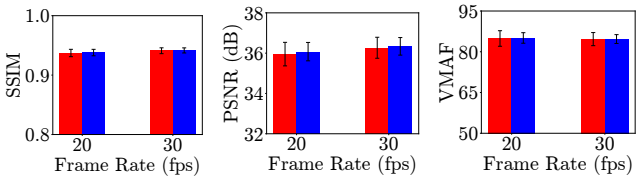
Performance Metrics. For each test run, we collect several performance and quality metrics, which are summarized below.

- **System Memory:** Total, Renderer, Upscaler
- **CPU Utilization:** Overall , Renderer, Upscaler
- **GPU**: Power Usage, Load Percent, Memory Usage
- **Processing Time:**[1] Render time, Upscale time
- **Quality:** SSIM, PSNR, VMAF

The system resource usage metrics are gathered from three separate tools. Metrics related to the overall utilization of individual resources are collected by the **HWiNFO** tool [20]. Process-specific CPU and memory utilization is collected by the Python library **psutil** [26]. In-process timings for different stages of rendering and upscaling are collected by the sample applications and reported as aggregated values at the end of each test run.

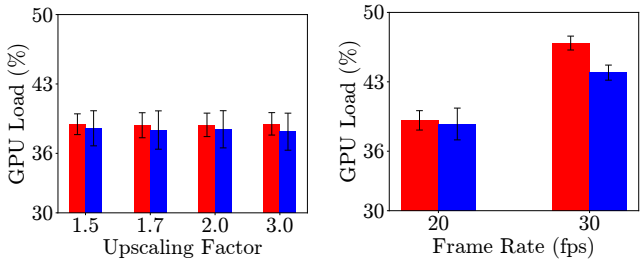
Quality comparisons are performed using SSIM (Structural Similarity Index) [31], PSNR (Peak Signal-to-Noise Ratio), and VMAF (Video Multi-Method Assessment Fusion) [36] metrics. SSIM and PSNR provide visual similarity to human perception and objective quality, respectively. VMAF leverages multiple metrics to assess video quality more comprehensively.

[1]These are collected inside the Renderer/Upscaler processes



(a) SSIM (b) PSNR (c) VMAF

Figure 7: Comparison of the quality achieved by the decoupled (Red) and current (Blue) approaches. Results for FSR with frame generation processing Sponza game.



(a) No Frame Generation (b) With Frame Generation

Figure 8: GPU utilization for the decoupled (Red) and current approaches (Blue). Results for DLSS processing Sponza.

Workstation Specifications. For all our tests, we use a workstation with i9-14900K CPU, 64 GB DDR5 Memory, and NVIDIA RTX 4060 GPU with 8 GB memory. This workstation provided sufficient performance to run all tests smoothly.

4.3 Evaluation on Individual Gaming Stations

We analyze the quality of upscaling produced by the proposed decoupled approach and compare it against the current, fully integrated, upscaling approach, which requires changing the source code of each game. We also analyze the overheads imposed by the proposed approach.

Quality Analysis. Accurately computing the SSIM [31], PSNR, and VMAF [36] quality metrics requires alignment of camera movement across different test runs; otherwise, the produced frames will have different pixel values and will not be comparable. While it is possible to keep the camera static, doing so would hinder the proper evaluation of the upscalers. To consider the dynamic aspects of game scenes, we move the camera in a torus curve to create a relatively complex camera movement. During the test run, we save the front buffer (presented frame) as a JPEG file without any compression at every animation step. Then, we combine the frames into a MOV container and save the video file without re-encoding the frame data.

To ensure our experiments are conducted as consistently as possible, we automate the entirety of our experiments and quality evaluation, including the camera movement of the renderer. We also fix the animation step time to achieve pixel-perfect frame alignment between test runs. This means that regardless of how long it takes to render a particular frame, we always capture the same scene setup. We run each test for 30 seconds, with a 30-second cool-down period, and we repeat the process three times. We run *every* permutation of the configuration parameters in §4.2.

We present representative *samples* of our results in Figure 6 and Figure 7; other results are similar. In these figures, we plot average results across all runs as well as the 95% confidence intervals as error bars. The results in Figure 6 for the DLSS upscaler show that the quality achieved by the proposed decoupled approach is indistinguishable from the quality of the current approach for all spatial upscaling ratios considered and all three quality metrics. Figure 7 demonstrates similar results for the FSR upscaler with frame generation enabled. That is, the proposed approach of decoupling upscalers from renderers does not negatively impact the upscaling quality, while it facilitates utilizing upscalers with different games. In addition to analyzing the quality using various objective metrics, we have also visually inspected the produced frames with and without our transparent upscaling method to ensure that there is no distinguishable quality difference.

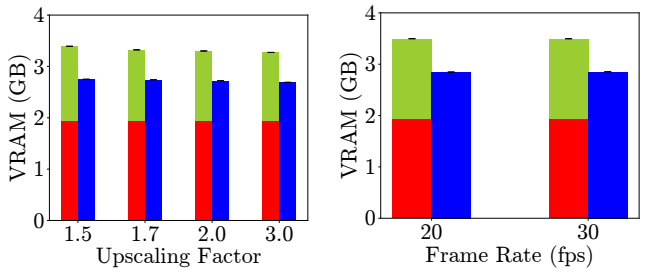
Overhead Analysis. To assess the overheads of the proposed approach, we terminate all applications running on the workstation and keep only our renderer and upscaler processes. We measure the various system utilization metrics mentioned in §4.2 using HWiNFO [20] and psutil [26] tools.

We present a sample of our results in Figure 8 and Figure 9, showing the GPU utilization and CPU memory usage, respectively. The proposed decoupling approach runs the renderer and upscaler as separate processes, which requires synchronization and copying of some data structures. This marginally increases the GPU utilization as shown in Figure 8. For example, with no frame generation (Figure 8a), the average increase in GPU utilization is less than 1%. Even when the upscaler generates additional frames (Figure 8b), which requires more processing, the increase in GPU utilization because of decoupling is up to 2.54%. We note similar increases in the CPU load have been observed in our experiments; figures are omitted due to space limitations.

In Figure 9, we analyze the memory requirements. We measure and plot the memory of the renderer and upscaler of the decoupled approach separately. The current approach, however, integrates the renderer and upscaler, and thus, its memory is not separated. As the figure shows, the decoupled approach requires less than 1 GB of additional memory. We also analyze the GPU memory requirements (figure not shown). Our results indicate that the decoupled approach requires up to 927 MB of additional GPU memory. The additional memory is because the decoupled approach creates two independent processes for the renderer and upscaler instead of one. Each process allocates its memory and resources. This is in addition to the shared buffers for exchanging required information between the renderer and upscaler. For practical deployments, multiple code optimizations can be performed to reduce the memory footprint of decoupling; our proof-of-concept prototype focused more on demonstrating the feasibility of decoupling.

4.4 Evaluation in Cloud Gaming

Implementation in Cloud Gaming using Media-over-QUIC. We implement the proposed decoupled upscaling approach in a cloud gaming platform to demonstrate its practicality. We are not aware of other works in the literature that utilize upscaling to reduce the extensive computing resources required by cloud gaming.



(a) No Frame Generation (b) With Frame Generation

Figure 9: CPU Memory usage for the decoupled approach and current approach (Blue). Memory of the decoupled approach is divided into two parts: Renderer (Red) and Upscaler (Green). Results for FSR processing Sponza.

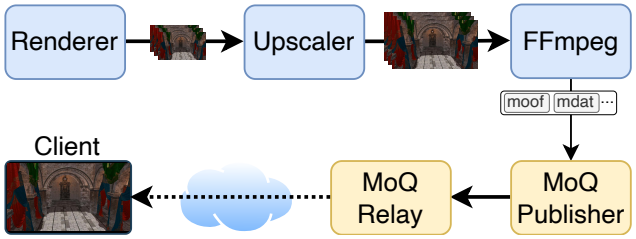


Figure 10: Overview of our cloud gaming testbed.

We consider the emerging Media-over-QUIC (MoQ) platform, which is being actively developed by the Internet Engineering Task Force (IETF) for efficient and low-latency streaming environments [16]. It is built on top of the widely deployed QUIC transport protocol, which is used in popular streaming services such as YouTube. Compared to WebRTC and other existing frameworks, MoQ offers several advantages, including better scalability and efficient congestion control. It also offers the ability to prioritize packets, such as player inputs or critical game frames, ensuring that essential data is delivered promptly, which is crucial for cloud gaming, where latency directly affects gameplay.

Our MoQ cloud gaming testbed is illustrated in Figure 10. The game renderer produces frames at low resolution and frame rate, which consume less computing resources. The upscaler, running as a separate process, improves the spatial resolution and generates new frames. The result of the upscaler is frames in RGBA32 pixel format. We pipe this raw pixel data into an FFmpeg module, which encodes and packages it into a fragmented MP4 container format. The encoded video stream is then piped through two other modules: MoQ Publisher and MoQ Relay. Both are based on the open-source implementation of MoQ [18]. MoQ Publisher publishes the stream using HTTP/3 (on top of QUIC). MoQ Relay takes the stream from the MoQ Publisher and serves it to clients, while performing functions such as adaptation, metadata processing, and caching. Clients connect to the MoQ Relay using a web browser that supports WebTransport, which provides low-latency and bidirectional communication over QUIC. The browser demultiplexes the stream, extracts video segments, and displays them.

Quality Analysis. We first subjectively validated the visual quality of the rendered frames on the receiver side after going through the entire pipeline in Figure 10. We observed no glitches, freezes, or visual distortions. This served as a sanity check for our implementation.

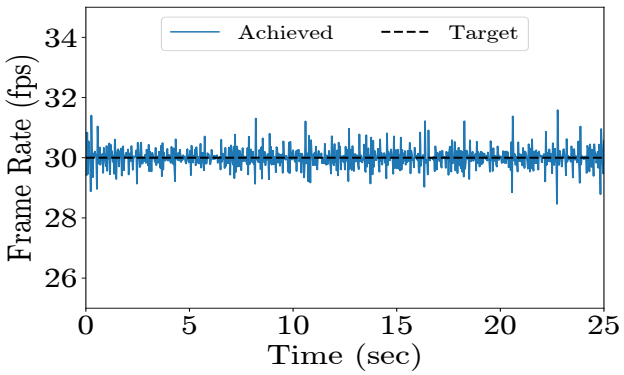


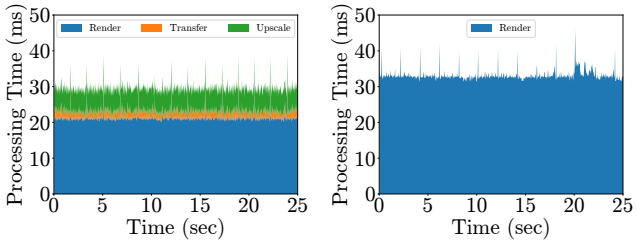
Figure 11: Performance of the FSR upscaler in the cloud gaming testbed implemented using the decoupled approach. Results show that our approach achieves the target frame rate.

To objectively analyze performance, we measure the frame rate achieved at the receiver. We set a target frame rate of 30 fps and configure the game engine to render at this rate with low resolution. Then, we upscale the resolution by 3.0x using FSR. We chose a 3.0x upscaling ratio because the FSR documentation recommends it as the maximum resource-saving option. There is no limit imposed on the upscaling ratio otherwise.

Following upscaling, frames are then encoded and transmitted over the MoQ testbed to the receiver. We record the arrival time of each frame at the receiver and compute the achieved frame rate over an extended period of time. The results, shown in Figure 11, confirm that our decoupled approach implemented in a cloud gaming testbed can achieve the target frame rate.

Timing Analysis. We analyze the time used by the renderer and upscaler for each frame. We record the timestamps as a frame passes through key stages of the pipeline. These stages include the start and end times for rendering and the start and end times for upscaling. Since our solution performs rendering and upscaling in separate processes, we also record the texture transfer time from the renderer to the upscaler. The results are shown in Figure 12a, where each stage is stacked on top of the previous stage to show the accumulated processing time for individual frames. The majority of the processing is done in the rendering stage, which accounts for more than 65% of the total processing time. Upscaling is much faster compared to rendering but still takes about 6-7 ms to perform. Transferring shared data between the renderer and upscaler takes about 1-2 ms.

To put these numbers in perspective, we plot, in Figure 12b, the time taken by the renderer for each frame without using any upscaling. In this case, the renderer produces the full resolution. As the figure indicates, the renderer takes, on average, slightly more time than the total time in Figure 12a, by about 2-3 ms per frame. That is, upscaling can actually reduce the total latency, which is especially important in cloud gaming where latency is critical. In addition, Figure 12b shows more occasional spikes in frame processing time



(a) Decoupled Upscaling **(b) No Upscaling**

Figure 12: Timing analysis from our cloud gaming testbed.

than the case with upscaling. This means the decoupled upscaling can provide a smoother performance.

Benefits of Upscaling in Reducing Computing Resources. We study the potential benefits of using upscalers to reduce computing resources. Specifically, we analyze two video games: Forza Horizon 5 and F1 23. Both are graphically rich car racing games. We first play each game without any upscaler, and we measure the fps achieved by the GPU on our workstation. We use CapFrameX [4] to record frame timings. Then, we run the game with different upscaler configurations. Then, we repeat the whole experiment for the other game.

A sample of our results is shown in Figure 13 for the F1 23 game using the FSR upscaler, where we plot the overall average frame rate achieved by the renderer across the whole experiment time. The figure shows that significant savings in computing resources could be realized by using upscalers. For example, without using any upscalers, our machine could render, on average, about 40 fps. Whereas, when we render at low resolution and then use FSR to upscale the resolution by 2.0x or 3.0x, the machine could almost double or triple the average number of rendered frames. This is achieved with almost no impact on the visual quality in the case of 2.0x and minimal impact in the case of 3.0x, based on our subjective analysis of the displayed frames during the experiments.

In Figure 14, we plot the achieved frame rate by the renderer across time with and without using upscalers. The results in the figure are for DLSS with frame generation enabled. We also plot the average across 1-sec periods as thicker lines. Two observations can be made on this figure. First, significant gains, in terms of the number of rendered frames per second, can be achieved by using upscalers. This aligns with the aggregate results in Figure 13, which were obtained with different upscaler and video games. Second, the achieved frame rate, with and without upscaler, fluctuates with time. This is because both the visual complexity of frames and the temporal motion across frames change during the game. This fluctuation indicates that the required computing resources in cloud gaming vary. Since our proposed approach decouples the renderer from upscaler, it offers a flexible and easy method to control the configuration parameters of the upscaler. This, in turn, enables cloud providers to dynamically manage these parameters across different gaming sessions to maximize the utilization of their computing resources and achieve the target quality of experience.

In summary, our cloud gaming testbed demonstrated that the proposed decoupled upscaling approach can successfully process and deliver high-quality gaming streams in real time, and it does not introduce additional latency. In addition, our experiments show

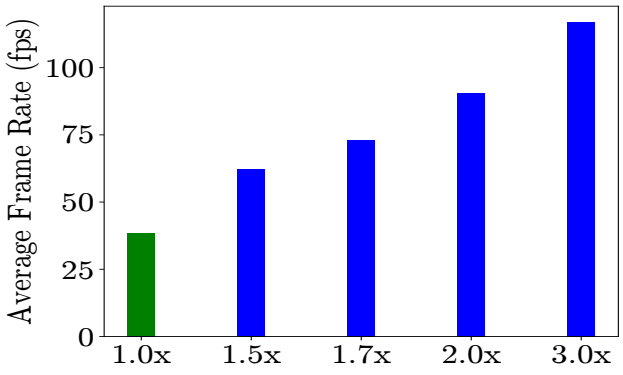


Figure 13: Performance gains from using upscalers. Results for FSR processing F1 game.

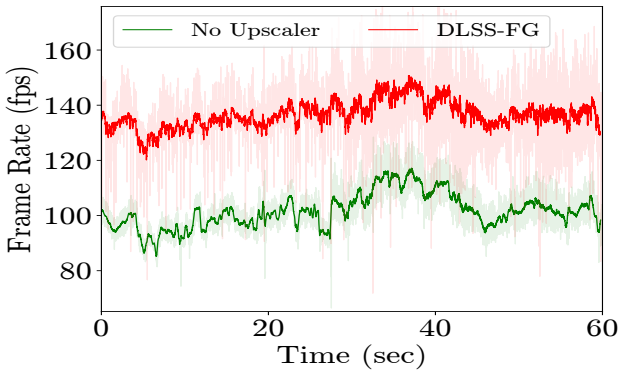


Figure 14: Performance gains from using upscalers. Results for DLSS with frame generation processing Forza game.

using upscalers in cloud gaming can save computing resources and/or serve more gaming sessions with the same resources.

5 CONCLUSION

Current upscalers require tight integration with the source code of video games, increasing the development cost and limiting their wide deployment. We presented a method to decouple upscalers from game renderers, which facilitates transparently using various upscalers without the need to change the source code of games. It also enables the utilization of upscalers in cloud gaming to save computing resources, which has not been done before. We demonstrated the practicality of the proposed method by implementing it in an open-source rendering game engine and showed its effectiveness with two popular upscalers. In addition, we developed a cloud gaming testbed using the Media-over-QUIC (MoQ) framework. The results from this testbed demonstrated that the proposed decoupled upscaling approach can successfully process and deliver high-quality gaming streams in real time. Our experiments also showed that using upscalers in cloud gaming can save substantial computing resources and/or serve more gaming sessions with the same resources.

REFERENCES

[1] Saeed Anwar, Salman Khan, and Nick Barnes. 2021. A Deep Journey into Super-resolution. *Comput. Surveys* 53, 3 (5 2021), 1–34. https://doi.org/10.1145/3390462

[2] Keith Bugeja, Kurt Debattista, and Sandro Spina. 2019. An asynchronous method for cloud-based rendering. *The Visual Computer* 35 (12 2019), 1827–1840.

[3] James Bulman and Peter Garraghan. 2020. A cloud gaming framework for dynamic graphical rendering towards achieving distributed game engines. In *Proceedings of the 12th USENIX Conference on Hot Topics in Cloud Computing (HotCloud)*. 1.

[4] CapFrameX. 2024. CapFrameX: Frametimes Capture and Analysis Tool. https://www.capframex.com/. Accessed 2024-08-23.

[5] Sharon Choy, Bernard Wong, Gwendal Simon, and Catherine Rosenberg. 2012. The brewing storm in cloud gaming: A measurement study on cloud to end-user latency. In *Proceedings of the 11th Annual Workshop on Network and Systems Support for Games (NetGames)*. 1–6.

[6] AMD Corporation. 2024. AMD FidelityFX^{TM} Cauldron Framework. https://gpuopen.com/fidelityfx-cauldron-framework/.

[7] AMD Corporation. 2024. AMD FidelityFX^{TM} Super Resolution (FSR). https://www.amd.com/en/products/graphics/technologies/fidelityfx/super-resolution.html.

[8] AMD Corporation. 2024. AMD FSR GitHub Repository. https://github.com/GPUOpen-LibrariesAndSDKs/FidelityFX-SDK. GitHub repository, Accessed 2024-08-23.

[9] Intel Corporation. 2024. Intel X^{e} Super Sampling (XeSS). https://www.intel.com/content/www/us/en/products/docs/discrete-gpus/arc/technology/xess.html.

[10] NVIDIA Corporation. 2024. NVIDIA Deep Learning Super Sampling (DLSS). https://www.nvidia.com/en-us/geforce/technologies/dlss/.

[11] TingxingTim Dong, Hao Yan, Mayank Parasar, and Raun Krisch. 2022. RenderSR: A Lightweight Super-Resolution Model for Mobile Gaming Upscaling. In *Proceedings of IEEE Conference on Computer Vision and Pattern Recognition (CVPR'22) Workshops*. New Orleans, LA, 3086–3094. https://doi.org/10.1109/CVPRW56347.2022.00348

[12] Open 3D Foundation. 2024. Open 3D Engine. https://o3de.org/.

[13] Epic Games. 2024. Unreal Engine. https://www.unrealengine.com/.

[14] Philippe Graff, Xavier Marchal, Thibault Cholez, Stephane Tuffin, Bertrand Mathieu, and Olivier Festor. 2021. An Analysis of Cloud Gaming Platforms Behavior under Different Network Constraints. In *Proceedings of the 17th International Conference on Network and Service Management (CNSM)*. 551–557. https://doi.org/10.23919/CNSM52442.2021.9615562

[15] Hua-Jun Hong, Fan-Chiang Tao-Ya, Cheng-Hsin Hsu, Kuan-Ta Chen, Chun-Ying Huang, and Cheng-Hsin Hsu. 2014. GPU consolidation for cloud games: Are we there yet?. In *Proceedings of the 13th Annual Workshop on Network and Systems Support for Games (NetGames)*. 1–6. https://doi.org/10.1109/NetGames.2014.7008969

[16] IETF. 2024. Media over QUIC (MoQ). https://datatracker.ietf.org/group/moq/about/.

[17] Youngjin Kim, Yubin Choi, Young Choon Lee, Hyuck Han, and Sooyong Kang. 2022. E-Render: Enabling UHD-Quality Cloud Gaming Through Edge Rendering. *IEEE Access* 10 (7 2022), 72107–72119.

[18] kixelated. 2023. moq-rs: Rust library for Media over QUIC. https://github.com/kixelated/moq-rs. GitHub repository, accessed 2024-08-23.

[19] Li Lin, Xiaofei Liao, Guang Tan, Hai Jin, Xiaobin Yang, Wei Zhang, and Bo Li. 2014. LiveRender: A Cloud Gaming System Based on Compressed Graphics Streaming. In *Proceedings of the 22nd ACM International Conference on Multimedia (MM)*. 347–356.

[20] Martin Malik. 2024. HWiNFO: System Information and Diagnostics Tool. https://www.hwinfo.com/.

[21] Ioannis Pantazopoulos and Spyros Tzafestas. 2002. Occlusion culling algorithms: A comprehensive survey. *Journal of Intelligent and Robotic Systems: Theory and Applications* 35, 2 (2002). https://doi.org/10.1023/A:1021175220384

[22] Giacomo Parolini, Dario Maggiorini, Davide Gadia, and Laura Anna Ripamonti. 2022. Distributed Rendering for Video Games via Object Streaming. In *Proceedings of the IEEE 42nd International Conference on Distributed Computing Systems Workshops (ICDCSW)*. Bologna, 209–214.

[23] PCGuide. 2024. Our graphics settings guide for Star Wars Outlaws on PC. https://www.pcguide.com/software/guide/best-graphics-settings-for-star-wars-outlaws/. Accessed 2024-08-23.

[24] Jon Peddie. 2019. *Ray Tracing: A Tool for All*. Springer.

[25] Zhengwei Qi, Jianguo Yao, Chao Zhang, Miao Yu, Zhizhou Yang, and Haibing Guan. 2014. VGRIS: Virtualized GPU Resource Isolation and Scheduling in Cloud Gaming. *ACM Transactions on Architecture and Code Optimization* 11, 2 (6 2014), 1–25. https://doi.org/10.1145/2632216

[26] Giampaolo Rodola. 2024. Psutil. https://github.com/giampaolo/psutil. GitHub repository, accessed 2024-08-23.

[27] Ryan Shea, Di Fu, and Jiangchuan Liu. 2015. Cloud Gaming: Understanding the Support From Advanced Virtualization and Hardware. *IEEE Transactions on Circuits and Systems for Video Technology* 25, 12 (12 2015), 2026–2037. https://doi.org/10.1109/TCSVT.2015.2450172

[28] Xinkun Tang, Ying Xu, Feng Ouyang, Ligu Zhu, and Bo Peng. 2023. A Cloud-Edge Collaborative Gaming Framework Using AI-Powered Foveated Rendering and Super Resolution. *International Journal on Semantic Web and Information Systems (IJSWIS)* 19, 1 (4 2023), 1–19.

[29] Unity Technologies. 2024. Unity Engine. https://unity.com/products/unity-engine.

[30] Deniz Ugur. 2024. TransparentSR. https://github.com/DenizUgur/TransparentSR. GitHub repository, accessed 2024-08-23.

[31] Z. Wang, A.C. Bovik, H.R. Sheikh, and E.P. Simoncelli. 2004. Image Quality Assessment: From Error Visibility to Structural Similarity. *IEEE Transactions on Image Processing* 13, 4 (4 2004), 600–612. https://doi.org/10.1109/TIP.2003.819861

[32] Songyin Wu, Sungye Kim, Zheng Zeng, Deepak Vembar, Sangeeta Jha, Anton Kaplanyan, and Ling-Qi Yan. 2023. ExtraSS: A Framework for Joint Spatial Super Sampling and Frame Extrapolation. In *Proceedings of ACM SIGGRAPH*. New York, NY, USA, 1–11. https://doi.org/10.1145/3610548.3618224

[33] Lei Xiao, Salah Nouri, Matt Chapman, Alexander Fix, Douglas Lanman, and Anton Kaplanyan. 2020. Neural supersampling for real-time rendering. *ACM Transactions on Graphics* 39, 4 (7 2020). https://doi.org/10.1145/3386569.3392376

[34] Sipeng Yang, Qingchuan Zhu, Junhao Zhuge, Qiang Qiu, Chen Li, Yuzhong Yan, Huihui Xu, Ling-Qi Yan, and Xiaogang Jin. 2024. Mob-FGSR: Frame Generation and Super Resolution for Mobile Real-Time Rendering. In *Proceedings of the ACM SIGGRAPH 2024 Conference Papers*. 1–11.

[35] Xu Zhang, Hao Chen, Yangchao Zhao, Zhan Ma, Yiling Xu, Haojun Huang, Hao Yin, and Dapeng Oliver Wu. 2019. Improving Cloud Gaming Experience through Mobile Edge Computing. *IEEE Wireless Communications* 26, 4 (4 2019), 178–183.

[36] Zhi Li, Anne Aaron, Ioannis Katsavounidis, Anush Moorthy, and Megha Manohara. 2016. Toward A Practical Perceptual Video Quality Metric.



TVMC: Time-Varying Mesh Compression Using Volume-Tracked Reference Meshes

Guodong Chen*
Northeastern University
Boston, MA, USA
chen.guod@northeastern.edu

Filip Hácha
University of West Bohemia
Plzeň, Czech Republic
hachaf@kiv.zcu.cz

Libor Váša
University of West Bohemia
Plzeň, Czech Republic
lvasa@kiv.zcu.cz

Mallesham Dasari
Northeastern University
Boston, MA, USA
m.dasari@northeastern.edu

Abstract

Time-varying meshes (TVMs), characterized by their varying connectivity and number of vertices, hold significant potential in AR/VR applications. However, their practical use is challenging due to their large file sizes and the complexity of time-varying topology. Many time-varying mesh compression methods attempted to exploit redundancy between consecutive meshes to compress TVMs more efficiently, however, most face difficulties in establishing stable vertex and surface correspondence between the frames of a TVM. We propose TVMC, a novel TVM compression method that leverages volume tracking and extracts high-quality reference meshes for inter-frame prediction. Specifically, we use as-rigid-as-possible volume tracking to align consecutive TVMs and track volume centers, followed by multidimensional scaling to refine reference centers. This allows us to precisely deform a group of frames to the reference space and extract the reference mesh which is then deformed to approximate each mesh in the group to get displacement fields for TVM compression. Extensive experiments show that TVMC outperforms state-of-the-art methods (e.g., Google Draco, V-DMC 4.0, etc.), with bitrates of 4-6 Mbps compared to 9-12 Mbps for Draco and 10-15 Mbps for V-DMC 4.0. It reduces the decoding time by 66.1% compared to Draco and enables an increased group of frames (up to 15) without significant distortion.

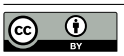
CCS Concepts

• **Computing methodologies** → **Computer graphics**.

Keywords

Time-varying mesh, volume tracking, mesh deformation, inter-surface mapping, mesh compression

ACM Reference Format:
Guodong Chen, Filip Hácha, Libor Váša, and Mallesham Dasari. 2025. TVMC: Time-Varying Mesh Compression Using Volume-Tracked Reference Meshes. In *ACM Multimedia Systems Conference 2025 (MMSys '25), March 31-April 4, 2025, Stellenbosch, South Africa.* ACM, New York, NY, USA, 11 pages. https://doi.org/10.1145/3712676.3714440



This work is licensed under a Creative Commons Attribution 4.0 International License.
MMSys '25, Stellenbosch, South Africa
© 2025 Copyright held by the owner/author(s).
ACM ISBN 979-8-4007-1467-2/2025/03
https://doi.org/10.1145/3712676.3714440

1 Introduction

With the rapid development of 3D sensing technologies [20], it has become increasingly feasible to produce complex 3D content with a high level of detail and quality. One of the most common and efficient ways of 3D content representation is triangle mesh sequences with changing topology and connectivity, also called time-varying meshes (TVMs). Differing from meshes in a 3D animation, time-varying meshes consist of a set of mesh frames that include time-varying topology, vertex numbers, vertex positions, and face connectivity, and each of them demands massive geometry and texture information to render high-quality visuals. Because of this, TVMs require a substantial amount of data for storage and transmission, necessitating an efficient compression method, as real-time streaming is impractical without using advanced networks [13], which are not available today.

Popular mesh compression tools, such as Google Draco [11], using methods such as TFAN [28] or EdgeBreaker[36], perform well in terms of compression ratio and latency by compressing each mesh *individually*. Each mesh in the sequence is treated as a separate entity during the compression process, so there is no need to manage dependencies or relationships between meshes. This feature provides Draco flexibility in handling different types of meshes; however, it does not exploit the inter-frame redundancy across the sequence of mesh frames and results in very high data rates for high-quality complex mesh sequences.

To improve mesh compression ratio, many other works have attempted to utilize spatial and temporal correlation to encode and decode meshes more efficiently. Some existing methods [1, 24, 26, 39] focus on dynamic mesh sequences with consistent topology across different frames, leveraging temporal correlation to improve compression performance. However, these approaches require the meshes to maintain a consistent topology and connectivity, similar to 3D animation, which limits their applicability to mesh sequence compression in real-world applications.

The Moving Picture Experts Group (MPEG) has developed several mesh compression standards, including IC (Interpolator Compression) [5], MESHGRID [37], and FAMC (Frame-based Animated Mesh Compression) [27]. Similarly, these standards are limited to processing dynamic mesh sequences with constant connectivity and cannot handle TVMs with changing topology. To address this issue, the MPEG 3D Graphics Coding (3DG) group recently issued a Call for Proposals (CfP) [30] for the Video-based Dynamic Mesh

Coding (V-DMC) standard. In response, Apple Inc. proposed video and subdivision-based mesh coding (VSMC) [26], which was selected as the foundation for the V-DMC standard. However, the temporally consistent re-meshing process required by VSMC is not always feasible, which imposes constraints on TVMs. Thus, efficient and effective compression of TVMs remains an open challenge.

Recently, embedded deformation has been used in 3D mesh compression [17, 21]. TVMs lack explicit vertex correspondence between frames and usually have varying connectivity and numbers of vertices over time, so establishing inter-frame correspondence and representing inter-frame differences is challenging. Embedded deformation addresses this problem by deforming a selected keyframe into the target frame of similar shape while preserving its topology. However, these approaches face fundamental limitations when self-contact regions (Section 3.3) exist in the keyframe, often resulting in severe visual artifacts and inaccuracies in decoded meshes.

To address the above limitations, we propose TVMC, a novel time-varying mesh compression method using volume-tracked reference meshes. TVMC not only exploits inter-frame redundancy to achieve a high compression ratio but also significantly reduces the deformation artifacts and distortions caused by self-contact regions. Specifically, we first adopt ARAP volume tracking to identify volume centers' positions within each mesh frame and the largest distances between each other during the whole sequence, along with a measurement of how tightly each pair of centers is bound together called *affinity* (Section 3.1). These affinities are then processed using Multidimensional Scaling (MDS) to derive a new set of centers called *reference centers* existing in reference space (Section 3.2). With the stable and accurate centers correspondence, we adopt center affinity deformation that can separate self-contact components, which is hard to achieve via traditional Radial Basis Functions (RBFs) mapping or key-vertices-based embedded deformation (Section 3.3). The affinity and Euclidean distance between centers is employed to calculate the *mesh self-contact degree* (Section 3.4). We deform meshes with a low self-contact degree to the reference space and use Poisson surface reconstruction to extract the volume-tracked reference mesh without self-contact regions (Section 3.5). Finally, we use center affinity deformation again to deform the reference mesh to approximate any mesh in the TVMs and compute displacements using subdivision surface fitting (Section 3.6). We show that our TVMC method outperforms Google Draco [11], V-DMC 4.0 [32], and recent embedded deformation-based methods such as [21] in terms of geometry compression ratio and decoding time. We highlight TVMC's performance from our evaluation as follows:

- TVMC achieves higher geometry compression rates compared to the state-of-the-art methods Draco and V-DMC 4.0, at a framerate of 30 fps and a D2-PSNR of around 80, with bitrates of 4 to 6 Mbps vs. 9 to 12 Mbps for Draco and 10 to 15 Mbps for V-DMC 4.0, respectively.
- TVMC supports real-time applications, reducing the decoding time by 66.1% compared to Draco.
- TVMC significantly increases the number of frames in a group of frames (GoF) (GoF = 15 vs. GoF = 5 in [21]) that can be compressed together without leading to rapid accumulated distortion.

2 Related Work and Limitations

There have been numerous works focused on 3D mesh compression [25, 34]. Based on the constraints of mesh sequence, we roughly divide them into static mesh compression [11, 28], dynamic mesh compression [1, 24, 26, 39], and time-varying mesh compression [7, 12, 15–19, 21, 29, 40]. Static mesh compression treats each mesh independently, it utilizes intra-redundancy to compress a single mesh. On the other hand, dynamic mesh compression compresses a sequence of meshes by storing a base mesh or key mesh along with a displacement field, which is feasible since dynamic meshes have a constant number of vertices and a constant connectivity. In contrast, time-varying mesh compression deals with mesh sequences where the geometry information changes and the topology and connectivity also vary, making it a more complex problem to solve.

2.1 Static Mesh Compression

Static mesh compression involves compressing vertices' positions, connectivity, textures, and other attributes. A prominent algorithm, Edgebreaker [36], developed by Rossignac in 1999, offers an efficient approach for manifold 3D mesh compression. In the Edgebreaker algorithm, each new triangle is adjacent to an already encoded one, enabling efficient compression of vertex positions and attributes such as normals. Using parallelogram prediction, Edgebreaker stores the difference between predicted and actual values, typically small, rather than absolute values. Using Edgebreaker, the state-of-the-art mesh compression tool, Draco [11], balances simplicity, efficiency, and performance well. However, Draco's limitation is that it compresses each mesh independently, which prevents it from efficiently compressing TVMs. This means it cannot utilize the temporal correlations that may be present across multiple meshes.

2.2 Dynamic Mesh Compression

Dynamic mesh is a sequence of static meshes with the same topology and connectivity that describes a temporal development of some physical surface, a common representation format in 3D animation. In 2021, the MPEG 3DG group issued a Call for Proposals on dynamic mesh coding [30], aiming to develop a new standard under the Visual Volumetric Video-Based Coding (V3C) standard, known as the V-DMC standard. Then the VSMC scheme proposed by Apple [26] was selected as the basis for the V-DMC standard. V-DMC uses inter-frame coding for dynamic meshes, representing a mesh with a decimated base mesh and a set of displacements generated from subdivision surface fitting. It estimates a motion field by tracking corresponding vertices between the current mesh and the reference mesh. The reference base mesh is then deformed to approximate the current base mesh, and ultimately subdivided and adjusted to fit the current mesh to get updated displacements for reconstruction.

However, to leverage temporal correlation, V-DMC requires the input mesh sequence to maintain a constant topology and connectivity. Although VSMC proposed a time-consistent re-meshing method to relax the constraint on mesh types, it is not always feasible, making the representation of inter-frame differences between the frames of a TVM still challenging.

2.3 Time-varying Mesh Compression

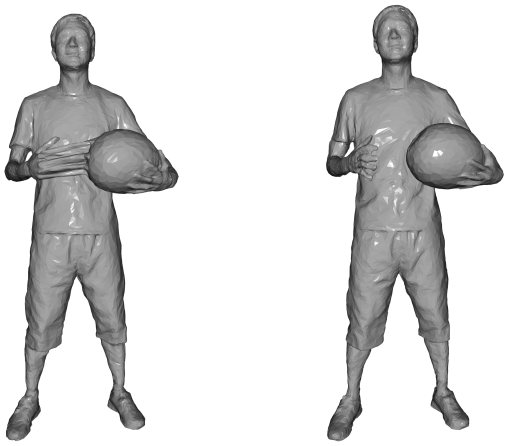
Modern 3D reconstruction techniques use a combination of RGB-D cameras, infrared (IR) cameras, and structured light sources to generate detailed 3D meshes. These meshes are typically time-varying, with different numbers of vertices, connectivity, and topology in each frame. So, how to process and utilize them efficiently is a major focus of research in this field.

Early work [41] used the first mesh frame in a given sequence as the 'reference mesh'. This first mesh is converted into a semi-regular normal mesh, followed by motion estimation to map it onto the following meshes. However, it accumulates errors frame by frame and brings significant artifacts and distortions.

In 2015, Collet et al. [6] estimated a feasibility score of each frame being a keyframe. They selected a relatively small set of meshes as keyframes and enforced that each keyframe covers a continuous frame range. In their keyframe choosing algorithm, they used larger surface area, lower-genus surface, and more connected components as the standard of identifying keyframe, which solved the self-contact problem to a certain extent. However, since they used a non-rigid ICP algorithm [23] to fit a keyframe to its neighboring frames, the fitting error is unavoidable with only vertex information. Furthermore, the algorithm repeats this process on the remaining frames until every frame is associated with a keyframe, which is computationally expensive.

Some other methods use embedded deformation [38] to deform selected reference mesh to each frame while maintaining topology. Based on the movements of strategically selected key points, a geometric transformation operator is applied to each vertex in the previous mesh to 'predict' the following meshes.

Hoang et al. [17] proposed an optimization-based technique to identify the optimal number and positions of key nodes for embedded mesh deformation. This approach minimizes the number of nodes required while ensuring relatively high-quality reconstruction. Similarly, Jin et al. from the KDDI Group were inspired by the embedded deformation proposed in [38] to construct an embedded graph representation for performing difference mapping between meshes [21]. However, these methods apply transformations to every vertex in the reference mesh based on merely geometry information, which can cause significant distortions when the reference mesh contains self-contact regions. Figure 1 illustrates this issue: if a mesh of a person placing their fingers on a basketball is deformed to a subsequent frame where the fingers and the basketball are separate, the vertices that were temporarily missing due to self-contact will also be missing in the deformed mesh, which leads to severe visual distortions where fingers still stick to the basketball (shown in Figure 1a).



(a) With self-contact regions **(b) Without self-contact regions**

Figure 1: Impact of self-contact regions on deformation quality. Both meshes are deformed to the same target mesh. For Figure 1a, the fingers touch the basketball, leading to deformation distortions caused by self-contact regions. For Figure 1b, the fingers and basketball are separate, resulting in a more artifact-free deformation.

3 TVMC Design

To address the challenges mentioned above, we propose a volume-tracked reference mesh-based time-varying mesh compression framework called TVMC. We use volume centers to build stable center correspondence rather than relying on traditional surface and vertices correspondence. We also adopt center affinity-based mesh deformation to deform a mesh by controlling its volume centers, which we demonstrate is more accurate than embedded deformation through key point control. TVMC creates a general, self-contact-free, volume-tracked reference mesh, significantly reducing deformation distortions caused by the self-contact regions. Extracted volume-tracked reference mesh can serve as the basis of many mesh compression methods that require a reference mesh, and center affinity-based mesh deformation can be a superior alternative to embedded deformation.

Figure 2 shows the simplified workflow of TVMC. The process begins with feeding a TVM sequence to As-Rigid-As-Possible (ARAP) volume tracking [9], which uniformly distributes a set of centers within the enclosed volume of each mesh frame. A Global Optimization step [8] is optional to improve the quality of the set of volume centers by eliminating several abnormal centers. The final tracked centers are then processed using Multidimensional Scaling (MDS) [4] to derive a set of reference centers that serve as the basis for subsequent operations. The input mesh sequence is then fed into Affinity Evaluation Module to calculate the self-contact degrees. We employ Mesh Editing [14] to deform meshes with a low self-contact degree to the reference space around reference centers by utilizing center affinity. Then the Reference Mesh Extraction Module adopts subdivision surface fitting to reduce the deformation deviation and generates a volume-tracked reference mesh through Poisson surface reconstruction [22]. This volume-tracked reference mesh is then deformed to approximate each mesh in the group while maintaining its topology and connectivity. Finally, we use the Subdivision Surface Fitting Module again to compute displacements between each reconstructed mesh and the volume-tracked reference mesh, enabling us to reconstruct each frame using the volume-tracked reference mesh and displacements.

3.1 As-Rigid-As-Possible Volume Tracking

Consider a TVM sequence, $S = \{M_0, M_1, \ldots, M_{N-1}\}$, where N is the total number of frames in the sequence. Let $M(t) = (V_t, E_t, F_t)$ be a static mesh at time t. V_t, E_t, and F_t represent the sets of vertices,

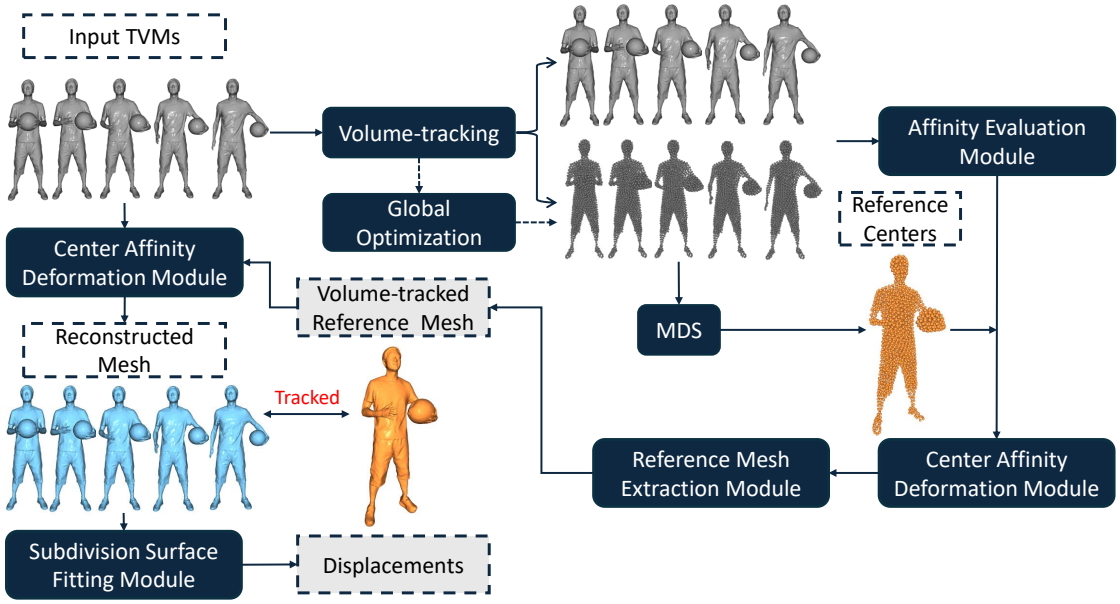


Figure 2: Simplified workflow of TVMC. TVMC consists of several key steps represented by colored boxes. The dotted boxes indicate the outputs of specific operations, while the grayed dotted boxes highlight the only two elements requiring compression. Gray meshes are the input TVMs, accompanied by their corresponding gray centers. The reference centers are shown in orange, along with the orange reference mesh, which represents the volume-tracked reference mesh. This reference mesh is deformed to the blue reconstructed meshes. The term "tracked" refers specifically to only changes in vertex positions, while maintaining the same connectivity and topology.

edges, and faces (triangles), respectively. We apply As-Rigid-As-Possible Volume Tracking and set the mode to *max affinity based tracking*, as proposed in [8], to the TVM sequence S. The method identifies a configurable number of K points set C, referred to as centers. Each center represents a small volume surrounding it with positions that vary over time. Each center follows a certain trajectory $c_i = [c_i(0), c_i(1), \ldots, c_i(N-1)], c_i(f) \in \mathbb{R}^3$, where $c_i(f)$ is the position of the i-th center in the f-th frame. For any frame f, we define $C_f = \{c_1(f), c_2(f), \ldots, c_K(f)\}$ as the set of all centers' positions in that frame. Briefly, the uniformly distributed centers are generated as follows. First, after setting a resolution for the longest axis, each mesh is converted to a tight regular square voxel grid. Then the fast winding number [2] is used to define the so-called indicator function (IF), identifying which voxels of the grid are enclosed by the mesh using the following equation:

$$IF(x) = \begin{cases} 1 & \omega_f(x) > \mu \\ 0 & \omega_f(x) \leq \mu \end{cases}, \tag{1}$$

where $\omega_f(x)$ represents the fast winding number and μ is a threshold. In this paper, we set $\mu = 0.5$. The threshold of fast winding number provides some robustness to non-watertight meshes so TVMC does not strictly require fully watertight meshes. Enhancing the fast winding number algorithm and distinguishing open surfaces from relatively enclosed objects before volume tracking can improve TVMC's performance when handling inputs with open surfaces. Next, the algorithm samples K random voxels with $IF(x) = 1$ and denotes them centers. Each of the remaining voxels is then connected with its nearest center, and every center is moved to its surrounding neighbors' centroid iteratively. This approach ensures uniform distribution of the selected centers in the first frame. For each subsequent frame, the set of centers from the previous frame will first be linearly extrapolated, then an optimization energy function is minimized to ensure uniformity and smoothness of the next sets of centers.

We refer the reader to the original paper for a detailed description of the As-Rigid-As-Possible Volume Tracking method [8, 9].

3.2 Reference Centers Generation: Multidimensional Scaling

Next, we generate reference centers as the basis of the volume-tracked reference mesh. Reference centers are a specialized set of volume centers located in the reference space. Similar to C, they represent the area inside the mesh surface, but the reference mesh they represent is adjusted to a self-contact-free shape. In order to get relative positions of centers without self-contact, we first build a distance matrix from the largest distances of centers. Let d_{ij} be the largest distance, a_{ij} be the *affinity* between two centers c_i and c_j across the whole TVM. Here, affinity is a function of the maximum distance between the centers, representing how strongly two entities are bound together. The largest distance and *affinity* between two centers c_i and c_j is calculated using the Euclidean distance:

$$d_{ij} = \max_f \|c_i(f) - c_j(f)\|, \tag{2}$$

$$a_{ij} = \exp(-(\alpha d_{ij})^2), \tag{3}$$

where $f \in [0, N-1]$ and α is a weight decided by the global scale of the input TVM. The main idea behind maximum-distance-based affinities is that if two centers are far enough apart in any part of the sequence, they probably are not related to each other in the rest of the sequence, no matter how close they may get. Such centers should then not affect the deformation of the same part of the surface.

The largest distance matrix D is then defined as $D = [d_{ij}]_{K\times K}$. Once the distance matrix D is computed, it is used as the input of Multidimensional Scaling (MDS). MDS seeks to find a set of points $r_1, r_2, \ldots, r_K$ in a 3-dimensional space that minimizes the stress function given by:

$$\min \sigma(R) = \sqrt{\frac{\sum_{i<j} \left(d_{ij} - \|r_i - r_j\|\right)^2}{\sum_{i<j} d_{ij}^2}}. \tag{4}$$

Here, $R = [r_1, r_2, \ldots, r_K]$ represents the reference centers we want in the 3-dimensional space and $\|r_i - r_j\|$ is the Euclidean distance between points r_i and r_j in the set of reference centers R. In R, connected centers should remain contiguous, while those that are, in fact, disconnected should be apart since we enforce the maximum distance over the length of the sequence. Finally, we perform a rigid registration of R that aligns R with C_f so that the volume-tracked reference mesh retains as many static regions as possible from the original meshes in the group. We use Singular Value Decomposition (SVD) to do this and call the space where the reference centers are located the reference space.

3.3 Self-contact Degree Evaluation

As mentioned in Section 2.3, deformations in self-contact regions can lead to severe distortions. To create a high-quality volume-tracked reference mesh, it is essential to gather as comprehensive surface information as possible and minimize the potential impact caused by self-contact regions. To this end, we exclude a certain number of the meshes in the group during the creation of the volume-tracked reference mesh, which allows us to increase the group size and improve the overall performance of TVMC.

Simply relying on the variation of the number of vertices, triangles, or vertex positions in TVMs to identify self-contact regions is not feasible. By utilizing center trajectories and affinity, we can estimate the likelihood of the presence of self-contact regions. Centers that are tightly bound together have a large affinity between them, while the further they move apart, the lower the affinity between them will be. Therefore, a pair of centers with low affinity but also small Euclidean distance suggests that these two centers are only temporarily close, which indicates an increased likelihood of self-contact region appearance.

Motivated by this observation, we calculate the mesh self-contact degree $SD = [sd_0, sd_1, \ldots, sd(N-1)]$ using the following formula that adds up the product of affinity between all centers and their current Euclidean distance:

$$sd_f = -\sum_{i<j} (a_{ij} \cdot \|c_i(f) - c_j(f)\|). \tag{5}$$

The smaller the self-contact degree is, the more likely it is that the self-contact regions exist.

3.4 Center Affinity Deformation

For any frame f, we have C_f that defines the set of all centers' positions in f. With the set of reference centers R, it is possible to construct a transformation $\hat{T_{i,f}}$ for each center c_i from the space of frame f to the reference space. Firstly, we introduce the concept of embedded deformation [38] introduced in [17, 21], which allows mesh deformation while preserving the natural properties of the original meshes to a certain extent. We then provide a detailed description of applying embedded deformation with center information and incorporate center affinity to perform center affinity-based deformation.

In [17], a set of key points $KP = [p_0, p_1, \ldots, p_{P-1}] \in \mathbb{R}^3$ is generated to control the areas around each key point, where P is the number of key points. Then a key point matching or registration method is applied to get key point correspondence between frame $f-1$ and frame f. For each vertex v_i in frame $f-1$, its position is updated according to the movements of its K nearest key points using the following equation:

$$v_i' = \sum_{j=0}^{K-1} w_{ij}(R_j(v_i - p_j) + T_j + p_j), \tag{6}$$

where w_{ij} is a distance-dependent influence weight. R_j and T_j are determined by solving:

$$\arg\min_{R_j,T_j}(E_{geometry} + \alpha E_{rotation} + \beta E_{smoothness}), \tag{7}$$

where $E_{geometry}$, $E_{rotation}$, and $E_{smoothness}$ penalize geometric error, rotation error, and smoothness error, respectively, and α and β are two weights. In [21], a mesh is first decimated into a base mesh. The vertices of the base mesh take place of the key points and the remaining steps follow a similar approach.

However, these embedded deformation methods update a vertex's position entirely depending on spatial Euclidean distance. The process of choosing K nearest key points would unavoidably choose key points that are close in terms of Euclidean distance but belong to different parts of the mesh. To address this issue, we adopt an improved deformation method called center affinity deformation. Center affinity deformation respects the topology by relying on center affinity instead of Euclidean distance to select the most suitable centers. The mesh deformation will significantly align with the transformation of its centers, while maintaining its topology.

In terms of transformation $\hat{T_{i,f}}$, we use dual quaternions to represent rigid transformations. We can represent a dual quaternion as $A + \epsilon B$, where A and B are ordinary quaternions and ϵ is the dual unit, which satisfies $\epsilon^2 = 0$. An ordinary quaternion can be represented as $a + b\mathbf{i} + c\mathbf{j} + d\mathbf{k}$, which satisfies $\mathbf{i}^2 = \mathbf{j}^2 = \mathbf{k}^2 = \mathbf{ijk} = -1$. Leveraging the properties of dual quaternions, we can describe a transformation with a rotation quaternion R and a transformation vector $T = (x, y, z)$ in the format of $A = R + (x/2, y/2, z/2, 0)\epsilon$. We use dual quaternions because they are easy to blend and inverse, which makes them convenient and efficient for center-affine deformations.

Specifically, for each vertex v_i in frame f, its nearest center $c_{(1),f}$ is determined as:

$$c_{(1),f} = \arg\min_{c_{(1),f}} \|c_{(1),f} - v_i\|. \tag{8}$$

We assume that the nearest center $c_{(1),f}$ can best reflect the motion trajectory of the vertex v_i. Then we find the next $K-1$ 'nearest' centers with the highest affinity to the center $c_{(1),f}$. The blended transformation for updating vertices can be expressed as:

$$T_i = \frac{\sum_{j=1}^{k} w_j \cdot \hat{T_{j,f}}}{\sum_{j=1}^{k} w_j}. \tag{9}$$

With the normalization of dual quaternion, the vertex v_i can be updated to:

$$v_i' = \frac{T_i}{\|T_i\|}(v_i). \tag{10}$$

Based on a smooth minimum function called Boltzmann operator [3], we get w_j from the following equations:

$$d_j = \|c_{(j),f} - v_i\|, j \in 1, 2, \ldots, K, \tag{11}$$

$$w_j' \approx \frac{\exp(\frac{-d_j}{\sigma d_1 + \epsilon})}{\sum_{k=1}^{K} \exp(\frac{-d_k}{\sigma d_1 + \epsilon})}, \tag{12}$$

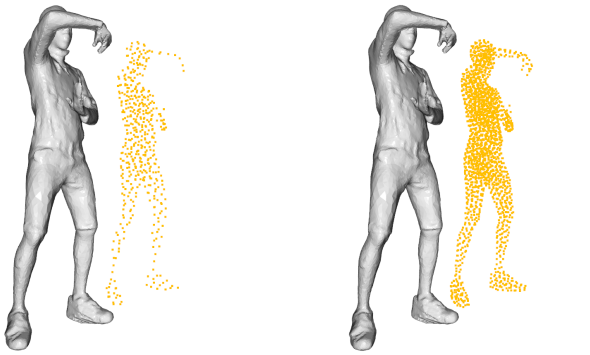
$$w_j = \frac{w_j' - \min_x(w_x')}{\sum_{k=1}^{K}(w_k' - \min_x(w_x'))}. \tag{13}$$

Distances d_j represent the distance between the j-th affine center to v_i, a small value ϵ is in the denominator to replace the calculation of a limit, thus simplifying the evaluation. σ controls the smoothness of the weights. After the normalization step in Equation 13, we obtain a set of weights that sum to one, which keeps all the centers inside the mesh surface during the deformation if the surface is sufficiently finely sampled.

In contrast to the original embedded deformation weights, the weights calculated in this way satisfy the interpolation condition, i.e., if the position of the deformed point v is the same as the position of the center c_i, then its weight $w_i = 1$ and the weights of the other centers $w_j = 0 \forall j \neq i$. For a more detailed description and other applications of center affinity deformation, we refer the reader to [14].

3.5 Volume-Tracked Reference Mesh Extraction

To improve the overall performance, TVMC divides the frames of the input TVM into several groups of frames and each group contains *GoF* meshes. As mentioned in Section 2.3, deformation in self-contact regions often leads to noticeable distortions, which can negatively impact the results of center affinity deformation and, consequently, the quality of the volume-tracked reference mesh. So, for each group, we drop $GoF/2$ meshes with a high possibility of containing self-contact regions by calculating their self-contact degrees introduced in Section 3.3, and adopt center affinity deformation to deform all the remaining frames to the reference space. Although our center affinity deformation method can deform meshes with different shapes to the same reference space with high accuracy, slight surface offsets exist between these deformed meshes. So, we employ subdivision surface fitting to improve the quality of our extracted volume reference mesh.



(a) ARAP volume tracking with 500 centers

(b) ARAP volume tracking with 2000 centers

Figure 3: Comparison of different numbers of centers. The orange points represent the volume centers of the gray mesh. In Figure 3b, 2000 centers capture the human body more accurately compared to the 500 centers shown in Figure 3a.

The subdivision surface fitting module first simplifies a mesh using mesh decimation methods, in this paper we use Quadric Error Metric (QEM) decimation [10]. Selecting the deformed meshes with the least self-contact degree as the key mesh, we feed the key mesh and all the decimated meshes to the subdivision surface fitting module. These decimated meshes are subdivided using one of the subdivision schemes in [35]. Here, we adopt the mid-point subdivision scheme for the entire process as it is computationally efficient and preserves the original mesh's topology.

Then, each vertex in these subdivided meshes will be updated to its nearest vertex in the key mesh. We use KDTrees to implement this in this paper. Subsequently, we only keep all the vertices and use this dense vertex set and Poisson surface reconstruction [22] implemented by Open3D [43] to extract a single surface mesh, with a setting of $depth = 9$. Based on the average numbers of vertices and triangles, we use QEM decimation again to simplify this to a certain degree and call this simplified mesh volume-tracked reference mesh. For the group of *GoF* meshes, this volume-tracked reference mesh contains complete components and surfaces to approximate each mesh with the help of center affinity deformation described in Section 3.4.

In TVMC, the quality of the volume-tracked reference mesh and overall compression performance is influenced by the number of centers we generated from ARAP volume tracking. Figure 3 shows the difference in visual results of a mesh from "Levi" sequence with different numbers of centers. Denser centers can better represent finer parts, such as the hands and lower legs, as compared between Figure 3a and Figure 3b, allowing ARAP volume tracking to be more effective. A smaller number of centers typically results in greater distortions when representing narrow regions, reducing the quality of the reference mesh and negatively impacting compression RD performance. However, using a larger number of centers increases computational costs and poses challenges to MDS in generating a high-quality set of reference centers. In this paper, the number of centers is set from 1000 to 4000 for all sequences.

3.6 Time-varying Meshes Compression

TVMC deforms the volume-tracked reference mesh to a geometric approximation f' of the frame f using the positions of reference centers and the positions of the centers C_f of the frame f, and ensures that f' has the same topology as the volume-tracked reference mesh. Therefore it is feasible to use simple displacement coordinates $dis_i \in \mathbb{R}^{n\times 3}, i \in 1, 2, \ldots, GoF$ to represent each frame in the group, where n is the vertex number of the reconstructed meshes.

The volume-tracked reference mesh is compressed using the state-of-the-art mesh compression tool Draco [11] in relatively high quality with settings of $qp = 14, cl = 7$. Displacements are stored in *PointClouds* and we use Draco to compress them with the setting of *pointcloud*. Note that the compression of displacements and the volume reference mesh are separate, any other compression method such as MPEG TMC13 version 20.0 [31] or video coding method can be adopted.

On the decoder side, the volume-tracked reference mesh and displacements are decoded and they are fed into the Reconstruct Module to simply update the vertices positions of the volume-tracked reference mesh while maintaining its connectivity.

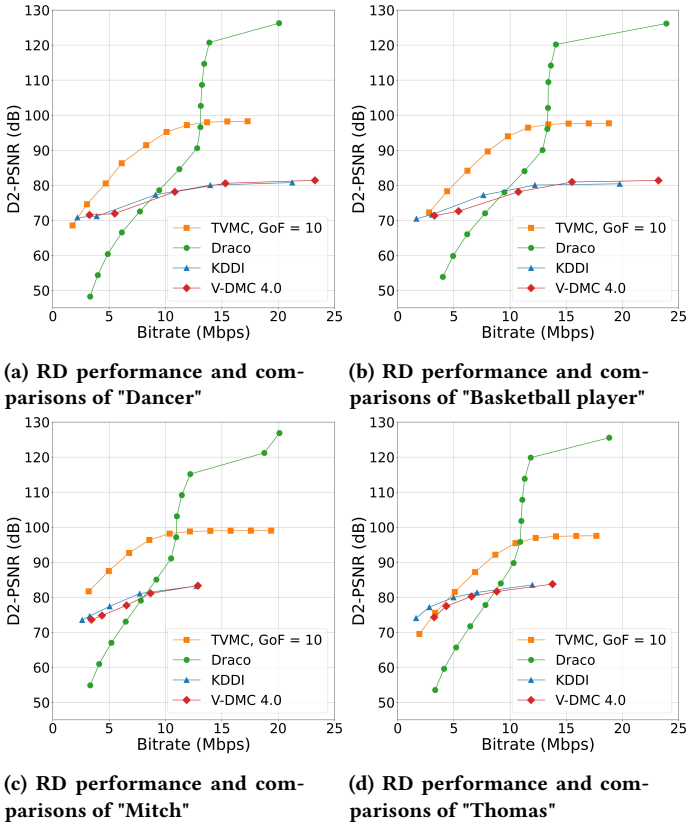
4 Experiments and Analysis

We conducted our experiments on a Lambda Vector equipped with an AMD Ryzen Threadripper 7960X 24-core 4.20 GHz CPU.

4.1 Datasets and Metrics

In this section, we evaluate TVMC on four complex TVM sequences that are defined and used by MPEG V-DMC [30]: "Dancer", "Basketball player", "Mitch", and "Thomas". These 4 TVM sequences, sourced from [42], represent human 3D objects with geometric and texture information. As our focus in this paper is on the compression of geometry information, we exclude the texture information and conduct experiments on 30 frames of each TVM. Specifically, we select the first 30 frames for "Dancer", "Basketball player", and "Mitch", for "Thomas" we select 120-th to 149-th frames because these 30 frames exhibit significant self-contact regions variation. We selected "Basketball Player" and "Thomas" specifically because they feature more self-contact regions, which present additional challenges for compression and can demonstrate TVMC's ability to efficiently deal with self-contact problems.

To evaluate the quality of decoded meshes, we use the state-of-the-art PSNR metrics as defined by the MPEG group [33] along with the root mean square error (RMSE) to quantify the degree of similarity between two sets of geometric data. PSNR metrics are well-established and are widely used in evaluating the performance of mesh compression techniques. In contrast, the structural similarity index measure (SSIM) is designed for 2D image data and is less suitable for evaluating 3D mesh geometry, so we do not consider SSIM in this work. The PSNR metrics include point-to-point error (D1) and point-to-plane error (D2). For 3D meshes, D2-PSNR is generally regarded as a more accurate evaluation metric, as it takes into account the normal vector of the underlying surface when calculating errors. We also evaluate the encoding and decoding time of TVMC and compare it with Draco [11], because our goal is to meet the real-time application requirement of at least 30 fps



(a) RD performance and comparisons of "Dancer"

(b) RD performance and comparisons of "Basketball player"

(c) RD performance and comparisons of "Mitch"

(d) RD performance and comparisons of "Thomas"

Figure 4: Objective RD performance evaluation on 4 TVMs: "Dancer", "Basketball player", "Mitch", and "Thomas". We vary quantization parameter qp between 7 to 16 to generate the curves for TVMC while varying qp between 7 to 19 for Draco to reach a similar range of bitrates. The KDDI and V-DMC 4.0 curves are sourced from [21] for the same dataset.

for decoding. Moreover, we include subjective evaluations to show TVMC's performance.

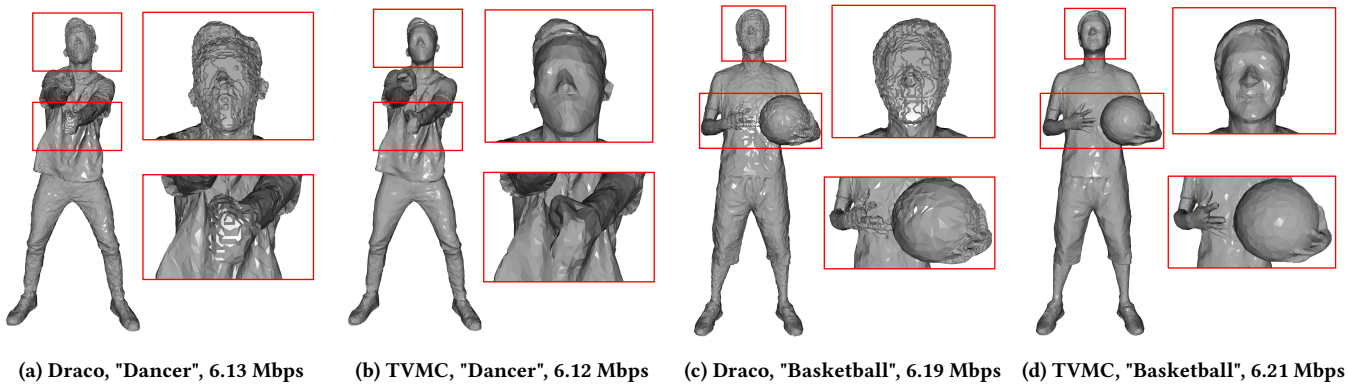
4.2 Experiments Details

When applying ARAP volume tracking for all sequences, we set the mode to *max affinity based tracking* and volume grid resolution to 512. Additionally, we use $eps = 1e-10$ for MDS to generate reliable sets of reference centers. For center affinity deformation, we employ dual quaternions as a representation of rigid transformations. The parameter α mentioned in Section 3.2 for calculating center affinity is in the range from 10^{-2} to 10^{1}. The value of parameter σ for calculating distance-dependent influence weight when adopting center affinity deformation in Section 3.3 is set to 2.

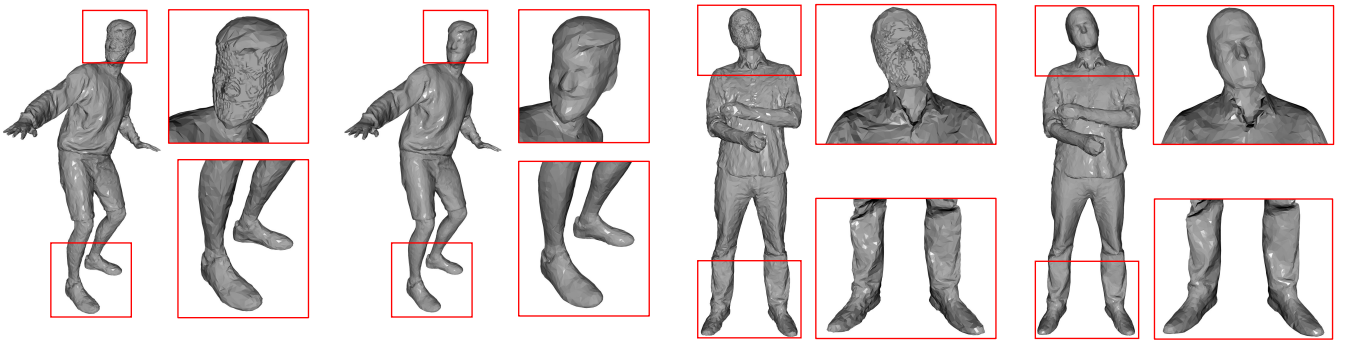
In TVMC, we use Draco to compress our generated volume-tracked reference mesh and displacements. Draco uses the parameter qp to control the level of geometric information quantization. To balance the trade-off between compression efficiency and quality, we set $qp = 14$ for the reference mesh and vary qp from 7 to 16 for displacements to achieve the target bitrates. We avoid $qp \leq 6$ as it introduces unacceptable artifacts.

Table 1: Objective comparisons between TVMC and Draco for the qualitative results shown in Figure 5. D2-PSNR is calculated using the PSNR metrics defined by the MPEG group [33], where higher PSNR values indicate better quality. RMSE values are presented as the log10 of the raw RMSE, smaller number indicates a smaller average difference between the original and reconstructed mesh. Table 1 also includes average encoding and decoding times, measured with a *GoF* setting of 10. Additionally, we illustrate the bitrate proportion allocated to displacements (for *GoF* frames total).

TVMs	TVMC					Draco			
	D2-PSNR	RMSE	Encoder	Decoder	Portion	D2-PSNR	RMSE	Encoder	Decoder
Dancer	84.10	-3.63	13.38 s	15.20 ms	93.08%	66.55	-2.76	181.70 ms	46.20 ms
Basketball player	84.14	-2.65	21.12 s	14.60 ms	93.14%	66.01	-1.74	229.27 ms	47.07 ms
Mitch	87.46	-3.78	10.65 s	13.40 ms	93.36%	66.99	-2.76	139.33 ms	35.40 ms
Thomas	81.50	-3.54	10.88 s	12.60 ms	93.52%	65.65	-2.75	137.93 ms	35.60 ms



(a) Draco, "Dancer", 6.13 Mbps **(b) TVMC, "Dancer", 6.12 Mbps** **(c) Draco, "Basketball", 6.19 Mbps** **(d) TVMC, "Basketball", 6.21 Mbps**



(e) Draco, "Mitch", 5.20 Mbps **(f) TVMC, "Mitch", 4.97 Mbps** **(g) Draco, "Thomas", 5.20 Mbps** **(h) TVMC, "Thomas", 5.09 Mbps**

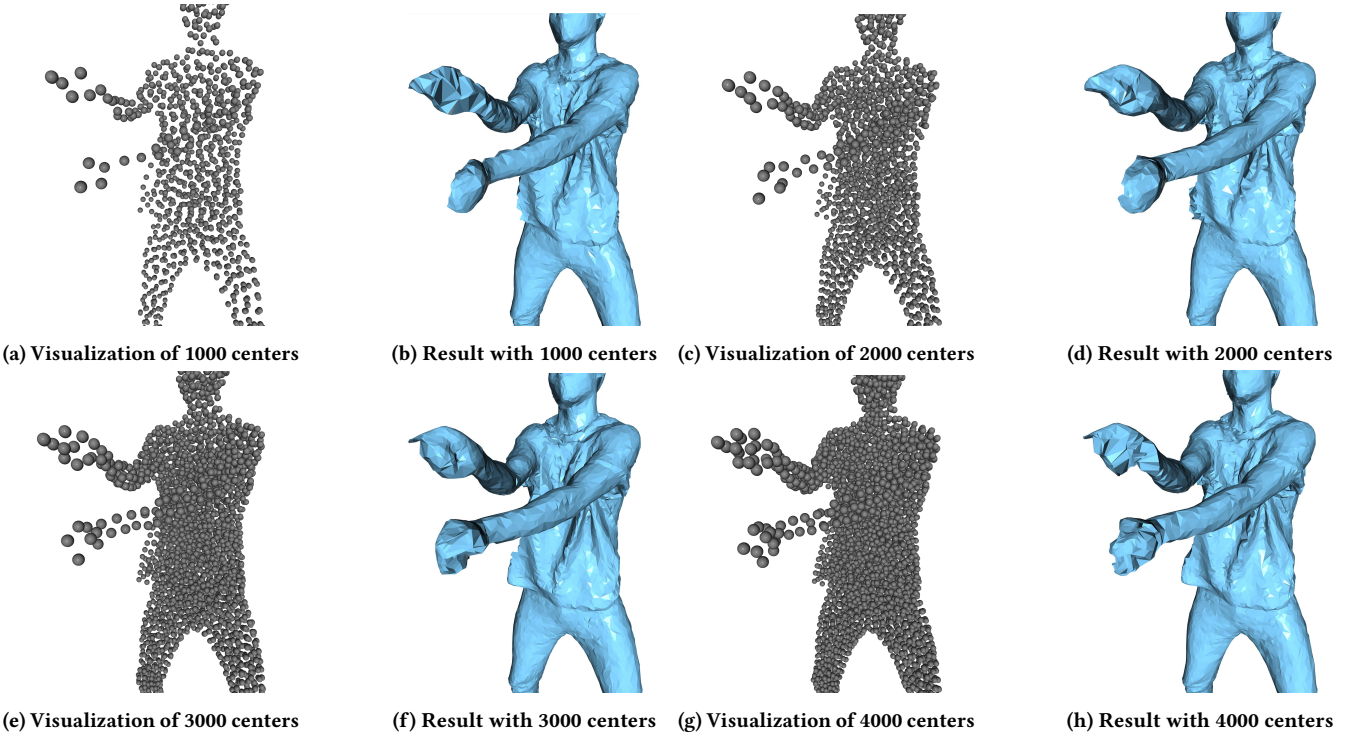
Figure 5: Subjective evaluation of "Dancer", "Basketball player", "Mitch", and "Thomas". Results using Draco and TVMC are compared at bitrates between about 5 and 6. Key differences in the face, hands, and feet are highlighted and zoomed within the red frames for closer comparison. Specific bitrates for each result are indicated in the captions of each subfigure.

4.3 Experiments Results

In TVMC, the number of centers (Section 4.3.2) and Group of Frames length (GoF) (Section 4.3.3) are two main parameters. We show full experiment results of RD performance for each TVM and compare them with Google Draco [11], V-DMC 4.0 [32], and an embedded deformation-based method proposed by KDDI group [21]. Note that we directly use the results of V-DMC 4.0 and the embedded deformation-based method presented in [21]. In the remainder of this section, we evaluate how different numbers of centers and GoF settings influence TVMC.

4.3.1 Comparisons. Figure 4 shows the objective evaluation of RD performance in D2-PSNR vs. bitrates. We compare TVMC with three baselines: Draco [11], the embedded deformation-based method proposed by KDDI group [21], and V-DMC 4.0 [32]. We set the number of centers to 1000, 2000, 3000, and 4000, and the GoF values to 5, 10, and 15 for all 4 TVM sequences. Due to page limits, we present only the results with 2000 volume centers and a GoF of 10.

From Figure 4, we can see that TVMC outperforms Draco, KDDI's method, and V-DMC 4.0, with the bitrates varying from 5 Mbps to 10 Mbps. Take "Dancer" as an example, TVMC can achieve a D2-PSNR of 80 with a bitrate less than 5 Mbps, while Draco, KDDI's method, and V-DMC require about 10 Mbps, 14 Mbps, and 15 Mbps to get



(a) Visualization of 1000 centers (b) Result with 1000 centers (c) Visualization of 2000 centers (d) Result with 2000 centers

(e) Visualization of 3000 centers (f) Result with 3000 centers (g) Visualization of 4000 centers (h) Result with 4000 centers

Figure 6: Subjective evaluation with different numbers of centers. Volume centers are represented with black spheres, and their corresponding reconstructed meshes are drawn in blue.

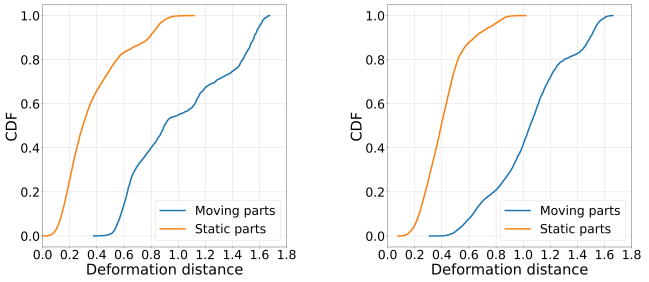
the same quality, respectively. For a bitrate larger than about 13 Mbps, Draco can achieve better D2-PSNR, however, reconstructed meshes with D2-PSNR larger than 80 are of high-enough quality. In terms of geometry compression, TVMC achieves a relatively high quality with much lower bitrate requirements, which is vital for 3D video streaming and storage. TVMC is a lossy compression method that improves compression ratio by using mesh decimation, which sacrifices some quality. Additionally, mesh deformation inevitably introduces some loss.

Note that distortions caused by self-contact regions cannot be directly reflected by evaluation metrics, since the errors are computed based on Euclidean distance between vertices. To assess these distortions, visual comparisons must be made subjectively. In Figure 5, we show the reconstructed meshes using Draco and TVMC at approximately equal bitrates. With a relatively low bitrate, about 5 Mbps to 6 Mbps, Draco causes obvious distortions on the parts of faces, hands, or feet, while TVMC performs much better. The detailed descriptions of corresponding objective evaluations and encoding and decoding time are presented in Table 1.

Table 1 also demonstrates that TVMC supports real-time applications by presenting its encoding and decoding time compared with Draco. TVMC requires, on average, 13.95 ms to decode a frame with precomputed and compressed displacements. In contrast, even under relatively ideal conditions, any compression method that decodes meshes through deformation may require over 100 ms per frame for decoding and reconstruction. From the experiments, we get results that TVMC can reduce decoding time by 66.1% compared to Draco. Although TVMC requires longer encoding times, these processes can be performed offline. Reducing the decoding time, on the other hand, is more critical to supporting real-time rendering and visualization, ultimately enhancing the user quality of the experience.

Figure 7 shows the Cumulative Distribution Functions (CDF) of the square root of the displacements' 2-norm, i.e., the deformation distance that each vertex of volume-tracked reference mesh moves. Ideally, only the vertices in the moving parts should shift, and the displacements of vertices in the static parts should be zero. However, due to a series of operations such as decimation, compression, and fitting, these values are not exactly zero but are very close. For "Thomas", in the static parts, over 85% of the deformation distances are less than or equal to 0.6, while in the moving parts, only about 10% of the deformation distances fall below 0.6. Thus, the information entropy is low, which benefits the compression process and helps TVMC achieve lower bitrates with relatively high quality.

4.3.2 Impact of the number of centers. We now investigate how the number of centers can affect TVMC's performance. As shown in Figure 6, increasing the number of centers from 1000 to 4000 results in more gray spheres (volume centers) representing the hands of the "Dancer". However, the quality of the reconstructed mesh does not always improve with a higher number of centers. In Figure 6a, only about five centers are used to represent each hand. This insufficient number of centers cannot adequately support stable center affinity deformation, leading to noticeable distortions after applying subdivision surface fitting. When the number of centers increases to 2000, the visual quality improves significantly



(a) CDF of "Basketball player" (b) CDF of "Thomas"

Figure 7: Cumulative distribution function (CDF) of deformation distance for "Basketball player" and "Thomas". The blue lines represent moving parts, while the orange lines represent static parts. Moving parts refer to main regions that change when the volume-tracked reference mesh is deformed to one of the meshes in the group, whereas static parts remain relatively unchanged. The distances are adjusted for relative scaling, as different TVMs have varying scales.

compared to 1000 centers since the additional centers provide better support for representation and deformation of the hands.

From Figure 6f, we can see that the result with 3000 centers does not differ significantly from Figure 6d, however, the increase in the number of centers slightly reduces the quality of the left and right hands. This is because more centers may introduce more challenges for MDS in generating a stable set of reference centers. When the number of centers increases, MDS must consider a larger number of pairwise distances, resulting in higher computational cost and potential errors in the optimization process. Additionally, Time-varying meshes usually have complex or irregular geometry distributions, making it more challenging to find a solution that balances all pairwise distances without introducing artifacts. With an excessive number of centers, MDS-generated reference centers are more likely to contain abnormal outputs, significantly reducing TVMC's overall performance, as shown in Figure 6h. Since the center affinity deformation relies on the movements of the nearest centers and $K-1$ centers with the highest affinity, a large number of centers can, in fact, harm the quality of center affinity deformation.

It is important to note that the optimal number of centers is highly dependent on the particular characteristics of the TVM sequence. Fine-tuning experiments can help increase TVMC's performance.

4.3.3 Impact of Group of Frames (GoF). In Figure 8, we show the RD performance under different settings of GoF ranges in 5, 10, and 15. We demonstrate that the varying GoF will not negatively impact the performance of our method, there is no significant difference in D2-PSNR performance under different GoF settings, with only slight fluctuations. This highlights that our method avoids error accumulation when compressing a group of meshes, a common issue in many reference-based compression methods such as [17, 21]. In contrast, our method can reach a GoF of 15, while the methods above struggle to exceed a GoF of 5 without a rapid increase in distortion. This feature is provided by the volume-tracked reference mesh, created from reference space, representing a self-contact-free shape. We use volume-tracked reference mesh to replace traditional

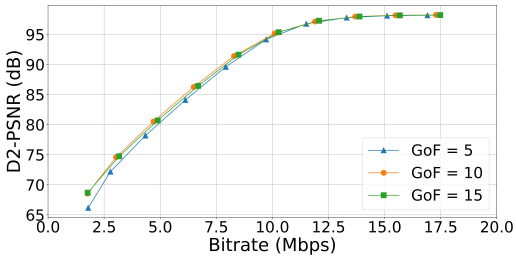


Figure 8: RD performance of "Dancer" under different GoFs of 5, 10, and 15.

reference mesh, cooperating with our center affinity deformation, avoiding the distortion accumulation problem efficiently.

Although TVMC efficiently addresses the issue of accumulated distortion, and can increase the GoF to 15 or more, using a GoF over 20 can lead to noticeable distortions. This is because deformations can cause deviations, and deformed frames for creating the volume-tracking reference mesh may not fully align as a cohesive surface, which degrades the quality of Poisson reconstruction, especially in narrow areas of the mesh, such as the wrists, lower legs and fingers. Since the subsequent displacement calculations and mesh decoding are based on the volume-tracked reference mesh, the overall performance of TVMC is decreased. Furthermore, as the GoF increase exceeds a certain limit, the meshes within the group exhibit greater shape variations. Deforming the reference mesh into a very different shape often results in noticeable distortions.

5 Conclusion

This paper proposes TVMC, a novel time-varying mesh compression approach that achieves high-efficiency mesh compression by utilizing volume centers and creating a self-contact-free volume-tracked reference mesh. With this self-contact-free volume-tracked reference mesh, TVMC enables low-bitrate mesh geometry compression while maintaining a high reconstruction quality. While TVMC utilizes the volume enclosed by the mesh surface, it does not strictly require fully watertight meshes. We evaluate TVMC on popular TVMs captured with real-world RGB-D sensors and demonstrate that it outperforms state-of-the-art methods such as Draco and V-DMC 4.0. Moreover, TVMC addresses the issue of accumulated distortions and increases the GoF size. Finally, TVMC enables real-time applications by achieving an average decoding time of less than 20 ms per frame. The main limitation of TVMC is that currently it only supports geometry information compression. Our future work includes upgrading TVMC to enable textured TVM compression.

Acknowledgments

This work was supported by Northeastern University, and the Czech Science Foundation under grants 23-04622L. We thank the anonymous reviewers and the shepherd Ketan Mayer-Patel for their thoughtful and constructive feedback.

References

[1] Rachida Amjoun and Wolfgang Straßer. 2009. Single-rate near lossless compression of animated geometry. *Computer-Aided Design* 41, 10 (2009), 711–718.

[2] Gavin Barill, Neil G Dickson, Ryan Schmidt, David IW Levin, and Alec Jacobson. 2018. Fast winding numbers for soups and clouds. *ACM Transactions on Graphics (TOG)* 37, 4 (2018), 1–12.
[3] L. Boltzmann. 1868. *Studien über das Gleichgewicht der lebendigen Kraft zwischen bewegten materiellen Punkten: vorgelegt in der Sitzung am 8. October 1868.* k. und k. Hof- und Staatsdr. https://books.google.com/books?id=vxWqmgEACAAJ
[4] Ingwer Borg and Patrick JF Groenen. 2007. *Modern multidimensional scaling: Theory and applications.* Springer Science & Business Media.
[5] Mikaël Bourges-Sévenier. 2002. An introduction to MPEG-4 animation framework extension (AFX). In *Proceedings. International Conference on Image Processing*, Vol. 3. IEEE, III–III.
[6] Alvaro Collet, Ming Chuang, Pat Sweeney, Don Gillett, Dennis Evseev, David Calabrese, Hugues Hoppe, Adam Kirk, and Steve Sullivan. 2015. High-quality streamable free-viewpoint video. *ACM Transactions on Graphics (ToG)* 34, 4 (2015), 1–13.
[7] Alexandros Doumanoglou, Dimitrios S Alexiadis, Dimitrios Zarpalas, and Petros Daras. 2014. Toward real-time and efficient compression of human time-varying meshes. *IEEE Transactions on Circuits and Systems for Video Technology* 24, 12 (2014), 2099–2116.
[8] Jan Dvořák, Filip Hácha, and Libor Váša. 2023. Global optimisation for improved volume tracking of time-varying meshes. In *International Conference on Computational Science*. Springer, 113–127.
[9] Jan Dvořák, Zuzana Káčereková, Petr Vaněček, Lukáš Hruda, and Libor Váša. 2022. As-rigid-as-possible volume tracking for time-varying surfaces. *Computers & Graphics* 102 (2022), 329–338.
[10] Michael Garland and Paul S. Heckbert. 1997. Surface simplification using quadric error metrics. In *Proceedings of the 24th Annual Conference on Computer Graphics and Interactive Techniques (SIGGRAPH '97)*. ACM Press/Addison-Wesley Publishing Co., USA, 209–216. https://doi.org/10.1145/258734.258849
[11] Google. 2018. Draco. https://google.github.io/draco/.
[12] Danillo Bracco Graziosi. 2021. Video-based dynamic mesh coding. In *2021 IEEE International Conference on Image Processing (ICIP)*. IEEE, 3133–3137.
[13] Himanshu Gupta, Max Curran, Jon Longtin, Torin Rockwell, Kai Zheng, and Mallesham Dasari. 2022. Cyclops: An fso-based wireless link for vr headsets. In *Proceedings of the ACM SIGCOMM 2022 Conference*. 601–614.
[14] Filip Hácha, Jan Dvořák, Zuzana Káčereková, and Libor Váša. 2024. Editing mesh sequences with varying connectivity. *Computers & Graphics* 121 (2024), 103943.
[15] Seung-Ryong Han, Toshihiko Yamasaki, and Kiyoharu Aizawa. 2007. Time-varying mesh compression using an extended block matching algorithm. *IEEE Transactions on Circuits and Systems for Video Technology* 17, 11 (2007), 1506–1518.
[16] Seung-Ryong Han, Toshihiko Yamasaki, and Kiyoharu Aizawa. 2008. Geometry compression for time-varying meshes using coarse and fine levels of quantization and run-length encoding. In *2008 15th IEEE International Conference on Image Processing*. IEEE, 1045–1048.
[17] Huong Hoang, Kunyao Chen, Truong Nguyen, and Pamela Cosman. 2023. Embedded Deformation-based Compression for Human 3D Dynamic Meshes with Changing Topology. In *Proceedings of the IEEE/CVF International Conference on Computer Vision*. 2252–2262.
[18] Junhui Hou, Lap-Pui Chau, Ying He, and Nadia Magnenat-Thalmann. 2014. A novel compression framework for 3D time-varying meshes. In *2014 IEEE International Symposium on Circuits and Systems (ISCAS)*. IEEE, 2161–2164.
[19] Junhui Hou, Lap-Pui Chau, Minqi Zhang, Nadia Magnenat-Thalmann, and Ying He. 2014. A highly efficient compression framework for time-varying 3-D facial expressions. *IEEE Transactions on Circuits and Systems for Video Technology* 24, 9 (2014), 1541–1553.
[20] Tao Jin, Mallesham Dasa, Connor Smith, Kittipat Apicharttrisorn, Srinivasan Seshan, and Anthony Rowe. 2024. Meshreduce: Scalable and bandwidth efficient 3d scene capture. In *2024 IEEE Conference Virtual Reality and 3D User Interfaces (VR)*. IEEE, 20–30.
[21] Xudong Jin, Jianfeng Xu, and Kei Kawamura. 2024. Embedded Graph Representation for Inter-Frame Coding of Dynamic Meshes. In *ICASSP 2024-2024 IEEE International Conference on Acoustics, Speech and Signal Processing (ICASSP)*. IEEE, 4070–4074.
[22] Michael Kazhdan, Matthew Bolitho, and Hugues Hoppe. 2006. Poisson surface reconstruction. In *Proceedings of the fourth Eurographics symposium on Geometry processing*, Vol. 7.
[23] Hao Li, Bart Adams, Leonidas J Guibas, and Mark Pauly. 2009. Robust single-view geometry and motion reconstruction. *ACM Transactions on Graphics (ToG)* 28, 5 (2009), 1–10.
[24] Guoliang Luo, Zhigang Deng, Xin Zhao, Xiaogang Jin, Wei Zeng, Wenqiang Xie, and Hyewon Seo. 2020. Spatio-temporal segmentation based adaptive compression of dynamic mesh sequences. *ACM Transactions on Multimedia Computing, Communications, and Applications (TOMM)* 16, 1 (2020), 1–24.
[25] Adrien Maglo, Guillaume Lavoué, Florent Dupont, and Céline Hudelot. 2015. 3d mesh compression: Survey, comparisons, and emerging trends. *ACM Computing Surveys (CSUR)* 47, 3 (2015), 1–41.
[26] Khaled Mammou, Jungsun Kim, Alexis M. Tourapis, Dimitri Podborski, and David Flynn. 2022. Video and Subdivision based Mesh Coding. In *2022 10th European Workshop on Visual Information Processing (EUVIP)*. 1–6. https://doi.org/10.1109/EUVIP53989.2022.9922888
[27] Khaled Mamou, Titus Zaharia, and Françoise Prêteux. 2006. A skinning approach for dynamic 3D mesh compression. *Computer Animation and Virtual Worlds* 17, 3-4 (2006), 337–346.
[28] Khaled Mamou, Titus Zaharia, and Françoise Prêteux. 2009. TFAN: A low complexity 3D mesh compression algorithm. *Computer Animation and Virtual Worlds* 20, 2-3 (2009), 343–354.
[29] Jean-Eudes Marvie, Maja Krivokuća, Céline Guede, Julien Ricard, Olivier Mocquard, and François-Louis Tariolle. 2022. Compression of time-varying textured meshes using patch tiling and image-based tracking. In *2022 10th European Workshop on Visual Information Processing (EUVIP)*. IEEE, 1–6.
[30] MPEG 2021. *Cfp for dynamic mesh coding.* MPEG. Online, ISO/IEC JTC1/SC29/WG7/N00231.
[31] MPEG. 2022. Mpeg-pcc-tmc13. https://github.com/MPEGGroup/mpeg-pcc-tmc13.
[32] MPEG 2023. *V-DMC TMM 4.0.* MPEG. ISO/IEC JTC1/SC29/WG7/N00595.
[33] MPEG 3DG 2019. *Common test conditions for point cloud compression.* MPEG 3DG. Geneva, Switzerland, ISO/IEC JTC1/SC29/WG11 N18474.
[34] Jingliang Peng, Chang-Su Kim, and C-C Jay Kuo. 2005. Technologies for 3D mesh compression: A survey. *Journal of visual communication and image representation* 16, 6 (2005), 688–733.
[35] Jrg Peters and Ulrich Reif. 2008. *Subdivision Surfaces* (1st ed.). Springer Publishing Company, Incorporated.
[36] Jarek Rossignac. 1999. Edgebreaker: Connectivity compression for triangle meshes. *IEEE transactions on visualization and computer graphics* 5, 1 (1999), 47–61.
[37] Ioan Alexandru Salomie, Adrian Munteanu, Augustin Gavrilescu, Gauthier Lafruit, Peter Schelkens, Rudi Deklerck, and Jan Cornelis. 2004. MESHGRID-A compact, multiscalable and animation-friendly surface representation. *IEEE transactions on circuits and systems for video technology* 14, 7 (2004), 950–966.
[38] Robert W Sumner, Johannes Schmid, and Mark Pauly. 2007. Embedded deformation for shape manipulation. In *ACM siggraph 2007 papers*. 80–es.
[39] Libor Vasa and Václav Skala. 2007. Coddyac: Connectivity driven dynamic mesh compression. In *2007 3DTV Conference*. IEEE, 1–4.
[40] Toshihiko Yamasaki and Kiyoharu Aizawa. 2010. Patch-based compression for time-varying meshes. In *2010 IEEE International conference on image processing*. IEEE, 3433–3436.
[41] Jeong-Hyu Yang, Chang-Su Kim, and Sang-Uk Lee. 2005. Progressive coding of 3D dynamic mesh sequences using spatiotemporal decomposition. In *2005 IEEE International Symposium on Circuits and Systems (ISCAS)*. IEEE, 944–947.
[42] Qi Yang, Joël Jung, Timon Deschamps, Xiaozhong Xu, and Shan Liu. 2023. Tdmd: A database for dynamic color mesh subjective and objective quality explorations. *arXiv preprint arXiv:2308.01499* (2023).
[43] Qian-Yi Zhou, Jaesik Park, and Vladlen Koltun. 2018. Open3D: A Modern Library for 3D Data Processing. *arXiv:1801.09847* (2018).

To Cap or not to Cap: Bandwidth Capping Effects on Network Interactions and QoE of Competing Short Video Streams

Nikolas Wehner
University of Würzburg
Würzburg, Germany
nikolas.wehner@uni-wuerzburg.de

Theo Karagioules
AT&T Labs – Research
Bedminster, NJ, USA
theo@research.att.com

Emir Halepovic
AT&T Labs – Research
Bedminster, NJ, USA
emir@research.att.com

Filip Simonovski
University of Augsburg
Augsburg, Germany
filip.simonovski@uni-a.de

Tobias Hoßfeld
University of Würzburg
Würzburg, Germany
tobias.hossfeld@uni-wuerzburg.de

Michael Seufert
University of Augsburg
Augsburg, Germany
michael.seufert@uni-a.de

ABSTRACT

Delivering popular short video streaming services like TikTok, Instagram Reels, or YouTube Shorts, poses substantial challenges for service providers and network operators. This is not only due to high download volumes but also due to high-volume pre-loading strategies that cause high bandwidth demand variations. These strategies, designed to reduce initial delays, can additionally lead to bandwidth excess when users swipe quickly through videos and consume only a fraction of downloaded content. This creates inefficiencies and unbalanced network resource utilization, particularly in competitive bandwidth environments. To address these challenges, we investigate the effectiveness of bandwidth capping in this paper, i.e., limiting the throughput of video flows in the network. We conduct measurement studies to analyze the impact of capping on network interactions and Quality of Experience (QoE) of three popular short video services in different scenarios. We find that capping substantially reduces download volume (15% – 45% median reduction) and bandwidth excess (18% – 52% mean reduction), while bandwidth utilization fairness improves. Meanwhile, QoE surprisingly remains nearly unaffected in most cases, with only minor statistical differences in a few scenarios.

CCS CONCEPTS

• **Networks** → **Network management**; **Network measurement**.

KEYWORDS

Short-form Video, QoE, Streaming, Traffic Management

ACM Reference Format:
Nikolas Wehner, Theo Karagioules, Emir Halepovic, Filip Simonovski, Tobias Hoßfeld, and Michael Seufert. 2025. To Cap or not to Cap: Bandwidth Capping Effects on Network Interactions and QoE of Competing Short Video Streams. In *The 16th ACM Multimedia Systems Conference (MMSys '25), March 31–April 4, 2025, Stellenbosch, South Africa.* ACM, New York, NY, USA, 11 pages. https://doi.org/10.1145/3712676.3714441

Permission to make digital or hard copies of all or part of this work for personal or classroom use is granted without fee provided that copies are not made or distributed for profit or commercial advantage and that copies bear this notice and the full citation on the first page. Copyrights for components of this work owned by others than the author(s) must be honored. Abstracting with credit is permitted. To copy otherwise, or republish, to post on servers or to redistribute to lists, requires prior specific permission and/or a fee. Request permissions from permissions@acm.org.
MMSys '25, March 31–April 4, 2025, Stellenbosch, South Africa
© 2025 Copyright held by the owner/author(s). Publication rights licensed to ACM.
ACM ISBN 979-8-4007-1467-2/25/03
https://doi.org/10.1145/3712676.3714441

1 INTRODUCTION

Short video services like TikTok, Douyin, YouTube Shorts and Instagram Reels, have become popular very quickly and gained billions of users [9]. Short videos are typically a few seconds to about a minute long and are mostly recorded on smartphones in vertical orientation, unlike long-form content, which is usually produced with professional equipment. Short video platforms generally present content to users algorithmically, based on individual preferences, their social interactions, or global content popularity. The main user interaction is swiping, i.e., selecting the next recommended video by moving the finger up or across the phone's touchscreen. Short videos generally follow HTTP adaptive streaming (HAS) principles, i.e., video quality can change within or between videos to adapt to varying network conditions as decided by the adaptive bitrate (ABR) algorithm. However, they exhibit distinct characteristics compared to long videos, particularly fewer encoding variants [42] and different QoE objectives. It is still highly important to deliver an uninterrupted video stream, but due to the shorter duration of videos, users now expect immediate access to content. Thus, eliminating initial delay and delays between videos has become the key QoE objective, but may be hard to attain in mobile networks where conditions can fluctuate.

Both network operators and short video services have to cope with the challenging combination of large amounts of generated traffic and stringent QoE objectives. In addition, short video traffic exhibits highly variable demand resulting from the on-off pattern of the pre-loading strategies [42], composed of large bursts and idle periods. These variations are especially challenging on links with high competition and might negatively impact other traffic, which can lead to reduced fairness in bandwidth utilization and QoE. Finally, the high-volume pre-loading behavior of apps can also result in high bandwidth excess. This occurs if users swipe to the next video before consuming most of the pre-loaded content, which contributes to more competition with other applications. Short video services in turn observe varying bandwidth on the bottleneck links and have to adapt rapidly to the conditions they contribute to themselves.

Similar but less pronounced issues occur for long video services and have recently been tackled with a cross-layer approach by

Spang et al. [32], which showed that smoothing video traffic with application-informed pacing can improve its interactions with other Internet traffic while maintaining the same or better QoE. However, this approach requires combining a custom ABR algorithm with limiting the server's sending rate, which is entirely within the domain of content and service providers. An alternative approach is to implement bandwidth capping only at the egress of a CDN edge node or inside the core or access network, to limit the throughput of video streams, thereby also smoothing traffic and promoting fairness at the network layer. This can be appealing, especially for network operators, as it allows to positively affect QoE independently of service providers, while saving network resources at the same time. However, there is no definitive answer yet on whether to cap or not to cap. In the best case, this might trigger a benign adaptation behavior of video services and could improve bandwidth utilization and QoE. In the worst case, QoE could be degraded by capping, which should be avoided. In this work, we investigate the impact of bandwidth capping on three popular, short video services – TikTok, Instagram Reels, and YouTube Shorts. While we focus on the impact of capping on bandwidth and QoE fairness within and between different short video services, we also analyze interactions with long video, using YouTube and Facebook Watch, and file download cross-traffic. We conduct automated measurement studies on smartphones competing for bandwidth on a bottleneck link, as typically found in mobile networks, and treat all services as black-boxes. Finally, we evaluate the impact of capping on unconsumed data (downloaded but not watched) considering different swiping behaviors. Our key insights into short video services are:

- Capping substantially reduces download volume (15% – 45% median reduction) and bandwidth excess (29% – 52% for two short video services, and 18% – 45% for one service).
- Capping improves bandwidth utilization fairness within and between short video services.
- Surprisingly, only a very small, service- and scenario-dependent penalty is observed on QoE metrics. One service showed no significant degradation in any QoE metric or capping scenario. One service was only affected in low capacity scenarios, where the worst-case initial delay exceeded 1 second and stalls increased by a factor of 3. One service consistently exhibited small noticeable reductions of visual quality by 6 – 9 VMAF points.
- Finally, capping allows short video services to be more friendly to other Internet traffic (file downloads and long-form videos), which in some cases benefit in performance.

Thus, we find that capping can be considered a reasonable traffic management tool with benefits for all stakeholders – network operators, service providers, and end users of video services.

Section 2 outlines related work. Section 3 presents the measurement methodology and experimental setup. We analyze network performance metrics in Section 4 and QoE performance in Section 5. In Sections 6 and 7, we investigate the impact of cross-traffic and swiping behavior, respectively. We discuss findings, limitations, and future work in Section 8 before Section 9 concludes.

2 RELATED WORK

Characteristics of Short Video Streaming Services. With the growing popularity of short video services, several works have been conducted to better understand their high-level characteristics, such as distribution of video duration and bitrate [4, 40].

Video Streaming Measurements. Various tools have been proposed for measurement-based performance evaluation of video streaming applications. A web browser plugin that collects data [16, 31, 33] can be automated to collect measurements using Selenium browser automation tool [22, 26]. A custom Android app can be utilized in a local testbed for network traffic and QoE monitoring [29, 35]. However, these approaches do not use native smartphone apps directly. To overcome this limitation, a wrapper [30] was introduced to leverage Android Debug Bridge (ADB) to automatically run measurements and interact with native app's UI elements. Zhu et al. [42] conducted automated measurement studies for short video services to examine pre-loading mechanisms and data consumption. In this work, we build upon prior approaches and present a comprehensive measurement framework for fully automated measurements, which is able to collect data both within native apps and in the network at the same time. It further allows for automated post-processing and analysis, ensuring accurate and consistent measurement results.

Quality of Experience (QoE). QoE is a widely recognized concept for assessing the subjectively perceived quality of Internet applications [3]. Seufert et al. [25] identified initial delay, stalling, and video quality as major QoE metrics for video streaming. In the case of short videos, initial delay is particularly important, as users expect instant playback when they swipe. This poses a major challenge as users' swiping behaviors are different. To tackle this challenge, various pre-loading strategies are used, following either a conventional [18, 39, 41] or a learning-based approach [7, 21, 38]. Many strategies result in pre-loading excessive content that may not ultimately be viewed by the user due to the unique nature of swiping behavior. The probabilistic model introduced in [41] addresses this issue by determining a maximum pre-load size, resulting in a 22.8% reduction in bandwidth excess. In [21], the authors formulate a QoE-data trade-off. By leveraging information about past users' viewing duration, they manage to reduce unconsumed data by up to 58% compared to existing methods.

QoE Fairness. The approaches presented so far are focused on a single user. However, due to the high popularity of video services, it is likely that multiple video streaming clients will share a bottleneck link [17]. Significant contributions to the understanding of how to measure and achieve fairness with respect to QoE are provided in [8, 24]. Improving fairness among competing video clients has also been a subject of research on congestion control. Minerva, a protocol introduced in [17] adjusts the bandwidth allocation of clients to maximize QoE fairness, resulting in a 24% improvement of the worst viewing experiences. Moreover, VSiM [37], a mobility-profiled QoE-driven bandwidth allocation strategy, enhances QoE fairness by 40% by using a central server that prioritizes bandwidth allocation based on users' trajectory. In contrast, the ABR algorithm TCPAL [28] has no centralized controller but still maintains QoE fairness even under fluctuating network conditions, while FlexStream uses a more centralized approach [2].

Network management. In modern network management, an important challenge for operators is to effectively orchestrate resources to maintain high QoE for users while ensuring (QoE) fairness across the network. Wamser et al. [34] analytically and simulatively evaluate a management concept that dynamically adjusts

network bandwidth according to the current traffic mix and application requirements. Their approach enhances QoE for video streaming and web browsing, demonstrating the potential of application-aware bandwidth allocation. To further optimize resource allocation, network slicing has emerged as an effective strategy, offering the flexibility to allocate network slices for specific services. Kwak et al. [14] presented a dynamic bandwidth slicing framework that optimizes resource allocation for video services, enhancing QoE and minimizing delays. Complementing this approach, research has also focused on bandwidth capping as a method to manage resources. For instance, in [13], the network and client work together to implement bandwidth capping, which helps reduce network congestion and enhances streaming performance.

3 METHODOLOGY

3.1 Measurement Study Design

Short Video Services. The goal of this work is to evaluate the benefits of network management for bandwidth utilization and QoE of short video streaming services in competition scenarios with a bottleneck link, e.g., in WiFi access networks or mobile network cells. More precisely, we want to investigate how different popular short video services – TikTok, Instagram Reels, and YouTube Shorts – are affected by capping on the bottleneck link. We treat these services as black boxes in terms of system design, adaptation logic, encoding, and congestion control. We anonymize the services for the remainder of this paper, referencing them randomly as Service 1-3, or S1-S3. To increase the realism of the measurement, and thus the generality of the results, we do not measure each service in isolation, with a single device on a bottleneck link as done in previous works. Instead, we measure the behavior and the resulting QoE using multiple devices in competition within and between services, which allows us to observe more realistic fluctuations of network conditions for the devices and services. In our testbed, we use six devices, with two devices for each service.

To allow for reliable, consistent, and reproducible measurements, interaction with the devices is automated using Appium [1]. The experiments start by clearing app caches, followed by starting the app. Then, each app is navigated to a predefined video playlist via the automated UI, as detailed in Section 3.3. All devices wait for a randomized duration of 0 to 5 seconds before starting playback. This prevents identical video startup sequences and avoids measurement bias from varying navigation times.

Since swiping behavior greatly impacts throughput requirements, we implement three swiping modes: *deterministic*, *half*, and *full*. In deterministic mode, a swipe occurs after each video is played for 15 seconds (regardless of its duration), reflecting the shortest observed median playback duration across services in our dataset and supported by prior findings [40]. The swipes in the other two modes occur at the half or full playback points (cf. Figure 2b).

Cross-Traffic. Since network management affects both short video flows and interactions with other cross-traffic, we consider two cross-traffic scenarios in this work. For this, we add two more smartphones to the testbed, which produce cross-traffic during the short video measurements. In the first scenario, we analyze the behavior of short video services in the presence of long videos. In this scenario, the two cross-traffic devices stream long videos, in particular, one streams YouTube and the other one Facebook Watch, representative black-box long video services. Again, we anonymize these services for the remainder of this work, referring to them as L1 and L2 in a randomized order. Each device streams a different video playlist. In the second scenario, we analyze the behavior of short video services in the presence of file downloads. Here, the two cross-traffic devices download a large ISO file from an Ubuntu server. For automating the file download, we use Selenium [23]. To start the file download, we open the Chrome browser on the end-devices and enter the download URL of an Ubuntu ISO. This ISO can never be completely downloaded during the experiment duration due to its size. File downloads are also loosely synchronized with app startups, i.e., they start around the same time as video playback.

These scenarios were selected to cover not only two other common service types, long video and file downloads, but also to include highly diverse cross-traffic characteristics. Similar to short videos, long videos have bursty traffic and short flow duration due to their typical on-off-behavior. During flow start they are fighting for their bandwidth share, subject to congestion control on transport layer (TCP/QUIC). Their network requirements are also affected by the ABR algorithms on application layer, and thus, they are able to adapt their network requirements to changing network conditions and competition. Contrary, file downloads are typically large, long-running flows. These flows exhibit converged TCP behavior, which means they aim to saturate the available capacity and will only adapt their network requirements in case of congestion as a result of TCP congestion control. Moreover, they are known to have advantages over short flows when competing for spare capacity, which can lead to a disproportionate share of capacity. By investigating these different cross-traffic characteristics, we are able to observe a wide range of the effects of capping on video streams.

Network Management. We consider three different bottleneck capacities (12, 16, or 20 Mbps) for the eight devices in our testbed, and compare two network management configurations: unconstrained and capping. In the unconstrained scenario, all video flows are assigned a low guaranteed download rate of 32 kbps to avoid starvation. All flows have a maximum allowed download rate equal to the overall capacity in the network. A fair bandwidth share between the eight devices for these capacities would thus result in 1.5 Mbps, 2.0 Mbps, and 2.5 Mbps for each device, respectively. In the capping configuration, we again guarantee all video flows a download rate of at least 32 kbps to avoid starvation, but cap the maximum allowed download rate to the fair shares of 1.5, 2.0, or 2.5 Mbps, respectively. The total network capacity equals eight times the capped download rates (12, 16, or 20 Mbps), making it comparable to the unconstrained scenario.

Performance Metrics. We evaluate the effect of bandwidth capping from both a network and a QoE performance perspective. In terms of network performance, we evaluate the peak throughput of the flows and the overall download volume per experiment. Additionally, we analyze the bandwidth utilization of each device, which we define as the fraction of traffic that was downloaded by this device compared to the theoretical maximum, i.e., the traffic that could be downloaded when fully utilizing the available capacity. To evaluate the QoE of the short video services, we focus on QoE factors that are widely accepted in literature [25], and by well-established video QoE models, e.g., the standardized video QoE model ITU-T

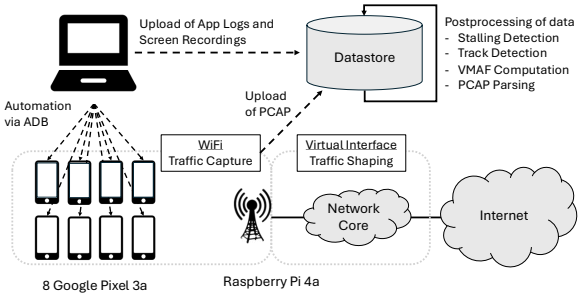


Figure 1: Simplified testbed setup.

P.1203 [10]. We assess the visual quality of the streamed videos by applying the Video Multimethod Assessment Fusion (VMAF) [15] method. It compares the quality of the streamed video to an undistorted reference video with a VMAF score of 100. The more the visual quality of the streamed video is reduced, the lower the resulting VMAF score. The smallest unit which can be perceived by a user – Just Noticeable Difference (JND) – is a score difference of ± 6 [20] for VMAF, with the implication that the larger the absolute VMAF difference, the more noticeable a quality switch would be. Apart from visual quality, the startup delay or initial delay, i.e., the delay before the video starts its playback, is another important QoE factor, especially for short video services, where users expect instantaneous playback start after swiping. Finally, we evaluate stalling, i.e., playback interruption due to rebuffering, which is considered the most important video streaming QoE factor [25, 27]. We present the total stalling duration here, representing the total waiting time during a single experiment run.

3.2 Measurement Setup

Figure 1 shows an overview of the testbed setup, which includes a PC, a Raspberry Pi 4a, eight Google Pixel 3a smartphones (Android 11), and an external server as a central data store. The PC automates app interactions on smartphones using Appium or Selenium with Android Debug Bridge (ADB) [5] and uploads logs and screen recordings to the datastore. The Raspberry Pi's WiFi interface acts as an access point for the smartphones, captures the network traffic in a PCAP file, and uploads it to the datastore. The Raspberry Pi's Ethernet interface routes all traffic to the Internet, allowing normal video service use. A virtual interface on the Raspberry Pi between these interfaces emulates the network core and is responsible for capping. Once the data is uploaded after the measurements, automated analysis identifies stalling events and ABR tracks from the screen recordings and computes VMAF for each video. To achieve this, we created a measurement framework featuring a dockerized core and various nodes that handle tasks such as Appium automation and screen recording post-processing. We use Apache Kafka and the publish-subscribe model for core-node communication.

Application Layer. In our testbed, all eight smartphones are connected via USB hub to the PC. Eight service nodes run on this PC, each with its own Appium or Selenium process for interacting with Android apps or performing file downloads on the respective devices. Each process distinguishes between the devices by using unique Android device IDs and ports, and then executes an automation script tailored for the configured service. We utilize the screen recording capabilities of Appium to record the video playbacks. During automation, we log timestamps of app interactions, e.g., swipes, and video IDs for each device individually. These logs help synchronize application and network data during analysis.

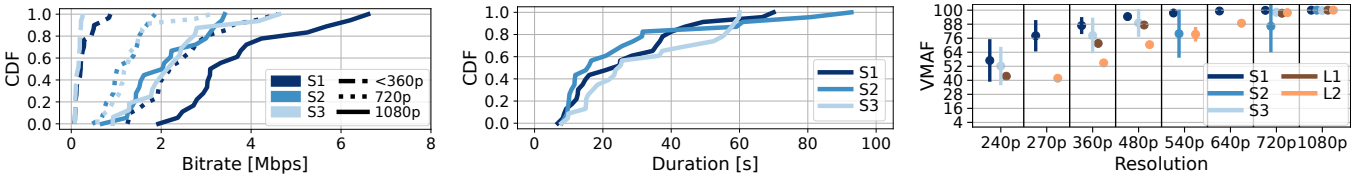
To coordinate the timings of the eight devices and ensure controlled competition, we use a synchronization node. This node synchronizes the service nodes by splitting the automation script into steps, i.e., app cache clearing, startup, and playback. Devices wait for others to complete each step before being signaled to proceed.

Network Layer. On the network side, we use one node for capping traffic with Linux `tc` and another for traffic capture with `tcpdump`, both on the WiFi interface of the Raspberry Pi 4a. Each smartphone is assigned a static IP address to facilitate capping. We use Hierarchical Token Bucket (HTB) for classful queuing to cap downlink traffic across different IP address ranges for each service. In detail, we define one class for the competing downlink video traffic and one class for all other non-competing traffic. Inheriting from the competing class, each device and video service creates another child class for capping device and video flows. We are able to exclude non-video traffic by marking all traffic flows belonging to specified service IP address ranges collected in advance.

Track Detection. The track detection node identifies the streamed quality levels over time to estimate QoE change with traffic capping. For YouTube, we monitor the track directly from the app itself using the built-in StatsForNerds option [12, 30]. Here, we immediately activate the StatsForNerds overlay after starting video playback. The overlay shows the video track, buffered video data, and more, and includes a copy button to log this information. By copying and logging this data periodically throughout video playback, we collect the relevant track information.

For all other services, we must rely on alternative means as they do not expose such statistics to the public. For this purpose, we utilize and adapt VideoEye [36], which requires to download all tracks of the original video from the video service platform as a reference. Then, VideoEye temporally aligns all screen recording frames with all frames of the reference video based on distances between downscaled, low-resolution versions of the frames. Finally, VideoEye computes the distances between screen recorded frames and all aligned reference frames for all reference tracks. At each point in time, the detected track of the screen recording is set to the reference track with the smallest distance.

In the original VideoEye evaluations, the authors had full control of the video sources and video player. In this work, we had to adapt VideoEye to make it compatible with all measured video services. First, the screen recorded frames and the reference frames must match pixel-wise. As the viewports of the apps usually do not fill the entire screen, we have to determine not only the measures of the viewport, but sometimes also how much of the source video is cut off due to viewport scaling. This requires exactly aligned cropping and scaling of screen recorded video and reference videos. Next, short video services usually overlay non-static text and icons on top of the actual video, e.g., the number of likes. To prevent these elements from skewing the error between screen recorded and reference frames, we mask them in both videos. Without masking, the pixel-wise error would be generally larger and could hide subtle



(a) CDF of short video bitrates. (b) CDF of short video duration. (c) VMAF for various video resolutions.

Figure 2: Statistical description of predefined short video playlists and long videos.

Table 1: Overview of conducted experiments. P1 and P2 correspond to playlists 1 and 2. The baseline scenario is marked in blue.

Traffic Management	Max. Bandwidth Per Device	Device 1	Device 2	Device 3	Device 4	Device 5	Device 6	CT Scenario	Device 7	Device 8	Swiping
Unconstrained	12/16/20 Mbps	S1 (P1)	S1 (P2)	S2 (P1)	S2 (P2)	S3 (P1)	S3 (P2)	Long Video	L1 (P1)	L2 (P1)	Deterministic
											Half
											Full
								FD	FD (Unc.)	FD (Unc.)	Deterministic
Capped	1.5/2.0/2.5 Mbps	S1 (P1)	S1 (P2)	S2 (P1)	S2 (P2)	S3 (P1)	S3 (P2)	Long Video	L1 (P1)	L2 (P1)	Deterministic
											Half
											Full
								FD	FD (Unc.)	FD (Unc.)	Deterministic

track differences. Finally, video players manipulate the color space of the videos by changing contrast and brightness, resulting in slightly different colors between screen recorded and reference videos. To address this, we use histogram matching to align the color space of the reference videos with the screen recorded video.

Stalling Detection. To identify stalling events during playback, we utilize the screen recordings and FFmpeg. FFmpeg provides a filter called `freezedetect`, which detects freezes by comparing differences between the components of the video frames. Independent of the service, we use the default noise tolerance (-60dB) and a custom duration of 0.5 seconds required to identify a set of frames as frozen. We mask all visuals not linked to the actual video, e.g., progress or loading bars, to avoid interference during detection. Finally, we post-process the detected stalling events by aggregating stalling events occurring rapidly after each other to a single event and by adding a single long stalling event in case the video never stopped freezing after startup.

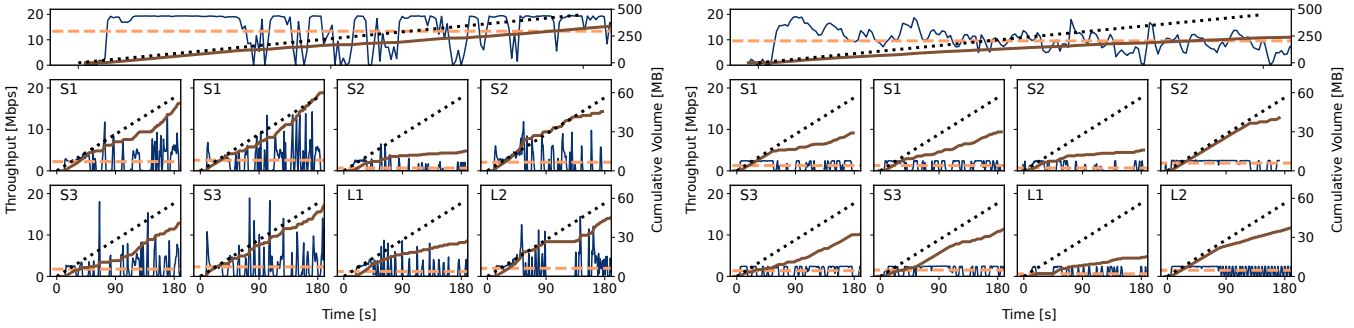
3.3 Experimental Design

Ideally, we would emulate typical short video app usage in our experiments: starting the app, selecting short videos if needed, and watching a sequence of recommended videos. However, pure recommendation-based streaming introduces randomness, which hinders reproducibility between iterations due to the differences between random videos. Thus, we generate a pre-defined playlist for each device. This ensures that each device consistently uses the same app and streams the same videos in every run, though playlists and videos vary between devices.

Playlist Characteristics. In the following, we briefly characterize the selections for the short video playlists and long videos with respect to bitrates, video duration, and VMAF in Figure 2. For each short video service, we select 24 videos with varying contents including high-motion (sports), medium-motion (dance clips and concerts), and low-motion videos (slowly wandering camera across a landscape), all with the aspect ratio of 9:16. We divide these videos into two playlists containing 12 videos each. We balance the network requirements of contents between the playlists per service and shuffle the contents only within the playlists.

The distribution of video bitrates for three selected resolutions, i.e., quality levels, of the video playlists are depicted in Figure 2a as CDF, where the x-axis denotes the video bitrate in Mbps. It shows that the bitrates range from 66 kbps to 6.6 Mbps and that S1 generally requires higher bitrates and network resources than the other services. Figure 2b shows the distribution of the video duration across the three services.The video duration ranges from around 6.8 seconds to around 92.8 seconds, while the median is 22.5 seconds. Generally, the video duration is comparable across the services. Finally, Figure 2c illustrates VMAF scores for various resolutions compared to a 1080p reference video. The markers show the average VMAF score across all measured videos of a service and the error bars indicate the standard deviation. We observe that the different services use different representation sets, and thus, different resolutions. For example, S3 offers only resolutions of 540p and above. We also find that, due to different encoding and compression settings, the visual quality of the resolutions are highly variable between services. For example, VMAF scores for S1 are already close to 100 for 540p, while for L1 they approach 100 only from 720p. We omit existing resolutions below 240p for clarity here.

Experiment Summary. We provide an overview of all conducted experiments in Table 1. Note that the maximum available bandwidth per device is always only one of the provided options, leading to 24 different experiment settings. Devices 7 and 8 always remain unconstrained in the file download (FD) cross-traffic scenario, even if the short video services are capped. From June to August 2024, we performed 50 consecutive runs for each of the 24 experiments, resulting in 1200 measurements in total. After filtering invalid or failed measurements, 1001 measurements remain with a mean of 37.07 and a minimum of 29 runs per experiment setting.



(a) Unconstrained. **(b) Capped.**

Figure 3: Visualization of network performance metrics for two exemplary measurement runs. Dark blue lines denote throughput. Brown lines denote cumulative download volume. Dashed orange line represents average throughput over measurement time. Dotted lines indicate maximum achievable download volume overall and per-device fair share.

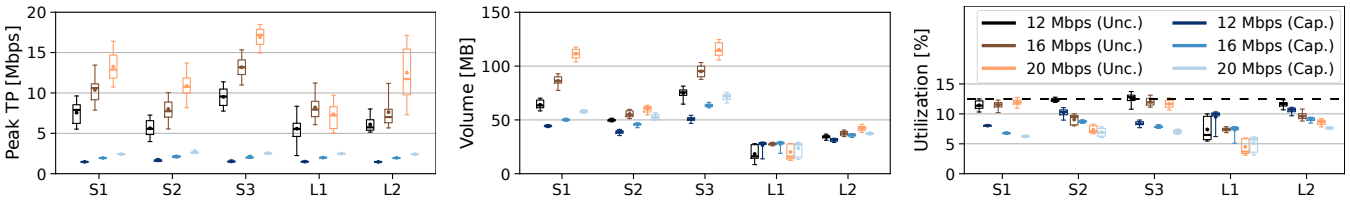
4 NETWORK PERFORMANCE

In this section, we analyze network traffic and the associated performance metrics. To illustrate the richness of insights we obtained from our experiments, Figures 3a and 3b present the results of two exemplary measurements for a bottleneck capacity of 20 Mbps in the unconstrained and capped conditions, respectively. Both runs face long video cross-traffic and deterministic swiping. The wide plot in the first row depicts overall traffic on the bottleneck link, while the second and third rows display per-device traffic. The plots show the measurement time in seconds [s] on the x-axis. The dark blue lines represent throughput in Megabits per second [Mbps], the brown lines indicate cumulative download volume in Megabytes [MB], and the orange lines show the average throughput over the measurement period, all displayed on the y-axis. Additionally, the dotted lines denote the maximum achievable download volume over time, both in regard to the overall link capacity of 20 Mbps (top row) and to the fair share per device of 2.5 Mbps (bottom rows).

The overall traffic in the unconstrained network (top of Figure 3a) shows that the average throughput over time was 13.37 Mbps (dashed orange line), which corresponds to an average bandwidth utilization of 66.8%. Initially, bandwidth competition peaks as services try to fill their buffers with the video currently being viewed and pre-loaded videos. Thus, we observe a period of peak download throughput (dark blue curve), in line with the expected initial buffering behavior of video streaming. This is followed by typical on-off traffic patterns of the devices, which in turn result in several utilization drops on the aggregate registered approximately at the 45 s mark and lasting until the end of the experiment. At this point, the slope of the cumulative download volume (brown curve) slightly decreases but maintains a linear trend until the end.

Looking at the individual devices in the bottom rows, we see that initially, all devices download around their fair share, although the second set of devices (right-most) of S1-S3 all show a large throughput spike above the fair share in the first seconds, an adverse effect of unconstrained competition. These devices also keep their bandwidth utilization around the fair share for the rest of the measurement run. For the other devices, i.e., the first (left-most) devices of S1-S3, the brown curve becomes much lower than the dotted black line indicating a reduction in their network requirements, which results in lower fairness in terms of bandwidth distribution. This behavior could be attributed to the ABR algorithm employed by the services, i.e., a reduction of the requested video bitrate. This suggests that their visual quality, and thus, their QoE will be lower, showcasing the unfavorable effect of competition on these three devices. Considering the cross-traffic, which is long video, we do not observe a large negative impact due to competition for this measurement run. After the initial burst phases, where both devices receive around their fair share, L1 shows the expected on-off pattern with throughput spikes every few seconds. In comparison, L2 has a longer initial burst phase up to around 90 seconds, followed by a 45-second pause, before initiating a second burst phase. Accounting for potential differences in system design between the two services, both behaviors can be considered normal.

In the capped scenario, shown in Figure 3b, the overall traffic reveals an average throughput of 9.56 Mbps, which corresponds to an average bandwidth utilization of 47.8%, substantially lower than in the unconstrained scenario. More specifically, compared to the unconstrained scenario (349 MB), the total downloaded volume was significantly lower, with a final value of 245 MB. Unlike the unconstrained scenario, the throughput (dark blue curve) never saturated the link capacity but remained mostly close to the average. This demonstrates that the network load and the overall burstiness of traffic can be effectively reduced by capping. Looking at the devices individually, we can observe that while the characteristic on-off behavior still exists, we encounter more extended on-phases, and sometimes even continuous, such as in the second device of S2. Both devices of S2 show a similar cumulative download volume (brown curve) compared to the corresponding unconstrained scenario in Figure 3a. S1 and S3, on the other hand, download substantially less compared to the unconstrained scenario, which depending on codecs and the effect of bitrate adaptation might also result in a reduced visual quality, and thus, a reduced QoE. Similar observations can be made for the cross-traffic, where L2 downloads around the same, but L1 downloads substantially less compared to the unconstrained scenario. With network management, fairness increases across devices and services when compared to unconstrained network utilization. Nonetheless, as a static network management approach, capping may affect QoE differently depending on the service's design, i.e., adaptation and pre-loading algorithms.



(a) Peak Throughput. **(b) Download Volume.** **(c) Bandwidth utilization.**

Figure 4: Comparison of network performance metrics.

Having shown exemplary measurement runs, we now focus on aggregated results for the remainder of this work. Figure 4 compares the distribution of network performance metrics, namely, peak throughput, download volume, and utilization. For each service and bandwidth setting, we display the observed distributions as box plots for both unconstrained (brown) and capped (blue) scenarios side by side. The different maximum download rates are indicated by color brightness: dark for 12 Mbps, medium for 16 Mbps, and light for 20 Mbps. Boxes span from the 25th to 75th percentile with thick lines indicating the median (50th percentile) and circles indicating the mean, while whiskers extend from the 5th to 95th percentile. The x-axis displays services and the y-axis corresponding metrics.

First, we observe that the peak throughput behaves as expected (cf. Figure 4a): in the unconstrained scenario, the throughput can theoretically reach up to the link capacity, though it never actually does due to competition. Naturally in the capped scenario, the peak throughput of all services is limited by the bandwidth cap.

Figure 4b displays the distribution for downloaded volume. For the short video S1-S3, it is evident that capping reduces the overall downloaded volume. While S1 and S3 generate from 30% to 45% less median volume compared to the unconstrained scenario, the volume difference is less pronounced for S2, where the median decrease is between 15% and 20%. We also observe that under capping, all short video services download similar amounts of data, thus achieving higher network fairness between services. Another important observation is that capping substantially reduces the variation of download volume of all services as indicated by the shorter boxes in the capping scenario. This shows that capping also leads to higher network fairness within services. While lower data volume also signals more efficient network usage and benefits to data-constrained users and rate plans, we will examine the impact on QoE when we look at Figure 5 in Section 5. For the long video L2, capping does not reduce the download volume much but again variation decreases. However, we can make an interesting observation for L1. Here, we see from the increased median in the darkest (12 Mbps) and lightest (20 Mbps) boxes that the download volume increases in the capping scenario compared to the unconstrained scenario. Thus, we find that capping has the potential to make short video flows more friendly neighbors, aligning with prior independent results in [32]. However, our results also indicate that the effect of capping may depend, aside from the design of the individual video applications as we established previously, on the type of video traffic as well. Since short and long video traffic may behave differently in capped scenarios, traffic management techniques tailored to video types could be more efficient and preferable.

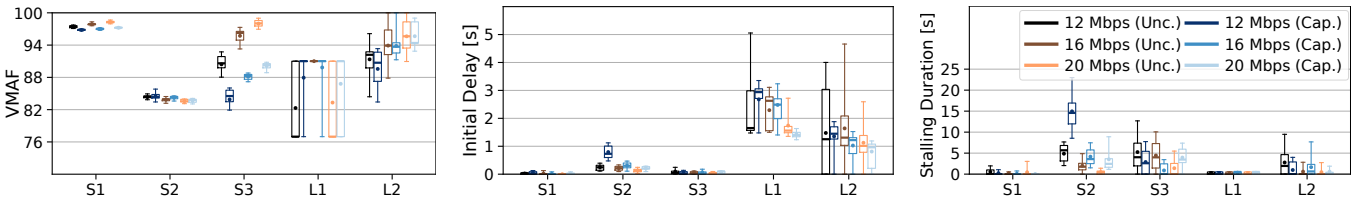
The box plots for bandwidth utilization in Figure 4c reveal lower fairness in the unconstrained scenario even though devices rarely exceed the fair share utilization at 12.5%, as indicated by the dashed line depicting the results of the capping scenario. We see large differences between services and also a high variation in utilization within services. As intended, with capping, the fairness is increased especially within services, as indicated by the shorter boxes. There are still fairness imbalances between services but to a lesser extent.

5 QOE PERFORMANCE

Network performance results revealed an increase in network-level fairness with capping but also a decrease in bandwidth utilization, potentially degrading application performance and QoE. Since QoE is crucial for end users, we will now analyze the resulting impact of capping on QoE for our baseline scenario. To assess the impact of capping on the application layer, we analyzed VMAF, initial delay, and total stalling duration, with results presented in Figure 5. Again, the results for the unconstrained scenario and the capping scenario are shown side by side (similarly to those in Figure 4).

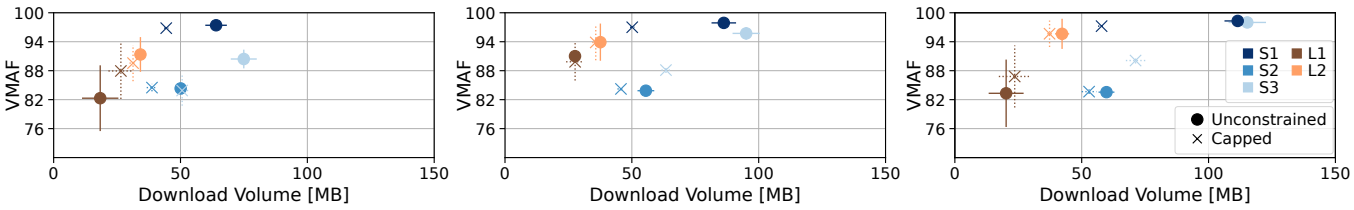
VMAF results (cf. Figure 5a) show that capping had minimal impact on the visual quality of S1 and S2 videos, while S3 experienced some decreases in VMAF. Note that the y-axis starts at 76 to better highlight the differences. The average VMAF decreases of S2 are between 6 and 9, and thus, above the JND threshold of 6 VMAF points [20]. This means that end users are expected to notice the reduced visual quality, which could negatively affect the QoE for this service. In the case of long video services, we see again that L1 benefits in VMAF values for 12 Mbps capacity (dark colors), while the VMAF scores with other capacities are mostly unaffected, which is in line with network performance metrics. For L2, we note that under the lowest capacity conditions, VMAF slightly decreases. Under higher capacities, we note similar VMAF scores and a less pronounced effect of capping. Further, the results are concentrated in a tight box (meaning higher fairness) around the median.

In summary, visual quality is not affected negatively in most capping scenarios, for all video services. Only S3 exhibits a consistent, but small reduction of visual quality, and potentially, overall QoE. To put these QoE results into perspective with the corresponding bandwidth utilization on the network layer, we analyze the trade-offs between cumulative download volume and VMAF scores for the different services in Figure 6. The x-axis represents the cumulative download volume, while the y-axis illustrates the corresponding VMAF scores. The averages in the unconstrained and capped scenarios are depicted with a circle and a cross, respectively. Horizontal and vertical lines show the standard deviations, solid



(a) VMAF. **(b) Initial delay.** **(c) Total stalling duration.**

Figure 5: Comparison of QoE performance metrics.



(a) 12 Mbps. **(b) 16 Mbps.** **(c) 20 Mbps.**

Figure 6: Trade-off between VMAF and download volume for each video service.

for unconstrained and dotted for capped. The analysis is conducted separately for the three available bottleneck capacities.

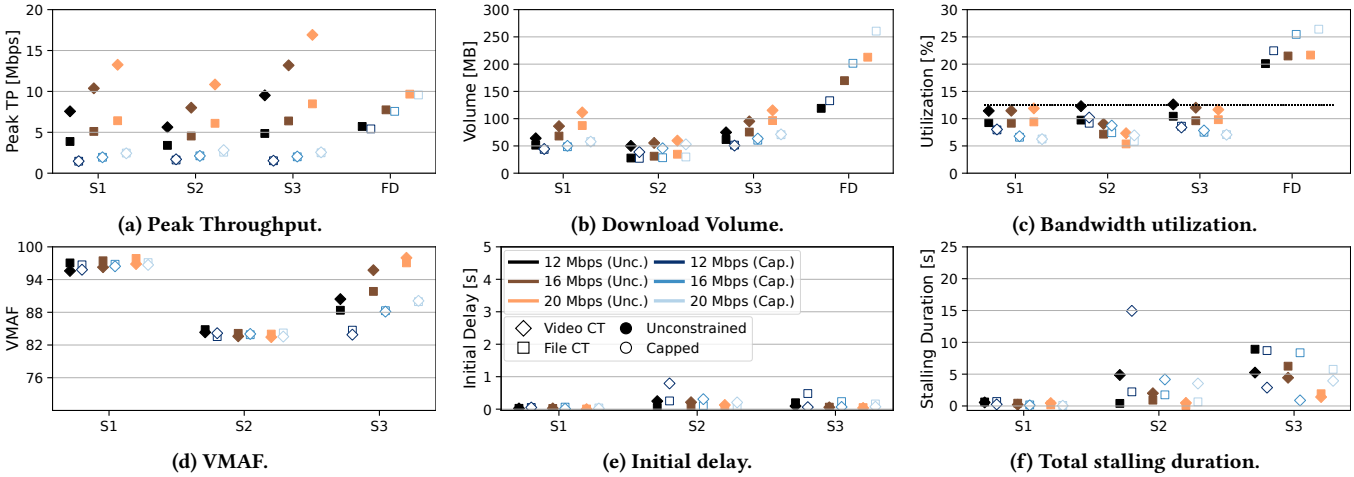
We can clearly see three behaviors with respect to the trade-off between VMAF and network utilization (download volume) as capping is applied: (1) substantial reduction of utilization without noticeable impact on visual quality, (2) small or insignificant change in both VMAF and utilization and (3) significant reduction in both VMAF and utilization. The first behavior resembles over-utilization of bandwidth without visual benefit prior to capping, which represents the ideal and best case of applying capping. In the figures, it can be observed as a horizontal shift leftward between unconstrained and capped values for S1. The second behavior may represent services that are already conservative in both network utilization and visual quality. Exemplars are S2 from the short video side, but also long video L1 and L2. This behavior appears as a minimal shift in any direction in the figures. Finally, the third behavior resembles maximized utilization and visual quality that are both sensitive to changes in network conditions. This is exemplified by S3 and appears as a diagonal shift towards lower values from unconstrained to capped points across both axes. This behavior is the least amenable to capping compared to prior two.

In summary, as four out of five services exhibit benign behaviors, we see mostly beneficial trade-offs with capping. However, this analysis explains only the network efficiency profile of services but must be combined with initial delay and stalling results for a comprehensive understanding.

Initial delay is especially important for short video streaming, as users expect no waiting time after swiping to the next video. In Figure 5b, we can generally see that all short video services (S1-S3) have much lower initial delays compared to the long video services (L1-L2). The initial delays of S1 and S3 are always below 100 ms, which is the limit for having the user feel that the system is reacting instantaneously [19], and they are not negatively affected by capping. Only S2 has an increase in initial delays with capping. Higher delay is especially prominent under the lowest bottleneck capacity, where startup times might exceed 1 second, which may be considered too long [40]. These delays should be clearly noticeable by users and could interrupt the users' flow of thoughts [19], and might thus negatively impact the QoE. For L1 and L2, we see a reduction of initial delay variation with capping. Additionally, for the lowest bottleneck capacity of 12 Mbps, we have an increase in initial delays, especially for L1, for which the median increases by more than 1 second. For the other capacity limits, the median values remain around the same with capping.

As with long videos, interruptions of the playback, also called stalling or rebuffering, are considered a major QoE factor for short-form videos. Figure 5c shows the analysis of the total stalling duration, i.e., the sum of durations of all stalling events within a video session. We observe almost no stalling for S1, even with capping, while S2 faces the most significant increase in total stalling duration, especially for the lowest bottleneck capacity where the average total stalling duration almost triples. For the long videos, however, we see that L1 experiences almost no stalling and remains unaffected by capping. L2 benefits from capping under the lowest and highest capacities, but has more stalling under medium capacity.

To sum up our findings on initial delay and stalling, we see again that services S1 and S3 adjust well to capping and can perform equally well as in unconstrained scenarios. Only S2 has statistically significant increased waiting times, in the worst case noticeable initial delay above 1 second and stalling increase up to a factor of 3, which could negatively affect the QoE. While long video services see a slight degradation of initial delay and stalling in some cases, they generally appear to perform well under capping. Overall, the results show that while it is possible to design short video services to maintain QoE under capping, there is still room for improvement.



(a) Peak Throughput. (b) Download Volume. (c) Bandwidth utilization.

(d) VMAF. (e) Initial delay. (f) Total stalling duration.

Figure 7: Cross-traffic impact on network and QoE performance metrics.

6 IMPACT OF CROSS-TRAFFIC

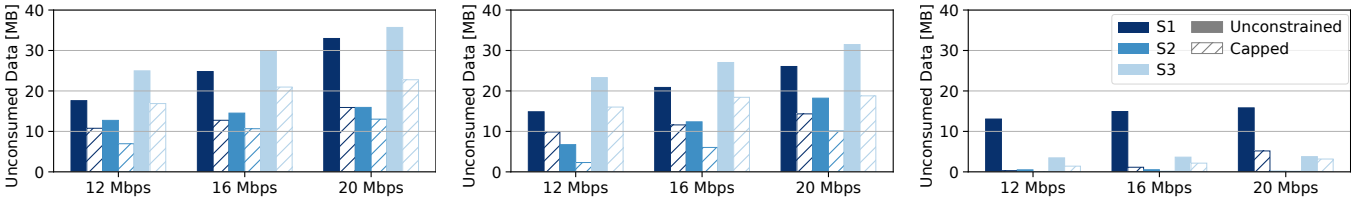
This section investigates the impact of capping on short video services under different cross-traffic conditions, including long video and file download traffic. File downloads remain unconstrained in the capping scenario, as we focus on the highly variable short video services. Figure 7 shows the performance of short video services (S1-S3) under different cross-traffic and capping scenarios, with respect to the different network-level and QoE metrics, in the deterministic swiping scenario. We also analyze the network performance changes for file downloads. Each marker in the figure represents the mean observed value, shaped by the cross-traffic type (diamond for long video and square for file download) and colored based on available capacity and traffic management scenario. Brown colors and filled markers correspond to the unconstrained scenario and blue colors and unfilled markers correspond to capping.

First, we observe that cross-traffic can impact the behavior of short video services at the network layer. While the on-off traffic pattern of long video allows short video services to utilize idle periods for faster data download, file download *throttles* short video services by continuously downloading in the background, leading to a much higher bandwidth utilization than long video (cf. Figure 4c and 7c). With file download cross-traffic, S1 and S3 exhibit the lower average peak throughput of up to 9 Mbps, while S2 up to only 6 Mbps in the unconstrained scenario. This is substantially lower compared to long video cross-traffic. When capping the available throughput per service, however, we observe that the differences between long video and file download cross-traffic scenarios can be reduced (S2) or entirely eliminated (S1 and S3). This indicates that capping allows more uniform, and thus, predictable network behavior from short video services irrespective of the characteristics of the cross-traffic. We note that with capping, the unconstrained file downloads exploit the remaining available bandwidth, leading to an increase in bandwidth utilization of 3 to 5 percentage points and higher download volumes with similar peak throughputs.

As we evaluate the application layer, we find that VMAF does not seem to be affected by cross-traffic, but only by traffic management. For the initial delays and the total stalling duration, we find that S1 is neither affected by cross-traffic nor by traffic management, S2 performs better in the file download cross-traffic scenario, while S3 performs better in the long video cross-traffic scenario, independent of traffic management. This behavior reflects differences in service design. While simple fixed rate limitation benefits some services and some scenarios, it might not be the best solution for other cases. Thus, studying how to improve management is required. In bandwidth competition scenarios, especially with cross-traffic, QoE could benefit from network management approaches that go beyond traffic-type agnostic capping. This includes using context information on traffic characteristics, considering the traffic mix, or accounting for differences in applications designs. Such cross-layer information exchange or joint network and service management would, however, require a framework to facilitate cooperation between network operators and video service providers.

7 IMPACT OF SWIPING BEHAVIOR

Lastly, we investigate how unconsumed data varies across different swiping scenarios. We define unconsumed data as the video data downloaded but not watched during a session. It is computed by subtracting the required bytes for watched videos from the total downloaded video bytes, using track information and bitrate metadata. The required video bytes are estimated using the identified track and video metadata, particularly the bitrate, where we assume uniform bitrate distribution across a video asset. Thus, the consumption computation is an approximation. Figure 8 depicts the mean observed unconsumed data for the short video services across different swiping and traffic management scenarios. Bars are colored by service, with filled bars representing the unconstrained scenario and hatched bars representing capping. The figure reveals that capping reduces the amount of unconsumed data across all swiping scenarios. The magnitude of reduction varies by service and available bandwidth, e.g., S1 and S3 benefit strongly from capping, while S2 benefits less. Figure 8a shows the unconsumed data savings when swiping occurs at deterministic intervals of 15 seconds (cf. [40]). Here, capping reduces the mean observed unconsumed data of S1 and S3 by 29% to 52%, and of S2 by 18% to 45%.



(a) Swiping in deterministic interval. **(b) Swiping at half of video.** **(c) Swiping at full video completion.**

Figure 8: Impact of swiping behavior on data overconsumption.

Smaller but noticeable reductions can be observed when swiping at half of the video. The savings are smaller with longer swiping intervals compared to the deterministic condition, as a larger portion of the downloaded video is watched. When swiping at full video playback, unconsumed data should be close to zero for all services, regardless of the traffic management scenario. While this is true for S2, S1 and S3 show non-zero unconsumed data, especially in the unconstrained scenario. This is caused by communication overheads and potential inaccuracies introduced by our methodology.

Our findings in this section indicate that capping can effectively reduce unconsumed data. These savings are higher if the swiping behavior is more frequent. Generally, unconsumed data reductions have positive effects on all stakeholders. They not only facilitate reduced network utilization, but reduce CDN/server load and related costs for video service providers. Finally, it is also highly valuable for end users with limited data plans.

8 DISCUSSION

In this study, we observe that capping increases network-level fairness between different services. Generally, the data volume and unconsumed data of short video services are both significantly reduced by capping, indicating benefits to all members of the ecosystem. For network operators, capping presents an opportunity for more efficient operations, while for content providers it can yield CDN savings. Above all, end users can rely on capping as a mechanism to preserve their data plan budgets, while also ensuring that network resources are not busy over-delivering video, but instead available for all their other needs. Comparing the effect of capping across different types of video traffic, we find that the performance of short video services becomes more robust against long video or file download cross-traffic, and we observe that capping reduces the variation in network load and the application layer traffic. This leads to generally more predictable behavior, and thus, renders managing a network simpler. In addition, network utilization fairness increases across all scenarios studied with a bandwidth cap. Nonetheless, a fixed rate capping, while being a simple network management approach, may yield a small penalty in QoE, albeit variable, according to each service's design, i.e., their adaptation and pre-loading algorithms. Generally, in terms of network performance, capping agrees with similar bandwidth regulation approaches, both directionally and quantitatively [13, 32]. For instance, the method proposed by Spang et al. [32] enhances network performance by reducing chunk throughput by up to 61% while improving QoE. Similarly, we found that capping achieves volume savings up to 45%. Furthermore, from a system design standpoint, capping is considered simpler, being decoupled from management at the application layer. Finally, most aspects of such a system have already been implemented in the past by network operators [6, 11].

As established throughout this study, our results indicate that bandwidth capping, i.e., simple fixed rate limitation, yields clear benefits for most services and scenarios studied. Nonetheless, we also observed a very small, service- and scenario-dependent penalty in QoE for end users. Our results suggest that optimizing QoE further would require network management techniques enhanced with context awareness of the video traffic type (short-form vs. long-form) and of each application's network requirements based on its system design. Applications, in turn, could incorporate awareness of capping techniques deployed at the network level. Thus, with respect to our future research goals, we aim to extend static rate caps toward dynamic rate caps or collaborative video traffic optimization, where network operators and video services synergize to optimize video traffic jointly. To do so, we envision that information regarding the traffic type and its QoE targets is exchanged, along with network resource availability and traffic profile requirements. Such cross-layer information exchange or joint network and service management would, however, require first the design and development of a framework to facilitate such cooperation between network operators and video service providers.

9 CONCLUSION

This work investigates whether capping of video flows, i.e., limiting their maximum network throughput, is beneficial from the perspectives of network operators, service providers, and end users. We perform a measurement study with real short video services, where six Android devices compete for bandwidth on a bottleneck link with two Android devices facilitating long video and file download cross-traffic scenarios. We analyze the effectiveness of capping on network utilization, bandwidth excess, and QoE. Our results indicate that capping is reasonable and beneficial for all stakeholders, as data volume and unconsumed data are reduced. Additionally, the fairness in the network improves. While some short video services maintain QoE, some sustain small QoE reduction under capping, meaning that there is room for improvement, but that improvement is not difficult to achieve. Overall, we find that the network becomes more predictable with capping, and thus, easier to manage. Consequently, the question should no longer be whether to cap or not to cap, as initially stated, but how to appropriately cap to maximize benefits, which we will explore as future work.

REFERENCES

[1] Appium. 2024. Appium Documentation. http://appium.io/docs/en/latest/ accessed 2024-09-24.

[2] Ibrahim Ben Mustafa, Tamer Nadeem, and Emir Halepovic. 2018. FlexStream: Towards Flexible Adaptive Video Streaming on End Devices using Extreme SDN. In *Proceedings of the 26th ACM International Conference on Multimedia* (Seoul, Republic of Korea) *(MM '18)*. Association for Computing Machinery, New York, NY, USA, 555–563. https://doi.org/10.1145/3240508.3240676

[3] Kjell Brunnström, Sergio Ariel Beker, Katrien De Moor, Ann Dooms, Sebastian Egger, Marie-Neige Garcia, Tobias Hossfeld, Satu Jumisko-Pyykkö, Christian Keimel, Mohamed-Chaker Larabi, et al. 2013. Qualinet White Paper on Definitions of Quality of Experience. (2013).

[4] Zhuang Chen, Qian He, Zhifei Mao, Hwei-Ming Chung, and Sabita Maharjan. 2019. A Study on the Characteristics of Douyin Short Videos and Implications for Edge Caching. In *Proceedings of the ACM Turing Celebration Conference-China*. 1–6.

[5] Android Developers. 2024. Android Debug Bridge (ADB) Documentation. https://developer.android.com/tools/adb accessed 2024-09-24.

[6] Jeremy Gillula. 2016. EFF Confirms: T-Mobile's Binge On Optimization is Just Throttling, Applies Indiscriminately to All Video. https://www.eff.org/deeplinks/2016/01/eff-confirms-t-mobiles-bingeon-optimization-just-throttling-applies accessed 2024-09-16.

[7] Jianchao He, Miao Hu, Yipeng Zhou, and Di Wu. 2020. LiveClip: Towards Intelligent Mobile Short-form Video Streaming with Deep Reinforcement Learning. In *Proceedings of the 30th ACM Workshop on Network and Operating Systems Support for Digital Audio and Video*. 54–59.

[8] Tobias Hoßfeld, Lea Skorin-Kapov, Poul E Heegaard, and Martin Varela. 2016. Definition of QoE Fairness in Shared Systems. *IEEE Communications Letters* 21, 1 (2016), 184–187.

[9] Mansoor Iqbal. 2024. TikTok Revenue and Usage Statistics (2024) - Business of Apps. https://www.businessofapps.com/data/tik-tok-statistics/ accessed 2024-09-16.

[10] ITU-T. [n. d.]. Recommendation P.1203: Parametric Bitstream-based Quality Assessment of Progressive Download and Adaptive Audiovisual Streaming Services over Reliable Transport.

[11] Arash Molavi Kakhki, Fangfan Li, David Choffnes, Ethan Katz-Bassett, and Alan Mislove. 2016. BingeOn Under the Microscope: Understanding T-Mobile's Zero-Rating Implementation. In *Proceedings of the 2016 Workshop on QoE-based Analysis and Management of Data Communication Networks*. 43–48.

[12] Theodoros Karagkioules, Dimitrios Tsilimantos, Stefan Valentin, Florian Wamser, Bernd Zeidler, Michael Seufert, Frank Loh, and Phuoc Tran-Gia. 2018. A Public Dataset for YouTube's Mobile Streaming Client. In *2018 Network Traffic Measurement and Analysis Conference (TMA)*. IEEE, 1–6.

[13] Vengatanathan Krishnamoorthi, Niklas Carlsson, and Emir Halepovic. 2018. Slow But Steady: Cap-based Client-network Interaction for Improved Streaming Experience. In *2018 IEEE/ACM 26th International Symposium on Quality of Service (IWQoS)*. IEEE, 1–10.

[14] Jeongho Kwak, Joonyoung Moon, Hyang-Won Lee, and Long Bao Le. 2017. Dynamic Network Slicing and Resource Allocation for Heterogeneous Wireless Services. In *2017 IEEE 28th Annual International Symposium on Personal, Indoor, and Mobile Radio Communications (PIMRC)*. IEEE, 1–5.

[15] Zhi Li, Anne Aaron, Ioannis Katsavounidis, Anush Moorthy, and Megha Manohara. 2016. Toward a Practical Perceptual Video Quality Metric. *The Netflix Tech Blog* 6, 2 (2016).

[16] Ricky KP Mok, Edmond WW Chan, and Rocky KC Chang. 2011. Measuring the Quality of Experience of HTTP Video Streaming. In *12th IFIP/IEEE International Symposium on Integrated Network Management (IM 2011) and Workshops*. IEEE, 485–492.

[17] Vikram Nathan, Vibhaalakshmi Sivaraman, Ravichandra Addanki, Mehrdad Khani, Prateesh Goyal, and Mohammad Alizadeh. 2019. End-to-end Transport for Video QoE Fairness. In *Proceedings of the ACM Special Interest Group on Data Communication*. 408–423.

[18] Duc Nguyen, Phong Nguyen, Vu Long, Truong Thu Huong, and Pham Ngoc Nam. 2022. Network-aware Prefetching Method for Short-form Video Streaming. In *2022 IEEE 24th International Workshop on Multimedia Signal Processing (MMSP)*. IEEE, 1–5.

[19] Jakob Nielsen. 1994. *Usability Engineering*. Morgan Kaufmann.

[20] Jan Ozer. 2017. Finding the Just Noticeable Difference with Netflix VMAF. *Streaming Learning Center* (2017).

[21] Nguyen Tien Phong, Truong Thu Huong, Pham Ngoc Nam, Truong Cong Thang, and Duc Nguyen. 2023. Joint Preloading and Bitrate Adaptation for Short Video Streaming. *IEEE Access* (2023).

[22] Anika Schwind, Michael Seufert, Özgü Alay, Pedro Casas, Phuoc Tran-Gia, and Florian Wamser. 2017. Concept and Implementation of Video QoE Measurements in a Mobile Broadband Testbed. In *2017 Network Traffic Measurement and Analysis Conference (TMA)*. IEEE, 1–6.

[23] Selenium. 2024. Selenium Documentation. https://www.selenium.dev/ accessed 2024-09-24.

[24] Michael Seufert. 2021. Statistical Methods and Models based on Quality of Experience Distributions. *Quality and User Experience* 6, 1 (2021), 3.

[25] Michael Seufert, Sebastian Egger, Martin Slanina, Thomas Zinner, Tobias Hoßfeld, and Phuoc Tran-Gia. 2014. A Survey on Quality of Experience of HTTP Adaptive Streaming. *IEEE Communications Surveys & Tutorials* 17, 1 (2014), 469–492.

[26] Michael Seufert, Raimund Schatz, Nikolas Wehner, and Pedro Casas. 2019. Quicker or Not? - An Empirical Analysis of QUIC vs TCP for Video Streaming QoE Provisioning. In *2019 22nd Conference on Innovation in Clouds, Internet and Networks and Workshops (ICIN)*. IEEE, 7–12.

[27] Michael Seufert, Nikolas Wehner, and Pedro Casas. 2018. Studying the Impact of HAS QoE Factors on the Standardized QoE Model P.1203. In *2018 IEEE 38th International Conference on Distributed Computing Systems (ICDCS)*. IEEE, 1636–1641.

[28] Michael Seufert, Nikolas Wehner, and Pedro Casas. 2019. A Fair Share for All: TCP-inspired Adaptation Logic for QoE Fairness Among Heterogeneous HTTP Adaptive Video Streaming Clients. *IEEE Transactions on Network and Service Management* 16, 2 (2019), 475–488.

[29] Michael Seufert, Nikolas Wehner, Florian Wamser, Pedro Casas, Alessandro D'Alconzo, and Phuoc Tran-Gia. 2017. Unsupervised QoE Field Study for Mobile YouTube Video Streaming with YoMoApp. In *2017 Ninth International Conference on Quality of Multimedia Experience (QoMEX)*. IEEE, 1–6.

[30] Michael Seufert, Bernd Zeidler, Florian Wamser, Theodoros Karagkioules, Dimitrios Tsilimantos, Frank Loh, Phuoc Tran-Gia, and Stefan Valentin. 2018. A Wrapper for Automatic Measurements with YouTube's Native Android App. In *2018 Network Traffic Measurement and analysis conference (TMA)*. IEEE, 1–8.

[31] Seufert, Michael and Wehner, Nikolas and Wieser, Viktoria and Casas, Pedro and Capdehourat, Germán. 2020. Mind the (QoE) Gap: On the Incompatibility of Web and Video QoE Models in the Wild. In *2020 16th International Conference on Network and Service Management (CNSM)*. IEEE, 1–5.

[32] Bruce Spang, Shravya Kunamalla, Renata Teixeira, Te-Yuan Huang, Grenville Armitage, Ramesh Johari, and Nick McKeown. 2023. Sammy: Smoothing Video Traffic to Be A Friendly Internet Neighbor. In *Proceedings of the ACM SIGCOMM 2023 Conference*. 754–768.

[33] Barbara Staehle, Matthias Hirth, Rastin Pries, Florian Wamser, and Dirk Staehle. 2010. YoMo: A YouTube Application Comfort Monitoring Tool. *New Dimensions in the Assessment and Support of Quality of Experience for Multimedia Applications, Tampere, Finland* 6 (2010).

[34] Florian Wamser, Andreas Blenk, Michael Seufert, Thomas Zinner, Wolfgang Kellerer, and Phuoc Tran-Gia. 2015. Modelling and Performance Analysis of Application-aware Resource Management. *International Journal of Network Management* 25, 4 (2015), 223–241.

[35] Florian Wamser, Michael Seufert, Pedro Casas, Ralf Irmer, Phuoc Tran-Gia, and Raimund Schatz. 2015. YoMoApp: A Tool for Analyzing QoE of YouTube HTTP Adaptive Streaming in Mobile Networks. In *2015 European Conference on Networks and Communications (EuCNC), pages=239–243*. IEEE.

[36] Shichang Xu, Eric Petajan, Subhabrata Sen, and Z Morley Mao. 2020. What You See Is What You Get: Measure ABR Video Streaming QoE via On-device Screen Recording. In *Proceedings of the 30th ACM Workshop on Network and Operating Systems Support for Digital Audio and Video*. 60–66.

[37] Yali Yuan, Weijun Wang, Yuhan Wang, Sripriya S Adhatarao, Bangbang Ren, Kai Zheng, and Xiaoming Fu. 2022. VSiM: Improving QoE Fairness for Video Streaming in Mobile Environments. In *IEEE INFOCOM 2022-IEEE Conference on Computer Communications*. IEEE, 1309–1318.

[38] Guanghui Zhang, Ke Liu, Haibo Hu, and Jing Guo. 2021. Short Video Streaming with Data Wastage Awareness. In *2021 IEEE International Conference on Multimedia and Expo (ICME)*. IEEE, 1–6.

[39] Haodan Zhang, Yixuan Ban, Xinggong Zhang, Zongming Guo, Zhimin Xu, Shengbin Meng, Junlin Li, and Yue Wang. 2020. APL: Adaptive Preloading of Short Video with Lyapunov Optimization. In *2020 IEEE International Conference on Visual Communications and Image Processing (VCIP)*. IEEE, 13–16.

[40] Yuming Zhang, Yan Liu, Lingfeng Guo, and Jack YB Lee. 2022. Measurement of A Large-scale Short-video Service Over Mobile and Wireless Networks. *IEEE Transactions on Mobile Computing* 22, 6 (2022), 3472–3488.

[41] Chao Zhou, Yixuan Ban, Yangchao Zhao, Liang Guo, and Bing Yu. 2022. PDAS: Probability-driven Adaptive Streaming for Short Video. In *Proceedings of the 30th ACM International Conference on Multimedia*. 7021–7025.

[42] Shangyue Zhu, Theo Karagioules, Emir Halepovic, Alamin Mohammed, and Aaron D Striegel. 2022. Swipe Along: A Measurement Study of Short Video Services. In *Proceedings of the 13th ACM Multimedia Systems Conference*. 123–135.

RemoteVIO: Offloading Head Tracking in an End-to-End XR System

Qinjun Jiang
University of Illinois Urbana-Champaign
Urbana, USA
qinjunj2@illinois.edu

Yihan Pang
University of Illinois Urbana-Champaign
Urbana, USA
yihanp2@illinois.edu

William Sentosa
University of Illinois Urbana-Champaign
Urbana, USA
sentosa2@illinois.edu

Steven Gao
University of Illinois Urbana-Champaign
Urbana, USA
hongyig3@illinois.edu

Muhammad Huzaifa
University of Illinois Urbana-Champaign
Urbana, USA
huzaifa2@illinois.edu

Jeffrey Zhang
University of Illinois Urbana-Champaign
Urbana, USA
jfzhang2@illinois.edu

Javier Perez-Ramirez*
Ofinno
Reston, USA
jperez-ramirez@ofinno.com

Dibakar Das
Intel Corporation
Portland, USA
dibakar.das@intel.com

David Gonzalez-Aguirre
Intel Labs
Hillsboro, USA
david.i.gonzalez.aguirre@intel.com

Brighten Godfrey
University of Illinois Urbana-Champaign and Broadcom
Urbana, USA
pbg@illinois.edu

Sarita Adve
University of Illinois Urbana-Champaign
Urbana, USA
sadve@illinois.edu

Abstract

Power consumption, and the resulting limitation to computational load, is a first-order constraint in designing comfortable all-day-wear extended reality (XR) devices that can provide rich immersive experiences. This paper concerns reducing XR device power consumption by offloading head tracking, one of the top CPU and power consumers, to a remote server. We present RemoteVIO, the first open-source end-to-end XR system that offloads head tracking (visual inertial odometry or VIO) to a remote server. Our work distinguishes itself from past studies on computation offloading in XR by properly addressing two under-explored but critical aspects: 1) a comprehensive evaluation of *user experience* in a complete end-to-end XR system and 2) a quantification of the *net power savings* on real hardware.

Through an Institutional Review Board (IRB) approved study, we find that RemoteVIO provides a satisfactory user experience under typical network conditions, but often degrades for network round trip time above 200 milliseconds (ms). We also demonstrate the first measured power savings from offloading head tracking on real hardware: compared with on-device tracking, RemoteVIO reduces CPU power by up to 52%, CPU+network power by up to 39%, and end-to-end full system power by up to 20%. Of equal importance, we examine the traditional approach of evaluating XR offloading techniques with datasets and quantitative metrics. Our results reveal that traditional head tracking metrics do not correlate with user experience, questioning the use of such metrics in XR systems research and underscoring the importance of using end-to-end systems that allow for user experience studies.

CCS Concepts

• **Computing methodologies** → **Virtual Reality**; Perception; • **Computer systems organization** → *Real-time system architecture, Power-efficient system design*; • **Human-centered computing** → *Ubiquitous and mobile computing*.

Keywords

Extended Reality, VR/AR, Head Tracking, Offloading, Low Power, Wireless Networks

1 Introduction

Recent advances in virtual, augmented, and mixed reality (VR/AR/MR), collectively referred to as extended reality (XR), have the potential to transform most industries and human activities [4–6, 12, 20]. However, with the end of conventional CMOS scaling [22, 61, 69], power consumption has become a first order design constraint to achieving the potential of XR. Power consumption directly affects an XR device's form factor, weight, and battery life, as well as the user's thermal discomfort through the battery and cooling support required.

*The work was conducted while the author was affiliated with Intel Labs.



This work is licensed under a Creative Commons Attribution 4.0 International License.
MMSys '25, March 31-April 4, 2025, Stellenbosch, South Africa
© 2025 Copyright held by the owner/author(s).
ACM ISBN 979-8-4007-1467-2/2025/03
https://doi.org/10.1145/3712676.3714442

It also determines the richness of the immersive experience through the compute capacity afforded within the power constraint. Unfortunately, there is currently an orders-of-magnitude gap between the power, compute capacity, and quality of experience of current and desired XR devices [37]. Specifically, desired compute power consumption for comfortable all-day wear XR devices is a few hundred milliwatts [67] while today's commercial devices typically consume 10s of Watts [31, 64]. The battery lives of the recent Meta Quest Pro and Apple Vision Pro are reported to be around two hours [7, 70] with the latter requiring an external battery pack [7].

A tempting approach to reduce power consumption in an XR device is hardware acceleration. However, as there are many components in an XR system (e.g., head/hand/eye tracking, scene reconstruction, reprojection or warping, spatial audio encoding and decoding, etc.) and the algorithms in these components continue to evolve in significant ways, developing a new hardware accelerator for each algorithmic advancement in every component is unsustainable. An alternative solution is to offload intensive computations to edge or cloud servers. Offloading does not suffer from the drawbacks of hardware acceleration as it is hardware independent and can readily adapt to changing algorithms. Since most XR use cases today have access to wireless networks through Wi-Fi or cellular mobile hotspots, offloading becomes an appealing strategy for enabling a low-power, mobile XR experience.

This paper explores reducing XR device power consumption by offloading head tracking computations to remote servers. Head tracking is one of the most CPU-intensive tasks (e.g., it has been shown to use 35% to 55% of all CPU cycles in the XR system [35–37]), making it an ideal target for offloading to reduce on-device power usage. However, to ensure the effectiveness of such an offloaded XR system, two critical aspects must be thoroughly evaluated.

The first critical aspect is **the end-to-end user experience**. A typical XR system generates and displays virtual content by interpreting the user's environment, actions, and movements using sensor inputs such as cameras and inertial measurement units (IMUs). We term the entire sensor to display process as the *end-to-end or E2E system*, and the user's experience of the displayed content as the *end-to-end or E2E user experience*. The E2E XR experience is intimate to users, especially in fully immersive VR, where users wear head-mounted displays (HMDs) and are immersed in a virtual world with a wide field of view displayed directly in front of their eyes. In this context, any performance degradation within the E2E system has the potential to result in a poor user experience or even physical discomfort. Therefore, ensuring an acceptable E2E user experience is the primary requirement for any useful XR system design.

Offloading head tracking presents significant challenges to the E2E user experience. Head tracking estimates the direction and position of the user's head, collectively referred to as the pose. Accurate and timely pose delivery is crucial for generating images and sounds that match the user's current viewpoint, directly impacting the user's quality of experience. Delays in the pose delivery due to high or unpredictable network latencies could potentially make the E2E user experience unacceptable.

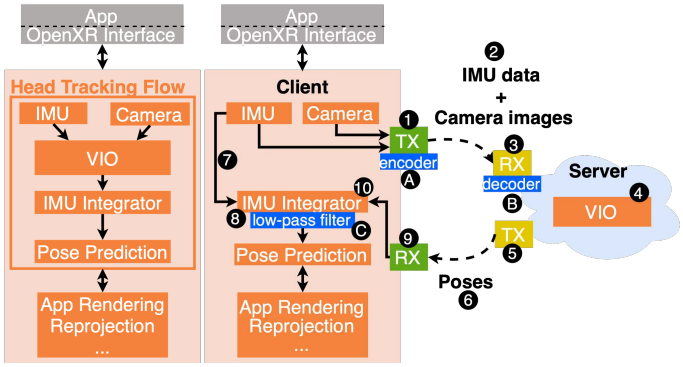
Unfortunately, existing work on offloading head tracking in XR [10, 19, 32, 43] does not assess the E2E user experience. Most work [10, 19, 43] does not build the E2E system, which eliminates the possibility of evaluating E2E user experience. Instead, these works build partial systems consisting of only components relevant to the tasks to be offloaded, and calculate metrics such as Absolute Trajectory Error (ATE) to gauge the pose accuracy against the ground truth. While suitable for evaluating algorithmic improvements, the efficacy of this approach for system-level XR research where user experience is paramount remains unclear. FleXR [32] develops and implements a more complete system with a flexible offloading architecture. However, their evaluations for head tracking in VR rely on datasets from the robotics domain and metrics like pipeline latency ($\sim$80-350 ms) and throughput (< 30 frames per second, or FPS), which do not adequately reflect the E2E user experience. Unlike previous work, our work is the first to **build an E2E immersive XR system with offloaded head tracking**, and the first to **evaluate whether offloading head tracking can preserve a satisfactory E2E user experience**.

The second critical aspect in ensuring the effectiveness of offloaded head tracking is **the net power savings**. Offloading in XR does not necessarily bring power benefits [24] because of the overhead it introduces, such as wireless communication, data serialization, and de/compression. However, existing work often infers power savings based on observed resource utilization (e.g. CPU/GPU load) rather than directly measuring the E2E system power consumption on real hardware, thereby overlooking the overhead from data transmission and codecs [10, 19, 43]. FleXR [32] reports standalone energy consumption of Ethernet with synthetic traffic but not for the E2E system. In contrast to prior work, our work is the first to **evaluate whether offloading head tracking can provide power savings on real hardware with real wireless networks.**

Overall, our work makes the following contributions.

- We design, implement, and evaluate *RemoteVIO*, the first E2E XR system that offloads head tracking (VIO) to a remote server. RemoteVIO enables the first results on the E2E user experience and power savings of an XR system with offloaded head tracking. We release our implementation as open-source[1] to motivate future work in this area to use an E2E system.
- We conduct a user study with 30 participants to evaluate the impact of offloading head tracking on the visual E2E user experience in an immersive VR environment. Across 246 trials comparing RemoteVIO experiences under a variety of network conditions with on-device head tracking, we find that RemoteVIO delivers acceptable E2E experience for typical wireless network latencies (up to about 200 ms) for 99% of the trials. Stressing the network latency further increases the number of unacceptable experiences – latencies of 400ms or beyond show unacceptable experiences for 19% of the trials.
- We study the effectiveness of the standard tracking metrics of ATE and Relative Pose Error (RPE) as proxies for E2E user experience. We find that ATE and RPE do not correlate with the E2E user experience results from our user study. For a more thorough evaluation of these metrics, we perform a sweep of various network latency values with both widely used datasets and new datasets we record specifically for VR, providing further validation of the limitations of these metrics.
- We present the first quantification of power savings from offloading VIO, including the overhead of communication on 5G

[1] https://github.com/ILLIXR/ILLIXR



(a) On-device VIO (b) RemoteVIO

Figure 1: (a) An end-to-end XR system with on-device VIO. We highlight the head tracking flow which produces high-frequency high-accuracy poses for the rest of the system. (b) *RemoteVIO* where VIO is offloaded to a remote server. We implement plugins on the client and server to handle data transmission, reception, and camera frame (de)compression.

and Wi-Fi networks and that of H.264 compression on commodity hardware accelerators [56]. We find that, for the cases we studied, RemoteVIO reduces power consumption for the CPU by up to 52% (avg 42%), for CPU+network by up to 39% (avg 20%), and for the full system by up to 20% (avg 13%).

2 Background

2.1 Head Tracking

Head tracking is a crucial component in XR that enables immersive and responsive user experiences by continuously monitoring the position and orientation (the 6-Degrees-of-Freedom or 6DoF pose) of the user's head. To achieve accurate and high-frequency tracking, XR systems typically rely on a fusion of estimates from multiple tracking techniques that operate on one or more sensors. A typical head tracking flow in XR is presented in Figure 1a. VIO algorithms use inertial measurement unit (IMU) data (acceleration and angular velocity) and camera images to generate a 6DoF pose. VIO offers good accuracy, but its frequency is constrained by the high computation latencies and the relatively low camera frequencies of tens of Hertz (Hz). Relying solely on VIO is therefore insufficient to meet the demands of the more frequent pose queries in an XR system (e.g., 120Hz or 144Hz for asynchronous reprojection or timewarp to produce the final images for display). In contrast, inertial based techniques such as IMU integration are lightweight and able to work at hundreds of Hz, yet can have bad drifting error without timely correction from the VIO poses. Combining VIO and IMU integration enables an XR system to meet both the accuracy and frequency requirements for pose queries.

Additionally, downstream XR components frequently require future pose estimates. For instance, rendering and reprojection are performed based on the anticipated pose at the moment the image will be displayed. Pose prediction algorithms address these needs by estimating the user's future position and orientation, leveraging current velocity or acceleration data under the assumption that human movements maintain momentum for short periods [27].

2.2 ILLIXR

RemoteVIO builds upon ILLIXR, an open-source full-system XR research testbed [1, 36, 37]. ILLIXR implements a typical E2E XR workflow all the way from the sensor inputs to the final display. It models computation units such as sensor capturing, head tracking, application rendering, and other major functionality in XR as separately compiled, dynamically linked plugins. The communication between plugins is centrally managed by ILLIXR's runtime.

For applications, ILLIXR supports a native interface and the recent OpenXR API (implemented with Monado [16]), thereby supporting various applications and game engines such as Godot [45] and Unreal [25]. For live experiments, ILLIXR supports several sensors for live inputs, computes on desktop and embedded class platforms (Section 4), and sends rendered pixels to the display of commercial headsets. Thus, ILLIXR offers a unique open platform for end-to-end XR research with the opportunity to study the impact of system optimizations on live XR experiences.

3 RemoteVIO

In this section, we first highlight the challenges in designing an XR system with head tracking offloaded and our solutions to each of them in RemoteVIO. We next describe our system design and implementation of RemoteVIO as an end-to-end system.

3.1 Challenges and Solutions

Challenge 1: Network Latency. XR, especially head tracking, is very sensitive to latency. Long and unpredictable network delays in wireless connections can make offloaded head tracking feel unresponsive and degrade the user experience.

Solution 1a. On-device IMU Integration. As discussed in Section 2.1, the head tracking flow uses both slow but accurate VIO and fast but error-prone IMU integration. Our design, illustrated in Figure 1b, moves the compute-intensive VIO component to the remote server ❹ while keeping the lightweight IMU integration ❿ on the device. When a new VIO pose is not available, the IMU integrator keeps generating estimated poses at hundreds of Hz IMU frequency ❽ to satisfy the high-frequency pose requests from the downstream system. It does so by integrating the IMU-reported acceleration and angular velocity ❼ into translational and rotational changes, and adds the changes on top of the last VIO pose. When the new VIO pose is received from the network, the IMU integrator updates its base pose with the new VIO pose ❿ and biases, which helps maintain the accuracy of its generated poses. This design results in the client getting VIO poses with much larger delays (than on-device VIO) due to the network latency. The IMU integration thus has to operate over a longer period, commensurate with the network latency. The key question is whether the extended integration period will affect E2E user experience in a perceptible way.[2] With our E2E system, we answer this question thoroughly with direct evaluation on the user experience (Sec. 4.2 and 5.1).

Solution 1b. Low-Pass Filtering. We observe that when the IMU integrator updates its base pose whenever a new VIO pose becomes available, a small, rapid, and unintended movement—known as jitter—often occurs in the trajectory. This happens because right

[2]The work in [19] uses similar observations but differs from our work as described in Section 6.

before and after a new VIO pose is received, the IMU integrator estimates two adjacent poses: one based on the older VIO pose and a longer period of IMU data, and the other based on the newer VIO pose, using a shorter period of IMU data. Due to network latency, the difference between the two poses can increase because of the longer accumulation of IMU integration errors, resulting in more noticeable jitter. To mitigate the jitter, we apply a low-pass filter [13] to the IMU integrated poses Ⓒ which effectively smooths out the jitter with negligible overhead.

Challenge 2: Network bandwidth. Cameras on current XR devices typically capture stereo images of reasonable resolution at frequencies in the 10s of Hz for tracking [7, 52, 71]. Transmitting them on the network can easily consume tens to hundreds of Mbps of bandwidth (e.g., 115Mbps for the stereo EuRoC dataset [11]). This single XR component could therefore use much of the available Wi-Fi bandwidth and overload 4G or 5G networks.

Solution 2. Compression of camera images. Our design therefore includes an encoder Ⓐ (decoder Ⓑ) to compress (decompress) camera data before (after) transmission (reception). Specifically, we perform H.264 compression/decompression on the raw camera stream using GStreamer [68], accelerated by NVIDIA NVENC and NVDEC [56]. The compression and decompression functions reside in the device_TX and server_RX plugins (discussed later) respectively (Ⓐ and Ⓑ in Figure 1b). We control the compression ratio by adjusting the compression bitrate. We conducted a bitrate sweep for EuRoC datasets (results omitted here due to space constraints) and found that a target bitrate of 2Mbps (4Mbps for stereo data) provides a balance between low bandwidth and small tracking errors using ATE. Therefore, we use 2Mbps bitrate for other experiments. Previous studies in robotics and autonomous vehicles have found negligible impact of moderate compression on odometry accuracy [58, 59]. However, our key concerns are whether compression leads to user-perceptible degradation and whether the codec adds significant latency and power overhead. Again, using an E2E system for our design and evaluation enables understanding the impact of the compression algorithm on final user experience and potential power benefits.

3.2 End-to-End System Design

Our system, RemoteVIO, employs a client-server model as shown in Figure 1b. VIO is relocated to the server while other components remain on the device (client). The client establishes two connections to the server – one for sending IMU data and camera images and the other for receiving poses – using TCP sockets with the default Linux settings and the TCP_NODELAY flag enabled.[3] We develop plugins to handle the transmission and reception on both the device-side (device_TX ❶ , device_RX ❾) and the server-side (server_TX ❺ , server_RX ❸) as in Figure 1b.

The device_TX ❶ plugin collects and serializes the IMU data and camera frames ❷ using Protocol Buffers [2]. It then sends the serialized message to the server via a TCP socket. The server_RX ❸ plugin receives the message, deserializes it, and feeds the IMU data and camera frames to VIO ❹ to calculate the pose. The new pose is handled by server_TX ❺ , which serializes the pose ❻ and sends it to device_RX ❾ over the network. Finally, device_RX ❾ deserializes the pose and sends it for use in IMU integrator ❿ .

We use open-source state-of-the-art CPU implementations for VIO and IMU integrator – OpenVINS [27] and GTSAM [18, 23] respectively. We expect the results to be extendable to other algorithms and implementations as well.

To create a complete XR experience, we implement our design on top of ILLIXR [36, 37]. We configure our system with the following critical components in addition to all of the components directly associated with head tracking. First, the system runs applications that produce the rendered images for display. We use two game engines – Unreal [25] and Godot [45] – which interface with ILLIXR through the Monado OpenXR implementation[16]. Second, the system executes asynchronous reprojection, also known as timewarp [72]. This process uses the predicted pose (discussed in Section 2.1) for the next vsync (display time) to reproject the rendered frame, correcting for the latency that has occurred since the renderer received its initial pose estimate. Reprojection is crucial as it significantly reduces visual latency, making it a vital component for the XR visual subsystem by continually updating the frame to align with the user's latest pose.

4 Evaluation Methodology

4.1 Experimental Platforms

High-end and embedded class devices. We conduct our experiments on two hardware platforms as the client. We use a high-end desktop PC with an Intel(R) Core i9-10900K CPU and a discrete NVIDIA GeForce RTX 3090 GPU. The powerful desktop provides sufficient resources for the on-device VIO configuration so that any difference in user experience or quality metrics with RemoteVIO can be attributed to offloading. For power measurements, however, we use an embedded class system, NVIDIA Jetson AGX Xavier development board [55], hereafter referred to as "Jetson," to represent a device with a power profile similar to modern headsets (Section 4.4).

Cloud-mode and Local-mode. We configure the *cloud-mode* to represent the realistic scenario for a system with VIO offloaded. The XR client is either the desktop PC or the Jetson. The offload server runs on a GPU-enabled server in the Amazon Web Services (AWS) data center [8] that is closest to our client location, a distance of around 300 miles. The client is connected to either Wi-Fi or 5G. For Wi-Fi, we connect to our on-site network, which delivers around 600 Mbps and 400 Mbps for upload and download respectively as measured through Speedtest and 21.5ms mean packet round-trip-time (RTT) as measured through ping [39].

For 5G experiments, we USB-tether a 5G-capable phone (Samsung S22 or Motorola Edge plus 2022) and share its network with the XR device. The phone is connected to TMobile's 5G Ultra Capacity which is based on the mid-band 5G, delivering about 70 Mbps upload and 300 Mbps download with an average of 63.7ms RTT to the cloud server.

To systematically and repeatably study the impact of network latency and bandwidth on RemoteVIO, we also set up a proxy system as follows. Both the client and the server run on the same high-end

[3]TCP is an appropriate choice for the communication protocol as VIO requires reliable, in-order delivery of the compressed image stream. There are many TCP variants and UDP-based protocols that provide similar service, which may be useful to explore in the future particularly when low latency transport is required and packet dropping is tolerable. As demonstrated here, however, the default TCP algorithm in Linux (the CUBIC algorithm) is sufficient for our purposes.

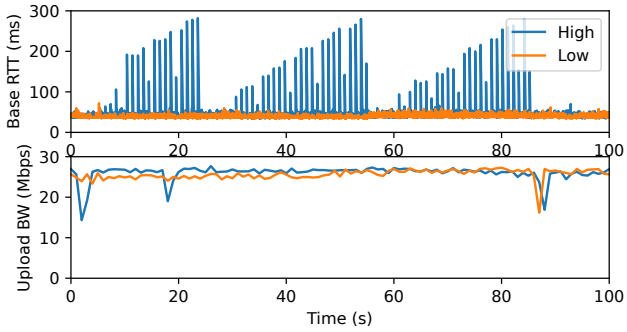


Figure 2: Network RTT and upload bandwidth for traces emulating 5G under low and high variability. Mean RTT for 5G-low is 39.36ms. 5G-high has a 49.84ms mean with spikes up to 282ms.

desktop PC and communicate through sockets binding to localhost. We either use the Linux Traffic Control (TC) to emulate the communication network based on fixed latency and bandwidth constraints or DChannel's trace-driven 5G emulator [63] to emulate 5G based on real-world traces. We call this mode the *local-mode*.

The user study in Section 4.2 is conducted under eight network conditions: live Wi-Fi and 5G in cloud-mode, and six network conditions in local-mode to ensure a more controlled experimental environment. Specifically, we use TC to set four levels of fixed network RTT of 50ms, 100ms, 200ms, and 400ms. Previous work has shown that the RTT from user clients to the nearest data centers (on WiFi or cellular) is within 200ms on most places on earth in most situations [17]. Our user study thus provides a good coverage of the spectrum of common network conditions, as well as more extreme network scenarios. Additionally, we use the DChannel emulator [63] to emulate a 5G network and examine the impact of 5G network dynamicity on RemoteVIO. The emulator delays traffic based on time-varying RTT and bandwidth values from a trace. For our study, we select the LowBand-Stationary trace, representing Verizon's 5G network in a stationary indoor setting with full signal strength — typical of XR end-user conditions. From this trace, we extract two 100-second segments with different RTT profiles to represent good (5G-low) and poor (5G-high) conditions. Figure 2 shows the RTT and upload bandwidth over time for both traces.

Sensors and Display. We use a Valve Index headset [71] as the display and connect it to our desktop PC and the Jetson.[4] For experiments with live sensor inputs, we attach a ZED mini camera [65] to the front of the headset. As the user walks around in our lab space, the camera feeds the system with IMU samples at 500Hz and stereo grayscale images of 672 × 376 pixels at 30 frames per second (FPS) in real-time, and the headset displays frames at 144Hz.

4.2 User Study

To understand users' perception of offloading VIO in the immersive VR environment, we conducted an IRB-approved user study with a total of 30 participants (13 females and 17 males, aged 21-57, recruited via IRB-approved advertisement on our campus social media and mailing lists). 8 participants have no prior XR experience, while 19, 2, and 1 interact with XR rarely (a few days per year), occasionally (a few days most months), and frequently (several days most weeks), respectively. All participants have normal or corrected-to-normal vision.

During the study, participants wore a Valve Index headset configured as described in Section 4.1 and were presented with two VR applications built on Unreal: Zen Garden (featuring a beautiful mansion and natural environment that the user could walk in) [21] and Flight Simulator (developed within our lab, with the user controlling an airplane). We adopted a Comparative Category Rating (CCR) method [34]. For each application, participants had a random subset of eight trials [5] where each trial contained two experiences under two setups: **baseline experience** performed VIO on the client XR device without offloading and **RemoteVIO** offloaded VIO under one of eight network conditions (Section 4.1). The participants were initially instructed to perform movements across all three axes of rotation and translation, followed by freely exploring Zen Garden and piloting in Flight Simulator. After each trial, participants were asked to fill out a questionnaire to assess their perception of the differences between the two experiences. The questionnaire contains the following questions:

- Q1: Did you find any difference in the two XR experiences? (Yes/No)
- Q2: Which experience did you like better? (Experience 1/2)
- Q3: For the experience ranked worse, how significant is the degradation in comparison to the better experience?
  - (a) Only slightly worse or almost the same
  - (b) Worse but the experience is acceptable to me
  - (c) Much worse and not acceptable to me
- Q4: What caused the perceived degradation? (Jitter, lag, jerkiness, drift, lost tracking, others)[6]

Importantly, participants were kept unaware of which experience was the baseline and which was RemoteVIO to avoid biasing their responses. The order of the two experiences in each trial was randomized as well for this purpose.

We categorized the responses to the questionnaire and assigned a numeric rating to the RemoteVIO experience under test:

- Rating 4: No worse than the baseline or indistinguishable: when the user answers "No" to Q1; or when the user answered "Yes" to Q1, but deemed the RemoteVIO experience to be better in Q2.
- Rating 3: Slightly worse than or almost the same as the baseline: when the user answered "Yes" to Q1 and (a) to Q3.
- Rating 2: Worse than the baseline but the experience is acceptable: when the user answered "Yes" to Q1 and (b) to Q3.
- Rating 1: Much worse than the baseline and the experience is unacceptable: when the user answered "Yes" to Q1 and (c) to Q3.

During the trials with live Wi-Fi or 5G, we monitored the network condition and noticed one case of unusual latency spike with Wi-Fi (1566ms RTT on average with a maximum of 3783ms). The user

[4]Ideally, experiments would be conducted directly on a standalone XR headset. However, state-of-the-art XR headsets are closed and proprietary, preventing a customized XR runtime from running directly on their hardware. Additionally, collecting power measurements on these platforms is not feasible due to proprietary restrictions.

[5]We limited the number of trials per participant to keep the study at a reasonable length and prevent fatigue.

[6]The different artifacts are explained in the questionnaire. Research staff are available for questions during the study as well.

experience for this trial was rated as much worse and not acceptable, motivating better adaptive solutions for edge cases.

Overall, we conducted 248 trials with a total of 496 experiences (248 each for RemoteVIO and baseline). Of the 496 experiences, users reported "Lost tracking" in two cases. This was anomalous (and unrelated to offloading). As these experiments cannot be repeated to determine/fix the source of the anomaly, we dropped the two trials from our reported results in Section 5.1.

4.3 Standard Tracking Accuracy Evaluation

While user studies are the gold standard for evaluating XR systems [3, 49, 74], they are expensive, time-consuming, and require significant engineering to build usable E2E systems. Consequently, we aim to assess the impact of offloading on standard metrics to determine if these metrics can serve as proxies for user experience when evaluating the performance of offloading head tracking.

Metrics. We report Absolute Trajectory Error (ATE) and Relative Pose Error (RPE) (separately, for translation and rotation)[78], standard metrics used in the literature to evaluate the accuracy of head tracking [9, 19, 33, 38, 48, 60, 66, 73]. ATE takes the Root Mean Square Error (RMSE) of positional difference between the estimated and ground truth trajectories at corresponding time steps. RPE, focusing on the relative movement, is measured by taking the RMSE of the difference between the motion vector of corresponding pose pairs. For each experiment, we collect the IMU integrated poses at the IMU frequency and calculate the ATE and RPE with respect to the ground truth. We use the evo package [29] to align the trajectories and then measure the ATE and RPE.

Datasets. We use three Vicon Room 1 and three Vicon Room 2 datasets from the widely used EuRoC MAV Dataset [11] recorded with drones, referred to as V1/V2_{01,02,03} respectively. The 01, 02, and 03 each represent a certain level of tracking difficulty (easy, medium, difficult) based on the drone speed, moving patterns, lighting conditions, blurriness in images, etc. These datasets contain stereo grayscale images of 752 $\times$ 480 pixels at 20 FPS and synchronized IMU measurements at 200Hz. Because the EuRoC datasets are recorded by drones and may not represent realistic movements of a person wearing an XR headset, we also record six additional datasets in two different computer labs using a ZED mini camera attached to the front of a Valve Index VR headset. The ZED mini camera captures stereo images of 672 $\times$ 376 pixels at 30 FPS and IMU measurement at 500Hz. The recorded datasets cover common use scenarios in VR: (1) "walk" where the user is walking slowly and looking around at an interesting site; (2) "static" where the user is sitting and looking at some monitor (e.g., website browsing, video gaming); (3) "game" where the user is playing some active VR game that involves frequent body movements (e.g., fitness games, dodging a bullet); (4) "slow" where the user is moving slowly and smoothly in both rotational and translational motions; (5) and (6) "fast1" and "fast2" where the user's movements are intense and fast in order to intentionally stress the system.

Thus, in total, we use 12 datasets – 6 from EuRoC and 6 from XR experiences. However, the latter six lack the high quality ground truth information usually obtained from motion capture systems. For assessing tracking accuracy for these trajectories in offloaded configurations, we use the poses generated by the on-device IMU integrator, without offloading, as a proxy for the ground truth.

	WiFi	5G
Airplane Mode (Network disabled)	0.658W	
Standby Mode (Network enabled, no traffic)	0.819W	0.909W
Activation Mode (Transmitting small traffic)	0.943W	1.328W
Offloading Mode (RemoteVIO running in cloud-mode)	1.253W	1.734W

Table 1: Wireless power consumption across network modes.

Latency Sweep. We perform a network latency sweep from 0 to 1000ms over all the 12 datasets in the local-mode.

We find that the ATE/RPE measurements are not affected by the XR application that runs on the device, so for our dataset experiments we use GLdemo, a lightweight OpenGL-based VR application available in the ILLIXR project.

Correlating with user experience. To explore the relationship between ATE/RPE and user experience, we record IMU, camera streams, and poses from RemoteVIO trials in the user study. We replay the IMU and camera streams on the baseline system (on-device VIO) to collect a reference trajectory as a proxy for ground truth and compute ATE/RPE for the RemoteVIO trials. We plot the ATE/RPE against the corresponding user ratings in a scatter plot and calculate Spearman's correlation coefficient [62] to evaluate the relationship between these metrics and actual user experience.

4.4 Power Consumption

We conduct power measurements on the Jetson using GLdemo and Platformer (a stylized, vibrant environment with floating platforms and simple geometries) [28] with nine datasets (six EuRoC datasets and three VR datasets) for repeatability. For each combination, we measure the power consumption of the Jetson with VIO running onboard or offloaded to the AWS server. To resemble the power envelope of commercial XR devices, we configure the Jetson to operate at a CPU frequency of 2.2 GHz, a GPU frequency of 1.3 GHz, a memory controller frequency of 1.06 GHz, and display frequency of 60Hz.

During these experiments, we continuously monitor the Jetson's power rails to measure the power consumption of various subsystems, including the CPU, GPU, DDR (DRAM), SoC (on-chip microcontrollers, encoder and decoder accelerators), and SYS (display ports, storage, and I/O) [37, 57]. The aggregate of these values yields the total power consumption.

Since our intended scenario is a wireless headset, we need to characterize the power consumption of data transmission over the wireless network. To achieve this goal, we tether the Jetson to a Motorola Edge plus 2022 and read the phone's power rails [54] at various modes as shown in Table 1.[7] To isolate the wireless communication power, we maintain the phone in an idle state without running any applications and ensure that the display remains off.

[7]The Jetson has no built-in wireless capability but can be equipped with a Wireless Network Card to enable access to Wi-Fi; however, this makes it hard to isolate network power from the rest of the system power. Therefore, we choose to use the tethered phone for both Wi-Fi and 5G power measurements.

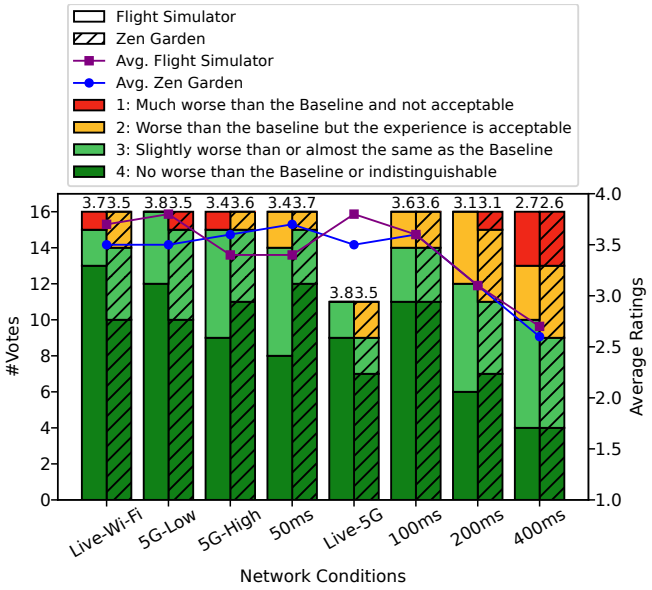


Figure 3: Users' ratings for RemoteVIO compared to the Baseline at different network conditions for Flight Simulator (solid left bars) and Zen Garden (right hatched bars). On top of each bar is the average rating for that application at that network condition. Higher numbers indicate better experiences. Additionally, the average ratings are plotted as line graphs, showing very similar average ratings when the average RTT is below 100ms. However, as the network RTT increases beyond 100ms, the ratings worsen significantly.

Each measurement lasts for one minute, during which we sample instantaneous voltage and current every 100ms. Subsequently, we calculate the average power consumption in Watts (W).

Since current XR devices require access to the Internet to use most features, we safely assume that the radio will be kept on and transmitting data even without VIO offloading. Therefore, the wireless power consumption for an XR device without VIO offloading will be similar to the power in activation mode, as shown in Table 1. To calculate the full system power for on-device VIO, we add the activation mode power to the power measured on the Jetson. For the scenarios where VIO is offloaded, we measure the wireless power in parallel as we measure the power on the Jetson.[8]

5 Results

5.1 User Experience From a User Study

5.1.1 Impact of Network Latency. Figure 3 reports the results of our user study described in Section 4.2. It covers the outcomes of 246 total trials comparing RemoteVIO with the baseline system (on-device VIO) across two applications and eight different network scenarios involving 30 users. For each network condition, except Live-5G, we conduct 16 trials per application. Only 11 trials are completed for Live-5G due to the loss of access to TMobile's 5G Ultra Capacity device midway through the study. We also report the average ratings on top of each bar. Higher numbers mean better ratings of the experience.

[8]We report only the 5G power numbers in the paper. For WiFi, the offloading mode has an average incremental power consumption of 0.3 W over activation mode, compared to 0.4 W for 5G, indicating a lesser impact on overall power savings.

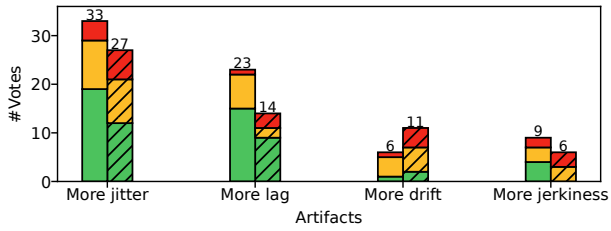


Figure 4: Users' votes on the artifacts that degrade their experience. The legend is the same as Figure 3.

Figure 3 shows that of all 246 RemoteVIO experiences, 209 (85%) are deemed indistinguishable from, similar to, or only slightly worse than the baseline experience (dark and light green bars at the bottom, rating 4 and 3); 27 (11%) are worse but the experience is still acceptable (yellow bars, second from the top, rating 2); and only 10 (4%) are deemed unacceptable (red bars on top, rating 1). Of the 10 unacceptable experiences, 8 occur at high RTT – one at 200ms, six at 400ms, and one for Live-Wi-Fi where the average/peak application latency is 1566ms/3783ms (an unusually high network congestion scenario).

Considering the impact of latency more specifically, for average network RTT no more than 200ms (5G-low, 5G-high, 50ms, 100ms, 200ms, and 54 live cases), 211 out of 214 experiences (99%) are acceptable. The experiences preserve acceptable quality even under scenarios where there are dynamic variations of RTT, such as 5G-low, 5G-high, and live Wi-Fi and 5G. A one-way ANOVA test [53] shows no significant difference in their mean ratings, with a significance level of 5%. In contrast, experiences with 400ms or higher RTT (including one Live case) see dramatic decrease in acceptable cases (79%). The one-way ANOVA test [53] has a p-value of less than 0.02 when including user responses with network RTT greater than or equal to 400ms, indicating that the ratings are indeed statistically much worse and the experience is degraded significantly at 400ms and above.

Overall, our user study shows that RemoteVIO provides a satisfactory experience until 200ms RTT even with significant variability in the latency (as in the 5G traces), and starts to degrade significantly at 400ms and beyond.

5.1.2 Reasons for Experience Degradation. In the user study questionnaire, we ask participants to identify the reasons that contribute to the degradation of their experience when they rate one experience as worse than another. Figure 4 presents the responses. The reasons in decreasing order of the total votes (combined across all applications) are more jitter (46%), more lag (29%), more drift (13%), and more jerkiness (12%).

The results have several implications. First, jitter and lag are the most perceptible artifacts when VIO is offloaded. Future efforts to improve systems with offloaded head tracking should prioritize reducing these two artifacts. Second, the effects of these artifacts on user experience are not well captured by commonly used metrics such as ATE and RPE, as will be quantitatively shown in Section

5.2.1. Therefore, developing a quantitative metric that better correlates with user experience and can accurately detect and quantify these artifacts is necessary. Finally, we observe that for a majority of cases where users report experiencing "more jitter" and "more lag," the rating for the overall experience is "slightly worse than or almost the same as the Baseline." This suggests that users have a relatively high tolerance for jitter and lag. It would be valuable to validate this finding using a broader range of applications and determine the threshold of perceptibility and tolerance for these artifacts in a way that can be captured by the quantitative metrics.

5.2 Standard VIO Metrics

5.2.1 Correlation between ATE/RPE and user experience. Figure 5 provides scatter plots showing the users' ratings of RemoteVIO experiences alongside the corresponding computed ATE/RPE (using the methodology in Section 4.3).

The figures show no obvious correlation between ATE/RPE and user ratings of their experiences, with a Spearmans' correlation coefficient of less than 0.4 for all cases. For example, a user rating of 1 (unacceptable) sees ATE value of 8.9 centimeters (cm)/1.6 degree for a 400ms RTT case, while a user rating of 4 (the best) shows much higher ATE value of 14 cm/2.8 degree for a 50 ms RTT case. The Live-Wi-Fi unacceptable case in Figure 3 with unusually congested network and high latency shows ATE (RPE) values of only 6.7 cm (2.14 cm) and 0.9 degree (0.67 degree). This in fact corresponds to the lowest level of ATE and RPE we observe from the latency sweep in the next section and clearly demonstrates the unsuitability of ATE and RPE as metrics for XR systems research.

5.2.2 Impact of Latency on ATE/RPE. Although the previous section already calls to question the validity of ATE/RPE as proxies for user experience in evaluating RemoteVIO, we report the impact of network latency on these widely used metrics for completeness. We use the local-mode with TC injected latency and perform a latency sweep from 0 to 1000ms. Also, we use both the widely adopted EuRoC datasets and our own recorded datasets to better represent XR use cases. Though we expect network latencies to typically remain below 200ms [17, 63], the higher latency sweep enables understanding of the robustness of the system and the metrics' behavior in presence of system stressors such as remote server congestion or high network RTT due to device mobility. As discussed in Section 4.3, we do the latency sweep for 12 datasets, during which we also observe the visual quality on a monitor. [9]

Figure 6 shows the ATE, both translational and rotational, for different injected latencies in local-mode (Section 4.1) for all 12 datasets. (RPE is omitted for space and has similar trends.) We first observe that the ATE values at 0ms (i.e., no injected latency) are dataset dependent and cover a wide range from 3.47cm (V2_02) to 17.8cm (slow1). The visual experience, however, is similarly smooth and pleasant for all cases. Thus, the results so far do not provide a clear ATE threshold that can be used to discriminate between acceptable and unacceptable visual experiences.

We next try to determine how *increases* in ATE (rather than absolute values) correlate with rising latency and impact on visual quality. We see that the increase in ATE with increasing latencies is also dataset dependent. For example, the ATE value increases extremely slowly for V1_01 (by 1cm and 0.01 degree respectively when the injected latency is 1000ms) and much faster for fast2 (by almost 10cm and 2.2 degrees when the injected latency is 1000ms). Other datasets fall in between the two extremes.[10] Nevertheless, Figure 6 shows that, overall, the range of ATE at different latencies stays mostly within the range seen at 0ms. However, the visual experience indicates that latencies above 200ms typically result in noticeable degradation. This degradation in the visual quality is corroborated by the user study results, but is not revealed by obvious changes in ATE.

The results in this section show that we cannot reliably use ATE or RPE to predict the user experience in an XR system. That is, there is no clearly identifiable ATE/RPE threshold or ATE/RPE-indicated latency value that guarantees a smooth user experience. This is because ATE and RPE focus more on the overall similarity of trajectories and average performance over the entire path, whereas artifacts like jitter are sporadic, subtle, and often missed by these metrics. These results underscore the need for either new metrics or full E2E system evaluation with a user study to understand the impact of offloading VIO on XR user experience.

5.3 Power Savings from Offloading VIO

Figure 7 shows the power consumption of the Jetson for two applications and nine datasets. The figure compares the baseline or on-device VIO (left bar) and RemoteVIO where VIO is offloaded to AWS via 5G (right bar). Results when offloading via Wi-Fi are similar. Each bar shows power consumption distribution across different hardware components. At the top of the right bars, we show the percentage reduction in power for RemoteVIO for the full system and the CPU+network component (separated by /).

RemoteVIO leads to a significant reduction in power consumption, with an overall system decrease of 13% and an average reduction of 30% in the CPU plus network (5G) components. The most substantial impact is observed in the CPU, with an average reduction of 42%, which aligns with expectations since VIO is primarily CPU-bound.

In addition to the reduction in CPU power consumption, a slight increase in the power usage of GPU and other components is observed in some cases. This occurs because offloading VIO frees up CPU resources for system-level tasks, such as orchestrating memory transfers and synchronization between hardware components. As a result, the performance of other components is enhanced. For example, we observe an 18% increase in the rendering frame rate for Platformer, and the GPU power consumption increases accordingly by 175mW on average. The SYS component also exhibits a small average power increase of 75mW. While we lack visibility into the specific components within SYS, it is reasonable to speculate that the increased GPU and SoC (discussed in the next paragraph) power contribute to this overall rise.

[9] Viewing a dataset experiment on a headset usually causes discomfort because the sensor data is independent of the user's motion. Our observations of the visual quality on a monitor are to provide an informal assessment of the impact of latency for an experiment (the formal user study is already reported in Section 5.1).

[10] One exception is for "static" where the rotational error increases by 5 degrees from 0ms to 1000ms.

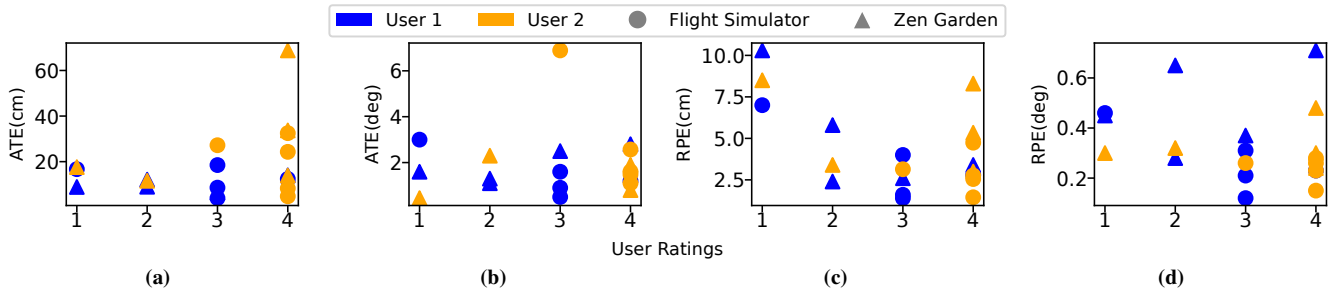


Figure 5: The relationship between two users' ratings of the experiences (in blue and orange) and (a) translational ATE, (b) rotational ATE, (c) translational RPE, (d) rotational RPE, for two applications (circle for FlightSimulator, triangle for ZenGarden). The scatter plots show no strong monotonic relationship (Spearman's correlation coefficient < 0.4) between the metrics and the user experience, suggesting ATE/RPE are inappropriate for evaluating head tracking in XR systems.

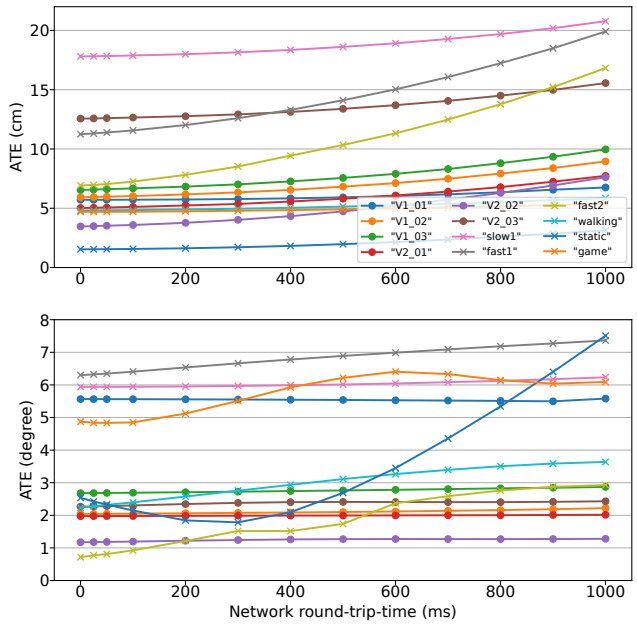


Figure 6: ATE (translational and rotational) across datasets with increasing emulated network RTT in Local-mode.

RemoteVIO also introduces power usage for image compression and wireless communication. The power for compression is categorized under SoC. Across all experiments, there is a 4.2% increase in SoC power with an absolute value of 73 mW. The wireless communication power is measured as described in Section 4.4. Incrementally beyond the activation mode (Section 4.4), RemoteVIO adds on average around 0.4 W to 5G power (and 0.3 W to Wi-Fi power), which is small compared to the total power savings of 2.2 W. Thus, RemoteVIO can still bring considerable power savings even accounting for compression and wireless communication power.

5.4 Summary of Results and Implications

In summary, our comprehensive evaluation of RemoteVIO addresses the two questions raised in Section 1: satisfactory user experience can be maintained under typical network conditions (i.e., network RTT below 200ms with variability) and offloading head tracking is an effective approach for reducing power consumption on XR devices. However, we observe significant degradation in user experience when network RTT exceeds the typical range. This decline in quality is not well reflected in the latency sweep experiments with datasets and quantitative metrics. As highlighted in Section 5.2, the increases in ATE and RPE with rising network RTT are minimal (also reported in [19]) and exhibit variability across different datasets. Such behaviors of ATE and RPE may lead researchers to overly optimistic conclusions about their system designs. Furthermore, ATE and RPE fail to capture key artifacts, such as jitter and lag, that contribute to user experience degradation. These findings emphasize the importance of conducting end-to-end system evaluations and user studies to ensure the practicality of XR system optimizations.

6 Related Work

Section 6.1 focuses on studies related to offloading head tracking in XR, highlighting how our approach differs from those works. Section 6.2 briefly discusses work on offloading other XR components.

6.1 Offloading Head Tracking in XR

SLAM-share [19] proposes offloading substantial portions of simultaneous localization and mapping (SLAM) to the cloud in a multi-user AR scenario. The SLAM-share design shares similarities with ours in utilizing IMU integration on the device to compensate for network latency. However, its evaluation of the tracking quality is conducted by calculating ATE on a standalone SLAM system without considering the user experience. The results indicate that ATE increases from 5.91 cm to 6.58 cm as network RTT rises from 0 to 1000 ms, based on a dataset from the robotics domain. The findings align closely with our own results on similar datasets (the six EuRoC datasets) in Figure 6. Yet, we show that ATE can behave very differently on datasets that are XR specific (the six datasets we collected). The user study on the E2E RemoteVIO further demonstrates that the experience is significantly degraded when the network RTT goes beyond 200 ms and ATE is not well correlated with the user experience. Therefore, user studies with E2E XR systems are crucial. (This observation also applies to non-offloading research where ATE or similar pose error metrics are commonly used to evaluate on-device tracking or pose prediction techniques for XR [9, 33, 38, 48, 60, 73].) Our study further distinguishes itself by evaluating the power consumption of the resulting E2E system on real hardware. In contrast,

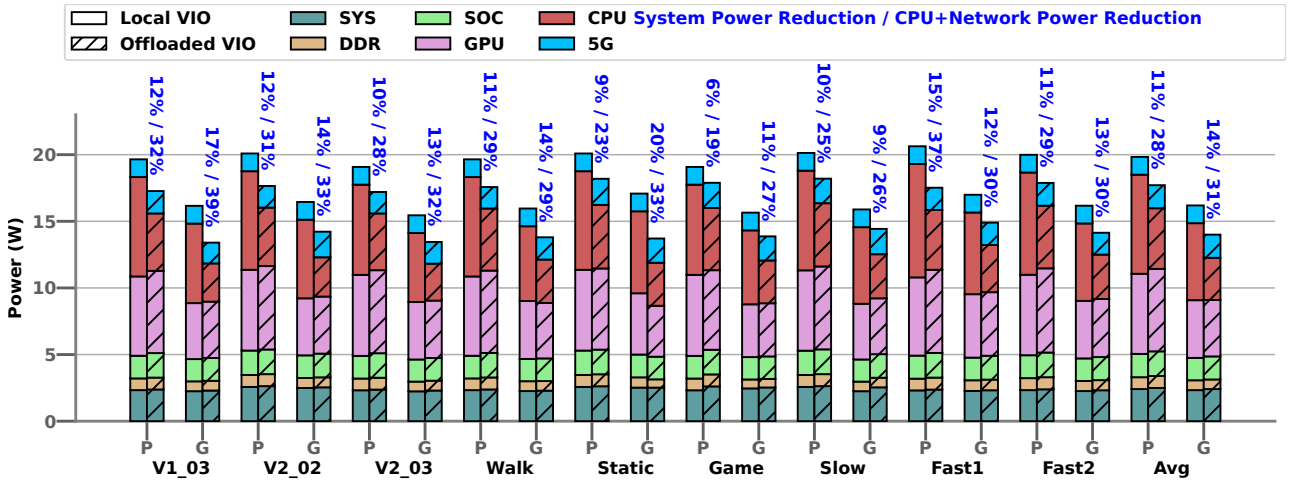


Figure 7: Power consumed on the Jetson with Baseline or on-device VIO (left bar) and RemoteVIO (right bar), for two applications, Platformer (P) and GLdemo (G), and nine datasets. Each bar shows the division of power into different hardware components. On top of the right hatched bars is the System Power Reduction and the CPU+Network Power Reduction, separated by /.

SLAM-share speculates the power savings based on the CPU usage and does not account for the communication and codec overhead.

Several other studies [10, 32, 43] have also explored offloading tracking tasks, such as SLAM or VIO. However, none evaluates the end-user experience. Additionally, the network conditions assumed are often constrained, with the server and client usually located close to each other and sharing low network RTT and high bandwidth. In contrast, our evaluation considers challenging real-world scenarios with live networks to cloud servers. Our offloading solution can also be seamlessly deployed on edge servers.

6.2 Offloading Other Tasks in XR

Besides head tracking, there is prior work on offloading other heavy components in XR to the edge or cloud for computational and power efficiency. Some popular targets for offloading are rendering [3, 26, 41, 44, 47–49, 51], object detection, recognition, and tracking[14, 15, 30, 42, 46, 50, 75–77], depth estimation [42, 43], and 3D scene reconstruction [40]. These components introduce unique challenges and power-saving opportunities compared to head tracking. For instance, offloading rendering is bandwidth-intensive due to high frequency and large frame sizes, and primarily reduces GPU power usage. Offloading different components can also be complementary; freed CPU resources from offloading head tracking, for example, can be reallocated to mesh decoding required by scene reconstruction offloading. Unfortunately, only a few existing works have properly evaluated user experience (e.g., [3, 26, 49]). Instead, most evaluations rely on task-specific quantitative metrics obtained from incomplete systems, which may not adequately reflect the impacts on the user experience, as demonstrated by our work. Meanwhile, given the extreme power constraints of mobile XR devices, it is important to ensure that the power overhead of offloading, along with any on-device computations required to mask offloading impacts [44, 48, 49], does not violate the power budget of the device. Our work offers an E2E system and a methodology to facilitate such work.

7 Conclusion and Future Work

This paper presents RemoteVIO, an end-to-end XR system that offloads the most computationally intensive part of head tracking. Our extensive user study reveals new findings. Offloaded head tracking delivers satisfactory user experience for typical wireless network conditions; however, contradicting previous work that does not measure E2E impacts, our study demonstrates that user experience deteriorates as network latencies increase, with noticeable degradation occurring at lower latency levels than those reported in prior work. Of equal importance, we find that widely used standard quantitative metrics are not well correlated with user experience. We also show power savings measured on real hardware, a significant result since power consumption is a first order obstacle to integrating more compute in modern headsets and a key motivator for offloading.

Future directions include addressing privacy concerns emerging from transmitting camera images of users' surroundings over the network, co-designed offloading of various XR components, supporting multiple users on the XR server, evaluating AR and MR workloads, and implementing adaptive techniques to enhance system robustness. Each of these directions presents distinct performance tradeoffs. We believe an E2E system will be essential for understanding these tradeoffs. To facilitate progress, we have made RemoteVIO open-source with modular and easily extendable offloading interfaces. Nevertheless, building a usable end-to-end XR system and performing extensive user studies for new optimizations is difficult. We therefore advocate the development of XR-specific metrics, input datasets, and application benchmarks that can reliably predict how optimizations affect user experience. These tools are crucial for advancing practical XR research and enabling real-world applications.

8 Acknowledgments

We thank our shepherd Mario Montagud Climent and other anonymous reviewers for their valuable feedback. This work is supported in part by the National Science Foundation under grants 2120464 and 2217144, the IBM-Illinois Discovery Accelerator Institute (IIDAI), and gifts from Cisco and T-Mobile.

References

[1] 2021. ILLIXR Illinois Extended Reality testbed. https://illixr.org.

[2] 2022. Protocol Buffers. https://developers.google.com/protocol-buffers.

[3] Ahmad Alhilal, Ze Wu, Yuk Hang Tsui, and Pan Hui. 2024. FovOptix: Human Vision-Compatible Video Encoding and Adaptive Streaming in VR Cloud Gaming. In *Proceedings of the 15th ACM Multimedia Systems Conference* (Bari, Italy) *(MMSys '24)*. Association for Computing Machinery, New York, NY, USA, 67–77. https://doi.org/10.1145/3625468.3647612

[4] Sepehr Alizadehsalehi, Ahmad Hadavi, and Joseph Chuenhuei Huang. 2020. From BIM to extended reality in AEC industry. *Automation in construction* 116 (2020), 103254.

[5] Ahmed Alnagrat, Rizalafande Che Ismail, Syed Zulkarnain Syed Idrus, and Rawad Mansour Abdulhafith Alfaqi. 2022. A review of extended reality (XR) technologies in the future of human education: Current trend and future opportunity. *Journal of Human Centered Technology* 1, 2 (2022), 81–96.

[6] Christopher Andrews, Michael K Southworth, Jennifer NA Silva, and Jonathan R Silva. 2019. Extended reality in medical practice. *Current treatment options in cardiovascular medicine* 21 (2019), 1–12.

[7] Apple. 2023. Apple Vision Pro. https://www.apple.com/apple-vision-pro/specs/.

[8] Amazon AWS. 2024. Amazon EC2 G4 Instances. https://aws.amazon.com/ec2/instance-types/g4/.

[9] Armand Behroozi, Yuxiang Chen, Vlad Fruchter, Lavanya Subramanian, Sriseshan Srikanth, and Scott Mahlke. 2024. SlimSLAM: An Adaptive Runtime for Visual-Inertial Simultaneous Localization and Mapping. In *Proceedings of the 29th ACM International Conference on Architectural Support for Programming Languages and Operating Systems, Volume 3*. 900–915.

[10] Ali J Ben Ali, Marziye Kouroshli, Sofiya Semenova, Zakieh Sadat Hashemifar, Steven Y Ko, and Karthik Dantu. 2022. Edge-SLAM: Edge-assisted visual simultaneous localization and mapping. *ACM Transactions on Embedded Computing Systems* 22, 1 (2022), 1–31.

[11] Michael Burri, Janosch Nikolic, Pascal Gohl, Thomas Schneider, Joern Rehder, Sammy Omari, Markus W Achtelik, and Roland Siegwart. 2016. The EuRoC micro aerial vehicle datasets. *The International Journal of Robotics Research* (2016). https://doi.org/10.1177/0278364915620033

[12] Leonor Adriana Cárdenas-Robledo, Óscar Hernández-Uribe, Carolina Reta, and Jose Antonio Cantoral-Ceballos. 2022. Extended reality applications in industry 4.0.–A systematic literature review. *Telematics and Informatics* 73 (2022), 101863.

[13] Géry Casiez, Nicolas Roussel, and Daniel Vogel. 2012. 1 € Filter: A Simple Speed-Based Low-Pass Filter for Noisy Input in Interactive Systems. In *Proceedings of the SIGCHI Conference on Human Factors in Computing Systems* (Austin, Texas, USA) *(CHI '12)*. Association for Computing Machinery, New York, NY, USA, 2527–2530. https://doi.org/10.1145/2207676.2208639

[14] Kaifei Chen, Tong Li, Hyung-Sin Kim, David E Culler, and Randy H Katz. 2018. Marvel: Enabling mobile augmented reality with low energy and low latency. In *Proceedings of the 16th ACM Conference on Embedded Networked Sensor Systems*. 292–304.

[15] Tiffany Yu-Han Chen, Lenin Ravindranath, Shuo Deng, Paramvir Bahl, and Hari Balakrishnan. 2015. Glimpse: Continuous, real-time object recognition on mobile devices. In *Proceedings of the 13th ACM Conference on Embedded Networked Sensor Systems*. 155–168.

[16] Collabora. 2024. Monado - OpenXR Runtime. https://monado.dev/.

[17] The Khang Dang, Nitinder Mohan, Lorenzo Corneo, Aleksandr Zavodovski, Jörg Ott, and Jussi Kangasharju. 2021. Cloudy with a chance of short RTTs: analyzing cloud connectivity in the internet. In *Proceedings of the 21st ACM Internet Measurement Conference*. 62–79.

[18] Frank Dellaert and GTSAM Contributors. 2022. *borglab/gtsam*. https://doi.org/10.5281/zenodo.5794541

[19] Aditya Dhakal, Xukan Ran, Yunshu Wang, Jiasi Chen, and K. K. Ramakrishnan. 2022. SLAM-Share: Visual Simultaneous Localization and Mapping for Real-Time Multi-User Augmented Reality. In *Proceedings of the 18th International Conference on Emerging Networking EXperiments and Technologies* (Roma, Italy) *(CoNEXT '22)*. Association for Computing Machinery, New York, NY, USA, 293–306. https://doi.org/10.1145/3555050.3569142

[20] Sanika Doolani, Callen Wessels, Varun Kanal, Christos Sevastopoulos, Ashish Jaiswal, Harish Nambiappan, and Fillia Makedon. 2020. A review of extended reality (xr) technologies for manufacturing training. *Technologies* 8, 4 (2020), 77.

[21] Epic. 2020. Epic Zen Garden. https://www.unrealengine.com/marketplace/en-US/product/epic-zen-garden.

[22] Hadi Esmaeilzadeh, Emily Blem, Renee St. Amant, Karthikeyan Sankaralingam, and Doug Burger. 2011. Dark silicon and the end of multicore scaling. In *Proceedings of the 38th Annual International Symposium on Computer Architecture* (San Jose, California, USA) *(ISCA '11)*. 365–376. https://doi.org/10.1145/2000064.2000108

[23] Christian Forster, Luca Carlone, Frank Dellaert, and Davide Scaramuzza. 2015. *IMU preintegration on Manifold for Efficient Visual-Inertial Maximum-a-Posteriori Estimation*.

[24] Meta Community Forums. 2024. Quest 3 Power Usage Tests and Approximate Battery Life. https://communityforums.atmeta.com/t5/Talk-VR/Quest-3-Power-Usage-Tests-and-Approximate-Battery-Life/td-p/1094433.

[25] Epic Games. 2024. Unreal Engine. https://www.unrealengine.com/en-US.

[26] Steven Gao, Jeffrey Liu, Qinjun Jiang, Finn Sinclair, William Sentosa, Brighten Godfrey, and Sarita Adve. 2025. XRgo: Design and Evaluation of Rendering Offload for Low-Power Extended Reality Devices. In *Proceedings of the 16th ACM Multimedia Systems Conference* (Stellenbosch, South Africa) *(MMSys'25)*. Association for Computing Machinery, New York, NY, USA.

[27] Patrick Geneva, Kevin Eckenhoff, Woosik Lee, Yulin Yang, and Guoquan Huang. 2019. OpenVINS: A Research Platform for Visual-Inertial Estimation. *IROS 2019 Workshop on Visual-Inertial Navigation: Challenges and Applications* (2019).

[28] Godot. 2020. Platformer 3D. https://github.com/godotengine/godot-demo-projects/tree/master/3d/platformer.

[29] Michael Grupp. 2017. evo: Python package for the evaluation of odometry and SLAM. https://github.com/MichaelGrupp/evo.

[30] Kiryong Ha, Zhuo Chen, Wenlu Hu, Wolfgang Richter, Padmanabhan Pillai, and Mahadev Satyanarayanan. 2014. Towards wearable cognitive assistance. In *Proceedings of the 12th annual international conference on Mobile systems, applications, and services*. 68–81.

[31] David Heaney. 2024. Apple Vision Pro Battery Capacity Reveals Its True Purpose. https://www.uploadvr.com/apple-vision-pro-battery-capacity/.

[32] Jin Heo, Ketan Bhardwaj, and Ada Gavrilovska. 2023. FleXR: A System Enabling Flexibly Distributed Extended Reality. In *Proceedings of the 14th ACM Multimedia Systems Conference* (Vancouver, BC, Canada) *(MMSys '23)*. Association for Computing Machinery, New York, NY, USA, 1–13. https://doi.org/10.1145/3587819.3590966

[33] Tianyi Hu, Fan Yang, Tim Scargill, and Maria Gorlatova. 2024. Apple vs. Meta: A Comparative Study on Spatial Tracking in SOTA XR Headsets. In *Proceedings of the 30th Annual International Conference on Mobile Computing and Networking*.

[34] Zixia Huang, Ahsan Arefin, Pooja Agarwal, Klara Nahrstedt, and Wanmin Wu. 2012. Towards the understanding of human perceptual quality in tele-immersive shared activity. In *Proceedings of the 3rd Multimedia Systems Conference* (Chapel Hill, North Carolina) *(MMSys '12)*. Association for Computing Machinery, New York, NY, USA, 29–34. https://doi.org/10.1145/2155555.2155560

[35] Muhammad Huzaifa. 2023. *Design and Evaluation of Extended Reality Systems*. Ph. D. Dissertation. University of Illinois Urbana-Champaign.

[36] Muhammad Huzaifa, Rishi Desai, Samuel Grayson, Xutao Jiang, Ying Jing, Jae Lee, Fang Lu, Yihan Pang, Joseph Ravichandran, Finn Sinclair, Boyuan Tian, Hengzhi Yuan, Jeffrey Zhang, and Sarita V. Adve. 2021. ILLIXR: Enabling End-to-End Extended Reality Research. In *2021 IEEE International Symposium on Workload Characterization (IISWC)*. 24–38. https://doi.org/10.1109/IISWC53511.2021.00014

[37] Muhammad Huzaifa, Rishi Desai, Samuel Grayson, Xutao Jiang, Ying Jing, Jae Lee, Fang Lu, Yihan Pang, Joseph Ravichandran, Finn Sinclair, Boyuan Tian, Hengzhi Yuan, Jeffrey Zhang, and Sarita V. Adve. 2022. ILLIXR: An Open Testbed to Enable Extended Reality Systems Research. *IEEE Micro* 42, 4 (2022), 97–106. https://doi.org/10.1109/MM.2022.3161018

[38] Gazi Karam Illahi, Ashutosh Vaishnav, Teemu Kämäräinen, Matti Siekkinen, and Mario Di Francesco. 2023. Learning to predict head pose in remotely-rendered virtual reality. In *Proceedings of the 14th Conference on ACM Multimedia Systems*. 27–38.

[39] iputils Maintainers. 2024. iputils - Network utilities including ping. https://github.com/iputils/iputils

[40] Tao Jin, Mallesham Dasa, Connor Smith, Kittipat Apicharttrisorn, Srinivasan Seshan, and Anthony Rowe. 2024. Meshreduce: Scalable and bandwidth efficient 3d scene capture. In *2024 IEEE Conference Virtual Reality and 3D User Interfaces (VR)*. IEEE, 20–30.

[41] Zhihui Ke, Xiaobo Zhou, Dadong Jiang, Hao Yan, and Tie Qiu. 2023. CollabVr: Reprojection-Based Edge-Client Collaborative Rendering for Real-Time High-Quality Mobile Virtual Reality. In *2023 IEEE Real-Time Systems Symposium (RTSS)*. IEEE, 304–316.

[42] Z. Jonny Kong, Qiang Xu, and Y. Charlie Hu. 2024. ARISE: High-Capacity AR Offloading Inference Serving via Proactive Scheduling. In *Proceedings of the 22nd Annual International Conference on Mobile Systems, Applications and Services* (Minato-ku, Tokyo, Japan) *(MOBISYS '24)*. Association for Computing Machinery, New York, NY, USA, 451–464. https://doi.org/10.1145/3643832.3661894

[43] Z Jonny Kong, Qiang Xu, Jiayi Meng, and Y Charlie Hu. 2023. AccuMO: Accuracy-centric multitask offloading in edge-assisted mobile augmented reality. In *Proceedings of the 29th Annual International Conference on Mobile Computing and Networking*. 1–16.

[44] Zeqi Lai, Y Charlie Hu, Yong Cui, Linhui Sun, Ningwei Dai, and Hung-Sheng Lee. 2019. Furion: Engineering high-quality immersive virtual reality on today's mobile devices. *IEEE Transactions on Mobile Computing* 19, 7 (2019), 1586–1602.

[45] Juan Linietsky, Ariel Manzur, and contributors. 2024. Godot Engine. https://godotengine.org.

[46] Luyang Liu, Hongyu Li, and Marco Gruteser. 2019. Edge Assisted Real-time Object Detection for Mobile Augmented Reality. In *The 25th Annual International*

Conference on Mobile Computing and Networking (Los Cabos, Mexico) *(MobiCom '19)*. Association for Computing Machinery, New York, NY, USA, Article 25, 16 pages. https://doi.org/10.1145/3300061.3300116

[47] Luyang Liu, Ruiguang Zhong, Wuyang Zhang, Yunxin Liu, Jiansong Zhang, Lintao Zhang, and Marco Gruteser. 2018. Cutting the cord: Designing a high-quality untethered vr system with low latency remote rendering. In *Proceedings of the 16th Annual International Conference on Mobile Systems, Applications, and Services*. 68–80.

[48] Xing Liu, Christina Vlachou, Mao Yang, Feng Qian, Lidong Zhou, Chendong Wang, Lifei Zhu, Kyu-Han Kim, Gabriel Parmer, Qi Chen, et al. 2020. Firefly: Untethered multi-user {VR} for commodity mobile devices. In *2020 USENIX Annual Technical Conference (USENIX ATC 20)*. 943–957.

[49] Edward Lu, Sagar Bharadwaj, Mallesham Dasari, Connor Smith, Srinivasan Seshan, and Anthony Rowe. 2023. Renderfusion: Balancing local and remote rendering for interactive 3d scenes. In *2023 IEEE International Symposium on Mixed and Augmented Reality (ISMAR)*. IEEE, 312–321.

[50] Jiayi Meng, Z. Jonny Kong, Y. Charlie Hu, Mun Gi Choi, and Dhananjay Lal. 2022. Do we need sophisticated system design for edge-assisted augmented reality?. In *Proceedings of the 5th International Workshop on Edge Systems, Analytics and Networking* (Rennes, France) *(EdgeSys '22)*. Association for Computing Machinery, New York, NY, USA, 7–12. https://doi.org/10.1145/3517206.3526267

[51] Jiayi Meng, Sibendu Paul, and Y. Charlie Hu. 2020. Coterie: Exploiting Frame Similarity to Enable High-Quality Multiplayer VR on Commodity Mobile Devices. (2020), 923–937. https://doi.org/10.1145/3373376.3378516

[52] Meta. 2024. Meta Quest 2 Tech Specs. https://www.meta.com/quest/products/quest-2/tech-specs.

[53] Prabhaker Mishra, Uttam Singh, Chandra M Pandey, Priyadarshni Mishra, and Gaurav Pandey. 2019. Application of student's t-test, analysis of variance, and covariance. *Annals of cardiac anaesthesia* 22, 4 (2019), 407–411.

[54] Arvind Narayanan, Xumiao Zhang, Ruiyang Zhu, Ahmad Hassan, Shuowei Jin, Xiao Zhu, Xiaoxuan Zhang, Denis Rybkin, Zhengxuan Yang, Zhuoqing Morley Mao, et al. 2021. A variegated look at 5G in the wild: performance, power, and QoE implications. In *Proceedings of the 2021 ACM SIGCOMM 2021 Conference*. 610–625.

[55] NVIDIA. 2024. NVIDIA Jetson Xavier. https://www.nvidia.com/en-us/autonomous-machines/embedded-systems/jetson-xavier-series/.

[56] NVIDIA. 2024. NVIDIA Video Codec SDK. https://developer.nvidia.com/video-codec-sdk.

[57] NVIDIA. 2024. Software-based power consumption modeling. https://docs.nvidia.com/jetson/archives/r34.1/DeveloperGuide/index.html#page/.

[58] Li Qingqing, J Penia Queralta, Tuan Nguyen Gia, Hannu Tenhunen, Zhuo Zou, and Tomi Westerlund. 2019. Visual odometry offloading in internet of vehicles with compression at the edge of the network. In *2019 Twelfth International Conference on Mobile Computing and Ubiquitous Network (ICMU)*. IEEE, 1–2.

[59] Li Qingqing, Jorge Pena Queralta, Tuan Nguyen Gia, and Tomi Westerlund. 2019. Offloading monocular visual odometry with edge computing: Optimizing image quality in multi-robot systems. In *Proceedings of the 2019 5th International Conference on Systems, Control and Communications*. 22–26.

[60] Tim Scargill, Ying Chen, Tianyi Hu, and Maria Gorlatova. 2023. SiTAR: Situated Trajectory Analysis for In-the-Wild Pose Error Estimation. In *2023 IEEE International Symposium on Mixed and Augmented Reality (ISMAR)*. 283–292. https://doi.org/10.1109/ISMAR59233.2023.00043

[61] Robert R Schaller. 1997. Moore's law: past, present and future. *IEEE spectrum* 34, 6 (1997), 52–59.

[62] Philip Sedgwick. 2014. Spearman's rank correlation coefficient. *Bmj* 349 (2014).

[63] William Sentosa, Balakrishnan Chandrasekaran, P. Brighten Godfrey, Haitham Hassanieh, and Bruce Maggs. 2023. DChannel: Accelerating Mobile Applications With Parallel High-bandwidth and Low-latency Channels. In *20th USENIX Symposium on Networked Systems Design and Implementation (NSDI 23)*. USENIX Association, Boston, MA, 419–436. https://www.usenix.org/conference/nsdi23/presentation/sentosa

[64] Omar Sohail. 2023. Apple Vision Pro Battery Capacity Said To Be Equivalent To An External Powerbank, But Runtime Does Not Match Current Figures. https://wccftech.com/apple-vision-pro-battery-capacity-equal-to-powerbank/.

[65] Stereolabs. 2024. ZED Mini - Mixed-Reality Camera. https://www.stereolabs.com/zed-mini/.

[66] Jürgen Sturm, Nikolas Engelhard, Felix Endres, Wolfram Burgard, and Daniel Cremers. 2012. A benchmark for the evaluation of RGB-D SLAM systems. In *2012 IEEE/RSJ international conference on intelligent robots and systems*. IEEE, 573–580.

[67] Dean Takahashi. 2018. Oculus chief scientist Mike Abrash still sees the rosy future through AR/VR glasses. https://venturebeat.com/2018/09/26/oculus-chiefscientist-mike-abrash-still-sees-the-rosy-future-through-ar-vr-glasses/.

[68] GStreamer Team. 2024. GStreamer: open source multimedia framework. https://gstreamer.freedesktop.org.

[69] Thomas N. Theis and H.-S. Philip Wong. 2017. The End of Moore's Law: A New Beginning for Information Technology. *Computing in Science & Engineering* 19, 2 (2017), 41–50. https://doi.org/10.1109/MCSE.2017.29

[70] Alan Truly. 2022. How long does the Quest Pro battery really last? Here's Meta's answer. https://www.digitaltrends.com/computing/how-long-does-quest-pro-battery-last-metas-answer/.

[71] Valve. 2024. Valve Index Headset. https://www.valvesoftware.com/en/index/headset.

[72] Johannes Marinus Paulus Van Waveren. 2016. The asynchronous time warp for virtual reality on consumer hardware. In *Proceedings of the 22nd ACM Conference on Virtual Reality Software and Technology*. 37–46.

[73] Junyi Wang and Yue Qi. 2023. Scene-independent Localization by Learning Residual Coordinate Map with Cascaded Localizers. In *2023 IEEE International Symposium on Mixed and Augmented Reality (ISMAR)*. IEEE, 79–88.

[74] Nan Wu, Kaiyan Liu, Ruizhi Cheng, Bo Han, and Puqi Zhou. 2024. Theia: Gaze-driven and Perception-aware Volumetric Content Delivery for Mixed Reality Headsets. In *Proceedings of the 22nd Annual International Conference on Mobile Systems, Applications and Services*. 70–84.

[75] Wenxiao Zhang, Bo Han, and Pan Hui. 2018. Jaguar: Low latency mobile augmented reality with flexible tracking. In *Proceedings of the 26th ACM international conference on Multimedia*. 355–363.

[76] Wenxiao Zhang, Bo Han, and Pan Hui. 2022. Sear: Scaling experiences in multi-user augmented reality. *IEEE Transactions on Visualization and Computer Graphics* 28, 5 (2022), 1982–1992.

[77] Wenxiao Zhang, Sikun Lin, Farshid Hassani Bijarbooneh, Hao-Fei Cheng, Tristan Braud, Pengyuan Zhou, Lik-Hang Lee, and Pan Hui. 2022. Edgexar: A 6-dof camera multi-target interaction framework for mar with user-friendly latency compensation. *Proceedings of the ACM on Human-Computer Interaction* 6, EICS (2022), 1–24.

[78] Zichao Zhang and Davide Scaramuzza. 2018. A Tutorial on Quantitative Trajectory Evaluation for Visual(-Inertial) Odometry. In *2018 IEEE/RSJ International Conference on Intelligent Robots and Systems (IROS)*. 7244–7251. https://doi.org/10.1109/IROS.2018.8593941

Low-Latency Volumetric Video Conferencing in Congested Networks Through L4S

Matthias De Fré[1], Jeroen van der Hooft[1], Chia-Yu Chang[2], Koen De Schepper[2], Patrice Rondao Alface[2], Danny De Vleeschauwer[2], Tim Wauters[1], Peter Steenkiste[3], Filip De Turck[1]
[1]IDLab, Ghent University - imec, Ghent, Belgium
[2]Nokia Bell Labs, Antwerp, Belgium
[3]Carnegie Mellon University, Pittsburg, United States of America

ABSTRACT

Current networking solutions are unable to satisfy the low-latency requirements of real-time volumetric video conferencing when faced with heavy congestion scenarios. Traditional congestion controllers use packet loss or the change in round-trip time (RTT) to estimate the bandwidth. Commonly, this method is too slow as congestion has already occurred and the receiving user has already experienced a latency spike. Low latency, low loss and scalable throughput (L4S), recently published as RFC 9330, wants to alleviate this problem by aiming for sub 1 ms queuing delay for low-latency traffic by using accurate explicit congestion notification (AccECN) packet marking to notify applications of early congestion. We propose an L4S-based pipeline for volumetric video delivery, which achieves a more consistent latency under congestion compared to web real-time communication (WebRTC). In addition, L4S bandwidth estimation achieves a 45% faster convergence compared to Google congestion control (GCC) estimation, commonly used in WebRTC. Furthermore, in our detailed evaluation setup the L4S application experiences no packet loss, while the WebRTC-based version suffers from irrecoverable packet loss, resulting in 3% of frames being undecodable.

CCS CONCEPTS

• **Information systems** → **Multimedia streaming**; • **Human-centered computing** → *Virtual reality*.

KEYWORDS

Volumetric Video, Virtual Conferencing, Virtual Reality, WebRTC, L4S

ACM Reference Format:
Matthias De Fré[1], Jeroen van der Hooft[1], Chia-Yu Chang[2], Koen De Schepper[2], Patrice Rondao Alface[2], Danny De Vleeschauwer[2], Tim Wauters[1], Peter Steenkiste[3], Filip De Turck[1]. 2025. Low-Latency Volumetric Video Conferencing in Congested Networks Through L4S. In *ACM Multimedia Systems Conference 2025 (MMSys '25), March 31-April 4, 2025, Stellenbosch, South Africa.* ACM, Stellenbosch, South Africa, 11 pages. https://doi.org/10.1145/3712676.3714443

Permission to make digital or hard copies of all or part of this work for personal or classroom use is granted without fee provided that copies are not made or distributed for profit or commercial advantage and that copies bear this notice and the full citation on the first page. Copyrights for components of this work owned by others than the author(s) must be honored. Abstracting with credit is permitted. To copy otherwise, or republish, to post on servers or to redistribute to lists, requires prior specific permission and/or a fee. Request permissions from permissions@acm.org.
MMSys '25, March 31-April 4, 2025, Stellenbosch, South Africa
© 2025 Copyright held by the owner/author(s). Publication rights licensed to ACM.
ACM ISBN 979-8-4007-1467-2/2025/03…$15.00
https://doi.org/10.1145/3712676.3714443

1 INTRODUCTION

The growing interest in virtual reality (VR) has led to its integration into various applications, such as virtual conferencing [26]. Traditional two dimensional (2D) implementations of these applications limited users to basic, non-immersive 2D perspectives. Proposals for six degrees-of-freedom (6DoF) video address this limitation by providing rotational and positional freedom [2]. However, current solutions fail to offer the low and consistent latency required for high interactivity, which is critical, as increased latency can lead to user frustration and a significant decline in the quality of the virtual conference experience [18].

The content of 6DoF applications can be divided into two distinct categories: image-based video and volumetric video formats [33]. Image-based solutions, such as light fields, generate a new frame based on numerous images to create a new view based on the user's viewing angle, captured by either a lenslet camera or a setup with many cameras [22]. However, this process leads to higher bitrates, increased latency and lower rendering performance at the client-side [33]. In contrast, volumetric video uses three dimensional (3D) objects such as meshes, consisting of vertices and a texture, or point clouds, consisting of individual points with a color, to represent objects within the video [4]. Capturing these objects can be done using a single specialized camera, resulting in a single red, green, blue and depth (RGBD) image, or with multiple specialized camera views that must be stitched together to create a single 3D object [3]. Compared to image-based video, volumetric video-based approaches yield bitrates that are significantly lower. Furthermore, the use of existing advanced rendering techniques increases overall quality. The simplicity of the capture process, together with its superior bandwidth and lower computational complexity, makes volumetric video an ideal candidate for 6DoF videos.

To ensure low latency in streaming, various protocol suites like low-latency dynamic adaptive streaming over HTTP (LL-DASH) and low-latency HTTP live streaming (LL-HLS) are often considered, yielding latencies ranging from 1 to 5 seconds in live streaming scenarios [33]. For true real-time communication, user datagram protocol (UDP)-based solutions like web real-time communication (WebRTC) are preferred, aiming to minimize latency by eliminating chunking overhead caused by LL-DASH and LL-HLS segments [32]. Additionally, WebRTC-based applications enable transmission of encoded frames through push-based methods, rather than the pull-based approach of transmission control protocol (TCP) formats, further improving the overall latency [5, 25].

The chosen streaming format alone does not guarantee both low latency and high quality of experience (QoE). Due to varied network conditions among users, content quality has to be adapted

to available bandwidth to ensure real-time delivery without packet loss. Unlike traditional video, volumetric video offers numerous viewing angles, allowing for quality adjustments based on the user's field of view (FoV) and position. Higher quality can be allocated to objects in proximity or directly within the FoV, while peripheral or distant objects are delivered at a lower quality [34, 14].

Quality adaptation relies on accurate bandwidth estimation. Traditional TCP methods typically employ a capacity-seeking approach: the bandwidth is progressively increased until packet loss is encountered, after which the transmission window is reduced [7]. Alternatively, other approaches exists which leverage variations in round-trip time (RTT) as indicators of network congestion [9]. For UDP, estimation happens at the application level using loss-based feedback messages that are exchanged between the sender and receiver. However, classic estimators react after congestion has taken place. In contrast, low latency, low loss and scalable throughput (L4S) aims to keep queuing delays below 1 ms by marking packets with accurate explicit congestion notification (AccECN) whenever there are early signs of congestion. This mechanism ensures prompt congestion response and consistent low latency [7]. Although the behavior of L4S is advantageous for real-time applications, it has yet to be evaluated for volumetric video conferencing.

The contributions of this paper are twofold. First, we propose an L4S-based streaming pipeline to support real-time communication for immersive applications, with a focus on maintaining a low latency in congested networks. This pipeline delivers consistent and uninterrupted volumetric video conferencing without high-latency spikes, which is essential for enhancing the user experience.. Second, we perform a comparative analysis between our proposed pipeline and an open-source WebRTC-based volumetric video solution [14], evaluating their respective performance in terms of latency, quality and behavior of the bandwidth estimator. The results show that the L4S-based solution achieves a similar, but more consistent, latency, faster bandwidth estimator convergence, and no packet loss, compared to the WebRTC-based approach when faced with competing traffic.

The remainder of this paper is organized as follows. Section 2 provides an overview of the current state of the art concerning volumetric video streaming. Section 3 introduces our proposed L4S-based approach to facilitate low-latency streaming of volumetric video. Section 4 presents the evaluation methodology and discusses the obtained results. Finally, Section 5 concludes the paper.

2 BACKGROUND AND RELATED WORK

This section presents an overview of the related work on real-time streaming of volumetric video. First, we introduce the concept of volumetric video and how it differs from traditional 2D video. Following this, we describe traditional transport protocols and solutions and the problems that occur with these solutions. Then, we introduce L4S-based transport as a possible solution to existing transport problems. Finally, we give an overview of the currently used quality adaptation methods for volumetric video streaming.

2.1 Volumetric Video

Because of the added depth dimension, volumetric video presents a substantially higher level of complexity in both transmission and encoding compared to traditional 2D video. The added complexity presents a challenge in maintaining low latency, making it essential to carefully select the type of volumetric content and encoder to ensure smooth, real-time video delivery.

While most 3D applications typically utilize meshes due to their advanced feature set, significant challenges emerge when applying them to volumetric video. The primary issue is that mesh capturing and processing introduce excessive latency, rendering them unsuitable for real-time virtual conferencing. In contrast, point cloud capture offers a more efficient alternative, as it eliminates the need to convert the captured data into a mesh. Additionally, it avoids computationally expensive mesh optimizations aimed at reducing object complexity [1]. Furthermore, point clouds have proven effective in volumetric video delivery. VR2Gather, a real-time volumetric video application, uses point clouds to enable real-time, bidirectional experiences, such as immersive museum visits or remote consultations [35].

No matter the representation, volumetric video typically results in bitrates that remain impractical for contemporary networks. For instance, large point clouds, such as those in the 8i dataset [12], can reach up to 5.2 Gb/s [33]. To address this problem, numerous point cloud codecs have been proposed to compress the data [20]. However, in virtual conferencing, minimizing encoding latency is essential to achieving a frame rate above 15 FPS, which is necessary for an acceptable user experience. As a result, highly bandwidth-efficient codecs such as video point cloud coding (V-PCC) and geometry-based point cloud compression (G-PCC) are unsuitable, as they have encoding and decoding times of multiple seconds for a single frame [38]. Alternative codecs, including cwipc [11] and Draco [16], achieve single-frame encoding times below 100 ms, which is viable for real-time volumetric video conferencing, albeit at the cost of increased bandwidth consumption [14]. However, these resulting bitrates are low enough to support volumetric video conferencing with a limited number of users. Furthermore, as the encoded frames are still quite large, they will need to be fragmented into a large number of packets. Since neither codec supports partial decoding, every packet must be received for successful playback. This requirement becomes challenging if many packets are dropped due to network congestion, highlighting the importance of selecting an appropriate network protocol.

2.2 TCP-based vs UDP-based content delivery

For on-demand video, HTTP-based implementations are often preferred due to their inherent reliability, ensuring that the whole video can be received by the user. Within the realm of HTTP streaming, protocol suites such as dynamic adaptive streaming over HTTP (DASH) and HTTP live streaming (HLS) have gained widespread adoption. These formats segment video content temporally and encode these segments at various bitrates and quality levels. This approach introduces a latency ranging anywhere from 1 to 18 seconds for live streaming scenarios [37], rendering it unsuitable for real-time applications. In contrast, WebRTC, a real-time communication suite utilizing UDP, achieves sub-second delays, making it a more suitable choice for real-time volumetric video streaming [25].

Unlike TCP-based solutions, such as LL-DASH, UDP-based approaches lack inherent flow control and bandwidth estimation

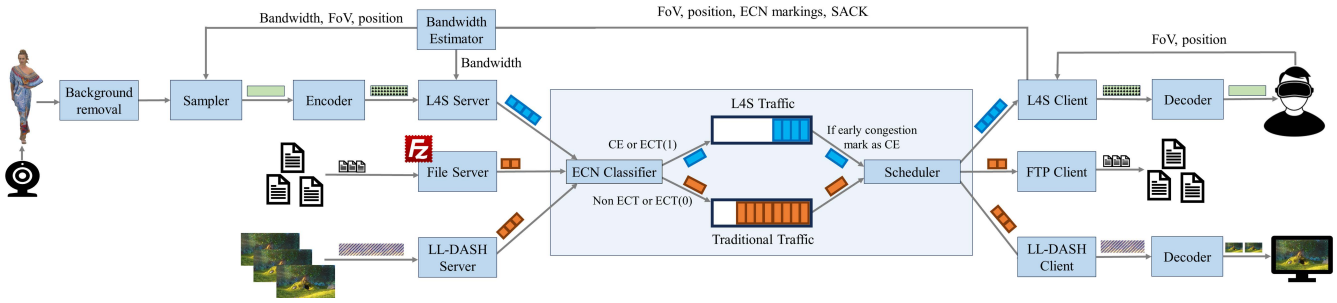


Figure 1: System architecture for an L4S-based volumetric video streaming solution sharing the network with non-L4S flows.

mechanisms. As a result, congestion controllers such as Google congestion control (GCC) [10] and Self-Clocked Rate Adaptation for Multimedia (SCReAM) [24], are implemented at the application layer to fulfill the purpose of bandwidth estimation and flow control. The primary objective of these controllers is to maintain a good balance between latency and high throughput by accurately estimating available bandwidth and pacing packets accordingly [13]. Similar to DASH-based solutions, the application can then use the bandwidth to select the transmitted quality on a per-peer basis, so that the video stream remains below the supplied bitrate [14]. In scenarios with a limited number of users, a unique quality representation can be generated for each user based on their available bandwidth, with the video encoder adjusting compression or reducing quality as needed to keep the stream within the bandwidth constraints of each user [27]. However, this approach significantly limits scalability, as the need to create a distinct representation for each user constrains the maximum number of participants. In this paper, the primary focus will be on the individual encoder to better illustrate the bandwidth estimation. Furthermore, for volumetric video, quality adaptation can be further enhanced by using the position and rotational information to prioritize the most important details [34].

For traditional transport, a significant portion of the latency comes from the queuing delay, with increased throughput raising the likelihood of higher, more variable queuing delay [29]. This is caused by congestion and queue buildup, particularly when all flows attempt to use their maximum bandwidth. Congestion eventually results in queue overflow and packet loss until the queue begins to drain [17]. The packet loss and increased delay are signals to the congestion controller that it should reduce its send rate in order to reduce network congestion. However, flows quickly increase send rates, restarting the process. Additionally, the queuing discipline at bottleneck links affects congestion reaction time; classic active queue management (AQM) disciplines delay packet drops for up to 20 ms after a burst before dropping the packet, potentially exacerbating delay [7]. These slow adaptation times, compounded by loss, cause buffering, increased latency and frame loss in streaming until network flows finally adjust their send rates. In the context of volumetric video, this increased buffering and latency can contribute to cybersickness [31]. Furthermore, due to the high bitrates of volumetric video, packet loss has a greater impact. Even a small percentage of packet loss can necessitate retransmitting a significant number of packets, each with its own risk of being lost again, resulting in further delays.

2.3 L4S-Based Transport

The emergence of real-time interactive applications like cloud gaming and virtual reality has intensified the demand for network protocols that prioritize both low latency and high throughput [8]. Traditional streaming solutions and congestion control mechanisms often struggle to achieve this balance, as latency reduction can come at the expense of reduced throughput, leading to a reduced user experience [19].

L4S attempts to address the problems that occur in highly congested networks. At the protocol level, L4S uses either TCP or UDP to transport data and acts as a congestion controller for these protocols. L4S also operates within the network architecture, enabling the identification and prioritization of latency-sensitive traffic flows by using the AccECN field of network packets [30]. This approach allows for differentiated services, ensuring minimal queuing delays for these flows without compromising overall network efficiency too much. L4S leverages AccECN to signal early congestion to senders, enabling them to adjust their transmission rates dynamically before the queuing delay becomes too high. This early congestion notification allows senders to react faster to upcoming congestion events, and prevents the queue from overflowing which would result in packets being dropped. Having a low queuing delay target also ensures that the latency within the network is less variable compared to traditional queuing systems [7].

To ensure coexistence with existing traffic, a dual queue setup is commonly used [15]. With this type of AQM, it becomes possible to split low-latency traffic from classic network traffic using the AccECN field. The queue containing the classic traffic works as a traditional queue, filling up until eventual packet loss. The low-latency queue will not immediately drop traffic, but will instead mark packets as congestion experienced (CE). Additionally, to prevent bursty traffic in the queue, each L4S application is required to space it's packets equally in a pacing interval. In the context of volumetric video, which has large bitrates, this pacing interval should be chosen carefully to maintain a balance between low latency and burstiness. In this paper, we will leverage L4S-based transport to create a volumetric video pipeline. To the best of the authors' knowledge, this is the first paper to study and evaluate L4S-based volumetric video in detail.

3 L4S-BASED VOLUMETRIC VIDEO DELIVERY

In this section, we first present the proposed volumetric video delivery architecture, followed by an explanation of the employed

quality adaptation scheme. Next, we provide an overview of the L4S components used, along with their implementation details.

3.1 Volumetric Video Delivery Architecture

The first step of the pipeline shown in Figure 1 involves capturing of the volumetric video, typically a person in virtual conferencing scenarios. In the context of this paper, it is assumed that a single camera is utilized, although this framework can be easily extended to accommodate multiple cameras by incorporating an additional component that is responsible for merging the views from all cameras. A specialized depth camera captures an RGBD image of the user, which is subsequently converted into a point cloud representation. This process may include an optional sampling step to reduce the number of points in the point cloud. However, caution must be exercised to ensure that sampling does not introduce a significant amount of latency. After sampling, it is necessary to encode the point cloud to reduce the bitrate, ensuring efficient transmission in networks with lower bandwidths. Similar to the sampling component, the encoding process must be performed with minimal latency to allow for higher frame rates, provided that sufficient bandwidth is available. Furthermore, it is important to note that current real-time state-of-the-art point cloud codecs do not support partial encoding. As a result, packet loss has a more pronounced effect, as the entire frame must be received for successful decoding.

After encoding, the frame is processed through the L4S network stack, beginning with the server component. Each frame is divided into packets, each containing frame data and a header for reconstruction at the receiver. The server is tasked with ensuring compliance with L4S requirements, such as implementing pacing and L4S-compliant bandwidth estimation. The next stage of the L4S stack is the queuing mechanism of the network, which must differentiate between L4S and traditional network traffic, handling each traffic type accordingly. This process may involve assigning different priorities or allocating distinct bandwidth shares to each flow type. For L4S traffic, it is critical to implement mechanisms that marks L4S packets so that they can anticipate congestion early enough to mitigate its effects. In contrast, traditional flows, such as large file downloads or LL-DASH streaming, typically have less stringent latency requirements than real-time virtual conferencing, which demands minimal latency. Further details regarding the specific implementation of the L4S components are discussed in the following subsections.

Upon delivery, the frames are decoded and displayed to the user. In most cases, rendering is carried out using a game engine, such as Unity, on a high-performance desktop PC. The rendered content is then transmitted to a head-mounted display (HMD) either through a high-speed wired connection or via a wireless solution, such as 5 GHz Wi-Fi or 5G.

3.2 Bandwidth Estimation

Bandwidth estimation is performed by leveraging the number of marked packets to calculate the marking probability. Based on this information, and using a target RTT of 25 ms, the system adjusts the bandwidth, either increasing or decreasing it accordingly. In the event of packet loss, either due to severe congestion or suboptimal network conditions, the L4S congestion controller falls back to a classic TCP-Reno congestion controller. As a result, the estimated bandwidth is halved upon detecting packet loss, helping to maintain low latency by reducing the need for retransmissions beyond the already lost packets. However, this approach leads to a reduction in video quality due to the lowered throughput.

In comparison to GCC, this L4S-based approach is more sensitive to network instability. GCC employs a lower threshold for packet loss, which mitigates the impact of minor losses that result from faulty equipment [36]. This lower boundary allows GCC to maintain stability in the presence of occasional packet loss, whereas the proposed approach responds more aggressively in such conditions.

3.3 Packet Acknowledgment

The delivery of packets is confirmed using selective acknowledgments (SACK) messages. In addition to containing the acknowledgments, it also includes information on the number of bytes marked as CE. Subsequently, these AccECN markings, along with the target RTT, are utilized by the congestion controller to estimate the available bandwidth. Moreover, the data contained within the SACK message enables the application to identify which packets require retransmission in the event of packet loss.

3.4 Quality Adaptation

An essential aspect of the architecture, present across multiple components, is the concept of quality adaptation. The objective is to optimally calculate the highest achievable quality within the constraints of limited available bandwidth. In the case of volumetric video, this calculation must also account for the user's FoV position and rotation. Achieving this, however, necessitates the coordination of several components that exchange information with one another.

In our solution, point cloud frames are adapted using an existing individual encoding approach [14]. This encoding method determines the required sampling rate for each user and uniformly reduces the number of points in the point cloud according to this rate. This approach facilitates a smooth scaling of frame size and enables more effective comparisons between the employed congestion control algorithms. Furthermore, it produces streams that are visually comparable to those generated by a multiple description coding (MDC)-based approach for use cases with up to 10 users, at a lower bitrate [14]. Alongside the estimated bitrate, both the user's position and FoV are required. This data is transmitted from the client to the server along with packet acknowledgments within the SACK messages. The positional and rotational information is utilized by the sampler to limit the maximum achievable quality produced by the individual encoding approach. The sampler achieves this by dividing the user's FoV into multiple equal-sized regions, each with a defined maximum sampling rate, and then determining the region in which the object currently resides. This approach is based on the rationale that the objective quality of the object can be reduced without significantly affecting the perceived visual quality for the user. However, it is important to note that there will be a discrepancy between the used position and the current position of the user as the pipeline does not perform any viewport prediction. Additionally, the transmission of the quality decision, and the time for the next frame to become available (between 1 ms and 33 ms for 30 FPS) both add latency before the new quality becomes visible.

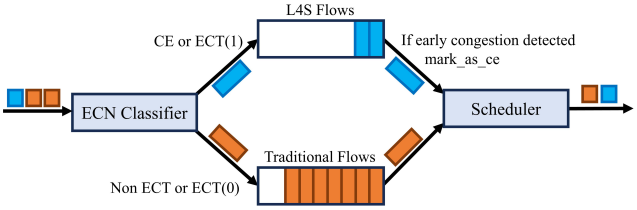


Figure 2: L4S compliant queue separating L4S and traditional traffic into two separate classes.

3.5 Retransmissions

As the point cloud codec does not support partial decoding, the loss of packets can have a significant impact on the visual quality. Because of this it is important to retransmit any packets that were lost so that the frames are able to be eventually decoded. The proposed architecture uses SACKs to identify which packets were dropped and need to be retransmitted to the client. The data for these dropped packets is retrieved from an in-flight buffer, which stores the data of frames currently in transit. However, this in-flight buffer only has limited capacity, and retains a frame until the client has acknowledged that it has received all packets of that frame.

Due to this limitation, new frames can only be admitted into the buffer when it is not full, leading to frames being dropped when the in-flight buffer is full. However, in contrast to WebRTC, which uses a round-robin buffer that removes frames once filled [21], this approach ensures that once a frame is transmitted, it will ultimately be delivered in its entirety. The size of the in-flight buffer is a critical parameter that must be carefully calibrated based on the specific requirements of the use case. A larger buffer enhances resilience against high round-trip times, but may also increase overall latency in scenarios with significant packet loss.

3.6 Packet Queuing

Any packet queuing configuration is allowed, provided it ensures that L4S traffic is isolated from the queuing latency experienced by classic traffic and that L4S packets are marked with CE in anticipation of upcoming congestion [6].

In Section 4, we will consider a queuing setup based on the principles shown in Figure 2. This queue uses the AccECN field to distinguish the L4S traffic from the classic traffic into the two separate classes of the overall queue. Link capacity is distributed between both queues and the aim is to achieve fairness between all flows in each traffic class. The packets within the low-latency queue are marked as CE whenever an early form of congestion is detected. This dual-class setup is designed to ensure the coexistence of L4S traffic with traditional traffic while maintaining low queuing delays for the L4S traffic. Additionally, the utilization of the two classes also does not negatively impact the performance of flows in the traditional queue as the scheduler is configured to prevent starvation of either queue.

3.7 Frame Pacing

To mitigate network congestion, an L4S application must minimize burstiness by reducing the number of packets transmitted back to back. In this L4S implementation, we propose to reduce the burstiness by spreading the packets of the frame over a fixed interval. The send time of each packet is calculated using Equation 1.

$$T_{\text{next}} = T_{\text{current}} + \frac{\text{pacing interval}}{\text{frame size}} \times \text{packet size} \tag{1}$$

This pacing mechanism, in conjunction with a fair queue (FQ) scheduler at the sender, ensures that frame data is transmitted gradually rather than in bursts, since the packets are evenly distributed throughout the specified pacing interval. In contrast to GCC, which transmits packets in bursts over intervals of 5 ms, this method exhibits greater fairness toward competing flows. By uniformly spreading the packets across the pacing interval, it maintains consistent burstiness regardless of the available bandwidth.

While the use of pacing introduces additional latency to the pipeline, it effectively prevents sudden spikes in the queuing time of the low-latency queue during frame transmission. This approach ensures that all flows within the low-latency queue can maintain consistent latency and sustain steady throughput, thereby enhancing the overall performance of those flows.

4 EVALUATION

This section contains an evaluation of the L4S-based implementation compared to an open-source WebRTC-based volumetric video pipeline [14]. First, we introduce the metrics that will be used in the evaluation. Following this, we describe the experimental setup and the implementation of the L4S compliant queuing setup that was used during the experiments. Then, the results of our experiments in terms of throughput, bandwidth estimation, transport latency, packet loss and visual quality are reported and discussed.

4.1 Evaluation Criteria

In our experiments, both the WebRTC-based and L4S-based implementations are evaluated against one or more competing TCP Cubic or TCP Prague flows, initiated by the *iperf* program. TCP Prague is a congestion control algorithm similar to the one in our L4S application, which aims for low, sub 1 ms, queuing delay by reacting to AccECN markings. These competing flows are introduced five seconds after the volumetric application has started. The results of our experiments are reported in two stages. First, we analyze the system's behavior during a startup phase lasting thirty seconds. This analysis allows us to draw significant conclusions regarding initial performance metrics, such as the time required for the congestion controller to converge and the growth rate of the bandwidth estimation. Next, we examine the system's behavior in the stable state, after the first thirty seconds have elapsed. In this stage, we are primarily interested in packet loss, transport latency, and visual quality. It is crucial to evaluate these metrics in the stable state, as the throughput will have stabilized, and the results will no longer be influenced by the initial convergence of the bandwidth estimate. Finally, we compare the total end-to-end latency of both flows.

4.2 Experimental Setup

Both the client and L4S server implementations are developed using the C++ programming language. To ensure that packets are not sent in bursts, the server uses a pacing interval of 20 ms and equally paces the packets over this interval using the *SO_TXTIME* socket option and *SCM_TXTIME* packet field to specify the send

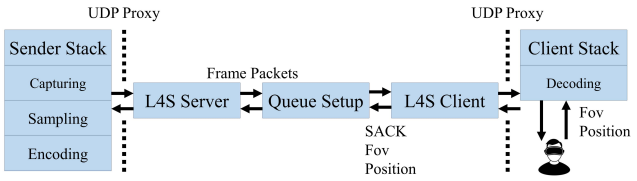


Figure 3: L4S stack using UDP sockets to transport data between Windows and Linux (L4S applications).

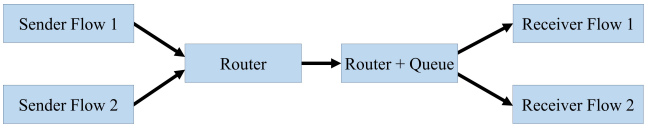


Figure 4: Experimental setup on the imec Virtual Wall [23].

time. To ensure that this send time is respected, a FQ is employed at the server side. The capture, encoding, decoding, and rendering components are implemented on a Windows system due to its superior support for VR application. However, given that the current L4S implementation requires capabilities such as setting packet send times and utilizing a sender side FQ, which are only supported by the Linux operating system, the decision was made to implement the networking components on a Linux machine. As shown in Figure 3, a UDP socket proxy is used to transmit data between the Windows-based modules and the L4S server running on Linux, as well as to receive data from the client. This mechanism introduces minimal latency due to the Windows and Linux systems being directly connected via Ethernet on the same network. All experiments utilize a sequence of a single person, captured with an Intel RealSense D455 camera, at a frame rate of 30 FPS with a resolution of 848 × 480 pixels. During the experiments, this precaptured volumetric video is played back at the same speed, using an accurate timer which sleeps for 33 ms between each frame. After background removal, the resulting point clouds contain an average of 125,000 points. The point clouds are then sampled based on the estimated bandwidth and user position by using the sampling method described in Section 3.4. To achieve real-time encoding at 30 FPS, each point cloud is individually encoded using the Draco [16] codec, with position quantization at 11 bits and KD-tree coding enabled. A speed parameter of 7 was selected to maintain an optimal balance between encoding speed and output bitrates. After the sampling and encoding, the resulting bitrates of the sequence vary between 12.5 Mb/s and 75 Mbit/s. In terms of congestion control, the WebRTC-based approach employs GCC, which is the most widely used congestion controller in WebRTC applications. Alternative congestion controllers, such as SCReAM [24], offer a different trade-off between throughput and latency, yielding results that differ from those presented in this section. Both the L4S bandwidth estimator and the GCC algorithm used by WebRTC use a minimum and initial bitrate of 12.5 Mb/s and a maximum rate of 90 Mb/s.

The experimental setup is shown in Figure 4. Throughput and latency experiments are carried out on the imec Virtual Wall, a testbed with a large number of interconnected bare metal nodes [23]. We use several connected nodes with the following hardware specifications; *CPU: Intel Xeon E5520 (2.27GHz), RAM: 12GB DDR3 (1066 MHz), NIC:* 1 Gb/s. Each node uses a patched Ubuntu 20.04 kernel to

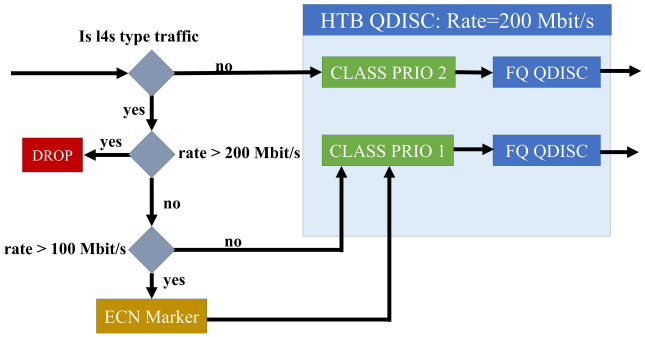


Figure 5: The detailed implementation of the L4S compliant queuing setup used in the experiments.

enable support for TCP Prague [28]. This includes the router node, which uses standard Linux features to perform the routing. All experiments are performed with 10 ms of delay added in both directions using traffic control, as well as 200 Mb/s bandwidth restriction on the outgoing interface of the queue.

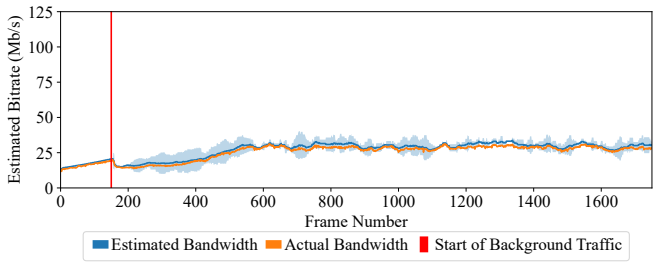
4.3 L4S-Compliant Queuing Setup

As described in Section 3.6, the queuing setup is an important part of ensuring that the L4S can properly work. For these experiments the queuing setup shown in Figure 5 was used. The first step of the queue is to separate L4S traffic from traditional flows. This separation is done by checking if the packet has been marked as AccECN-capable transport.

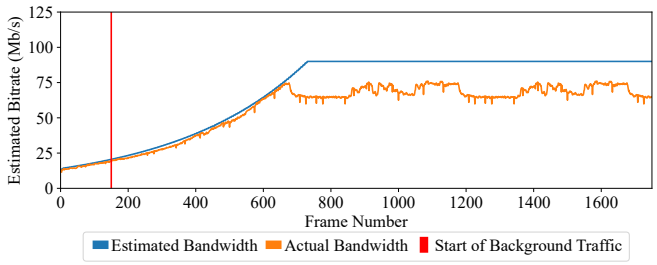
For a traditional flow, no additional processing is performed, and the packet is immediately placed into the queue. However, for L4S traffic, additional steps are taken. First, it is checked if the total rate of all L4S traffic exceeds 200 Mb/s. If this is the case, packets will be dropped immediately, and no AccECN marking will be done. In the next step, an AccECN marker is utilized, setting the AccECN field to CE if the total rate surpasses 100 Mb/s. This marking serves as a notification to the applications that there is a probability that congestion is imminent and that they should lower their send rate to prevent sudden peaks in latency. It is important to emphasize that the packets marked with CE are not dropped and that they proceed through the rest of the queue as normal.

In these experiments, it is assumed that all AccECN-capable flows will respond appropriately to early congestion warnings by reducing their transmission rates. This reduction is intended to prevent actual congestion from occurring. Specifically, by lowering their send rates, the total rate of L4S traffic is kept below 100 Mb/s, ensuring that traditional flows consistently have at least 100 Mb/s of bandwidth available. This mechanism prevents the traditional flows from experiencing starvation due to excessive L4S traffic, which is one of L4S's requirements.

Once packet marking is completed, both L4S and traditional flows are placed into the main hierarchy token bucket (HTB) queue. The L4S traffic is assigned to the higher priority class, giving it preferential access to available bandwidth. However, since packet marking occurs before priority assignment and is triggered at 100 Mb/s, L4S flows are capped at consuming a maximum of 100 Mb/s of the total 200 Mb/s bandwidth. This outcome assumes that all L4S flows appropriately adjust their send rates in response to packet marking to not cross this 100 Mb/s limit.

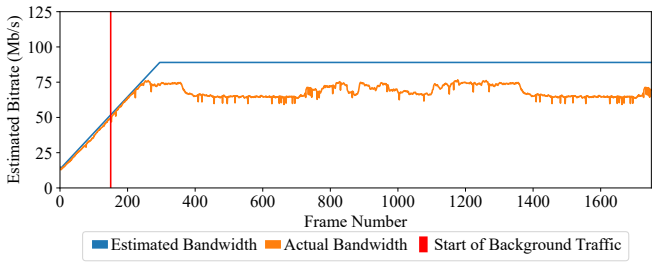


(a) Three competing TCP Cubic flows

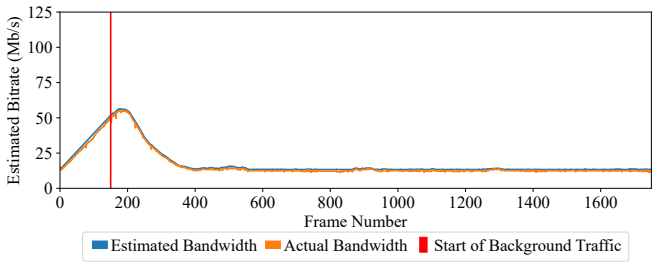


(b) Three competing TCP Prague flows

Figure 6: WebRTC estimated bitrate for different competing flows at 30 FPS.



(a) Three competing TCP Cubic flows



(b) Three competing TCP Prague flows

Figure 7: L4S estimated bitrate for different competing flows at 30 FPS.

The final component of the queue is a FQ placed at the end of each priority class. This FQ ensures fairness between different flows by operating in a round-robin manner. It cycles through all active flows, taking one or more packets from each before moving to the next. Without this FQ, the default queuing strategy, typically first-in, first-out (FIFO), would be employed. A FIFO approach would disproportionately benefit bursty traffic, as it could dominate the queue and overwhelm other flows, especially those that are more responsive to packet loss.

4.4 Achieved Throughput

Figure 6a shows the behavior of the bandwidth estimation and the resulting bitrates for the GCC algorithm used in the WebRTC application, when faced with three competing TCP Cubic flows. This result shows that the maximum bitrate stabilizes at 30 Mb/s once the estimator has converged. This behavior is also present when lowering the number of TCP Cubic flows to two. However, with only one TCP Cubic flow, there is a slight increase in throughput, with the maximum reaching 46 Mb/s. Notably, in this scenario, there are no L4S flows, meaning the entire 200 Mb/s is allocated to traditional traffic. This behavior implies that the expected throughput for WebRTC should be around 75 Mb/s when competing with one or two TCP Cubic flows, if it is assumed that the bandwidth is fairly distributed. The fact that WebRTC fails to achieve this expected throughput suggests that it struggles with the chosen implementation of GCC or queuing setup. In contrast, as shown in Figure 6b, when competing against TCP Prague flows, the WebRTC application is able to reach its maximum throughput. This behavior is because TCP Prague flows receive AccECN markings upon reaching 100 Mb/s of the total 200 Mb/s, prompting them to reduce their transmission rates.

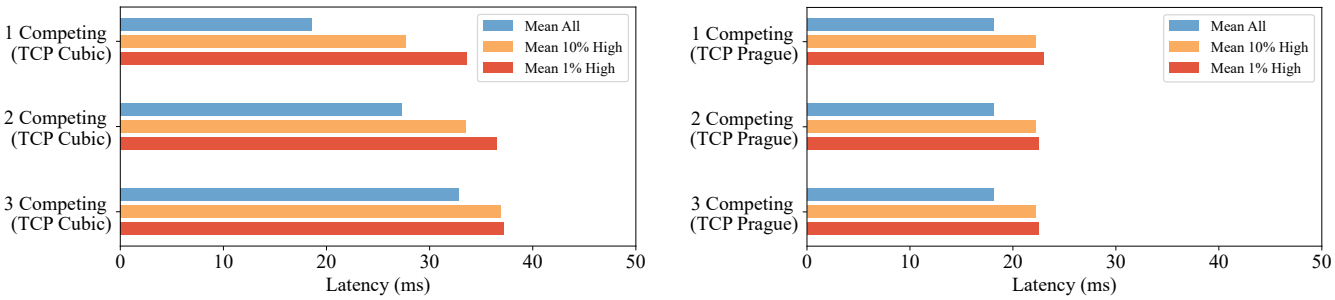
Using the L4S-based application, we observe the results shown in Figure 7. In case of competing TCP Cubic flows, the L4S-based application consistently reaches and maintains its maximum throughput. This outcome is expected, as the L4S application is the only flow in the 100 Mb/s priority queue dedicated to L4S traffic. However, when three TCP Prague flows are introduced, a number of packets receive AccECN markings, causing the throughput of the L4S application to decrease significantly to its minimum value of 12.5 Mb/s. This behavior ensures that the queuing delay is close to 1 ms. Reducing the number of competing flows leads to a higher throughput: with only one TCP Prague flow, the L4S application achieves an average throughput of 58 Mb/s, while with two flows it reaches 15 Mb/s.

4.5 Convergence of Bandwidth Estimation

While achieving maximum throughput in a stabilized environment is important, the time it takes to reach this stable state is also a key metric to consider. In these experiments, the initial bitrate for both implementations is set to 12.5 Mb/s, which corresponds to the minimum bitrate necessary to transmit the lowest quality. Starting with this minimal value provides a clear indication of how quickly the estimated bandwidth increases over time until it reaches the maximum estimated bitrate of 90 Mb/s.

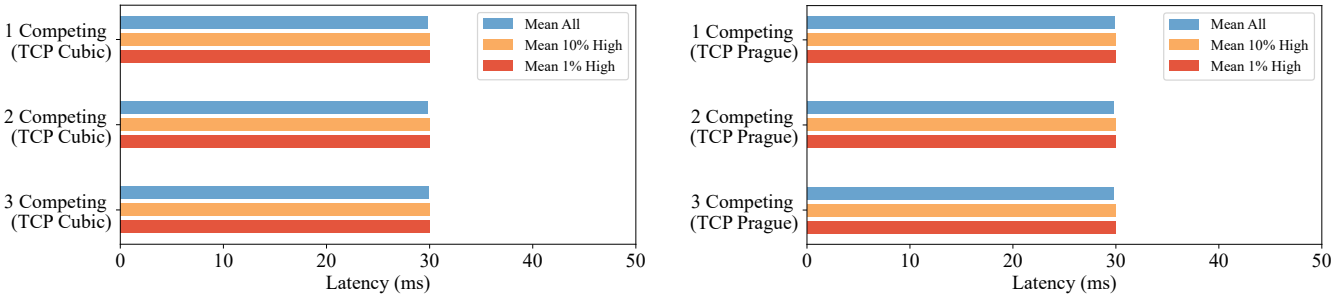
Based on the results presented in the previous section, it is evident that the WebRTC application only achieves its maximum throughput when competing with TCP Prague flows, while the L4S application reaches its maximum throughput when contending with TCP Cubic flows. Due to this fact, it is possible to use these results as an indication of the convergence speed.

As shown in Figure 6b, convergence for the WebRTC application occurs after approximately 22 seconds, which is significantly slower than the L4S-based implementation, where maximum throughput is achieved in just 10 seconds. Although the quality gradually improves during this period, the prolonged convergence of



(a) Three competing TCP Cubic flows **(b) Three competing TCP Prague flows**

Figure 8: WebRTC latency for different competing flows.



(a) Three competing TCP Cubic flows **(b) Three competing TCP Prague flows**

Figure 9: L4S latency for different competing flows.

the WebRTC-based implementation time substantially impacts the overall user experience. Furthermore, during periods of temporary congestion, the bitrate will decrease, requiring an extended recovery period to return to optimal levels, which further deteriorates the user's QoE. This slow convergence is observed when looking at the growth behavior of the estimation curve. In the case of L4S, the growth is clearly linear, whereas WebRTC exhibits a more gradual, very slow, non-linear increase.

A potential solution to address the slow convergence is to increase the initial bitrate, which would significantly reduce the time required to reach maximum throughput. However, due to the queuing structure, shown in Figure 5, that utilizes FQ at the end of each priority class, this approach poses certain risks. If a flow would send a sudden burst of data that exceeds its share of bandwidth, the likelihood of packet loss for that flow increases. In contrast, if the burst is lower than the data share, a large amount packet loss will occur for the other flows, as they would not have sufficient time to gradually reduce their transmission rates. Additionally, for L4S traffic, a more gradual increase is more fair towards the other L4S flows as they will not experience a sudden burst of packet markings.

4.6 Frame Loss

Due to the early congestion notifications in the L4S implementation, the L4S-based application avoids packet loss by marking packets with AccECN before any are dropped. In contrast, the WebRTC implementation suffers more from packet loss, which it attempts to mitigate through negative acknowledgments (NACKs) to identify and request the lost packets. For this reason, it is critical to focus on the percentage of frames that remain undecodable during transmission for WebRTC. Given that the WebRTC-based implementation uses a NACK retry count of one, each lost packet has only one opportunity for retransmission. Combined with the 1024-packet round-robin buffer for NACKs, this configuration increases the likelihood that certain frames may remain undecodable in favor of achieving a lower latency for the other frames. When competing with TCP Prague flows, the WebRTC implementation suffers no packet loss. However, when faced with TCP Cubic flows, the WebRTC-based implementation experiences non-recoverable packet loss, resulting in approximately 3% of frames being undecodable when contending with three competing TCP Cubic flows.

4.7 Peak Transport Latency

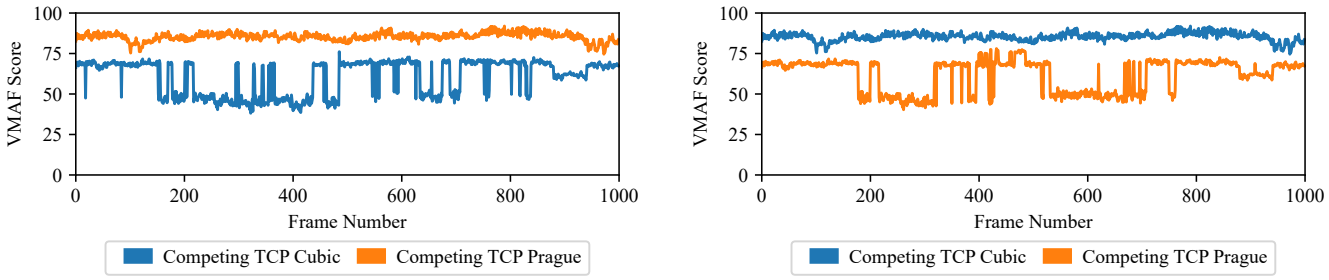
For the latency results, it is important to reiterate that 10 ms of bidirectional latency was introduced and that this latency is also included in the results of this section. As shown in Figure 8, the WebRTC implementation achieves lower latency than the L4S implementation when competing with TCP Prague flows. However, this advantage is only maintained in the presence of fewer than two TCP Cubic flows: when competing with three TCP Cubic flows, the latency for WebRTC surpasses that of the L4S implementation. Furthermore, Figure 8a shows that the latency, which mainly consists of queuing delay, for the WebRTC implementation continues to increase as the number of competing TCP Cubic flows rises, making it more difficult to predict the average latency when the number of competing flows is unknown. In addition to average latency, it is essential to examine peak latency, as these spikes are more noticeable to the end user. Both the WebRTC and L4S applications avoid significant latency spikes. However, in terms of peak latency, the WebRTC implementation only performs better when contending with a single TCP Cubic flow. From two TCP Cubic flows onward, L4S outperforms WebRTC.

Table 1: VMAF scores for both implementations when competing with different flows, lower intra-sequence scores indicate better quality consistency within a single iteration.

Competing flow	TCP Cubic				TCP Prague			
	VMAF		Intra-sequence VMAF		VMAF		Intra-sequence VMAF	
	Mean	Std dev	Mean	Std dev	Mean	Std dev	Mean	Std dev
WebRTC	61.98	3.26	10.53	0.83	85.39	0.00	2.57	0.00
L4S	84.97	0.33	3.16	0.52	45.17	0.04	2.41	0.34

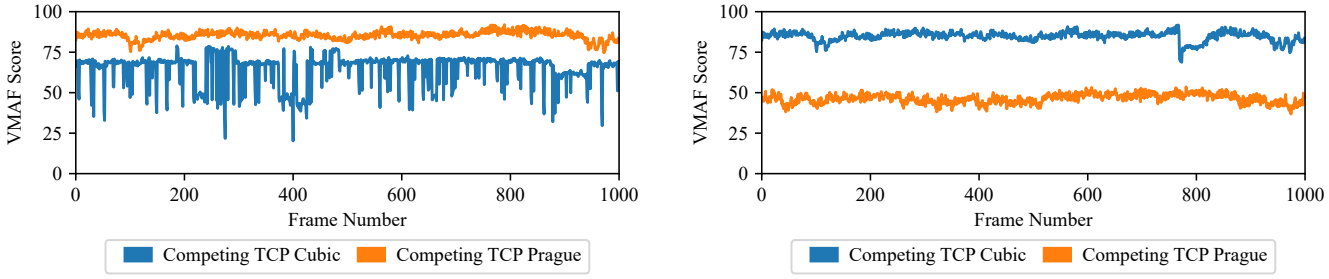
Table 2: Maximum and minimum VMAF scores for three iterations with a competing three competing TCP Cubic or TCP Prague flows.

Competing flow	TCP Cubic						Prague					
Iteration	1		2		3		1		2		3	
	Min	Max	Min	Max	Min	Max	Min	Max	Min	Max	Min	Max
WebRTC	24	77	11	79	18	79	74	91	74	91	74	91
L4S	74	91	74	91	74	91	37	54	37	53	37	64



(a) WebRTC **(b) L4S**

Figure 10: WebRTC and L4S VMAF for different competing flows (one concurrent) in the stable state of a single iteration.



(a) WebRTC **(b) L4S**

Figure 11: WebRTC and L4S VMAF for different competing flows (two concurrent) in the stable state of a single iteration.

Furthermore, as shown in Figure 9, the latency for L4S remains consistent across all configurations. For L4S, the total latency is primarily determined by the chosen parameters, with 10 ms resulting from the introduced bidirectional latency and an additional 20 ms from the selected pacing interval. This consistency is due to the fact that the L4S implementation effectively shifts the unpredictable latency from queuing to a controlled and predictable pacing latency, allowing it to maintain stable performance regardless of the number of competing flows, compared to the WebRTC-based implementation, where latency continues to increase as more flows are introduced. These results indicate that the proposed L4S approach is successful in maintaining the queuing delay of sub 1 ms.

4.8 VMAF Quality

Video Multi-Method Assessment Fusion (VMAF) is a comprehensive video quality metric that merges scores from various video evaluation methods to generate a unified score between 0 and 100. This metric reflects the user-perceived quality more accurately than methods based solely on geometric comparisons of frame differences, since these do not consider the full sequence of frames. In scenarios where frame loss is observed, which in these tests occurs exclusively for WebRTC, the preceding frame is used instead, resulting in a small freeze.

As shown in Table 1, the VMAF scores for WebRTC and the L4S implementation are comparable when competing with a TCP Cubic flow. However, in the presence of a TCP Prague flow, the L4S implementation performs significantly worse, primarily due to its inability to transmit at higher quality.

Table 3: Latency (ms) introduced by each of the transport-independent pipeline modules.

Sender		Receiver	
Camera	Encoding	Decoding	Rendering
104 ms	22 ms	14 ms	5 ms

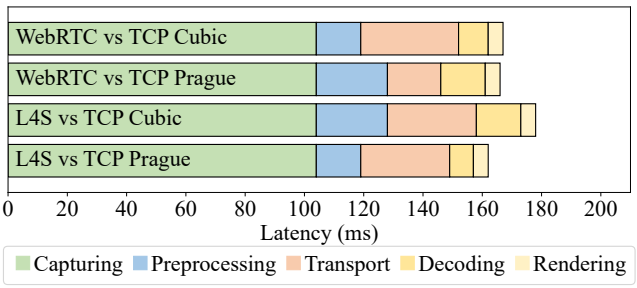


Figure 12: End-to-end latency for the WebRTC-based and L4S-based implementations when competing with three TCP Cubic or TCP Prague flows.

While the average VMAF score is an important metric for assessing overall visual quality, it is equally important to examine how the quality evolves during a single iteration. Large fluctuations in quality causes users to become less immersed, which has a significant impact on their experience. As illustrated in Figure 10a, when competing with only one TCP Cubic flow, the WebRTC-based application maintains a relatively stable VMAF score throughout the entire iteration. The minor drops observed can be attributed to variations in frame sizes within the used point cloud sequence. Similar behavior is observed for L4S in Figure 10b. However, for L4S this behavior occurs when competing with a TCP Prague flow instead of a TCP Cubic flow, due to those flows competing over the same bandwidth share.

However, these results change when increasing the number of competing flows to two. As depicted in Figure 11a, when competing with TCP Prague flows, the WebRTC-based implementation exhibits no significant variation, as it continues to utilize the 100 Mb/s share of the available bandwidth. In contrast, when contending with two TCP Cubic flows, the quality fluctuates more noticeably than when competing with only one flow. However, as shown in Figure 11b, the L4S-based implementation does not experience such fluctuations. Instead, it maintains a stable VMAF score, suggesting that although the average score is lower when competing with TCP Prague flows, it achieves a more consistent quality level across the iteration.

These findings are further reflected in Table 1, where the intra-sequence scores reveal significantly greater variance within a single iteration for the WebRTC implementation compared to L4S. Moreover, as shown in Table 2, the WebRTC-based implementation experiences more pronounced drops in visual quality, resulting in a lower minimum VMAF score per iteration than the L4S application. This discrepancy is primarily attributed to frame drops that occur exclusively in the WebRTC implementation.

4.9 End-to-End Latency Breakdown

Most components of the pipeline are independent of the adopted transport protocol. Table 3 shows the average latency for each of these components, with the camera contributing the most to the transport-independent end-to-end latency. However, when considering the impact of the transport protocol along with three competing flows, the results in Figure 12 reveal that both the WebRTC-based and L4S-based implementations achieve similar end-to-end latencies. The slight differences in encoding and decoding times can be attributed to higher throughput leading to longer processing times. Nevertheless, as noted in Section 4.7, the L4S implementation exhibits significantly more consistent latency compared to WebRTC, with the key advantage being that L4S latency remains unaffected by the number of competing flows.

5 CONCLUSION AND FUTURE WORK

In this paper, we propose a low latency, low loss and scalable throughput (L4S)-based architecture for volumetric video conferencing using point clouds. We perform a comprehensive comparison of the proposed architecture against an open-source, web real-time communication (WebRTC)-based volumetric video delivery pipeline [14], which utilizes WebRTC to enable real-time volumetric conferencing. Our L4S-based implementation demonstrates a more stable transport latency by shifting unpredictable queuing delays to a predictable latency through packet pacing. Consequently, the transport latency of the proposed L4S architecture is determined by the pacing interval and any fixed network latency, whereas the WebRTC-based implementation experiences increased latency as the number of competing Cubic flows grows. Moreover, the L4S implementation leverages accurate explicit congestion notification (AccECN) markings to receive early congestion warnings, adjusting its rate accordingly, which prevents most packet loss. In contrast, the WebRTC implementation suffers from packet loss, with 3% of frames becoming undecodable due to missing packets and the failure of negative acknowledgment (NACK)-based retransmission. Additionally, the L4S bandwidth estimator converges from 12.5 Mb/s to 90 Mb/s within 10 seconds, representing a 45% faster convergence rate compared to the Google congestion control (GCC) estimator used in the WebRTC-based implementation. However, implementing the L4S-based approach requires bottleneck routers to have a fair queuing setup that supports AccECN marking, which may be challenging for older or legacy networks.

In future work, we plan to extend the implementation based on L4S to support bidirectional communication. This extension will enable the implementation of a many-to-many communication scenario, providing an opportunity to evaluate the performance of L4S in environments with multiple senders and receivers. Such a scenario will yield valuable insights into the scalability and efficiency of L4S in more complex, real-world conferencing setups. Furthermore, since both the WebRTC and L4S-based implementations are designed to be content-independent, extending the system to support various types of volumetric video, such as mesh-based formats, would greatly enhance the modularity of the pipeline. This flexibility is further bolstered by the use of the simple sampling quality adaptation method and the Draco codec, which both support mesh-based encoding.

REFERENCES

[1] H. Afzal, D. Aouada, B. Mirbach, and B. Ottersten. 2018. Full 3D reconstruction of non-rigidly deforming objects. *ACM Transactions on Multimedia Computing, Communications, and Applications (TOMM)*, 14, 1s.

[2] S. J. Ahn, L. Levy, A. Eden, A. S. Won, B. MacIntyre, and K. Johnsen. 2021. IEEEVR2020: exploring the first steps toward standalone virtual conferences. *Frontiers in Virtual Reality*, 2, 648575.

[3] D. S. Alexiadis, D. Zarpalas, and P. Daras. 2012. Real-time, full 3-D reconstruction of moving foreground objects from multiple consumer depth cameras. *IEEE Transactions on Multimedia*, 15, 2, 339–358.

[4] M. Bassier, M. Vergauwen, and F. Poux. 2020. Point cloud vs. mesh features for building interior classification. *Remote Sensing*, 12, 14, 2224.

[5] N. Bouzakaria, C. Concolato, and J. Le Feuvre. 2014. Overhead and performance of low latency live streaming using MPEG-DASH. In *IISA 2014, The 5th International Conference on Information, Intelligence, Systems and Applications*. IEEE, 92–97.

[6] B. Briscoe, K. De Schepper, M. Bagnulo, and G. White. 2023. Low Latency, Low Loss, and Scalable Throughput (L4S) Internet Service: Architecture. RFC 9330. (Jan. 2023). DOI: 10.17487/RFC9330.

[7] B. Briscoe, K. De Schepper, O. Tilmans, M. Kühlewind, J. Misund, O. Albisser, and A. S. Ahmed. 2019. Implementing the'prague requirements' for low latency low loss scalable throughput (L4S). *Netdev 0x13*.

[8] W. Cai, R. Shea, C.-Y. Huang, K.-T. Chen, J. Liu, V. C. Leung, and C.-H. Hsu. 2016. A survey on cloud gaming: future of computer games. *IEEE Access*, 4, 7605–7620.

[9] N. Cardwell, Y. Cheng, C. S. Gunn, S. H. Yeganeh, and V. Jacobson. 2016. BBR: congestion-based congestion control: measuring bottleneck bandwidth and round-trip propagation time. *Queue*, 14, 5, 20–53.

[10] G. Carlucci, L. De Cicco, S. Holmer, and S. Mascolo. 2016. Analysis and design of the Google congestion control for web real-time communication (WebRTC). In *Proceedings of the 7th International Conference on Multimedia Systems*, 1–12.

[11] 2024. Cwi point cloud codec. (2024). https://github.com/cwi-dis/cwipc_codec.

[12] E. d'Eon, B. Harrison, T. Myers, and P. A. Chou. 2017. 8i voxelized full bodies-a voxelized point cloud dataset. *ISO/IEC JTC1/SC29 Joint WG11/WG1 (MPEG/JPEG) input document WG11M40059 / WG1M74006*, 7, 8.

[13] W. De Bruijn and E. Dumazet. 2018. Optimizing UDP for content delivery: GSO, pacing and zerocopy. In *Linux Plumbers Conference*.

[14] M. De Fré, J. van der Hooft, T. Wauters, and F. De Turck. 2024. Scalable MDC-based volumetric video delivery for real-time one-to-many WebRTC conferencing. In *Proceedings of the 15th ACM Multimedia Systems Conference*, 121–131.

[15] K. De Schepper, O. Albisser, O. Tilmans, and B. Briscoe. 2022. Dual queue coupled AQM: deployable very low queuing delay for all. *arXiv:2209.01078*.

[16] [SW] Google, Draco 2023. URL: https://google.github.io/draco/.

[17] W. Feng, D. Kandlur, D. Saha, and K. Shin. 1997. Techniques for eliminating packet loss in congested TCP/IP networks. Tech. rep. Citeseer.

[18] S. Garg, A. Srivastava, M. Glencross, and O. Sharma. 2022. A study of the effects of network latency on visual task performance in video conferencing. In *CHI Conference on human factors in computing systems extended abstracts*, 1–7.

[19] P. Graff, X. Marchal, T. Cholez, B. Mathieu, S. Tuffin, and O. Festor. 2024. Improving cloud gaming traffic QoS: a comparison between class-based queuing policy and L4S. In *Network Traffic Measurement and Analysis Conference (TMA 2024)*. IEEE, 10.

[20] D. Graziosi, O. Nakagami, S. Kuma, A. Zaghetto, T. Suzuki, and A. Tabatabai. 2020. An overview of ongoing point cloud compression standardization activities: Video-based (V-PCC) and geometry-based (G-PCC). *APSIPA Transactions on Signal and Information Processing*, 9.

[21] S. Holmer, M. Shemer, and M. Paniconi. 2013. Handling packet loss in WebRTC. In *2013 IEEE International Conference on Image Processing*. IEEE, 1860–1864.

[22] X. Hu, C. Wang, Y. Pan, Y. Liu, Y. Wang, Y. Liu, L. Zhang, and S. Shirmohammadi. 2021. 4DLFVD: a 4D light field video dataset. In *Proceedings of the 12th ACM Multimedia Systems Conference*, 287–292.

[23] [SW], imec Virtual Wall 2023. URL: https://doc.ilabt.imec.be/ilabt/virtualwall/.

[24] I. Johansson. 2014. Self-clocked rate adaptation for conversational video in LTE. In *Proceedings of the 2014 ACM SIGCOMM workshop on Capacity sharing workshop*, 51–56.

[25] A. B. Johnston and D. C. Burnett. 2012. *WebRTC: APIs and RTCWEB protocols of the HTML5 real-time web*. Digital Codex LLC.

[26] B. Kenwright. 2019. Virtual reality: where have we been? where are we now? and where are we going?

[27] R. Neff and A. Zakhor. 1997. Very low bit-rate video coding based on matching pursuits. *IEEE Transactions on circuits and systems for video technology*, 7, 1, 158–171.

[28] 2024. Patched l4s linux kernel. (2024). https://github.com/L4STeam/linux.

[29] M. P. Sarma. 2013. Performance measurement of TCP and UDP using different queuing algorithm in high speed local area network. *International Journal of Future Computer and Communication*, 2, 6, 682.

[30] D. Shan and F. Ren. 2018. Ecn marking with micro-burst traffic: problem, analysis, and improvement. *IEEE/ACM Transactions on Networking*, 26, 4, 1533–1546.

[31] J.-P. Stauffert, F. Niebling, and M. E. Latoschik. 2020. Latency and cybersickness: impact, causes, and measures. a review. *Frontiers in Virtual Reality*, 1, 582204.

[32] J. Tideström. 2019. Investigation into low latency live video streaming performance of WebRTC. (2019).

[33] J. van der Hooft, H. Amirpour, M. T. Vega, Y. Sanchez, R. Schatz, T. Schierl, and C. Timmerer. 2023. A tutorial on immersive video delivery: from omnidirectional video to holography. *IEEE Communications Surveys & Tutorials*.

[34] J. van der Hooft, T. Wauters, F. De Turck, C. Timmerer, and H. Hellwagner. 2019. Towards 6DoF HTTP adaptive streaming through point cloud compression. In *Proceedings of the 27th ACM International Conference on Multimedia*, 2405–2413.

[35] I. Viola, J. Jansen, S. Subramanyam, I. Reimat, and P. Cesar. 2023. Vr2gather: a collaborative, social virtual reality system for adaptive, multiparty real-time communication. *IEEE MultiMedia*, 30, 2, 48–59.

[36] D. Vucic and L. Skorin-Kapov. 2019. The impact of packet loss and google congestion control on QoE for WebRTC-based mobile multiparty audiovisual telemeetings. In *MultiMedia Modeling: 25th International Conference, MMM 2019, Thessaloniki, Greece, January 8–11, 2019, Proceedings, Part I 25*. Springer, 459–470.

[37] Wowza. 2021. 2021 Video Streaming Latency Report. Tech. rep. Wowza.

[38] M. Yang, Z. Luo, M. Hu, M. Chen, and D. Wu. 2023. A comparative measurement study of point cloud-based volumetric video codecs. *IEEE Transactions on Broadcasting*.

XRgo: Design and Evaluation of Rendering Offload for Low-Power Extended Reality Devices

Steven Gao*
University of Illinois at Urbana-Champaign
Urbana, Illinois, USA
hongyig3@illinois.edu

Jeffrey Liu*
University of Illinois at Urbana-Champaign
Urbana, Illinois, USA
jliu179@illinois.edu

Qinjun Jiang
University of Illinois at Urbana-Champaign
Urbana, Illinois, USA
qinjunj2@illinois.edu

Finn Sinclair
University of Illinois at Urbana-Champaign
Urbana, Illinois, USA
finnnorth@gmail.com

William Sentosa
University of Illinois at Urbana-Champaign
Urbana, Illinois, USA
sentosa2@illinois.edu

Brighten Godfrey
University of Illinois at Urbana-Champaign
Urbana, Illinois, USA
pbg@illinois.edu

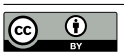
Sarita Adve
University of Illinois at Urbana-Champaign
Urbana, Illinois, USA
sadve@illinois.edu

Abstract

Extended reality (XR) devices must render high-quality 3D graphics at low latency to deliver truly immersive experiences. However, XR devices are severely power- and resource-constrained, limiting the quality of on-device (local) rendering. Offloading rendering to a powerful remote machine can enhance graphics quality, but network latency can degrade the overall experience. To mask latency, XR systems reproject the rendered frame to compensate for user motion since the rendered pose. Traditional reprojection, known as TimeWarp, uses a lightweight mechanism to compensate for latency in rotational motion, but not translational motion. Compensating for translational motion is more expensive, but is increasingly important at higher latencies.

We present XRgo, the first open-source XR runtime that supports application-transparent offloading of rendering to a remote server with client-side six-degrees-of-freedom (6-DoF) reprojection. XRgo uses OpenWarp, the first open-source and OpenXR-compatible lightweight 6-DoF mesh-based reprojection technique. Compared to on-device rendering, XRgo consistently renders at the target frame rate while consuming less power. User studies show that on average, OpenWarp provides a superior XR experience than TimeWarp. Further, OpenWarp only consumes $< 2\%$ more power than TimeWarp. Our findings reveal that the quality of experience depends on both network average latency and variability, highlighting the need for end-to-end evaluations under live network conditions.

*The first two authors contributed equally to the paper.



This work is licensed under a Creative Commons Attribution 4.0 International License.
MMSys'25, March 31–April 4, 2025, Stellenbosch, South Africa
© 2025 Copyright held by the owner/author(s).
ACM ISBN 979-8-4007-1467-2/25/03
https://doi.org/10.1145/3712676.3714444

CCS Concepts

• **Computing methodologies** → **Virtual reality**; **Perception**; Image-based rendering; • **Computer systems organization** → *Real-time system architecture*; • **Human-centered computing** → *Ubiquitous and mobile computing design and evaluation methods.*

Keywords

Extended reality, remote rendering, asynchronous reprojection, offloading, low power, wireless networks

ACM Reference Format:
Steven Gao, Jeffrey Liu, Qinjun Jiang, Finn Sinclair, William Sentosa, Brighten Godfrey, and Sarita Adve. 2025. XRgo: Design and Evaluation of Rendering Offload for Low-Power Extended Reality Devices. In *The 16th ACM Multimedia Systems Conference (MMSys'25), March 31–April 4, 2025, Stellenbosch, South Africa.* ACM, New York, NY, USA, 12 pages. https://doi.org/10.1145/3712676.3714444

1 Introduction

Extended reality (XR) devices enable the possibility of highly immersive experiences. However, truly immersive experiences require an XR headset to render high-quality 3D graphics with low latency while consuming low power. These devices must also have a lightweight form factor, thus introducing further constraints on computational power. Recently, offloading computation to a more powerful machine has introduced the potential for significant power savings. Remotely rendering scenes allows for increased geometric and shading complexity, as mobile-level devices are still largely limited to low polygon counts with forward shading. Remote machines can also benefit from techniques such as hardware-accelerated ray tracing [26] and the compute capacity to render recent graphics primitives (e.g., neural models) [25, 37] in real-time. However, the naïve solution of video streaming without compensation mechanisms suffers

from the inherent latency caused by network conditions to the point where the experience may be intolerable.

To mask this additional latency, the first line of compensation introduces post-rendering rotational reprojection, also known as TimeWarp (TW) [53], which reprojects (or warps) the rendered frame according to the user's current pose. However, TW can only account for differences in head rotation and cannot properly mask translational differences. The main challenge of reprojecting for translational motion is that new information can be introduced, e.g., object disocclusion. When this information is not available from the most recent frame, missing pixels must be filled in. The problem is that effective hole-filling techniques are usually expensive and cannot run at target frame rates. Any supplemental data used by these algorithms would also occupy more network bandwidth.

Recent advancements have introduced candidate solutions to the above challenges, but they are either proprietary, target mobile/desktop displays, simulate constant network latencies, and/or lack power measurements [7, 16, 29, 44]. It is unclear if these techniques and their evaluations directly translate to XR head-mounted displays (HMDs) under live network conditions. Furthermore, these algorithms also require modifications to the renderer, making it difficult to integrate them with existing applications immediately.

This paper presents the design and evaluation of XRgo, the first open-source end-to-end XR runtime that addresses the above concerns. Specifically, we make the following contributions:

- We present XRgo: e**x**tended **r**eality with **g**raphics **o**ffloaded, the first end-to-end open-source[1] XR runtime that supports application-transparent and OpenXR-compliant[2] offloading of rendering to a remote server with client-side six-degrees-of-freedom (6-DoF) asynchronous reprojection.
- XRgo uses OpenWarp, the first open-source[3] and OpenXR-compatible lightweight mesh-based 6-DoF reprojection algorithm. OW remains lightweight by using fixed-function rasterization hardware for hole-filling, leveraging depth data provided by the OpenXR API.
- We conduct the first user study paired with power measurements to compare quality and power between on-device rendering, offloading with TW, and offloading with OW.
- We demonstrate that, on average, OW provides better user experiences than TW, and only consumes $< 2\%$ more power.
- We show that the quality of experience depends on network latency and variability, differing considerably between live networks and constant latency simulations.

Our findings underscore the need for end-to-end systems and evaluations with network variability to study XR offloading.

2 Background and Related Work

2.1 Reprojection

Users are generally more sensitive to rendering latency when viewing through a headset than a traditional flat display. A commonly used metric is motion-to-photon (MTP) latency [54], the time from the user's motion with a new pose to the time that the displayed frame changes to reflect the new pose. Studies estimate that MTP latency should be less than 20 ms for virtual reality (VR) and 5 ms for augmented reality (AR) [12, 23]. XR runtimes attempt to mask higher end-to-end latencies through a combination of pose prediction and asynchronous reprojection (or warping). The application first renders a frame with the pose predicted for the display time. Then, right before display time, the runtime executes a reprojection pass using an updated pose [54]. If the application renderer takes too long and misses the display time, the runtime reprojects the previously rendered frame using the new pose. To minimize latency, it is important for the reprojection pass to run as close to the display time as possible. Asynchronous reprojection thus enables display responsiveness at the target frame rate even if the application renderer itself is running at intolerably high latencies.

Rotational reprojection [42, 53], mostly commonly TimeWarp (TW) [53], is an inexpensive reprojection technique that has seen widespread adoption. However, rotation-only TW only accounts for changes in the user's head rotation, and cannot compensate for differences in translational motion, which become noticeable at higher latencies [6]. To combat this, XR runtimes such as the Oculus Quest 2 have introduced translational reprojection techniques [7]. These runtimes are proprietary and the impact of translational reprojection on end-to-end XR performance or power has not been studied in the literature.

In the broader graphics literature, reusing information from previous frames has a long history of research [46]. Early work used forward reprojection techniques [35, 55], which suffered from holes from missing information. Nehab et al. presented a reverse reprojection cache [38] that could be used for reverse evaluation, but required additional computation to fill in holes. Yang et al. proposed a bidirectional, or hybrid, approach for reprojection into intermediate frames [58], but such an algorithm requires frames from the *future*, making it unsuitable for our purpose.

Some geometry-based warping algorithms [16, 34, 45, 47, 50] are similar to OW in that they render with proxy geometry, but generally require multiple reference frames to generate and composite the final frame. Many are also significantly more expensive and cannot run at full frame rate on a mobile-level device. Other image-based rendering algorithms use depth peeling [27, 44] to supply additional hole-filling information. In contrast, OW only requires the previous frame as input (minimizing bandwidth) and does not use an additional composition pass (minimizing compute), making it more suitable for an offloaded scenario. Our user studies from Section 6.1 demonstrate that these simplifications are acceptable.

Outatime is a cloud gaming system that compensates for network latency by speculatively pre-rendering frames [29]. Though not designed for XR, Outatime's error correction uses a coarse-mesh-based reprojection and hole-filling method similar to OW. However, as detailed in Section 3.3, OW uses a different overscanning mechanism to remain OpenXR-compatible. Our method has lower bandwidth requirements, with user studies indicating that it is acceptable.

More recent work also includes the use of motion vectors and/or optical flow during frame extrapolation [5, 14, 39, 57, 59]. These

[1] XRgo is implemented within ILLIXR and the code is available at https://github.com/ILLIXR/ILLIXR.
[2] OpenXR is a popular runtime-agnostic standard, permitting application portability across OpenXR runtime implementations.
[3] OpenWarp was developed as one of the authors' B.S. thesis [13] with the original standalone implementation at https://github.com/Zee2/openwarp.

techniques further use this information for the purpose of supersampling to a higher resolution, and may also assist with reprojecting dynamic objects. Although OW does not incorporate such information, the OpenXR API supports feeding motion vectors to the runtime; our offloading system makes it easy to implement and evaluate such algorithms on XR headsets.

2.2 Offloaded Rendering for XR

There is much recent research on XR systems with offloaded rendering [2, 4, 22, 24, 28, 30–32, 36, 43], proposing a variety of techniques to address network latency and bandwidth as discussed below. Our work is the first to implement 6-DoF reprojection in a complete XR system and study its impact through a user study paired with power measurements.

Like our work, several previous studies have explored offloading all application rendering to a server. The open-source ALVR [2] and ElectricMaple [43] XR systems implement application-transparent offloaded rendering with client-side asynchronous TW. Liu et al. [30] propose to minimize offloading latency by parallelizing frame encoding, pipelining the streaming process, and coordinating server-side rendering with client-side display times. The CollabVR system from Ke et al. [24] aims to minimize encoding latency and bandwidth by exploiting frame redundancy in stereo images to only transmit the left-eye image. The FovOptix system [4] studies foveated rendering and video encoding under live network conditions with VR headsets. The design and evaluation of XRgo and OW are orthogonal and reciprocally complement these works.

To handle responsiveness in mobile VR systems, many offloading systems choose to split rendering between the client and server. Some systems [22, 28, 31, 36] render the background scene as a panoramic view on the server, while rendering the foreground or other interactive objects on the client. The RenderFusion system introduced by Lu et al. [32] further introduces a partitioning scheme to dynamically balance local and remote rendering, optimizing quality according to network conditions and on-device rendering capacity. OW is complementary to these techniques and can enable more foreground objects to be rendered on the server by better handling parallax effects, reducing the power and computations on the client. XRgo may also benefit from the above techniques for locally rendering interactive or dynamic objects, as OW is better suited for static scenes (however, they are not application-transparent and break OpenXR compatibility).

2.3 ILLIXR Runtime

ILLIXR [1, 18, 19] is an open-source XR runtime and research testbed. It provides various state-of-the-art XR components (as *plugins*) and facilitates their scheduling and communication. Communication is handled via the Switchboard framework, following a publish-subscribe model with shared memory. Each plugin can publish or subscribe to event streams, referred to as *topics*. Plugins can either execute a callback whenever new data is published, or asynchronously read the most recent data from a given topic.

Plugins are compiled as dynamic libraries that are composed at runtime. This allows for a modular system that enables swap-in implementations as long as all data flow dependencies are respected. This framework is also naturally suitable for offloading because the actual flow of data is abstracted away from individual plugins. For example, as long as the reprojection plugin has access to some rendered image and its associated pose, it does not need to know whether this image was rendered locally or decoded over the network. ILLIXR supports native applications as well as OpenXR applications through integration with Monado [11].

Similar to the RemoteVIO system proposed by Jiang et al. [21], we build on the ILLIXR runtime to transmit topics over a network. However, we target the graphics pipeline, while RemoteVIO targets the head tracking pipeline.

3 XRgo System Design

This section first describes the high-level modifications to an XR runtime to enable offloading rendering. We then provide a theoretical formulation of our reprojection algorithm, and discuss implications to consider when reprojecting for offloaded XR rendering.

3.1 System Overview

Figure 1a illustrates the key rendering-related components of a traditional on-device XR runtime. Figure 1b illustrates XRgo, with the runtime split between the client device and the server.

XRgo augments the traditional on-device XR runtime by adding a network driver to transmit poses and images between the server and client. The computationally expensive application rendering is moved to the server, which requires poses streamed from the client. When a new frame is rendered, the server encodes the frame and transmits it back to the client along with the frame's associated pose. For reprojection techniques such as OW, XRgo also supports transmitting the rendered depth associated with the stereoscopic images. The challenge of rendering remotely is that MTP latency accumulates from pose transmission to the server, rendering, encoding, frame transmission to the client, and decoding. Naïvely, this latency is intolerable without any compensation mechanisms.

The client machine now maintains a lighter runtime, running pose estimation and streaming the head pose to the server. Upon receiving a new encoded frame from the server, the client decodes the frame and pushes it to a buffer pool. Before each display period, the asynchronous reprojection pass uses the most recently decoded frame to display a reprojected image to the user.

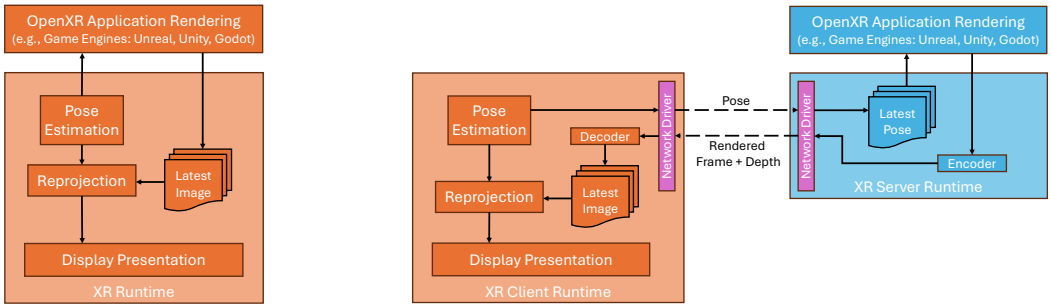
Compared to on-device rendering, the XRgo client-side decoder may be considered as the application renderer where higher latency is induced by the network and video codec. By reprojecting at the target frame rate using the latest pose, the client can effectively maintain a satisfactory MTP latency (with perceptual quality depending on the reprojection algorithm).

3.2 OpenWarp - 6-DoF Reprojection

We next describe OpenWarp, a lightweight, mesh-based 6-DoF reprojection algorithm.

Theoretical formulation. We start by introducing the basic warp equation that aims to reproject a source image from view 1 to view 2. For a given pixel at (u_1, v_1) with depth d_1, we can express its new image coordinates (u_2, v_2) with depth d_2 using Equation 1.

$$\begin{bmatrix} u_2 & v_2 & d_2 & 1 \end{bmatrix}^T = \mathbf{P}\mathbf{V}_2\mathbf{V}_1^{-1}\mathbf{P}^{-1} \begin{bmatrix} u_1 & v_1 & d_1 & 1 \end{bmatrix}^T . \quad (1)$$



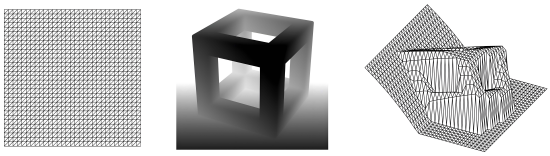
(a) Original on-device runtime (b) XRgo runtime at the client and server

Figure 1: (a) Overview of rendering-related components in the base XR runtime with on-device application rendering. (b) Overview of the XRgo runtime, highlighting the key architectural differences from the on-device runtime. Application rendering is shifted from the client device to a remote server. The client and server instances of the runtime interface through network backends that transmit poses (from the client) and encoded frames and depth data (from the server).

Here, $\mathbf{P}$ is the projection matrix and $\mathbf{V}_1, \mathbf{V}_2$ are the view matrices corresponding to view 1 and view 2 respectively. However, a naïve forward evaluation of this equation between a source and destination image is neither injective nor surjective.

The challenge with a non-injective mapping is that multiple pixels from the source image may fit into a single pixel on the destination image, thus requiring a post-warp merging process based on each of the computed depth values. On the other hand, the challenge with non-surjectivity is that no pixels from the source image may map onto another pixel on the destination image, thus requiring an additional pass to fill in these holes.

OpenWarp algorithm. We use a *reprojection mesh*, which computes the forward warp equation for each vertex of a planar mesh instead of each pixel. This allows us to distort the reprojection mesh into a continuous polygonal approximation of the rendered scene. Once we have this mesh approximation of the scene, we rasterize the mesh from a novel view. An example is visualized in Figure 2c.



(a) Base mesh (b) Depth image (c) Reprojection mesh

Figure 2: OW transforms a uniform grid of vertices (a) using the depth image (b) back into world space as a reprojection mesh (c), thus serving as a continuous scene approximation for rendering at different views.

Using a sparse reprojection mesh not only requires less computation than the per-pixel computation of a naïve per-pixel forward evaluation, but also takes advantage of two key stages of the graphics pipeline to directly address the non-injectivity and non-surjectivity of the warp equation. Firstly, even if multiple source pixels may map to the same destination pixel, merging is automatically handled by depth testing. On the other hand, even if some destination pixels have no direct mapping, the continuous representation of the reprojection mesh induces an automatic hole-filling during the fragment shader due to vertex attribute interpolation. Both of these advantages come directly from rasterization's hardware-accelerated fixed-function pipeline.

Depth adjustment. Since the reprojection mesh is a continuous representation, large discontinuities between foreground and background objects introduce an edge case that isn't well-handled by attribute interpolation. In these cases, if one vertex lies on the foreground while another lies in the background, interpolation may be obvious due to significant color differences (Figure 3a).



(a) Default depth (b) Adjusted depth

Figure 3: Without any depth adjustments in (a), the color interpolation between foreground and background is obvious. Correcting for this in (b) leads to small bump (overhang) artifacts. For backgrounds without complex patterns, these artifacts remain unobtrusive.

We handle this by checking depths within a small neighborhood of pixels. If any neighboring pixel's depth is significantly closer than the current pixel's depth (by some threshold M), the current pixel is identified as a boundary between the background and foreground. The boundary pixel's corresponding vertex depth is set to that of the closer pixel, ensuring that large stretches in the reprojection mesh mostly interpolate background colors alone. Figure 4 visualizes this depth adjustment. Since background vertices are shifted to the foreground, there is a small visible "overhang" around the edges of foreground objects, which may or may not be obtrusive depending on the scene (Figure 3b).

In most cases, overhang artifacts are more tolerable than the noticeable interpolation between contrasting background and foreground colors. Overhang artifacts also become less noticeable with finer reprojection mesh resolutions. Our results in Section 6.1 show that they are usually acceptable with our selected configurations.

3.3 Quality/Power/Bandwidth Tradeoffs

Our system design and 6-DoF reprojection algorithm introduce a set of tradeoffs to be considered when offloading rendering.

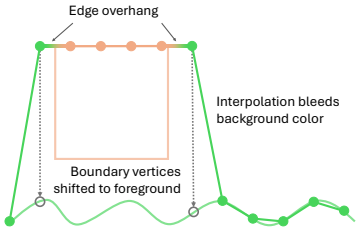


Figure 4: A 2D example scene and overlaid reprojection mesh. Given the orange box in the foreground and green waves in the background, the reprojection mesh forms an approximate scene. To avoid over-stretching between the orange and green vertices, the boundary background vertex is adjusted to match the foreground depth. This creates a slight overhang, where some background colors are closer than they should be, but reduces foreground-background stretching.

Video compression. The choice to offload rendering requires color and depth compression. When compressing at a given bitrate, two main encoding parameters affect image quality:

- *Frame rate*. The encoding frame rate is decoupled from the server-side rendering frame rate, and the most recently rendered frame is encoded when desired. At higher encoding frame rates, individual frames must be compressed more heavily, resulting in more compression artifacts and lower visual quality. At lower encoding frame rates, the client-side reprojection algorithm may need to compensate for longer latencies due to additional gaps in decoded frames.
- *Resolution*. Video encoding usually encodes at equal or lower resolution than the source. Increasing the rendered frame resolution increases the amount of information that must be compressed into a fixed number of bits, potentially reducing the image quality by introducing compressed artifacts.

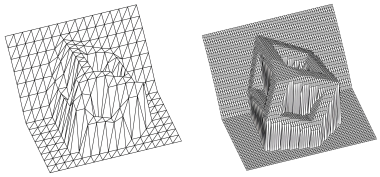
Network streaming. Although offloading removes the task of rendering from the client, it also introduces the overheads of transmitting poses, receiving frames, decoding the frames, and potentially more complex reprojection. All of these overheads increase device power consumption, so offloading should only be considered if these overheads incur less power than local rendering.

Depth compression. TW only requires color, while OW requires both color and depth. At a fixed bandwidth, TW can encode color at a higher bitrate because OW must transmit the depth as well. Thus, when there is little translational motion to compensate for, TW may have clearer visual quality compared to OW.

Overscanned field-of-view (FoV) and effective resolution. Since OW only uses the server-rendered frame to generate the reprojection mesh for re-rendering, the server-rendered frame's parameters can be decoupled from those of the final image.

For example, the server may render at an *overscanned* FoV for the client to generate a larger-than-current-frame reprojection mesh. The advantage of this approach is that more information at the borders is available during reprojection, which may potentially compensate for more latency. However, this also means that only a portion of each encoded frame is used during client-side reprojection. To maintain the same effective resolution as before (without overscanning), the encoded frame resolution may need to increase according to the rendered image's increased angular area. However, this incurs the video compression tradeoffs discussed earlier. In our selected configurations, we only overscan by 10% and find that increasing the frame resolution is not strictly necessary.

Grid Resolution. Due to the algorithm's use of vertex attribute interpolation, a finer grid resolution introduces less noticeable interpolation artifacts between vertices. An example is shown in Figure 5. However, this costs more client-side compute power.



(a) 16x16 mesh **(b) 64x64 mesh**

Figure 5: Resolution of the reprojection mesh affects the quality of the scene approximation. The (a) 16x16 mesh does not capture as many details as the (b) 64x64 mesh, but the former requires less compute (and power) for reprojection.

4 System Implementation

XRgo is implemented on top of the ILLIXR runtime (running on the server and the client) and supports OpenXR applications. The OpenXR application is simply run on the server and is not aware of any offloading scheme. The server-side runtime compresses the application frames and transmits them to the client for decoding (Section 4.1). Communication between the server and client is facilitated through a network backend in ILLIXR (Section 4.2).

Server: The server-side runtime composites and compresses the rendered frame submitted by the application. We modify the compositor to pass rendered frame depth, when available, through the entire composition pipeline. Our implementation then injects a post-composition ILLIXR plugin to compress and transmit (rather than display) the frame to the client.

Client: The client-side runtime implements an additional plugin to push poses to the server, and receive transmitted frame packets to decode. Decoded frames are pushed to a topic where a separate plugin, running at the target display rate, reprojects the most recently decoded frame according to the current pose.

4.1 Video Compression

We modify FFmpeg's [51] `hevc_nvenc` to avoid any redundant GPU-CPU memory copies during compression, such that all encoding/decoding operations remain on the GPU. Any unnecessary copies significantly increase the encoding/decoding pipeline latency. We encode both color and depth images via H.264 4:2:0 in BT.709 colorspace with extended luma range [9]. We cast the 32-bit depth to the same 8 bits across three RGB channels to conform to the codec's bit-depth. Although this is generally undesirable for encoding depth, we show in Section 6.1 that OW remains robust.

On our NVIDIA Jetson platform (discussed in Section 5.1), we implement a pipeline to decode the frames using the Jetson Multimedia API [40] and convert the decoded frames from multiplanar format to a linear RGBA format. The rest of the graphics pipeline uses the converted image for reprojection.

4.2 Network Backend

To transmit poses and encoded color and depth images, we extend ILLIXR's Switchboard *topics* [19] to support serialization and transmission over the network, which we call *networked topics*. Networked topics behave just like regular topics, except that they are initialized with hints that enable optimizations in the networking stack such as priority, ordering, packetization, etc. Since our encoding pipeline requires frames to be decoded in order of encoding, we implement a TCP network backend in Switchboard to guarantee an ordered stream. Although our case only transmits poses to the server and frames to the client, this enables Switchboard to flexibly transmit multiple topics as desired and could be used to enable the offloading of other computation in the future.

5 Evaluation Methodology

This section first presents the platforms we use and the baseline we compare against. We then describe the applications and metrics we use to evaluate our system.

5.1 Experimental Platform, Baseline, and Evaluated System Configurations

Headset display. We use the Valve Index [52] as our target display, powered by our system in various setups described below. The headset has two 1440×1600 displays with a refresh rate of up to 144 Hz. However, the 144 Hz mode is only available as an experimental option in the SteamVR runtime. Since most untethered VR headsets on the market run at 90 Hz, we also select 90 Hz as the target frame rate for both local and offloaded evaluations.

System under test. Since commercially available headsets are proprietary, we cannot run the XRGo runtime directly on them. Instead, we run the XRgo client-side runtime on an NVIDIA Jetson AGX Orin 64 GB Developer Kit [41], which is an embedded class system that approximates headset hardware. We use the 30 W power profile for the Jetson to emulate the thermal design power (TDP) of current headsets. To get a complete XR user experience, we connect the Jetson to the Valve Index headset, using the headset purely as a display. We run the XRgo server-side runtime on a high-end workstation equipped with an Intel Core i9-13900KF CPU and a discrete NVIDIA GeForce RTX 4090 GPU.

The server and client are connected with various network configurations described below. To evaluate the benefit of OW, we run the offloaded client-side configurations with both OpenWarp (referred to as OW) and TimeWarp (referred to as TW). To evaluate the benefit of offloading, we also test ILLIXR with TW running entirely on the Jetson with no offloading (referred to as Local).

Ideal baseline. We evaluate user experience with our various Jetson configurations by comparing against an ideal baseline. This baseline represents the best experience achievable with the headset tethered to a high-end workstation. We use the same workstation as the server in the XRGo configurations above. We run ILLIXR on this configuration and ensure that it can run at the target frame rate, while also using TW to reproject right before display (to minimize latency between render and display time). This setup represents the ideal quality of experience (QoE) to strive for.

Network configurations. In offloaded scenarios, we consider our client under the following set of live[4] and simulated (using tcconfig [17] with constant latency and no bandwidth constraints) network conditions (the server is always connected via Ethernet):

- Wi-Fi available in our lab, which provides approximately 500 Mbps for both upload and download bandwidth respectively, with an average 3.66 ms round-trip latency. However, there are occasional spikes of up to low-mid double digits due to varying network traffic. We compare this against a fixed constant network round-trip time of 20 ms.
- Trace-driven 5G emulation[5] following the approach from [48]. We collect time-varying latency and bandwidth traces using a Verizon 5G phone (Google Pixel 5) inside our lab that sends data to our workstation. The latency and bandwidth traces are collected separately. The traces last for 200 seconds (repeated indefinitely) and have 46.37 ± 9.87 ms (mean ± std) round-trip latency, 66.11 ± 0.79 Mbps upload bandwidth, and 225.23 ± 8.72 Mbps download bandwidth. We compare this against a fixed constant network round-trip time of 50 ms.

In total, each scene (application) we evaluate (described below) has 9 configurations: 1 for the local case and 8 across all network and reprojection permutations.

Other parameters: We match the headset display by rendering and encoding each eye buffer at 1440×1600 resolution. We encode at a total bitrate of 190 Mbps, a safe lower bound of the 5G trace (and WiFi) bandwidth. For TW, this entire bitrate can be allocated to encoding color. For OW, we partition this bitrate into 180 Mbps and 10 Mbps to encode color and depth respectively — we find that OW is relatively robust against heavily compressed depth.

For OW, we use a grid size of 360×400. We find that a neighborhood radius of 0.005 (with texture coordinates being normalized in $[0, 1]$) is appropriate for this grid size. We also overscan by 10% (Section 3.3) but use the same image resolution. This theoretically decreases the effective resolution of the transmitted image, but our user studies confirm that this is acceptable (Section 6.1).

5.2 Applications and Scenes

All scenes are exported from the Godot 4 Game Engine [3] with its Forward+ renderer. We target two different situations aiming to include diverse scene characteristics:

- **Bistro** [33], shown in Figure 6a, is an outdoor scene with primarily direct lighting, representing an architectural visualization with moderate graphics.
- **Spaceship** [20], shown in Figure 6b, is a semi-open scene with complex indirect lighting and geometry, representing a game environment with high-end graphics.

Importantly, neither application could meet the target frame rate on the Jetson in Local mode. We note that Unreal Engine 5 (UE5) can offer additional rendering features, but at the time of writing,

[4]Live refers to network conditions with time-varying latency and bandwidth, as in real networks. We used live Wi-Fi or 5G traces collected from actual 5G networks.
[5]We opted for trace-driven emulation instead of live 5G to reduce variability and ensure consistency across user tests, as the performance of live 5G networks can vary significantly at different times of the day (e.g., daytime vs. nighttime).



(a) Bistro scene **(b) Spaceship scene**

Figure 6: (a) The Bistro scene and (b) the Spaceship scene captured from a camera in the Godot Game Engine preview.

the Jetson Orin 64 GB does not support UE5 projects.[6] Using such applications could make it even more expensive to render on-device.

5.3 User Study on Quality of Experience

Since computed image metrics are not always representative of a user's actual quality-of-experience [49], we evaluate OW's perceptual quality with a user study. This study protocol is derived from prior work [21] and has Institutional Review Board approval.

5.3.1 Participant Recruitment and Demographic. We recruit 29 participants via email lists and direct contact. Participants are compensated with $10 for every 30 minutes, including any necessary breaks. 20 participants self-identify as male and 9 participants as female. Ages are 19-55 years old, with an average age of 26.38 ± 9.03 (mean ± std) years old. All participants have normal or corrected-to-normal vision. 8 participants report never interacting with XR, 16 report rarely interacting with XR (a few days a year), 4 report occasionally interacting with XR (a few days most months), and 1 report frequently interacting with XR (several days most weeks).

5.3.2 Experimental Protocol. Before any experiments, participants may freely use the baseline configuration to adjust the headset and familiarize themselves with the system. Here, they are placed in a warm-up scene, different from the two described in Section 5.2.

Once experiments begin, participants first explore the test scene presented through the baseline configuration. They are told that this is the ideal configuration that they will be asked to compare all test configurations against. They may request to revisit the baseline whenever they would like to refresh their point of reference (otherwise, it is not repeated to prevent fatigue). They are encouraged to make both rotational and translational movements, and are free to walk around the test space (a $\sim 4.5 \times 2.4$ meter rectangle).

After experiencing the baseline, the user begins trials of the test configurations described in Section 5.1, which are kept unknown to them. For each trial, the participant is asked to compare the test experience against the first baseline. To avoid bias, configurations are roughly counter-balanced by randomizing the scene order per user, the network condition per scene, and the reprojection algorithm per network condition. Participants may take breaks for as long as desired between trials to minimize fatigue, and may also choose to re-evaluate specific configurations if necessary after re-trying the baseline. Including breaks, each user spent around $60 - 90$ minutes to complete all 18 trials. The participant evaluates each trial with the following questionnaire [21], where Experience 1 is always the baseline while Experience 2 is the test configuration:

(1) Did you find any difference in the two XR experiences?
  (a) Yes, I can tell that they are different.
  (b) No, they are indistinguishable to me.
(2) Which experience did you like better?
  (a) Experience 1.
  (b) Experience 2.
(3) For the experience you ranked worse, how significant is the degradation in comparison to the better experience?
  (a) Only slightly worse or almost the same.
  (b) Worse but the experience is acceptable to me.
  (c) Much worse and not acceptable to me
(4) What contributed to the degradation you perceived (select one or more)? [7]
  (a) More jitters.
  (b) More lag.
  (c) More jerkiness.
  (d) More drift.
  (e) Lost tracking.
  (f) Other (please specify).
(5) Please share any additional comments or suggestions you may have regarding your experience.

Since we are interested in overall QoE, we only use responses to Questions 1, 2, and 3 to quantify ratings. Experience 2 delivering an indistinguishable experience or better (determined from Questions 1 and 2) receives a score of 4; if Experience 2 felt worse, the answer to Question 3 corresponds to a score of (a) $\rightarrow$ 3, (b) $\rightarrow$ 2, (c) $\rightarrow$ 1.

We do not factor Questions 4 or 5 into the rating value. Instead, we use Question 4 to potentially identify what may have affected user experience, but leave a rigorous evaluation of specific degradation causes to future work. We use Question 5 to understand unanticipated issues in our protocol, discussed in Section 6.1.4.

5.3.3 Statistical Analysis. We run a set of statistical tests to analyze the results of our user study. First, we apply the two-sample t-test and Wilcoxon Signed-Rank Test [56] to evaluate the average treatment effect. Then, we run Ordinal Logistic Regression (OLR) [15] to evaluate the effect size of each independent variable (reprojection algorithm, average network latency, and network variability).

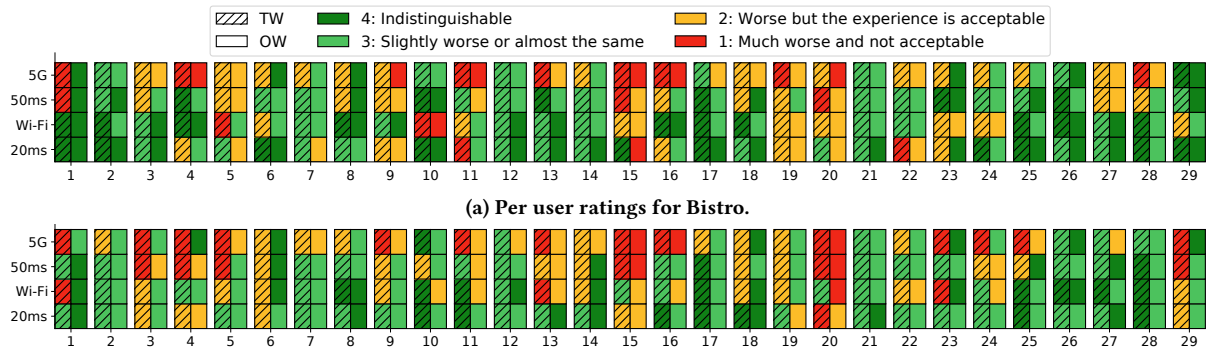
The two-sample t-test serves as a widely recognized reference for determining statistical significance. However, due to the ordinal nature of our dependent variable (user ratings), the t-test is not well-suited because 1) user rankings may not be normally distributed, 2) the population might not have equal variances, and 3) the real distance between two rankings may not be equal. The Wilcoxon test relaxes these assumptions and suits the ordinal user rankings.

We further employ OLR to explore the effect size of individual factors. OLR is suitable for modeling relationships where the dependent variable is ordinal and the intervals between them may not be equal. We employ the Brant test [8] to assess the proportional odds assumption, a prerequisite for OLR,[8] which assumes that the relationship between the predictors and the ordinal response variable is consistent across all outcome categories. The model estimates the log odds of the dependent variable being at or below a certain level

[6] https://forums.developer.nvidia.com/t/jetson-agx-orin-unreal-engin-vulkan-problem/277494

[7] Each artifact is explained in the footnotes of the questionnaire. Answers to this question are not factored into the quantitative rating.

[8] In our results, the p-values of the Brant test are all less than 0.05, confirming that the proportional odds assumption is met.



(a) Per user ratings for Bistro.

(b) Per user ratings for Spaceship.

Figure 7: Raw user study results with quantified and color-coded rating values from Section 5.3. Each user (x-axis) rated their experience with offloaded rendering on each network condition (y-axis) using both reprojection algorithms (paired bars, left is TW and right is OW). Parts (a) and (b) show results for Bistro and Spaceship respectively. Higher rating is better, with 4 (dark green) indicating that the test configuration is indistinguishable or better compared to the baseline, and 1 (red) indicating that the test configuration is unacceptable. For the local rendering configurations (not shown here), all users rated them as unacceptable, with the exception of user 28 who rated Bistro as worse but acceptable.

given the independent variables. We model average latency as a continuous variable and other independent variables as categorical. Network variability and reprojection are binary-encoded where 1 denotes live network and using OW for reprojection.

5.4 Power Measurements

We measure the total power consumption from all configurations described in Section 5.1, except for the simulated network latencies.

We use the Jetson Developer SDK's power measurement tool, `tegrastats`, to collect instantaneous voltage and current across power rails every 10 ms. The `GPU_SOC` rail measures GPU and SOC power, which includes various fixed-function hardware such as the video decoder; `CPU_CV` measures CPU power consumption as the CV cores are not utilized for our system; `I/O` measures the power supplied to display port, USB, and various other I/O functions; `DDR` measures the DDR memory power consumption.

To reduce variance in rendering power, we use a dataset to provide the trajectory rather than live experiments. We adopt the EuRoC MAV V1_02 trajectory [10]. This trajectory is replayed throughout all power trials, so the same poses are always rendered.

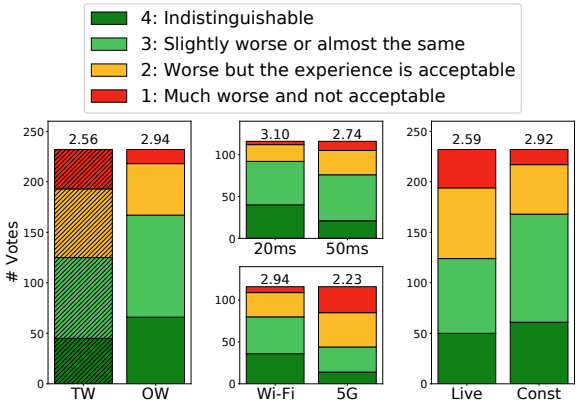
6 Results

This section presents the results of our user study (Section 6.1) and power measurements (Section 6.2).

6.1 User Study

Figure 7 presents the raw data from our user study, showing how each user (x-axis) rated their experience for each network condition (y-axis), each reprojection algorithm (left bar for TW and right bar for OW), and each scene (7a for Bistro and 7b for Spaceship). For better visualization, we use dark green, light green, yellow, and red colors to represent the ratings from best (4 for indistinguishable) to worst (1 for unacceptable).

We derive the following data from Figure 7 to inform our analysis and conclusions below. Figure 8 aggregates the number of votes for each rating to ascertain the overall impact of the reprojection



(a) Reprojection **(b) Latency** **(c) Variability**

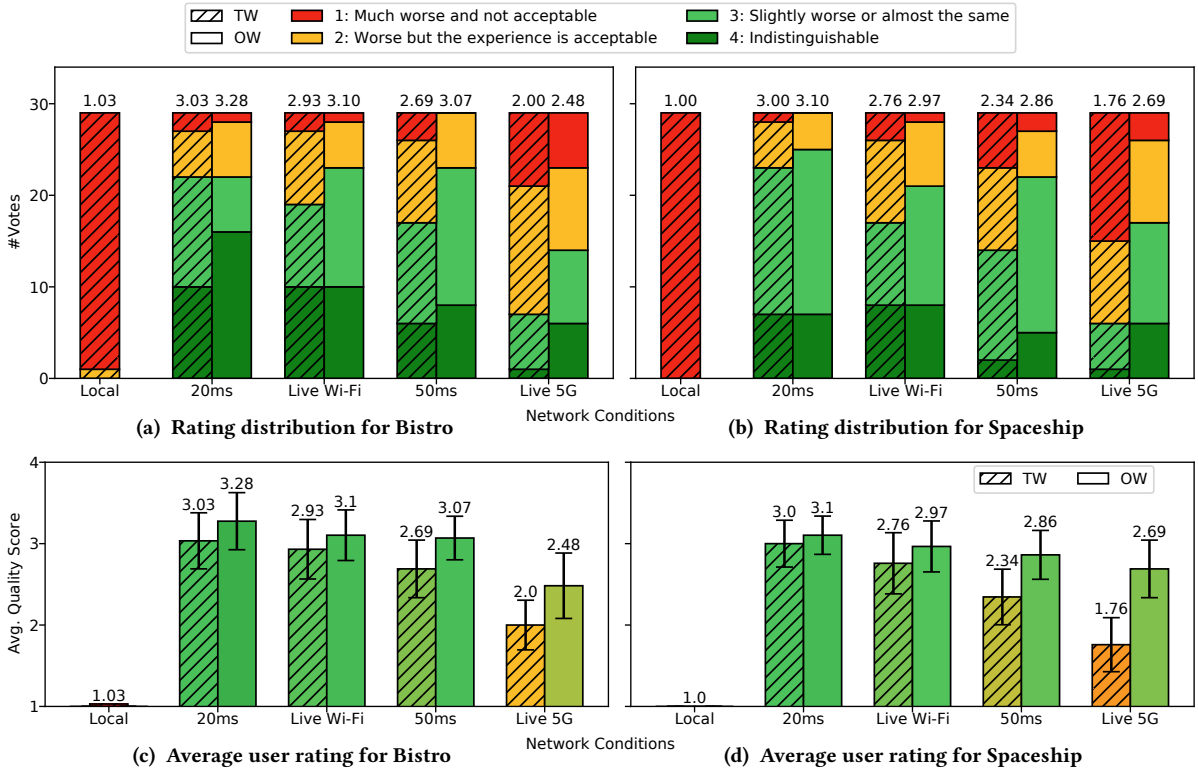
Figure 8: Aggregated votes by (a) reprojection, (b) network latencies, grouped by constant latencies on the top and live network on the bottom, and (c) network variability.

algorithm (Figure 8a), network latency (Figure 8b), and network variability (Figure 8c) on the user experience. Figures 9a and 9b aggregate the rating votes for each combination of reprojection algorithm, network latency, and network variability for the two scenes. Figures 9c and 9d show the corresponding average ratings with a 95% confidence interval. Table 1 shows results for the three statistical tests described in Section 5.3.3 to corroborate the statistical significance of our conclusions.

6.1.1 *Reprojection Algorithm.*

OW outperforms TW on average. Figure 8a shows the average user ratings for TW and OW as 2.56 and 2.94 respectively, aggregated across all experimental conditions. Overall, OW delivers an experience more similar to the baseline than TW. Table 1 shows almost-zero p-values for the t- and Wilcoxon tests, and a high OLR coefficient for OW, confirming statistical significance of our results.

Figure 8a also shows that, in our experiments, OW delivers more high-quality experiences and fewer unacceptable experiences compared to TW. For example, the dark/light green (high-quality)



(a) Rating distribution for Bistro

(b) Rating distribution for Spaceship

(c) Average user rating for Bistro

(d) Average user rating for Spaceship

Figure 9: Finer-grained rating distributions (a, b) and average ratings (c, d). 95% confidence intervals are plotted on the average ratings. For paired bars: left is TW, right is OW. The left column is Bistro and the right column is Spaceship. Higher is better.

	t-test p-value	Wilcoxon p-value	OLR coef	OLR p-value
Analysis for Aggregated Results in Figure 8				
OW vs. TW	0.000	0.000	0.7682	0.000
Avg. lat.			-0.0312	0.000
20ms vs. 50ms	0.001	0.000		
Wi-Fi vs. 5G	0.000	0.000		
Live vs. Const	0.000	0.000	-1.0135	0.000
Analysis for OW vs. TW at High Latency and Variability				
Lat. (5G + 50ms)	0.000	0.000		
Var. (Wi-Fi + 5G)	0.001	0.000		

Table 1: Statistical tests from Section 5.3.3. For the t- and Wilcoxon test, we only compare average latency under the same network variability condition (live or const). For OLR, we parameterize average latency as a continuous variable; bolded categorical variables are binary-encoded as 1. All p-values labeled 0.000 are <0.0005. For analysis of OW vs. TW at high latency and variability, data is aggregated by comparable latency (5G + 50ms) and variability (Wi-Fi + 5G).

portions constitute 72% of OW experiences vs. 54% for TW, while the red (unacceptable) part is only 6.0% for OW vs. 16.8% for TW.

6.1.2 *Latency.*

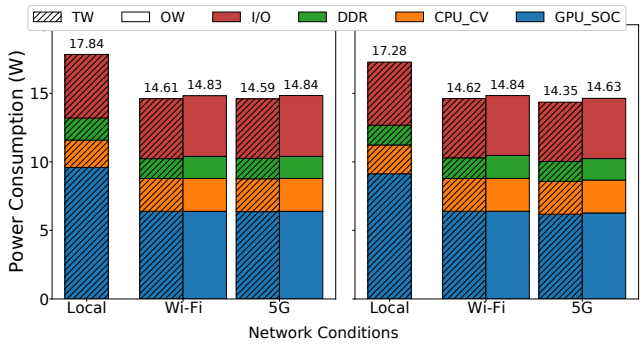
Higher latencies degrade user experience. Figure 8b examines the impact of network latency on user experience. The top compares constant latency networks at 20ms and 50ms latency, while the bottom compares live networks with an average latency of 3.66 ms for Wi-Fi and 46.37 ms for 5G. In both cases, higher latency shows a worse average rating — 2.74 vs. 3.10 for constant 50 ms vs. 20 ms and 2.23 vs. 2.94 for live 5G vs. Wi-Fi. Table 1 shows that these differences are statistically significant via the t- and Wilcoxon tests, with an OLR coefficient of −0.0311 that predicts, with high statistical significance, a negative relationship between QoE and latency.

OW outperforms TW even at higher latencies. Figure 9 shows that even though higher latencies negatively impact experience, OW outperforms TW at these latencies for both constant latency and live networks, providing a better-than-acceptable experience on average. At a constant 50ms latency, OW has average ratings of 3.07 and 2.86 for Bistro and Spaceship respectively as opposed to 2.69 and 2.34 for TW. For 5G, OW has average ratings of 2.48 and 2.69 for the two scenes compared to 2.00 and 1.76 for TW. Table 1 supports the statistical significance when comparing OW to TW under high latencies (5G + 50 ms).

6.1.3 *Live Network Conditions.*

Network variability degrades user experience. Figure 8c shows that on average, the live networks have a lower rating than the constant latency networks (2.59 vs. 2.92). Table 1 shows this difference is statistically significant, and the OLR coefficient confirms a strong negative relationship between QoE and network variability.[9]

[9]A variability comparison between live Wi-Fi and 20 ms networks is nuanced due to their different average latencies. Although Wi-Fi average latency is only 3.66 ms, Figure 9 shows that it still has slightly worse QoE than 20 ms constant latency. We intuit that network variability degrades the QoE, outweighing the lower latency. 5G and 50 ms networks, however, can be compared directly and Figure 9 shows that 5G has a consistently lower rating. A t-test and Wilcoxon test on 5G vs. 50 ms aggregated



(a) Power for Bistro **(b) Power for Spaceship**

Figure 10: Measured power on (a) Bistro and (b) Spaceship with local rendering, 5G offloading, and Wi-Fi offloading. For paired bars: left is TW, right is OW. Lower is better.

OW outperforms TW at high network variability. Figure 9 shows that OW has higher average ratings than TW under 5G and Wi-Fi for both scenes. For example, 5G ratings are 2.48 for OW vs. 2.00 for TW for Bistro. Corresponding values for Spaceship are 2.69 vs. 1.76. Wi-Fi also demonstrates higher ratings for OW over TW, but the difference is less pronounced. The tests in Table 1 establish the statistical significance that OW outperforms TW under network variability (Wi-Fi + 5G). When latency spikes, OW reveals more obvious stretching artifacts while TW users feel more abrupt movements due to high lag. Our results suggest that users, on average, prefer the OW artifacts over TW's staggered movement.

6.1.4 Constraints of Our Study. This section discusses the constraints of our study and potential future work. As detailed in Section 5.3, we asked the users to perform multiple trials with different configurations. Although the stimuli presentation schedule may introduce dependencies between trials, we aimed to mitigate this by randomizing presentation order per user and instructing them to only compare against the baseline. Due to the number of trials per user (16 excluding baseline), we did not schedule additional hidden references to prevent user fatigue. We also compensated participants based on time to encourage more careful evaluations.

We included users with diverse prior XR expertise but did not distinguish them by this demographic. Furthermore, due to the diversity of elements in the scenes, we also did not focus on scene dependency. We leave the study of these factors to future work.

Nonetheless, to the best of our knowledge, our study is the first published end-to-end user study that extensively (with 522 total trials) investigates reprojection algorithms for XR offloading.

6.2 Power Consumption

We report the end-to-end power measurements for both scenes in Figure 10. The Jetson measurements between 5G and Wi-Fi are roughly equal, so we discuss the measurements reported from the 5G experiments and note that similar observations apply to Wi-Fi.

In a locally rendered system with a 30 W TDP profile on the Jetson, both scenes fail to meet the target frame rate. Bistro averaged 20 application frames per second (FPS) and 45 reprojection FPS, and Spaceship averaged 8 FPS and 32 FPS respectively. All but one user rated this unacceptable (Figure 9). Furthermore, as Figure 10 shows, this unacceptable experience consumed around 3W more power than the offloading client running at full 90FPS.

We also benchmarked the application at 60 W TDP, the maximum on an Nvidia AGX Orin. While Bistro was able to make target frame rate most of the time, Spaceship failed to meet the target with 32 application FPS and 65 reprojection FPS on average. At 60 W TDP, Spaceship consumed 44.41 W and Bistro 44.16 W, which is around 30 W more than the thin client for rendering offload. Even with the higher energy budget of 60 W, from the authors' experience, the QoE of Spaceship running locally is still lower than that of offloading due to low application and reprojection frame rates.

Although with offloading, the client must communicate with the server, decode transmitted frames, and apply a potentially more complex reprojection at the target frame rate, we find this overhead is much smaller than the power required to render the scene locally at an acceptable frame rate. Figure 10 shows the power for the client with rendering offloaded is about 15W. Further, the power overhead of OW decoding additional depth images and rendering a reprojection mesh is small; when offloading on 5G, OW only consumes 1.81% more power than TW.

7 Conclusions

This paper presents XRgo, an end-to-end open-source XR runtime that enables application-transparent and OpenXR-compliant offloaded rendering with OpenWarp, a lightweight client-side 6-DoF reprojection algorithm. XRgo enables us to evaluate both the power and quality tradeoffs of OW against TW in an offloaded environment. Our user study demonstrates that OW's translational latency compensation significantly improves the experience compared to TW in an offloaded setting. Thus, the combination of offloading rendering with asynchronous OW delivers high-quality 3D graphics at low power and low effective latency.

The modularity of our system enables future studies in asynchronous reprojection techniques, as well as video codecs, streaming, and network protocols, in an end-to-end system setting. Our evaluation has considered static scenes, and we expect OW to have limited effectiveness with moving objects. A promising future direction is to enhance XRgo with a combination of OW and other compensation techniques, specifically those for dynamic and interactive scenes, to enable additive benefits (Section 2.2). In the future, given multiple configurations and compensation techniques with different quality-power-performance tradeoffs, a policy could be employed to dynamically choose when to offload and which configuration to use based on given resource constraints.

8 Acknowledgments

We thank the anonymous reviewers and our shepherd, Hans-Jürgen Zepernick, for their constructive feedback. We also thank Professor Han Zhao for valuable discussions of statistical analysis for our user study. This work is supported in part by the National Science Foundation under grants 2120464 and 2217144, the IBM-Illinois Discovery Accelerator Institute (IIDAI), and gifts from Cisco and T-Mobile.

across the two scenes and reprojection algorithms (not shown in Table 1) provided substantial statistical significance with p-values < 0.0001.

References

[1] 2021. ILLIXR Illinois Extended Reality testbed. https://illixr.org.

[2] 2024. Air Light VR (ALVR). https://github.com/alvr-org/ALVR.

[3] 2024. Godot Engine. https://godotengine.org/.

[4] Ahmad Alhilal, Ze Wu, Yuk Hang Tsui, and Pan Hui. 2024. FovOptix: Human Vision-Compatible Video Encoding and Adaptive Streaming in VR Cloud Gaming. In *Proceedings of the 15th ACM Multimedia Systems Conference* (Bari, Italy) *(MMSys'24)*. Association for Computing Machinery, New York, NY, USA, 67–77. https://doi.org/10.1145/3625468.3647612

[5] AMD. 2023. AMD FidelityFX Super Resolution 3. https://gpuopen.com/fidelityfx-super-resolution-3/.

[6] Michael Antonov. 2015. Asynchronous Timewarp Examined. https://developers.meta.com/horizon/blog/asynchronous-timewarp-examined/.

[7] Dean Beeler, Ed Hutchins, and Paul Pedriana. 2016. Asychrononus Spacewarp. https://developer.oculus.com/blog/asynchronous-spacewarp/.

[8] Rollin Brant. 1990. Assessing Proportionality in the Proportional Odds Model for Ordinal Logistic Regression. *Biometrics* 46, 4 (1990), 1171–1178. http://www.jstor.org/stable/2532457

[9] ITU-R Recommendation BT. 2015. *Parameter values for the HDTV standards for production and international programme exchange*. Technical Report. International Telecommunication Union.

[10] Michael Burri, Janosch Nikolic, Pascal Gohl, Thomas Schneider, Joern Rehder, Sammy Omari, Markus W Achtelik, and Roland Siegwart. 2016. The EuRoC Micro Aerial Vehicle Datasets. *The International Journal of Robotics Research* 35, 10 (2016), 1157–1163. https://doi.org/10.1177/0278364915620033 arXiv:https://doi.org/10.1177/0278364915620033

[11] Collabora. 2024. Monado - OpenXR Runtime. https://monado.dev/.

[12] Mohammed S. Elbamby, Cristina Perfecto, Mehdi Bennis, and Klaus Doppler. 2018. Toward Low-Latency and Ultra-Reliable Virtual Reality. *IEEE Network* 32, 2 (2018), 78–84. https://doi.org/10.1109/MNET.2018.1700268

[13] Sinclair Finn. 2020. *Spatio-temporal reprojection for virtual and augmented reality applications*. Bachelor's Thesis. University of Illinois at Urbana-Champaign. https://hdl.handle.net/2142/109178

[14] Jie Guo, Xihao Fu, Liqiang Lin, Hengjun Ma, Yanwen Guo, Shiqiu Liu, and Ling-Qi Yan. 2021. ExtraNet: Real-time Extrapolated Rendering for Low-latency Temporal Supersampling. *ACM Trans. Graph.* 40, 6, Article 278 (2021), 16 pages. https://doi.org/10.1145/3478513.3480531

[15] Frank E. Harrell. 2015. *Ordinal Logistic Regression*. Springer International Publishing, Cham, 311–325. https://doi.org/10.1007/978-3-319-19425-7_13

[16] Jozef Hladky, Michael Stengel, Nicholas Vining, Bernhard Kerbl, Hans-Peter Seidel, and Markus Steinberger. 2022. QuadStream: A Quad-Based Scene Streaming Architecture for Novel Viewpoint Reconstruction. *ACM Trans. Graph.* 41, 6, Article 233 (2022), 13 pages. https://doi.org/10.1145/3550454.3555524

[17] Tsuyoshi Hombashi. 2024. tcconfig. https://github.com/thombashi/tcconfig.

[18] Muhammad Huzaifa, Rishi Desai, Samuel Grayson, Xutao Jiang, Ying Jing, Jae Lee, Fang Lu, Yihan Pang, Joseph Ravichandran, Finn Sinclair, Boyuan Tian, Hengzhi Yuan, Jeffrey Zhang, and Sarita V. Adve. 2021. ILLIXR: Enabling End-to-End Extended Reality Research. In *2021 IEEE International Symposium on Workload Characterization (IISWC)*. 24–38. https://doi.org/10.1109/IISWC53511.2021.00014

[19] Muhammad Huzaifa, Rishi Desai, Samuel Grayson, Xutao Jiang, Ying Jing, Jae Lee, Fang Lu, Yihan Pang, Joseph Ravichandran, Finn Sinclair, Boyuan Tian, Hengzhi Yuan, Jeffrey Zhang, and Sarita V. Adve. 2022. ILLIXR: An Open Testbed to Enable Extended Reality Systems Research. *IEEE Micro* 42, 4 (2022), 97–106. https://doi.org/10.1109/MM.2022.3161018

[20] Jaanus Jaggo. 2023. Abandoned Spaceship Demo. https://godotengine.org/asset-library/asset/1733.

[21] Qinjun Jiang, Yihan Pang, William Sentosa, Steven Gao, Muhammad Huzaifa, Jeffrey Zhang, Javier Perez-Ramirez, Dibakar Das, David Gonzalez-Aguirre, Brighten Godfrey, and Sarita Adve. 2025. RemoteVIO: Offloading Head Tracking in an End-to-End XR System. In *Proceedings of the 16th ACM Multimedia Systems Conference* (Stellenbosch, South Africa) *(MMSys'25)*. Association for Computing Machinery, New York, NY, USA. https://doi.org/10.1145/3712676.3714442

[22] Teemu Kämäräinen, Matti Siekkinen, Jukka Eerikäinen, and Antti Ylä-Jääski. 2018. CloudVR: Cloud Accelerated Interactive Mobile Virtual Reality. In *Proceedings of the 26th ACM International Conference on Multimedia* (Seoul, Republic of Korea) *(MM'18)*. Association for Computing Machinery, New York, NY, USA, 1181–1189. https://doi.org/10.1145/3240508.3240620

[23] David Kanter. 2015. *Graphics Processing Requirements for Enabling Immersive VR*. Technical Report.

[24] Zhihui Ke, Xiaobo Zhou, Dadong Jiang, Hao Yan, and Tie Qiu. 2023. CollabVr: Reprojection-Based Edge-Client Collaborative Rendering for Real-Time High-Quality Mobile Virtual Reality. In *2023 IEEE Real-Time Systems Symposium (RTSS)*. 304–316. https://doi.org/10.1109/RTSS59052.2023.00034

[25] Bernhard Kerbl, Georgios Kopanas, Thomas Leimkühler, and George Drettakis. 2023. 3D Gaussian Splatting for Real-Time Radiance Field Rendering. *ACM Transactions on Graphics* 42, 4 (2023). https://repo-sam.inria.fr/fungraph/3d-gaussian-splatting/

[26] Emmett Kilgariff, Henry Moreton, Nick Stam, and Brandon Bell. 2018. NVIDIA Turing Architecture In-Depth. https://developer.nvidia.com/blog/nvidia-turing-architecture-in-depth/.

[27] Janghun Kim and Sungkil Lee. 2023. Potentially Visible Hidden-Volume Rendering for Multi-View Warping. *ACM Trans. Graph.* 42, 4, Article 86 (2023), 11 pages. https://doi.org/10.1145/3592108

[28] Zeqi Lai, Y. Charlie Hu, Yong Cui, Linhui Sun, and Ningwei Dai. 2017. Furion: Engineering High-Quality Immersive Virtual Reality on Today's Mobile Devices. In *Proceedings of the 23rd Annual International Conference on Mobile Computing and Networking* (Snowbird, Utah, USA) *(MobiCom'17)*. Association for Computing Machinery, New York, NY, USA, 409–421. https://doi.org/10.1145/3117811.3117815

[29] Kyungmin Lee, David Chu, Eduardo Cuervo, Johannes Kopf, Yury Degtyarev, Sergey Grizan, Alec Wolman, and Jason Flinn. 2015. Outatime: Using Speculation to Enable Low-Latency Continuous Interaction for Mobile Cloud Gaming. In *Proceedings of the 13th Annual International Conference on Mobile Systems, Applications, and Services* (Florence, Italy) *(MobiSys'15)*. Association for Computing Machinery, New York, NY, USA, 151–165. https://doi.org/10.1145/2742647.2742656

[30] Luyang Liu, Ruiguang Zhong, Wuyang Zhang, Yunxin Liu, Jiansong Zhang, Lintao Zhang, and Marco Gruteser. 2018. Cutting the Cord: Designing a High-quality Untethered VR System with Low Latency Remote Rendering. In *Proceedings of the 16th Annual International Conference on Mobile Systems, Applications, and Services* (Munich, Germany) *(MobiSys'18)*. Association for Computing Machinery, New York, NY, USA, 68–80. https://doi.org/10.1145/3210240.3210313

[31] Xing Liu, Christina Vlachou, Feng Qian, Chendong Wang, and Kyu-Han Kim. 2020. Firefly: Untethered Multi-user VR for Commodity Mobile Devices. In *2020 USENIX Annual Technical Conference (USENIX ATC 20)*. USENIX Association, 943–957. https://www.usenix.org/conference/atc20/presentation/liu-xing

[32] Edward Lu, Sagar Bharadwaj, Mallesham Dasari, Connor Smith, Srinivasan Seshan, and Anthony Rowe. 2023. RenderFusion: Balancing Local and Remote Rendering for Interactive 3D Scenes. In *2023 IEEE International Symposium on Mixed and Augmented Reality (ISMAR)*. IEEE, 312–321. https://doi.org/10.1109/ISMAR59233.2023.00046

[33] Amazon Lumberyard. 2017. Amazon Lumberyard Bistro, Open Research Content Archive (ORCA). http://developer.nvidia.com/orca/amazon-lumberyard-bistro

[34] William R. Mark, Leonard McMillan, and Gary Bishop. 1997. Post-rendering 3D Warping. In *Proceedings of the 1997 Symposium on Interactive 3D Graphics* (Providence, Rhode Island, USA) *(I3D'97)*. Association for Computing Machinery, New York, NY, USA, 7–ff. https://doi.org/10.1145/253284.253292

[35] Leonard McMillan and Gary Bishop. 1995. Head-tracked Stereoscopic Display Using Image Warping. In *Stereoscopic Displays and Virtual Reality Systems II*, Scott S. Fisher, John O. Merritt, and Mark T. Bolas (Eds.), Vol. 2409. International Society for Optics and Photonics, SPIE, 21 – 30. https://doi.org/10.1117/12.205865

[36] Jiayi Meng, Sibendu Paul, and Y. Charlie Hu. 2020. Coterie: Exploiting Frame Similarity to Enable High-Quality Multiplayer VR on Commodity Mobile Devices. In *Proceedings of the Twenty-Fifth International Conference on Architectural Support for Programming Languages and Operating Systems* (Lausanne, Switzerland) *(ASPLOS'20)*. Association for Computing Machinery, New York, NY, USA, 923–937. https://doi.org/10.1145/3373376.3378516

[37] Ben Mildenhall, Pratul P. Srinivasan, Matthew Tancik, Jonathan T. Barron, Ravi Ramamoorthi, and Ren Ng. 2021. NeRF: Representing Scenes as Neural Radiance Fields for View Synthesis. *Commun. ACM* 65, 1 (2021), 99–106. https://doi.org/10.1145/3503250

[38] Diego Nehab, Pedro V. Sander, Jason Lawrence, Natalya Tatarchuk, and John R. Isidoro. 2007. Accelerating Real-Time Shading with Reverse Reprojection Caching. In *Proceedings of the 22nd ACM SIGGRAPH/EUROGRAPHICS Symposium on Graphics Hardware* (San Diego, California, USA) *(GH'07)*. Eurographics Association, Goslar, DEU, 25–35.

[39] NVIDIA. 2022. NVIDIA DLSS 3. https://www.nvidia.com/en-us/geforce/technologies/dlss/.

[40] NVIDIA. 2024. Jetson Linux API Reference. https://docs.nvidia.com/jetson/l4t-multimedia/.

[41] NVIDIA. 2024. NVIDIA Jetson Orin. https://www.nvidia.com/en-us/autonomous-machines/embedded-systems/jetson-orin/.

[42] Edward Peek, Christof Lutteroth, and Burkhard Wünsche. 2013. More for Less: Fast Image Warping for Improving the Appearance of Head Tracking on HMDs. In *2013 28th International Conference on Image and Vision Computing New Zealand (IVCNZ 2013)*. 41–46. https://doi.org/10.1109/IVCNZ.2013.6726990

[43] PlutoVR and Collabora. 2024. ElectricMaple. https://github.com/PlutoVR/electric-maple.

[44] Bernhard Reinert, Johannes Kopf, Tobias Ritschel, Eduardo Cuervo, David Chu, and Hans-Peter Seidel. 2016. Proxy-guided Image-based Rendering for Mobile Devices. *Comput. Graph. Forum* 35, 7, 353–362.

[45] Gernot Schaufler. 1996. Exploiting Frame-to-frame Coherence in a Virtual Reality System. In *Proceedings of the IEEE 1996 Virtual Reality Annual International Symposium*. 95–102. https://doi.org/10.1109/VRAIS.1996.490516

[46] Daniel Scherzer, Lei Yang, Oliver Mattausch, Diego Nehab, Pedro V. Sander, Michael Wimmer, and Elmar Eisemann. 2012. Temporal Coherence Methods in Real-Time Rendering. *Computer Graphics Forum* 31, 8 (2012), 2378–2408.

https://doi.org/10.1111/j.1467-8659.2012.03075.x

[47] Andre Schollmeyer, Simon Schneegans, Stephan Beck, Anthony Steed, and Bernd Froehlich. 2017. Efficient Hybrid Image Warping for High Frame-Rate Stereoscopic Rendering. *IEEE Transactions on Visualization and Computer Graphics* 23, 4 (2017), 1332–1341. https://doi.org/10.1109/TVCG.2017.2657078

[48] William Sentosa, Balakrishnan Chandrasekaran, P. Brighten Godfrey, Haitham Hassanieh, and Bruce Maggs. 2023. DChannel: Accelerating Mobile Applications With Parallel High-bandwidth and Low-latency Channels. In *20th USENIX Symposium on Networked Systems Design and Implementation (NSDI 23)*. USENIX Association, Boston, MA, 419–436. https://www.usenix.org/conference/nsdi23/presentation/sentosa

[49] Rahul Singh, Muhammad Huzaifa, Jeffrey Liu, Anjul Patney, Hashim Sharif, Yifan Zhao, and Sarita Adve. 2023. Power, Performance, and Image Quality Tradeoffs in Foveated Rendering. In *2023 IEEE Conference Virtual Reality and 3D User Interfaces (VR)*. 205–214. https://doi.org/10.1109/VR55154.2023.00036

[50] Ferdi Smit, Robert van Liere, Stephan Beck, and Bernd Froehlich. 2009. An Image-Warping Architecture for VR: Low Latency versus Image Quality. In *2009 IEEE Virtual Reality Conference*. 27–34. https://doi.org/10.1109/VR.2009.4810995

[51] FFmpeg team. 2024. FFmpeg. https://ffmpeg.org/.

[52] Valve. 2024. Valve Index Headset. https://www.valvesoftware.com/en/index/headset.

[53] J. M. P. van Waveren. 2016. The Asynchronous Time Warp for Virtual Reality on Consumer Hardware. In *Proceedings of the 22nd ACM Conference on Virtual Reality Software and Technology* (Munich, Germany) *(VRST'16)*. Association for Computing Machinery, New York, NY, USA, 37–46. https://doi.org/10.1145/2993369.2993375

[54] Daniel Wagner. 2018. Motion to Photon Latency in Mobile AR and VR. https://medium.com/@DAQRI/motion-to-photon-latency-in-mobile-ar-and-vr-99f82c480926.

[55] Bruce Walter, George Drettakis, and Steven Parker. 1999. Interactive Rendering using the Render Cache. In *Eurographics Workshop on Rendering*, Dani Lischinski and Greg Ward Larson (Eds.). The Eurographics Association. https://doi.org//10.2312/EGWR/EGWR99/019-030

[56] Frank Wilcoxon. 1945. Individual Comparisons by Ranking Methods. *Biometrics Bulletin* 1, 6 (1945), 80–83. http://www.jstor.org/stable/3001968

[57] Songyin Wu, Sungye Kim, Zheng Zeng, Deepak Vembar, Sangeeta Jha, Anton Kaplanyan, and Ling-Qi Yan. 2023. ExtraSS: A Framework for Joint Spatial Super Sampling and Frame Extrapolation. In *SIGGRAPH Asia 2023 Conference Papers* (Sydney, NSW, Australia) *(SA'23)*. Association for Computing Machinery, New York, NY, USA, Article 92, 11 pages. https://doi.org/10.1145/3610548.3618224

[58] Lei Yang, Yu-Chiu Tse, Pedro V. Sander, Jason Lawrence, Diego Nehab, Hugues Hoppe, and Clara L. Wilkins. 2011. Image-based Bidirectional Scene Reprojection. In *Proceedings of the 2011 SIGGRAPH Asia Conference* (Hong Kong, China) *(SA'11)*. Association for Computing Machinery, New York, NY, USA, Article 150, 10 pages. https://doi.org/10.1145/2024156.2024184

[59] Sipeng Yang, Qingchuan Zhu, Junhao Zhuge, Qiang Qiu, Chen Li, Yuzhong Yan, Huihui Xu, Ling-Qi Yan, and Xiaogang Jin. 2024. Mob-FGSR: Frame Generation and Super Resolution for Mobile Real-Time Rendering. In *ACM SIGGRAPH 2024 Conference Papers* (Denver, Colorado, USA) *(SIGGRAPH'24)*. Association for Computing Machinery, New York, NY, USA, Article 64, 11 pages. https://doi.org/10.1145/3641519.3657424

LTS: A DASH Streaming System for Dynamic Multi-Layer 3D Gaussian Splatting Scenes

Yuan-Chun Sun[1], Yuang Shi[2], Cheng-Tse Lee[1], Mufeng Zhu[3], Wei Tsang Ooi[2], Yao Liu[3], Chun-Ying Huang[4], and Cheng-Hsin Hsu[1]
[1]National Tsing Hua University, Taiwan
[2]National University of Singapore, Singapore
[3]Rutgers University, NJ, USA
[4]National Yang Ming Chiao Tung University, Taiwan

Abstract

We present a novel DASH-based streaming system for dynamic 3D Gaussian Splatting (3DGS) scenes, addressing the challenges of streaming large amounts of 3DGS data over diverse and dynamic networks. Our **Layer, Tile, and Segment Adaptive streaming (LTS)** system combines three key features: (i) multi-layer streaming, which adapts to diverse client capabilities while balancing visual quality and bandwidth usage, (ii) tiled streaming, which reduces unnecessary data transmission by focusing on the user's viewport, and (iii) segment streaming, which divides dynamic 3DGS scenes into segments, letting clients request them dynamically to handle network fluctuations. Our experimental results demonstrate that our LTS system achieves superior performance in both live and on-demand streaming of dynamic 3DGS scenes compared to the baselines. For example, in live streaming, LTS could achieve up to 99.70% reduction in missing frames on average and deliver a maximum PSNR (Peak Signal-to-Noise Ratio) improvement of 10.08 dB. In on-demand streaming, LTS could reduce the freeze time by up to 92.01%, and increase the synthesized view quality by up to 5.14 dB in PSNR and 0.11 in SSIM (Structural Similarity Index). Our source codes are available at: https://github.com/AIINS-NTHU/LTS-DASH-Streaming-System-for-3DGS.

CCS Concepts

• **Information systems** → **Multimedia streaming**; • **Computing methodologies** → **Mixed / augmented reality**; **Image compression**; • **Networks** → **Data path algorithms**.

Keywords

Volumetric video, streaming, multi-layer representation, adaptive bitrate algorithm

ACM Reference Format:
Yuan-Chun Sun, Yuang Shi, Cheng-Tse Lee, Mufeng Zhu, Wei Tsang Ooi, Yao Liu, Chun-Ying Huang, and Cheng-Hsin Hsu. 2025. LTS: A DASH Streaming System for Dynamic Multi-Layer 3D Gaussian Splatting Scenes. In *ACM Multimedia Systems Conference 2025 (MMSys '25), March 31-April 4, 2025, Stellenbosch, South Africa.* ACM, New York, NY, USA, 12 pages. https://doi.org/10.1145/3712676.3714445

Permission to make digital or hard copies of all or part of this work for personal or classroom use is granted without fee provided that copies are not made or distributed for profit or commercial advantage and that copies bear this notice and the full citation on the first page. Copyrights for components of this work owned by others than the author(s) must be honored. Abstracting with credit is permitted. To copy otherwise, or republish, to post on servers or to redistribute to lists, requires prior specific permission and/or a fee. Request permissions from permissions@acm.org.
MMSys '25, March 31-April 4, 2025, Stellenbosch, South Africa
© 2025 Copyright held by the owner/author(s). Publication rights licensed to ACM.
ACM ISBN 979-8-4007-1467-2/2025/03
https://doi.org/10.1145/3712676.3714445

1 Introduction

3D Gaussian Splatting (3DGS) [19] has become one of the most popular 3D content formats in recent years, supporting novel view synthesis from multiple input views of a 3D scene. Such a capability enables 6 Degrees-of-Freedom (6-DoF) interactions, a Holy Grail of many immersive applications built around Extended Reality (XR) technologies, which seamlessly fuse the physical and virtual worlds.

Using 3DGS in immersive XR applications offers multiple advantages. First, it synthesizes visually appealing novel views, compared to 3D point clouds. Second, its fast rendering speed makes it well-suited for interactive applications consumed by Head-Mounted Displays (HMDs), compared to NeRF (Neural Radiance Fields) [25]. Third, its simple generation pipeline reduces the construction cost, compared to 3D meshes. Many of these immersive XR applications are distributed, e.g., film production [8], digital heritage [17, 35], robotic navigation [64], online education [2], tele-medicine [26], and remote collaboration [9], requiring 3DGS scenes to be exchanged among multiple parties.

Streaming dynamic 3DGS scenes, composed by multiple frames along the time domain, over the best-effort Internet for immersive XR applications is no easy task for two reasons: (i) the size of 3DGS frames that need to be generated, stored, and transmitted are large, and (ii) HMD users are vulnerable to cyber-sickness [43] that could be caused by late or incomplete 3DGS frames. More specifically, 3DGS frames are much larger than 3D point cloud counterparts because 3DGS frames consist of more attributes for the higher visual quality of synthesized novel views. For example, a single frame in Kerbl et al. [19] occupies about 270–734 MB, which is equivalent to 22.65–58.72 Gbps for streaming their dynamic versions at 10 frames per second (fps). Even with increasingly more studies on compacting and compressing 3DGS scenes [3, 5, 10, 20, 23, 28, 29, 34, 39, 40], the required bandwidth is still beyond the capacity of the network in many parts of the world.

This paper tackles the problem of streaming dynamic 3DGS scenes to heterogeneous clients over diverse and dynamic networks. To the best of our knowledge, dynamic 3DGS scenes have never been streamed with actual network protocols and standards. Instead, most existing dynamic 3DGS works propose representations of dynamic 3DGS scenes to support different scenarios, e.g.,

The work was partially supported by the NSTC of Taiwan (#112-2221-E-007-090-MY3).

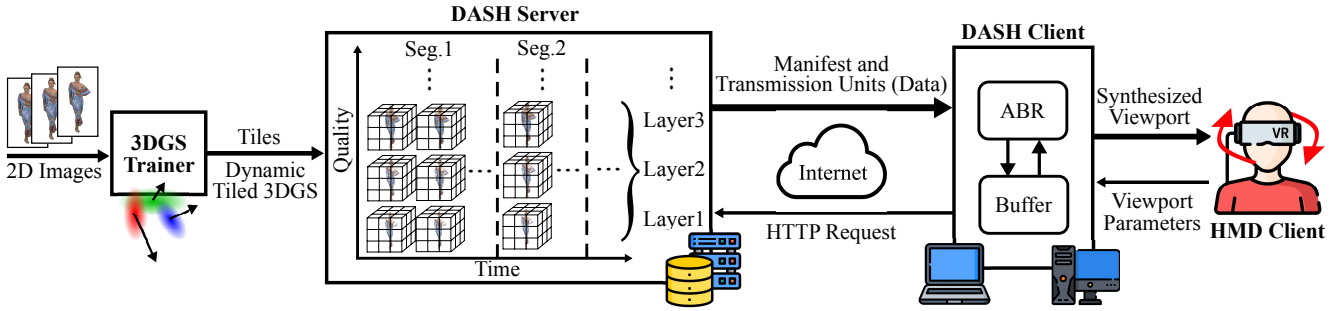


Figure 1: Overview of our LTS system for streaming dynamic 3DGS scenes.

on-the-fly 3DGS scene reconstruction [48], longer-duration 3DGS scenes [21], and resource-constrained mobile devices [54]. Only a very recent bitrate allocation work [49] mentioned RTP (Real-time Transport Protocol) and DASH (Dynamic Adaptive Streaming over HTTP) standards in their architectural design. However, they did not present a low-level design or implement an end-to-end system. In contrast, we develop, implement, and evaluate the very first dynamic 3DGS streaming system in this paper. In our work, the DASH standard is employed for (i) *live 3DGS scenes*, where late 3DGS frames are skipped as rebuffering events are prohibited, and (ii) *on-demand 3DGS scenes*, where clients pause for the packets of late 3DGS frames to arrive.

Our system relies on three principles. First, it is progressive. Different HMDs offer heterogeneous hardware resources, making one-size-fits-all 3DGS representations less suitable. Second, it is viewport-aware. It does not stream the entire 3DGS scene because each HMD client only needs a subset of Gaussian splats in its viewport at any moment. Third, it adapts to network conditions. Network bandwidth varies over time, leading to different instantaneous bitrates of dynamic 3DGS scenes. Following these principles, we propose to: (i) train each dynamic 3DGS scene into multiple *layers*, where higher layers depend on lower layers while providing incrementally better visual quality of synthesized novel views, (ii) spatially partition each dynamic 3DGS scene into multiple *tiles*, which are cuboids containing subsets of Gaussian splats, (iii) temporally divide each dynamic 3DGS scene into multiple *segments*, which are a set of consecutive 3DGS frames.

Multi-layers, tiles, and segments allow us to realize: (i) layered streaming to support heterogeneous HMD clients, (ii) viewport-dependent selection to reduce the number of requested Gaussian splats, and (iii) application-layer buffering to absorb bandwidth fluctuation. While tiling 3D information and segmenting videos can be applied easily to dynamic 3DGS frames, the current dynamic 3DGS approaches do not support layered and progressive representation. In this paper, we adapted a recent layered representation of a static 3DGS scene, LapisGS [39], to dynamic scenes and evaluated its effectiveness in streaming scenarios. Each triplet of <*layer*, *tile*, *segment*> specifies a basic *Transmission Unit (TU)* in our adaptive streaming system, named **Layer, Tile, and Segment adaptive streaming (LTS)**.

Fig. 1 presents an overview of our proposed LTS system. A 3DGS trainer produces a sequence of 3DGS frames consisting of a larger number of Gaussian splats from a set of input videos. These 3DGS frames are aggregated into multiple segments along the time domain. Each segment is represented in several layers for different quality levels. Moreover, each 3DGS scene is cut into tiles in the 3D space for view-dependent streaming. The processed content forms TU (data) files, which are stored on the DASH server. The DASH server also generates manifest files carrying metadata for the DASH client. DASH clients can be hosted on (more capable) HMDs or (nearby) computers. DASH client employs a buffer to absorb the bandwidth fluctuations for smoother playback, which is especially critical for on-demand 3DGS scenes. The crux of the DASH client is an *Adaptive BitRate (ABR)* algorithm, which selects TUs to request based on network conditions, viewport parameters, buffer dynamics, and other factors. Once the playback time arrives, the synthesized novel view for the HMD viewport is displayed.

This paper makes the following contributions:

- We built LTS based on the DASH standard for streaming live and on-demand dynamic 3DGS scenes over diverse and dynamic networks, which is the first of its kind.
- We proposed D-LapisGS, the first multi-layer dynamic 3DGS representation for LTS, to better cope with heterogeneity and dynamics among different networks, HMDs, and users.
- We formulated and solved the bitrate adaptation problem, and we implemented it as the ABR algorithm of LTS to optimally select the transmission units to request.

Our LTS system demonstrates outstanding performance in dynamic 3DGS streaming, excelling in both live and on-demand scenarios. In live streaming, LTS reduces missing frames by 99.7% and achieves a PSNR improvement of up to 10.08 dB, compared to the baseline without tiles. For on-demand streaming, LTS minimizes freeze time by as much as 92.01%, compared to the baseline without tiling and layering, yet maintaining high visual quality of 27.61 dB in PSNR and 0.903 in SSIM.

2 Related Work and Background

2.1 DASH Streaming of Dynamic 3D Scenes

Our LTS system has three unique features: layers, tiles, and segments, which were studied for content types other than 3DGS.

Layered Streaming. Scaling 3D representation into layered structures organizes videos into multiple layers, facilitating adaptation to varying network conditions and devices [1]. A base layer provides essential quality, while enhancement layers improve resolution, frame rate, or other attributes. This approach is also applied

to 3D point clouds for efficient compression. For instance, Guarda et al. [14] proposed a deep learning-based scaling method using interlaced sub-sampling for point cloud geometry coding. Similarly, Wang et al. [52] introduced a multiscale end-to-end learning framework for compressing 3D point clouds. Xie et al. [56] employed a scalable point cloud geometry compression framework that addresses the challenges of transmitting large-scale point clouds.

As for 3DGS, we are the first proposing a progressive layered approach [39], named LapisGS, for static 3DGS scenes, which encodes multiple levels of detail into a single-layered model, supporting adaptive streaming and seamless rendering. Our method demonstrates superior efficiency and flexibility, achieving substantial improvements in visual quality and model compactness. However, this method is tailored for static scenes and proves suboptimal for dynamic 3DGS streaming due to the inherent temporal relationships between successive frames caused by object movement.

Tiled Streaming. Tile-based streaming is a multimedia transmission method primarily used for interactive and immersive media streaming when a viewer can only view a subset of the content at a time. In zoomable videos [33] and 360° videos [13, 32, 36, 55, 62], for instance, the entire video frame is divided into smaller regions or *tiles*. The system dynamically selects the tiles to transmit based on the user's perspective or behavior, avoiding unnecessary data transmission. Tile-based techniques have proven highly effective for 3D content streaming, enabling efficient data management and improved streaming performance by optimizing bandwidth usage and enhancing visual quality [22]. For instance, Wang et al. [53] proposed a tile-based adaptive streaming method for point clouds to address the challenges of large data sizes and bandwidth constraints. Their approach allocates bitrates for 3D tiles based on several QoE (Quality of Experience) factors Meanwhile, Park et al. [30] proposed a system for streaming volumetric media, utilizing 3D tiles to optimize bandwidth by culling or adjusting tile details based on the user's view frustum and distance. Similarly, Yin et al. [58] utilized feature grids and residual filtering for adaptive volumetric video streaming. In the context of a large 3D scene, Forgione et al. [12] divided polygon soups representing a virtual environment into axis-aligned cells for streaming Their research indicates that the 3D tile-based method delivers high visual quality. However, 3DGS scenes are more complex than 360° videos and point clouds, which make the previous studies not directly applicable.

Segment Streaming. DASH [44] is a widely used video streaming protocol that allows the system to adaptively adjust video quality based on the user's available bandwidth. By dividing videos into small segments encoded at different bitrates, DASH ensures seamless playback by selecting the most appropriate segment quality in real-time, minimizing buffering and interruptions. DASH has been proven to significantly improve streaming quality, particularly in scenarios with fluctuating network conditions [37]. In recent years, DASH has been employed in many advanced applications, such as multi-view [45, 46, 51, 59] and 360° video streaming [16, 31, 57], where adaptive quality control is crucial for maintaining a smooth and immersive user experience. Moreover, research has extended DASH's capabilities to include 3D video streaming [18, 38, 60], demonstrating its flexibility and effectiveness in handling diverse content types, other than 3DGS.

2.2 Dynamic 3DGS Scenes

Dynamic 3DGS. 3DGS [19] renders photorealistic scenes in real-time by modeling a 3D scene as a collection of Gaussian primitives. Each Gaussian is defined by attributes such as its position x and a covariance matrix Σ, where covariance matrix $\Sigma = \boldsymbol{RSS}^T\boldsymbol{R}^T$ is derived from scaling matrix $\boldsymbol{S}$ and rotation matrix $\boldsymbol{R}$.

Building on this foundation, Luiten et al. [24] proposed Dynamic 3D Gaussians, which integrates dynamic scene reconstruction and 6-DoF tracking. This method is based on an analysis-by-synthesis framework, representing scenes as a collection of 3D Gaussians with variable positions and orientations while maintaining constant attributes such as color, size, and opacity. By incorporating physical constraints, including short-term local rigidity, local rotational similarity, and long-term local isometry losses, the approach ensures the motion of Gaussian elements remains physically consistent over time. This dynamic 3DGS framework maintains temporal coherence and enables compact dynamic representation without significant quality degradation, but it still remains insufficiently optimized for streaming applications.

Dynamic 3DGS Streaming. Research on dynamic 3DGS has grown rapidly, focusing on areas such as generation, rendering, compression, augmentation, regularization, AI-generated content, simulations, dynamic scene reconstruction, and storage [6]. Streaming dynamic 3DGS content has similarly emerged as a prominent area of research, as it offers significant potential for real-time 3D applications. One notable framework [48] leveraged existing 2D video streaming architectures by transmitting 2D images from the server and reconstructing 3DGS frames on client devices. Similarly, Liu et al. [21] addressed dynamic 3DGS streaming through a sliding window approach tailored for volumetric video streaming. Their method is particularly advantageous for frequently changing dynamic scenes, enabling smoother playback. Wang et al. [54] proposed a framework that converts dynamic 3D Gaussians into 2D videos, enabling efficient transmission and rendering through hardware video codecs. The method is designed specifically for mobile devices to support high-quality volumetric video playback. However, these studies did not explicitly address how to transmit dynamic 3DGS content over the network. In contrast, our earlier work [49] proposed a Rate-Distortion (R-D)-based streaming architecture designed to optimize 3DGS streaming by estimating the R-D characteristics of 3DGS frames and encoding them with adaptive parameters. This approach enhanced streaming efficiency and demonstrates strong performance in handling complex scenes. Nevertheless, we did not consider how to dynamically adjust streaming to mitigate delays caused by encoding large 3DGS frames.

3 The Proposed LTS System

3.1 Layered Streaming

LapisGS [39] successfully balanced visual fidelity with the compactness of the model using layered structure, and Dynamic 3DGS [24] ensures temporal coherence by incorporating physical constraints. The current problem at our hand is, however, more challenging as we have to create a *multi-layered* and *frame-dependent* representation for DASH-based dynamic 3DGS streaming. To overcome these limitations, we introduce Dynamic LapisGS (D-LapisGS in short), an innovative representation specifically tailored for efficient

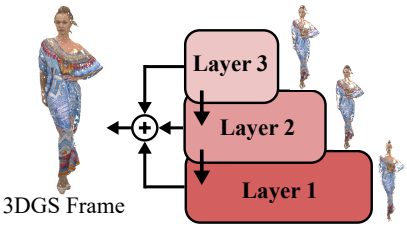


Figure 2: Layer structure of a 3DGS frame in LTS.

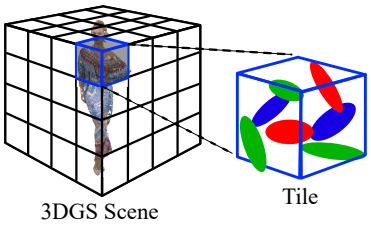


Figure 3: Spatial structure of a 3DGS frame in LTS.

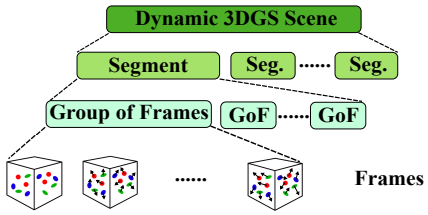


Figure 4: Temporal structure of a dynamic 3DGS scene in LTS.

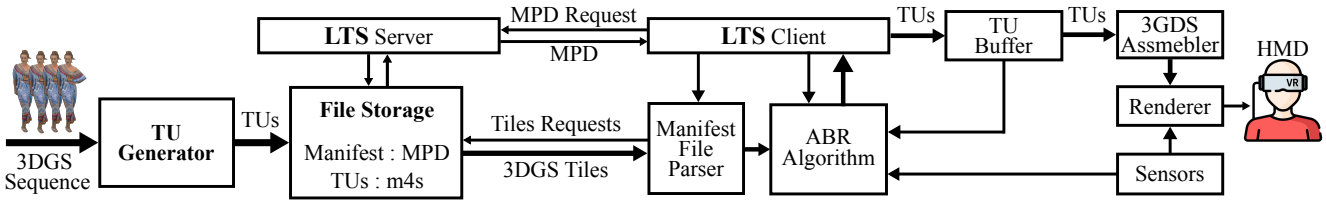


Figure 5: System architecture of our proposed LTS system.

dynamic 3DGS streaming. This approach synthesizes the fundamental concepts of Luiten et al. [24] and the LapisGS [39] to create a layered, frame-dependent architecture that better meets dynamic scene requirements.

D-LapisGS builds upon the progressive multiscale training methodology of its predecessor. The system initially trains a low-resolution base layer, followed by successive enhancement layers utilizing higher-resolution datasets, as shown in Fig. 2. These supplementary layers enhance the detail captured by previous layers while maintaining fixed parameters. The framework dynamically optimizes layer opacity to balance their relative contributions during subsequent training phases. This hierarchical structure enables the model to construct a coarse layout of the scene early on, while later layers progressively incorporate higher-frequency details, thereby enhancing convergence efficiency and minimizing redundancy across quality levels. To maintain temporal coherence, D-LapisGS treats Gaussians as particles governed by physical constraints, including local rigidity, rotational similarity, and long-term local isometry. These physical priors ensure coherent particle neighborhood movement and maintain spatial relationships over time, functioning as optimization regularizers.

3.2 Tiled Streaming

For 6-DoF immersive XR streaming, each HMD user moves freely in a 3D environment, where the user's viewport does not always cover the whole 3DGS scene. Hence, we incorporate tiled streaming that cuts the scene into multiple cuboids. As illustrated in Fig. 3, our scene is segmented into cuboids of uniform size based on the location of 3D Gaussian splats in the 3D coordinate system. More comprehensive and adatpvie tile segmentation approaches are also possible. By taking the HMD viewport parameters, such as position and orientation, as input, our system determines and only transmits the *visible* tiles to cut down the network and computing overhead. This approach reduces the latency and enhances the overall visual quality by focusing resources on the visible regions of the 3D environment.

3.3 Segmented Streaming

To adapt to diverse and dynamic network conditions over time, our system employs the segmentation approach in the DASH standard, as illustrated in Fig. 4. In particular, each dynamic 3DGS scene consists of a sequence of 3DGS frames, where G consecutive frames form a Group of Frames (GoF), encoded into one intra-coded and $G-1$ inter-coded frames. C consecutive GoFs are then aggregated into a segment, which is streamed together. Last, M segments constitute the dynamic 3DGS scene, which contains $G \times C \times M$ 3DGS frames in total. With segments, HMD clients in our system could dynamically determine which segments to request at what time based on the changing network bandwidth and other factors. By buffering additional segments during favorable network conditions, our system can mitigate the adverse effects of poor network conditions encountered shortly thereafter, particularly when streaming on-demand 3DGS scenes.

3.4 System Architecture

Fig. 5 illustrates the overall architecture of our LTS system, which is an extension of the DASH standard designed for 2D and 360-degree videos. We first introduce the LTS server, which starts with a TU generator that turns a dynamic 3DGS scene into a set of TUs. TUs are the minimum sets of bytes from a given dynamic 3D scene that can be requested and transmitted over the network. In particular, the TU generator creates multiple layers, which are encoded by an enhanced 3D point cloud codec [49]. The encoded frames are cut into tiles spatially and then form segments temporally. The resulting TUs are saved in the storage as `m4s` files, which is a widely used container file format in DASH streaming. We note that each TU can be uniquely identified by a triplet of its layer, tile, and segment

indexes. The TU generator also keeps track of the size of individual TUs and computes the expected quality of synthesized novel views, which are crucial to the client in deciding which TUs to request. Based on these results, the TU generator produces manifest files, called Media Presentation Description (MPD), consisting of codec information, playback duration, frame rate, bandwidth requirements, TU URLs, etc. These manifest files are also stored in the storage. The server implements an HTTP server that transmits the TU and manifest files to clients upon request.

Next, we present the LTS client, which is responsible for selecting the best TUs to request based on the network conditions, client capabilities, and HMD user behaviors. The client consists of an HTTP client, which retrieves manifest and TU files from the server. A manifest file parser analyzes manifest files to get metadata about the target dynamic 3DGS scene. TU files are first sent to the TU buffer, which could absorb the varying delays due to fluctuating network bandwidth. During playback, corresponding TUs are moved from the TU buffer to the 3DGS assembler, which then merges and decodes these TUs into 3DGS frames that can be rendered for HMD display based on the current HMD position and orientation (i.e., 6-DoF pose of the viewport) determined by the inertial and RF (Radio Frequency) sensors. The brain of the LTS client is its ABR algorithm, which instructs an HTTP client to request specific TUs based on: (i) the metadata from the manifest file parser, (ii) the estimated network bandwidth from the HTTP client, (iii) the buffer level from TU buffer, and (iv) the viewport position and orientation from HMD sensors. Given that the ABR algorithm is the crux of our LTS system, we detail its design in the next section.

4 Bitrate Adaptation Algorithm

4.1 Problem Statement

Our ABR algorithm consists of two steps: (i) TU selection and (ii) TU sorting. TU selection essentially solves an optimization problem to maximize the expected quality of synthesized novel views within a sliding allocation window of W segments subject to an estimated bit budget B. B is computed based on the throughput measurements of prior segments. To filter out high-frequency noise, we employ Exponentially Weighted Moving Average (EWMA), which assigns an empirically chosen weight of 20% to the most recent measurement. The optimization algorithm is solved once every allocation step, denoted as S segments, where $S \leq W$. We note that our optimization problem only considers visible TUs, as streaming other TUs is simply a waste of resources. Moreover, our optimization problem takes the already buffered TUs as inputs and only requests TUs that haven't been requested and transmitted. By repeatedly solving the optimization problem, our ABR algorithm adapts to the network, system, and user dynamics.

The second step considers the layer dependency, tile importance, and time urgency when sorting the set of selected TUs to be requested. The following sorting criteria are employed: (i) earlier segments, (ii) (within the same segment) lower layers, and (iii) (within the same layer) closer tiles are requested sooner. We request earlier segments sooner because they are closer to their playback time and are more urgent. Furthermore, we request lower layers sooner because, without layers beneath them, higher layers cannot be rendered. Last, inspired by Fang et al. [11], we compute the distance between each tile's center and the current HMD position and request closer tiles sooner. The intuition is that closer tiles tend to occupy larger fractions of the viewport, and humans are more likely to be attracted by them.

4.2 Mathematical Formulation

Next, we describe the formulation of our optimization problem. We let L, T, and W represent the numbers of layers, tiles, and segments, respectively, in the current allocation window. From the manifest files, we get to know the size of each TU, written as $R_{w,l,t}$, where $w \in \{1, 2, \ldots, W\}$, $l \in \{1, 2, \ldots, L\}$, and $t \in \{1, 2, \ldots, T\}$. The manifest files shall also provide the expected quality of synthesized novel views. Our pilot tests indicated that estimating per-tile quality at the server is challenging because many Gaussian splats span across multiple nearby tiles. Hence, we calculated the expected quality levels at the layer-segment granularity, which are computed using ground-truth images from the testing dataset. Specifically, for segment w ($w \in \{1, 2, \ldots, W\}$), the expected quality improvement of receiving layer l on top of all lower layers $l' < l$ is written as $Q_{w,l}$ ($l \in \{1, 2, \ldots, L\}$). Along the same line, we decided to perform a *visibility check* on the dynamic 3DGS scene before running the ABR algorithm. We use a boolean value $V_{w,l,t} \in \{0, 1\}$, where $w \in \{1, 2, \ldots, W\}$, $l \in \{1, 2, \ldots, L\}$, and $t \in \{1, 2, \ldots, T\}$ to capture visibility, where $V_{w,l,t} = 1$ iff this TU is visible in the HMD viewport. In addition, we need to know if a TU has already been buffered. We define $H_{w,l,t}$ in the same way as $V_{w,l,t}$, where $H_{w,l,t} = 0$ iff this TU has already been buffered. Our optimization formulation only considers downloading TUs whose $V_{w,l,t} = 1$.

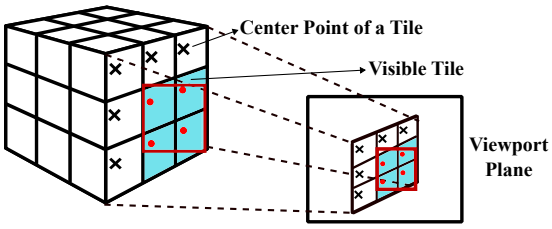


Figure 6: Illustration of visibility check.

The optimization formulation selects the layers and segments to request, as the invisible tiles have been determined and excluded. We use a boolean variable $x_{w,l}$, where $w \in \{1, 2, \ldots, W\}$ and $l \in \{1, 2, \ldots, L\}$ to denote whether this layer-segment should be requested: $x_{w,l} = 1$ iff it is chosen. Before moving on, we define an intermediate boolean variable $Y_{w,l}$, where $w \in \{1, 2, \ldots, W\}$ and $l \in \{1, 2, \ldots, L\}$ to indicate if any tiles in this layer-segment need to be requested. Note that $Y_{w,l}$ is actually a function of our inputs (and independent to our decision variables $x_{w,l}$), which can be computed by:

$$Y_{w,l} = \left\lceil \sum_{t'=1}^{T} V_{w,l,t'} \cdot H_{w,l,t'} \Big/ T \right\rceil, \quad (1)$$

$\forall\ w \in \{1, 2, \ldots, W\}$ and $l \in \{1, 2, \ldots, L\}$. Here $Y_{w,l} = 1$ iff this layer-segment has at least a tile that hasn't been buffered.

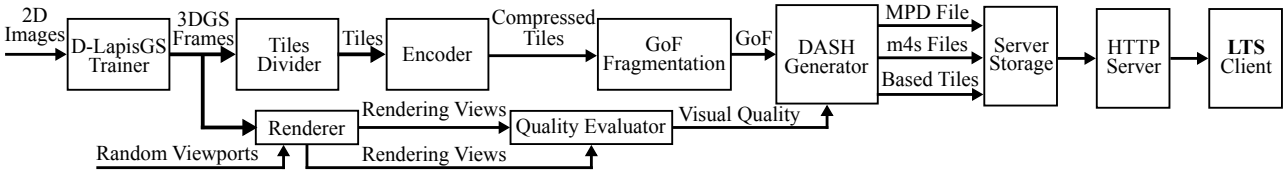


Figure 7: The LTS server implementations.

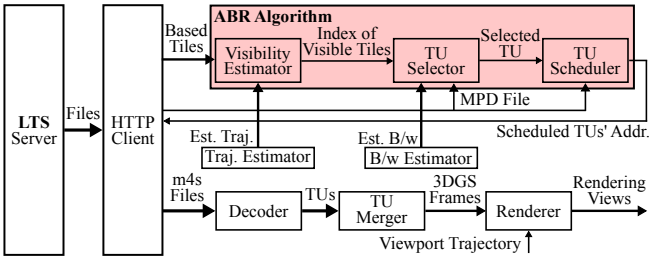


Figure 8: The LTS client implementations.

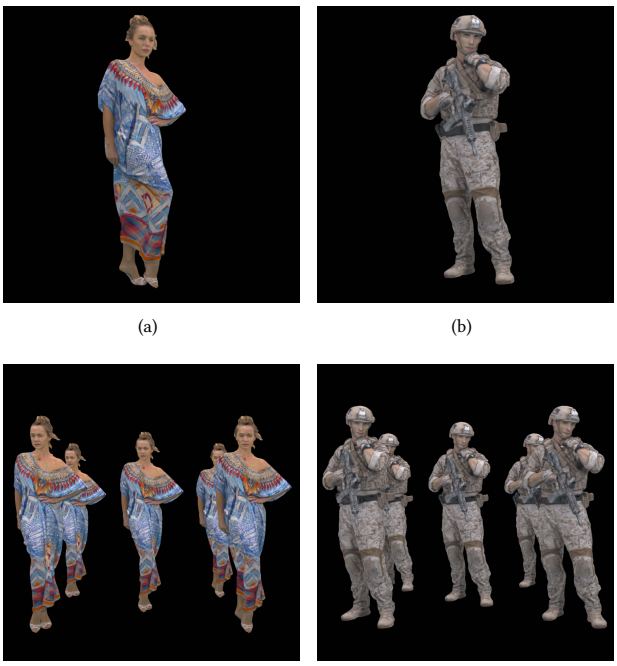


Figure 9: Sample rendered views from: (a) LongDress$_1$, (b) Soldier$_1$, (c) LongDress$_5$, and (d) Soldier$_5$

The optimization problem can be written as:

$$\mathbf{X}^* = \text{argmax}_{\mathbf{X}} \sum_{w=1}^{W} \sum_{l=1}^{L} \left[Q_{w,l}(1 - Y_{w,l}) + x_{w,l} Y_{w,l} \right] \cdot \alpha^{W-1} \quad (2a)$$

$$\text{s.t.} \sum_{w=1}^{W} \sum_{l=1}^{L} x_{w,l} \sum_{t=1}^{T} R_{w,l,t} \cdot H_{w,l,t} \leq B; \quad (2b)$$

$$x_{w,l} \leq Y_{w,l} \; \forall \; w = \{1, 2, \ldots, W\}, l = \{1, 2, \ldots, L\}; \quad (2c)$$

$$x_{w,l} \cdot Y_{w,l} \geq x_{w,l+1} \cdot Y_{w,l+1} \; \forall \; w = \{1, 2, \ldots, W\}, l = \{1, 2, \ldots, L-1\}; \quad (2d)$$

$$x_{w,l} \in \{0, 1\} \; \forall \; w = \{1, 2, \ldots, W\}, l = \{1, 2, \ldots, L\}. \quad (2e)$$

Here, the objective in Eq. (2a) considers the overall quality of the synthesized novel views, including both the buffered and selected layer-segments. We note that a factor $0 \leq \alpha \leq 1$ is used to prioritize the layer-segments with closer playback time; α is a system parameter. In terms of the constraints, Eq. (2b) ensures the total size of the requested tiles does not exceed the bit budget; Eq. (2c) makes sure that we don't request already buffered layer-segments; and Eq. (2d) validates the layer dependency, i.e., we request a layer only if all its lower layers are either buffered or also requested. Lastly, Eq. (2e) defines the boolean decision variables, which are collectively written as $\mathbf{X}$.

4.3 The Proposed ABR Algorithm

We propose an ABR algorithm with two steps: visibility check and full space search.

Visibility check. Before solving the optimization problem, we first check if each tile is visible in the HMD viewport. Here, a virtual camera projects every tile in a 3D coordinate system onto a 2D viewport plane. In addition, two 3D coordinate systems, world and camera, exist. Several viewport/camera parameters are needed to transform a point in the world coordinate system to the camera coordinate system and to project a point in the camera coordinate system to a 2D viewport plane. These parameters include the rotation matrix, translation matrix, focal length, principal point, and resolution.

Fig. 6 gives a high-level idea of our visibility check. We use the tile center $(x, y, z)_W$ in the world coordinate system to represent that tile. Following Szeliski [50], we transform $(x, y, z)_W$ to $(x, y, z)_C$ in the camera coordinate system using the rotation matrix and translation matrix. After that, we project $(x, y, z)_C$ to $(\phi, \psi)_V$ on the viewport plane using the focal length, principal point, and resolution. Comparing $(\phi, \psi)_V$ with the width and height of the HMD viewport allows us to determine if the tile is visible or not. Repeating the same check with all tiles, potentially in parallel, helps us to filter out invisible tiles.

Full space search. The formulation in Eq. (2) is a Binary Integer Programming (BIP) problem, which is non-convex by nature and is difficult to solve. Fortunately, the search space of our problem is rather small, as typically: (i) a small number of layers (L) and (ii) a rather short allocation window (W) are considered. This is because too many layers incur high training overhead, while too long allocation windows increase the estimation error on the network

bandwidth. Hence, we propose to solve our formulation optimally using a full space search. Throughout our experiments, we found our algorithm terminates within 40 ms, which is fairly reasonable as our problem is only solved once every few seconds at most.

5 Performance Evaluations

5.1 Implementations

We have implemented the proposed LTS system in a mixture of Python and C. Fig. 7 illustrates the server-side implementations of our LTS system. The server first turns 2D images into multi-layer 3DGS frames using the *D-LapisGS Trainer*. These frames are then divided into tiles through the *Tile Divider*. Each tile is compressed by an *Encoder*. We use an enhanced Draco [49] in its lossless mode for 3DGS compression. The compressed tiles are grouped into GoF via the *GoF Fragmentation* and subsequently passed to a *DASH Generator*. Meanwhile, the frames are also sent to *Renderer* to generate 2D views from 30 random viewpoints, which are evaluated by the *Quality Evaluator* for the estimated visual quality. The *DASH Generator* creates tiles based on the estimated qualities, TU size, and other parameters, such as codec, frame rate, and playback time. The results are stored in MPD and m4s files in the *Server Storage*, which is connected to the *HTTP Server* for data transmission.

Fig. 8 illustrates the client-side implementations of our LTS system. For brevity, we omit all the arrows representing requests. When the streaming starts, the client requests MPD file through the *HTTP Client*. After receiving MPD files, the system runs the ABR algorithm. The *Visibility Estimator* determines which tiles are visible based on the estimated trajectory provided by the *Trajectory Estimator*. If not otherwise specified, we employ user traces in our experiments, which can be considered as a very accurate estimator. Then, *TU Selector* selects the best TUs based on the available bandwidth estimated by the *Bandwidth Estimator* and MPD files. We implement the EWMA filter [4] for a throughput-based bandwidth estimator. The *TU Scheduler* determines the priority levels of individual TUs and outputs them in an ordered list. The HTTP server follows the list to request the `m4s` files. Once the client receives the files, the *Decoder* decompresses them into TUs and merges the TUs by the *TU Merger* to reconstruct the 3DGS frames. Finally, the *Renderer* renders the views following the client trajectory for visualization.

5.2 Setup

Dynamic 3DGS scenes. We employ two dynamic 3D sequences from the 8i point cloud dataset [7], *LongDress* and *Soldier*, in our experiments. To generate corresponding dynamic 3DGS scenes from them, we render each point cloud frame from multiple viewpoints into images and use them to train a 3DGS frame following the settings in Shi et al. [42]. Specifically, Open3D [63] version 0.15.1 is used for 2D rendering, where the width and height of the rendered images are both 600 pixels. For model training and testing, we generate 100 random views for training and another 200 random views for testing with the camera settings of Mildenhall et al. [25].

To vary the scene complexity level, we duplicate the considered 8i object d times and refer to them as LongDress$_d$ (or LD$_d$) and Soilder$_d$ (or SO$_d$). To get long enough dynamic 3DGS scenes, we repeat 30 3DGS frames 30 times, which leads to four 30-second scenes at 30 fps. In the rest of this paper, we present sample results from LD$_1$, SO$_1$, LD$_5$, and SO$_5$, as shown in Fig. 9. In our dynamic 3DGS scenes, each GoF consists of 30 frames, with 1 I-frame and 29 P-frames, encoded at full, half, and one-eighth resolutions for 3-, 2-, and 1-layer representations. For tiling, LD$_1$ and SO$_1$ have bounding boxes of $2.4\times2.4\times2.4$ m^3, while LD$_5$ and SO$_5$ have bounding boxes of $4.8\times4.8\times2.4$ m^3. The dimensions of the tiles in all dynamic 3DGS scenes are $0.6\times0.6\times0.6$ m^3. Table 1 gives the key statistics of these four scenes. Using these scenes, we generate four representations for comparisons: (i) our proposed D-LapisGS, (ii) LapisGS [39], (iii) D3DGS [24], and (iv) ordinary 3DGS [19]. Among them, LapisGS and D3DGS are the State-Of-The-Art (SOTA), while the resource-demanding 3DGS generates the highest-quality novel views serving as reference views for quality metrics. All training hyperparameters were maintained constant throughout all experimental stages.

HMD trajectories. As reported in previous studies [38, 41, 47, 61], the users tend to spend most of their time looking toward the frontal body of the avatar in volumetric videos, with particular focus on the avatar's face. To better evaluate the generalization ability of our system, we consider four 30-second trajectories, all following a spiral path rotating at 8 degrees per frame and pointing towards the center of the dynamic 3DGS scenes. The first two trajectories always maintain a distance of 2.5 m to the center, while the other two trajectories start with a distance of 5 m and gradually reduce it to 2.5 m. All four trajectories start at an elevation of 2 m, but two of them move up for 2 m in total, while the other two move down. We repeat all our experiments with these four diverse trajectories and report average performance among them.

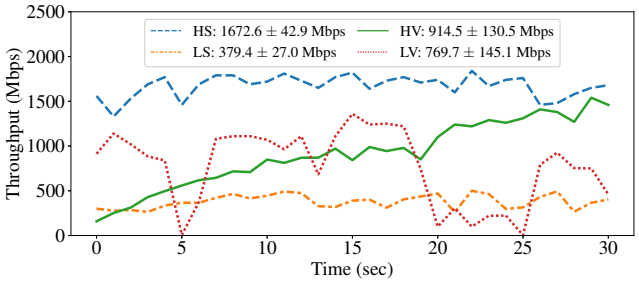


Figure 10: Throughput of four considered network traces from a 5G testbed.

Network traces. To emulate real network conditions, we employ the network traces from the Lumos5G dataset [27]. We first classify all the network traces into four categories:

- *High Stable (HS)*, whose average throughput is in the top 33% and whose throughput variance is in the bottom 10%.
- *Low Stable (LS)*, whose average throughput is in the bottom 33% and whose throughput variance is in the bottom 10%.
- *High Varying (HV)*, whose average throughput is in the top 33% and whose throughput variance is in the top 10%.
- *Low Varying (LV)*, whose average throughput is in the bottom 33% and whose throughput variance is in the top 10%.

We randomly select four network traces from each category and plot their throughput over time in Fig. 10. For evaluations, we emulate the network condition following the network traces.

Table 1: Dynamic 3DGS Scenes Used in Our Experiments (3-/2-/1-Layer)

Scene	LongDress_1	Soldier_1	LongDress_5	Soldier_5
Per-Frame No. Gau. (10^3)	255 / 137 / 66	306 / 166 / 78	1274 / 683 / 331	1528 / 830 / 392
Average Frame Size (Mb)	72.01 / 38.63 / 18.69	86.40 / 46.91 / 22.15	260.36 / 193.12 / 93.46	308.17 / 234.56 / 110.74
Bounding Box (m^3)	$2.4 \times 2.4 \times 2.4$	$2.4 \times 2.4 \times 2.4$	$4.8 \times 4.8 \times 2.4$	$4.8 \times 4.8 \times 2.4$

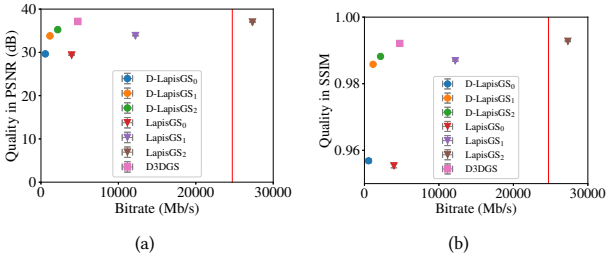


Figure 11: Comparing different representations and layers with the ordinary 3DGS: (a) quality in PSNR and (b) SSIM. The red vertical line indicates the size of the ordinary 3DGS.

Metrics. We evaluate our system using both visual quality and streaming performance metrics. PSNR and SSIM [15] are used to measure the pixel-level fidelity and the perceived visual quality of synthesized views, respectively. For streaming performance, we measure missing frames, which represent the number of frames that fail to meet their designated playback deadlines, and freeze time, which indicates the cumulative duration of rebuffering events required to accommodate delayed tile deliveries. Missing frames are particularly critical for live streaming scenarios, while freeze time is crucial for on-demand streaming applications.

Streaming scenarios. We consider two common scenarios: live and on-demand streaming. In live streaming, the server generates frames in real-time, while user focus more on timely delivery of frames. In contrast, in on-demand streaming, the server prestores all frames, the client can preload frames, and users ask for better visual quality.

ABR baselines. We are not aware of any ABR algorithm designed for DASH streaming of dynamic multi-layer 3DGS scenes. Therefore, we employ three ABR heuristics that request all TUs but with different playback criteria to serve as the baseline for our comparisons. When a baseline ABR algorithm is executed, it requests all TUs in its allocation window. Moreover, at the playback time of the current segment, the baseline algorithm checks its playback criterion to determine if the frames should be skipped (for live streaming) or the playout should freeze (for on-demand streaming). In particular, the considered baselines are: (i) *No Adaptation (NA)*, which requires all TUs of the current segment to be buffered, (ii) *No Tiling (NT)*, which requires all TUs belong to all tiles of at least one layer of the current segment to be buffered, and (iii) *No Layer (NL)*, which requires all TUs belong to the visible tiles of all layers of the current segment to be buffered.

Experiment environment. The LTS server and client are executed on workstations with AMD 32-core CPUs at 2.80 GHz and NVIDIA GTX-3090 GPUs with 24 GB VRAM. The 3DGS trainer is executed on a workstation with an NVIDIA H100 GPU. If not otherwise specified, we let $\alpha = 0.9$, $G = 30$, $C = 1$, $W = 3$, $S = 1$, $M = 30$, $L = 1$, and viewport resolution to be 1024 ×1024.

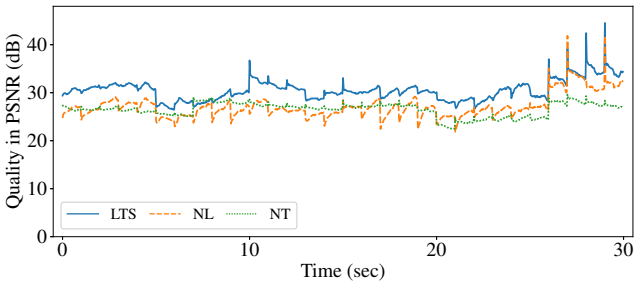


Figure 12: Sample visual quality from LongDress_1 under different ABR algorithms.

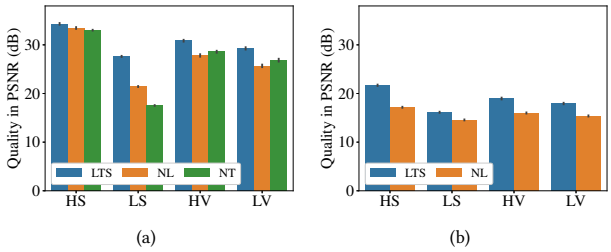


Figure 13: Visual quality with different network traces under different ABR algorithms: (a) simpler LongDress_1 and (b) more complex LongDress_5.

5.3 Results

Different representations. Fig. 11 shows the visual quality and overall streaming bitrate among different representations from LongDress_1. Compared with LapisGS, our proposed D-LapisGS can save 92.1% of bitrate with the highest layer and 70.8% with the lowest layer. Besides, the overall visual quality of our representation achieves 37.0 dB in PSNR and 0.991 in SSIM, which are fairly high. Compared with D3DGS, our proposed D-LapisGS representation consumes 54.60% less bitrate. This can be attributed to our improved training process: we consider not only the previous frames but also the lower layers, allowing us to get more compact 3DGS frames. In addition, our representation can provide different levels of detail

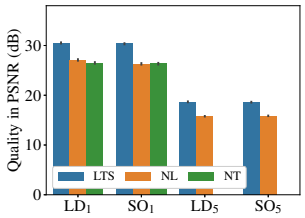


Figure 14: Visual quality from ABR algorithms in different scenes in PSNR.

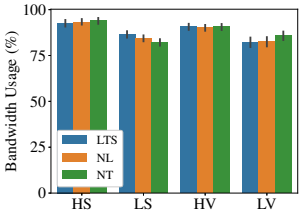


Figure 15: Bandwidth usage from individual ABR algorithms with different network traces.

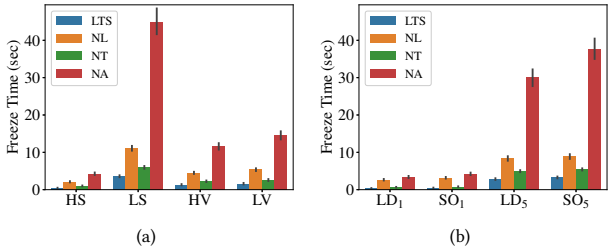


Figure 17: Average freeze time from ABR algorithms under: (a) different network traces and (b) different scenes.

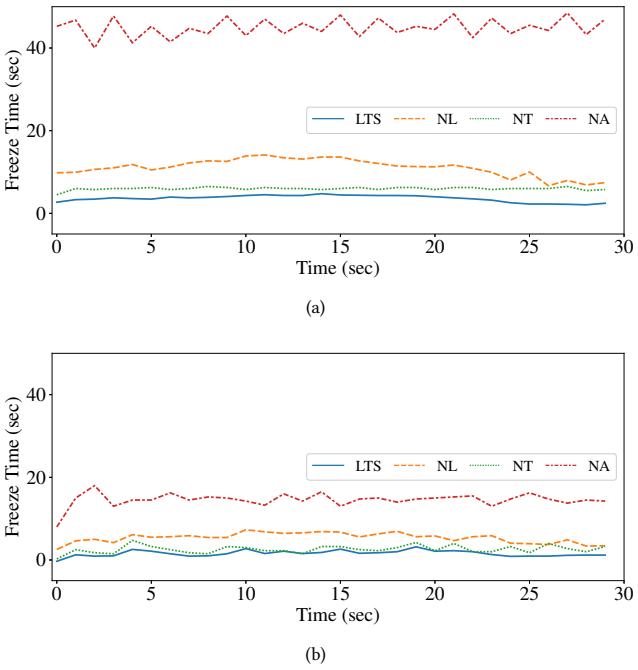


Figure 16: Freeze time under sample network traces: (a) HS and (b) LS.

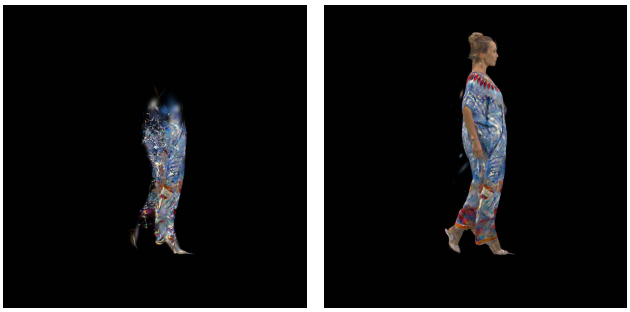


Figure 18: Sample rendered views by using: (a) NL and (b) LTS in on-demand streaming.

for more flexible 3DGS streaming to heterogeneous HMD clients over diverse networks.

Live streaming. According to our results, using NA for live streaming is not practical. Its bitrate can easily exceed the available bandwidth even under the best network conditions, i.e., none of the 3DGS frames can be rendered. Hence, we do not report the results from NA in live streaming. Besides, our results also show that NT does not work in more complex scenes, such as LongDress_5 and Soldier_5. In particular, even with the lowest layer alone, its bitrate often exceeds the available bandwidth, leading to too many missing frames. Hence, we do not report the missing frame results from NT either. In terms of missing frames from LTS and NL, we find that, on average, they only suffer from 1.88 missing frames from 900-frame traces, which is 99.7% less than NT. Adding to that, all missing frames with LTS happened in the LV trace. We note that in this trace, some extreme network conditions cause the bandwidth to drop to zero, creating an impossible mission for any ABR algorithms.

Fig. 12 shows the overall visual quality from LongDress_1 with different ABR algorithms. The results show that LTS achieves higher visual quality in PSNR and SSIM across all 3DGS frames. Besides, in Fig. 13, we show the visual quality of PSNR in a simple scene, LongDress_1, and a complex scene, LongDress_5, using different ABR algorithms under different network traces. The results reveal that LTS outperforms NL and NT by at most 6.20 dB and 10.08 dB in PSNR in the simple scene. It also outperforms NL by up to 4.52 dB in PSNR in the complex scene. Fig. 14 compares the overall visual quality among different scenes. We observe that LTS achieves 24.54 dB in PSNR and 0.851 in SSIM, beating: (i) NL by 4.06 dB in PSNR and 0.83 in SSIM, and (ii) NT by 3.418 dB in PSNR and 0.04 in SSIM. Fig. 15 presents the overall bandwidth usage in all network conditions. We find that in most network conditions, LTS utilizes the available bandwidth more efficiently, which can be attributed to its capability of intelligently selecting the tiles and layers.

On-demand streaming. Fig. 16 reports the per-frame average freeze time due to buffering. We observe that LTS always provides the shortest freeze time. We also give the average freeze time under different network conditions in Fig. 17(a). LTS has a 1.73-second freeze time on average, and a 3.60-second freeze time in the worst network condition. Besides, LTS reduces the freeze time by at most 70.56%, 57.57%, and 92.01% compared to NL, NT, and NA, respectively. While we could not give the figure on the average quality due

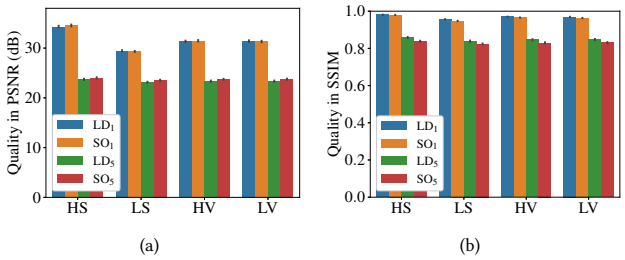


Figure 19: Average visual quality by using LTS in: (a) PSNR and (b) SSIM in on-demand streaming.

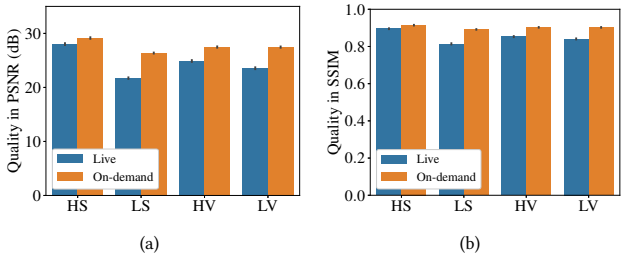


Figure 20: Average visual quality by using LTS for live and on-demand streaming under different network traces in: (a) PSNR and (b) SSIM.

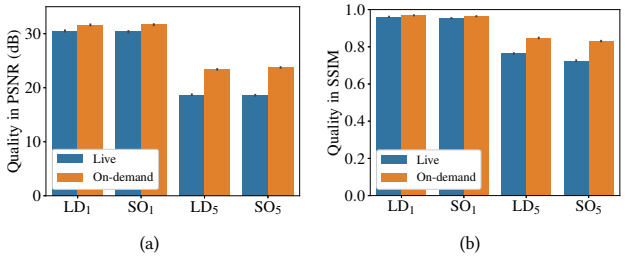


Figure 21: Average visual quality by using LTS for live and on-demand streaming under different 3DGS scenes in: (a) PSNR and (b) SSIM.

to space limitation, we note that LTS delivers an average quality of 27.61 dB in PSNR and 0.903 in SSIM, which are fairly high. Fig. 18 gives sample rendered views from LTS and NL. Fig. 18(a) reveals that with NL, only part of the 3DGS scene is streamed in high quality, which Fig. 18(b) shows that LTS is able to render a complete frame in the acceptable visual quality. The above results show that LTS provides a smooth dynamic 3DGS streaming experience for users with good visual quality, while other ABR algorithms result in long freeze time and inferior visual quality, which may drive the HMD users away from the services.

Next, we compare the average freeze time among different scenes in Fig. 17(b). We notice that, generally, all the ABR algorithms need longer freeze time to render more complex scenes. In addition, with tiling, LTS reduces the freeze time by up to 76.50%, and with layering, decreases the freeze time by up to 85.61%. Combining tiling and layering in our proposed LTS system cuts the freeze time by up to 91.31%.

Fig. 19 reports the average visual quality from LTS under different network conditions. We notice LTS performs better with simple scenes and under higher and more stable bandwidth (HS trace). Meanwhile, we also observe that LTS achieves shorter freeze time in those conditions (figures not shown due to space limitations). Take LongDress_5 as an example; streaming it causes 8X freeze time than streaming LongDress_1. We also observe insignificant visual quality differences in bandwidth conditions other than HS, although the freeze time do increase along with the decrease of average bandwidth.

Comparison between live and on-demand streaming. Figs. 20 and 21 compare the performance of LTS between live and on-demand streaming scenarios. The figures show that LTS achieves better visual quality in on-demand streaming, especially when the network bandwidth is low, or the scene is complex, with gaps of at most 5.14 dB in PSNR and 0.11 in SSIM. However, using on-demand streaming also leads to longer freeze time: at most, 246 seconds of freeze time is observed for 30-second traces.

6 Conclusion

In this paper, we introduce LTS, a DASH-based streaming system designed for dynamic 3DGS scenes. By integrating layering, tiling, and segmenting features, LTS achieves promising performance in both live and on-demand streaming. While streaming dynamic 3DGS scenes has never been done in the literature, by comparing to the current practice, our results show that LTS reduces missing frames during live streaming by up to 99.7%, while improving the PSNR by up to 4.06 dB and the SSIM by up to 0.83. In on-demand streaming, LTS reduces the freeze time by 92.01% while increasing the PSNR by 5.14 dB and the SSIM by 0.11. Adding to that, LTS effectively lowers bitrate requirements by up to 92.1% for the highest quality layer, demonstrating its superior efficiency in compressing dynamic 3DGS scenes. These evaluation results prove LTS's ability to stream adaptive, high-quality, and resource-efficient dynamic 3DGS scenes, making it a promising solution for immersive XR applications across diverse network conditions, dynamic systems, and heterogeneous HMD users.

This paper cannot cover all aspects related to streaming dynamic 3DGS scenes due to the space limit, and thus can be extended in several directions. For example, we primarily assess visual quality using objective metrics, while evaluating QoE of real users' trajectories through user studies is equally critical. Moreover, additional measurable QoE factors, such as motion-to-photon latency, which is the elapsed time between head movement and corresponding scene changes in the HMD, should be considered and analyzed for QoE modeling (especially, cybersickness) to further optimize LTS. Last, like other immersive streaming system LTS comes with quite a few system parameters related to tiles, layers, and segments. These parameters can be used to tradeoff system performance and resource consumption, and should be systematically and even dynamically adjusted.

References

[1] Edward H Adelson. 1995. Layered representations for vision and video. In *Proc. IEEE Workshop on Representation of Visual Scenes (In Conjunction with ICCV'95)*. 3–9.

[2] Ahmed Alnagrat, Rizalafande Che Ismail, Syed Zulkarnain Syed Idrus, and Rawad Mansour Abdulhafith Alfaqi. 2022. A review of extended reality (XR) technologies in the future of human education: Current trend and future opportunity. *Journal of Human Centered Technology* 1, 2 (August 2022), 81–96.

[3] Milena T. Bagdasarian, Paul Knoll, Yi-Hsin Li, Florian Barthel, Anna Hilsmann, Peter Eisert, and Wieland Morgenstern. 2024. 3DGS.Zip: A Survey on 3D Gaussian Splatting Compression Methods. *arXiv preprint arXiv:2407.09510* (June 2024).

[4] Lars Burgstahler and Martin Neubauer. 2002. New modifications of the exponential moving average algorithm for bandwidth estimation. In *Proc. of the 15th ITC Specialist Seminar*. C327.

[5] Yuanhao Cai, Yixun Liang, Jiahao Wang, Angtian Wang, Yulun Zhang, Xiaokang Yang, Zongwei Zhou, and Alan Yuille. 2025. Radiative Gaussian Splatting for Efficient X-Ray Novel View Synthesis. In *European Conference on Computer Vision*. 283–299.

[6] Guikun Chen and Wenguan Wang. 2024. A Survey on 3D Gaussian Splatting. *arXiv preprint arXiv:2401.03890* (January 2024).

[7] Eugene d'Eon, Bob Harrison, Taos Myers, and Philip A. Chou. 2017. 8i Voxelized Full Bodies, version 2 – A Voxelized Point Cloud Dataset. https://plenodb.jpeg.org/pc/8ilabs

[8] Epic Games, Inc. 2016. *RealityCapture in Ghost in the Shell*. https://www.capturingreality.com/RealityCapture-In-Ghost-In-The-Shell.

[9] FE Fadzli, MS Kamson, AW Ismail, and MYF Aladin. 2020. 3D telepresence for remote collaboration in extended reality (XR) application. In *IOP Conference Series: Materials Science and Engineering*. 012005.

[10] Zhiwen Fan, Kevin Wang, Kairun Wen, Zehao Zhu, Dejia Xu, and Zhangyang Wang. 2024. In *Advances in Neural Information Processing Systems*. 140138–140158.

[11] Jia-Wei Fang, Kuan-Yu Lee, Teemu Kämäräinen, Matti Siekkinen, and Cheng-Hsin Hsu. 2023. Will Dynamic Foveation Boost Cloud VR Gaming Experience?. In *Proc. of the 33rd Workshop on Network and Operating System Support for Digital Audio and Video*. 29–35.

[12] Thomas Forgione, Axel Carlier, Géraldine Morin, Wei Tsang Ooi, Vincent Charvillat, and Praveen Kumar Yadav. 2018. DASH for 3D Networked Virtual Environment. In *Proc. of the 26th ACM International Conference on Multimedia*. 1910–1918.

[13] Yu Guan, Chengyuan Zheng, Xinggong Zhang, Zongming Guo, and Junchen Jiang. 2019. Pano: Optimizing 360° Video Streaming with a Better Understanding of Quality Perception. In *Proc. of the ACM Special Interest Group on Data Communication*. 394–407.

[14] André FR Guarda, Nuno MM Rodrigues, and Fernando Pereira. 2020. Deep learning-based point cloud geometry coding with resolution scalability. In *2020 IEEE 22nd International Workshop on Multimedia Signal Processing (MMSP)*. 1–6.

[15] Alain Horé and Djemel Ziou. 2010. Image Quality Metrics: PSNR vs. SSIM. In *Proc. of IEEE International Conference on Pattern Recognition*. 2366–2369.

[16] Mohammad Hosseini and Viswanathan Swaminathan. 2016. Adaptive 360 VR video streaming based on MPEG-DASH SRD. In *2016 IEEE International Symposium on Multimedia (ISM)*. 407–408.

[17] Chiara Innocente, Luca Ulrich, Sandro Moos, and Enrico Vezzetti. 2023. A framework study on the use of immersive XR technologies in the cultural heritage domain. *Journal of Cultural Heritage* 62 (July-August 2023), 268–283.

[18] Jack Jansen, Shishir Subramanyam, Romain Bouqueau, Gianluca Cernigliaro, Marc Martos Cabré, Fernando Pérez, and Pablo Cesar. 2020. A pipeline for multiparty volumetric video conferencing: transmission of point clouds over low latency DASH. In *Proc. of the 11th ACM Multimedia Systems Conference*. 341–344.

[19] Bernhard Kerbl, Georgios Kopanas, Thomas Leimkühler, and George Drettakis. 2023. 3D Gaussian Splatting for Real-Time Radiance Field Rendering. *ACM Trans. Graph.* 42, 4 (November 2023), 139–1.

[20] Joo Chan Lee, Daniel Rho, Xiangyu Sun, Jong Hwan Ko, and Eunbyung Park. 2024. Compact 3D Gaussian Representation for Radiance Field. In *Proc. of the IEEE/CVF Conference on Computer Vision and Pattern Recognition*. 21719–21728.

[21] Bangya Liu and Suman Banerjee. 2024. SwinGS: Sliding Window Gaussian Splatting for Volumetric Video Streaming with Arbitrary Length. *arXiv preprint arXiv:2409.07759* (September 2024).

[22] Zhi Liu, Qiyue Li, Xianfu Chen, Celimuge Wu, Susumu Ishihara, Jie Li, and Yusheng Ji. 2021. Point cloud video streaming: Challenges and solutions. *IEEE Network* 35, 5 (September 2021), 202–209.

[23] Tao Lu, Mulin Yu, Linning Xu, Yuanbo Xiangli, Limin Wang, Dahua Lin, and Bo Dai. 2024. Scaffold-GS: Structured 3D Gaussians for View-Adaptive Rendering. In *Proc. of the IEEE/CVF Conference on Computer Vision and Pattern Recognition*. 20654–20664.

[24] Jonathon Luiten, Georgios Kopanas, Bastian Leibe, and Deva Ramanan. 2024. Dynamic 3D Gaussians: Tracking by Persistent Dynamic View Synthesis. In *International Conference on 3D Vision (3DV)*. 800–809.

[25] Ben Mildenhall, Pratul P Srinivasan, Matthew Tancik, Jonathan T Barron, Ravi Ramamoorthi, and Ren Ng. 2021. Nerf: Representing scenes as neural radiance fields for view synthesis. *Commun. ACM* 65, 1 (December 2021), 99–106.

[26] Tadatsugu Morimoto, Takaomi Kobayashi, Hirohito Hirata, Koji Otani, Maki Sugimoto, Masatsugu Tsukamoto, Tomohito Yoshihara, Masaya Ueno, and Masaaki Mawatari. 2022. XR (extended reality: virtual reality, augmented reality, mixed reality) technology in spine medicine: status quo and quo vadis. *Journal of Clinical Medicine* 11, 2 (January 2022), 470.

[27] Arvind Narayanan, Eman Ramadan, Rishabh Mehta, Xinyue Hu, Qingxu Liu, Rostand A. K. Fezeu, Udhaya Kumar Dayalan, Saurabh Verma, Peiqi Ji, Tao Li, Feng Qian, and Zhi-Li Zhang. 2020. Lumos5G: Mapping and Predicting Commercial mmWave 5G Throughput. In *Proc. of the ACM Internet Measurement Conference*. 176–193.

[28] KL Navaneet, Kossar Pourahmadi Meibodi, Soroush Abbasi Koohpayegani, and Hamed Pirsiavash. 2023. Compact3D: Compressing gaussian splat radiance field models with vector quantization. *arXiv preprint arXiv:2311.18159* (2023).

[29] Simon Niedermayr, Josef Stumpfegger, and Rüdiger Westermann. 2024. Compressed 3D Gaussian Splatting for Accelerated Novel View Synthesis. In *Proc. of the IEEE/CVF Conference on Computer Vision and Pattern Recognition*. 10349–10358.

[30] Jounsup Park, Philip A Chou, and Jenq-Neng Hwang. 2019. Rate-utility optimized streaming of volumetric media for augmented reality. *IEEE Journal on Emerging and Selected Topics in Circuits and Systems* 9, 1 (February 2019), 149–162.

[31] Dimitri Podborski, Emmanuel Thomas, Miska M Hannuksela, Sejin Oh, Thomas Stockhammer, and Stefan Pham. 2018. 360-degree video streaming with MPEG-DASH. *SMPTE Motion Imaging Journal* 127, 7 (2018), 20–27.

[32] Feng Qian, Bo Han, Qingyang Xiao, and Vijay Gopalakrishnan. 2018. Flare: Practical Viewport-Adaptive 360-Degree Video Streaming for Mobile Devices. In *Proc. of the 24th Annual International Conference on Mobile Computing and Networking*. 99–114.

[33] Ngo Quang Minh Khiem, Guntur Ravindra, Axel Carlier, and Wei Tsang Ooi. 2010. Supporting zoomable video streams with dynamic region-of-interest cropping. In *Proc. of the First Annual ACM SIGMM Conference on Multimedia Systems*. 259–270.

[34] Kerui Ren, Lihan Jiang, Tao Lu, Mulin Yu, Linning Xu, Zhangkai Ni, and Bo Dai. 2024. Octree-GS: Towards Consistent Real-time Rendering with LOD-Structured 3D Gaussians. *arXiv preprint arXiv:2403.17898* (Octorber 2024).

[35] Bruno Rodríguez-García, Henar Guillen-Sanz, David Checa, and Andres Bustillo. 2024. A systematic review of virtual 3D reconstructions of Cultural Heritage in immersive Virtual Reality. *Multimedia Tools and Applications* (April 2024), 1–51.

[36] Yago Sanchez de la Fuente, Robert Skupin, and Thomas Schierl. 2015. Compressed Domain Video Processing for Tile Based Panoramic Streaming Using SHVC. In *Proc. of the 3rd International Workshop on Immersive Media Experiences*. 13–18.

[37] Michael Seufert, Sebastian Egger, Martin Slanina, Thomas Zinner, Tobias Hoßfeld, and Phuoc Tran-Gia. 2014. A survey on quality of experience of HTTP adaptive streaming. *IEEE Communications Surveys & Tutorials* 17, 1 (September 2014), 469–492.

[38] Yuang Shi, Bennett Clement, and Wei Tsang Ooi. 2024. QV4: QoE-based Viewpoint-Aware V-PCC-encoded Volumetric Video Streaming. In *Proc. of the 15th ACM Multimedia Systems Conference*. 144–154.

[39] Yuang Shi, Simone Gasparini, Géraldine Morin, and Wei Tsang Ooi. 2024. LapisGS: Layered Progressive 3D Gaussian Splatting for Adaptive Streaming. *arXiv preprint arXiv:2408.14823* (August 2024).

[40] Yuang Shi, Simone Gasparini, Géraldine Morin, Chenggang Yang, and Wei Tsang Ooi. 2025. Sketch and Patch: Efficient 3D Gaussian Representation for Man-Made Scenes. *arXiv preprint arXiv:2501.13045* (January 2025).

[41] Yuang Shi and Wei Tsang Ooi. 2024. Perceptual Impact of Facial Quality in MPEG V-PCC-encoded Volumetric Videos. In *Proc. of the 16th International Workshop on Immersive Mixed and Virtual Environment Systems*. 71–74.

[42] Yuang Shi, Ruoyu Zhao, Simone Gasparini, Géraldine Morin, and Wei Tsang Ooi. 2024. Volumetric Video Compression Through Neural-based Representation. In *Proc. of the 16th International Workshop on Immersive Mixed and Virtual Environment Systems*. 85–91.

[43] Ashutosh Singla, Steve Göring, Dominik Keller, Rakesh Rao Ramachandra Rao, Stephan Fremerey, and Alexander Raake. 2021. Assessment of the Simulator Sickness Questionnaire for Omnidirectional Videos. In *2021 IEEE Virtual Reality and 3D User Interfaces (VR)*. 198–206.

[44] Thomas Stockhammer. 2011. Dynamic adaptive streaming over HTTP –: standards and design principles. In *Proc. of the Second Annual ACM conference on Multimedia Systems*. 133–144.

[45] Tianyu Su, Abbas Javadtalab, Abdulsalam Yassine, and Shervin Shirmohammadi. 2014. A DASH-based 3D multi-view video rate control system. In *2014 8th International Conference on Signal Processing and Communication Systems (ICSPCS)*. 1–6.

[46] Tianyu Su, Ashkan Sobhani, Abdulsalam Yassine, Shervin Shirmohammadi, and Abbas Javadtalab. 2016. A DASH-based HEVC multi-view video streaming system. *Journal of Real-Time Image Processing* 12, 2 (May 2016), 329–342.

[47] Shishir Subramanyam, Irene Viola, Alan Hanjalic, and Pablo Cesar. 2020. User centered adaptive streaming of dynamic point clouds with low complexity tiling. In *Proc. of the 28th ACM international conference on multimedia*. 3669–3677.

[48] Jiakai Sun, Han Jiao, Guangyuan Li, Zhanjie Zhang, Lei Zhao, and Wei Xing. 2024. 3DGStream: On-the-Fly Training of 3D Gaussians for Efficient Streaming of Photo-Realistic Free-Viewpoint Videos. In *Proc. of the IEEE/CVF Conference on Computer Vision and Pattern Recognition*. 20675–20685.

[49] Yuan-Chun Sun, Yuang Shi, Wei Tsang Ooi, Chun-Ying Huang, and Cheng-Hsin Hsu. 2024. Multi-frame Bitrate Allocation of Dynamic 3D Gaussian Splatting Streaming Over Dynamic Networks. In *Proc. of the 2024 SIGCOMM Workshop on Emerging Multimedia Systems*. 1–7.

[50] Richard Szeliski. 2022. 3D to 2D projections. In *Computer Vision: Algorithms and Applications*. Springer Nature, Chapter 2.1.4, 41–51.

[51] Dion Tanjung, Jong-Deok Kim, Dong-Hyun Kim, Jewon Lee, Soonchoul Kim, and Joon-Young Jung. 2023. Qoe optimization in dash-based multiview video streaming. *IEEE Access* 11 (August 2023), 83603–83614.

[52] Jianqiang Wang, Dandan Ding, Zhu Li, and Zhan Ma. 2021. Multiscale point cloud geometry compression. In *2021 Data Compression Conference (DCC)*. 73–82.

[53] Lisha Wang, Chenglin Li, Wenrui Dai, Junni Zou, and Hongkai Xiong. 2021. QoE-driven and tile-based adaptive streaming for point clouds. In *ICASSP 2021-2021 IEEE International Conference on Acoustics, Speech and Signal Processing (ICASSP)*. 1930–1934.

[54] Penghao Wang, Zhirui Zhang, Liao Wang, Kaixin Yao, Siyuan Xie, Jingyi Yu, Minye Wu, and Lan Xu. 2024. V^ 3: Viewing Volumetric Videos on Mobiles via Streamable 2D Dynamic Gaussians. *ACM Transactions on Graphics (TOG)* 43, 6 (November 2024), 1–13.

[55] Mengbai Xiao, Chao Zhou, Yao Liu, and Songqing Chen. 2017. OpTile: Toward Optimal Tiling in 360-degree Video Streaming. In *Proc. of the 25th ACM International Conference on Multimedia*. 708–716.

[56] Liang Xie, Wei Gao, Huiming Zheng, and Ge Li. 2024. SPCGC: Scalable Point Cloud Geometry Compression for Machine Vision. In *2024 IEEE International Conference on Robotics and Automation (ICRA)*. 17272–17278.

[57] Shou-Cheng Yen, Ching-Ling Fan, and Cheng-Hsin Hsu. 2019. Streaming 360° videos to head-mounted virtual reality using DASH over QUIC transport protocol. In *Proc. of the 24th ACM Workshop on Packet Video*. 7–12.

[58] Daheng Yin, Jianxin Shi, Miao Zhang, Zhaowu Huang, Jiangchuan Liu, and Fang Dong. 2024. FSVFG: Towards Immersive Full-Scene Volumetric Video Streaming with Adaptive Feature Grid. In *Proc. of the 32nd ACM International Conference on Multimedia*. 11089–11098.

[59] Dooyeol Yun and Kwangsue Chung. 2017. DASH-based multi-view video streaming system. *IEEE Transactions on Circuits and Systems for Video Technology* 28, 8 (April 2017), 1974–1980.

[60] Markos Zampoglou, Kostas Kapetanakis, Andreas Stamoulias, Athanasios G Malamos, and Spyros Panagiotakis. 2018. Adaptive streaming of complex Web 3D scenes based on the MPEG-DASH standard. *Multimedia Tools and Applications* 77, 1 (December 2018), 125–148.

[61] Emin Zerman, Radhika Kulkarni, and Aljosa Smolic. 2021. User behaviour analysis of volumetric video in augmented reality. In *2021 13th International Conference on Quality of Multimedia Experience (QoMEX)*. 129–132.

[62] Ali Zeynali, Mohammad H. Hajiesmaili, and Ramesh K. Sitaraman. 2024. BOLA360: Near-optimal View and Bitrate Adaptation for 360-Degree Video Streaming. In *Proc. of the 15th ACM Multimedia Systems Conference*. 12–22.

[63] Qian-Yi Zhou, Jaesik Park, and Vladlen Koltun. 2018. Open3D: A Modern Library for 3D Data Processing. *arXiv preprint arXiv:1801.09847* (January 2018).

[64] Xin Zhou, Zhepei Wang, Hongkai Ye, Chao Xu, and Fei Gao. 2020. Ego-planner: An esdf-free gradient-based local planner for quadrotors. *IEEE Robotics and Automation Letters* 6, 2 (December 2020), 478–485.

SAILS: A Synchronous Accessible Immersive Online Learning System for Young Learners

Yuran Sun, Zhuoying Zhang, Zhenxiao Luo, Alan William Dougherty, Man Ho Yip, Yi King Choi, Chuan Wu
The University of Hong Kong
yuransun@connect.hku.hk

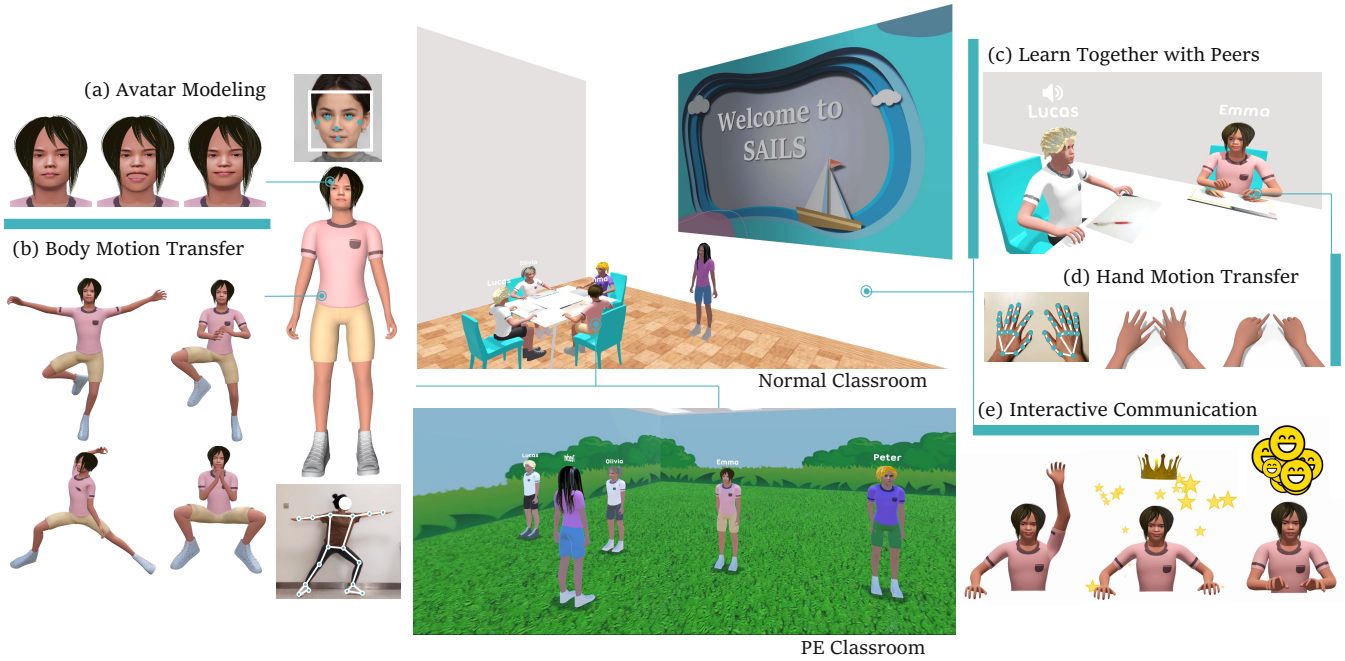


Figure 1: Core features of SAILS. The system supports two class types: a normal classroom and a PE classroom. Avatar modeling, based on a single input image, retains the user's facial features and serves as the fundamental representation of users within the application, as shown in (a). In the normal classroom, teachers can instruct using a virtual whiteboard while students sit around a table and learn together, observing each other's expressions, hand movements, and classwork. Various interaction methods are provided, such as chatting, raising hands, sending emojis, and offering rewards, as shown in (c), (d), and (e). In the PE classroom, students and teachers exercise together within a shared environment. Real-time body motion transfer allows everyone to view each other's movements, as shown in (b).

Abstract

Online distance learning emerged as a prominent means of education during the pandemic and is expected to continue as a long-lasting trend, enabling access to top education resources without the constraints of physical distance. However, conferencing software predominantly used for synchronous online teaching and learning is not suitable or sufficient for young students and teachers. The learning experiences of these students heavily rely on hands-on activities and interactions with peers and teachers that are challenging to replicate. Additionally, teachers face difficulties in effectively monitoring student progress. To address these challenges, we propose a Synchronous Accessible Immersive Online Learning System (SAILS). It offers an immersive and interactive platform for young learners by integrating real-life school activities with a virtual learning environment. Using the system, teachers can easily organize classes, assess students' work. We evaluated the effectiveness of SAILS through a user study involving 40 young learners from a kindergarten and several primary schools, along with their teachers. The results demonstrated a strong preference for our system among the participants, highlighting its ability to provide a more immersive and engaging online learning experience.

Permission to make digital or hard copies of all or part of this work for personal or classroom use is granted without fee provided that copies are not made or distributed for profit or commercial advantage and that copies bear this notice and the full citation on the first page. Copyrights for components of this work owned by others than the author(s) must be honored. Abstracting with credit is permitted. To copy otherwise, or republish, to post on servers or to redistribute to lists, requires prior specific permission and/or a fee. Request permissions from permissions@acm.org.
MMSys '25, March 31–April 4, 2025, Stellenbosch, South Africa
© 2025 Copyright held by the owner/author(s). Publication rights licensed to ACM.
ACM ISBN 979-8-4007-1467-2/25/03
https://doi.org/10.1145/3712676.3714446

CCS Concepts

• **Human-centered computing** → **User studies**; **Interactive systems and tools**; • **Applied computing** → **Interactive learning environments**; **Distance learning**.

Keywords

Distance education, Classroom, Immersive virtual environments, Education, Learning effect, Full-body avatar, Character animation

ACM Reference Format:

Yuran Sun, Zhuoying Zhang, Zhenxiao Luo, Alan William Dougherty, Man Ho Yip, Yi King Choi, Chuan Wu. 2025. SAILS: A Synchronous Accessible Immersive Online Learning System for Young Learners. In *The 16th ACM Multimedia Systems Conference (MMSys '25), March 31–April 4, 2025, Stellenbosch, South Africa.* ACM, New York, NY, USA, 11 pages. https://doi.org/10.1145/3712676.3714446

1 Introduction

During the pandemic, the way schools teach their students largely switched to online learning. Kindergarten or primary school children were provided with forms of online learning activities, such as pre-recorded videos or online classes. Many of their parents expressed concerns about learning efficiency and found their children lacked concentration and learning interests [39]. Even after the pandemic, synchronous distance education remains important, as it allows learners to participate in real-time classes from anywhere with an Internet connection, overcoming geographical barriers [13].

Existing solutions typically utilize conferencing tools such as Zoom [15] to facilitate synchronous online learning. However, these tools have limitations in engaging students and promoting interactions [7], especially for young learners. For instance, they do not effectively support hands-on and collaborative activities like drawing or physical education exercises [16]. In such scenarios, teachers are forced to check small windows one by one, making it difficult to monitor student participation intuitively and conveniently. While VR-based solutions offer better engagement, they require bulky headsets, which are not suitable for young children [21]. Therefore, it is essential to design a synchronous online learning system that is immersive, accessible, and specifically tailored to the needs of young students and their teachers.

We propose SAILS, a Synchronous Accessible Immersive Online Learning System. Our goal is to enable an interactive and immersive learning environment for young learners, fostering a sense of "togetherness" during classes while prioritizing practicality and affordability. The fundamental concept of our system is to replicate the physical classroom in a 3D virtual world and leverage both physical and virtual spaces to enhance the educational experience for young children in an online setting. The system allows students and teachers to join a virtual classroom using their home device (i.e., personal computers and webcams), where they are present alongside their classmates and teachers. By providing a structured environment that mimics traditional classroom settings, our system helps students concentrate more on their study tasks without frequent interruptions from teachers checking their work., while also facilitating the teacher's ability to conveniently monitor students' learning progress.

Our innovative approach seamlessly integrates physical school activities with virtual learning experiences. When students physically works on their learning activities at their desk, their work is automatically shared with others in the virtual classroom. Meanwhile, the teacher can present teaching materials on a virtual whiteboard and effortlessly monitor each student's work by "walking" around, all in the virtual space. Additionally, our system includes a virtual physical education (PE) classroom, allowing students to exercise together in the same virtual environment with their teacher leading the session from the front, rather than being confined to separate small windows as in traditional video conferencing systems.

The main contributions of our system are summarized as follows:

▹ We design a Synchronous Accessible Immersive Online Learning System, named SAILS, for young learners. It runs on a single home computer equipped with two RGB cameras. The system mimics traditional classroom settings, allowing multiple students to sit around tables and share the individual work that they are working on the physical table, creating a virtual "learning-together" experience in 3D space. We also implement a PE classroom, where students can exercise together virtually.

▹ We have developed realistic 3D head modeling and implemented real-time facial expression and body/hand movement transfer in SAILS. Users are represented by their virtual avatars within the system. These avatars can reflect users' appearance and be dynamically animated according to users' actual movements. Our system also integrates various multimedia components, such as voice chat, live streaming videos, and clicking-based interactivity features to enhance the virtual teaching and learning experience. By involving young learners and teachers in the iterative design process, we ensure that the system is tailored to their needs and preferences, enhancing usability and user satisfaction.

▹ We conducted a user study involving 40 kindergarten and primary school students aged from 5 to 12, along with their teachers, to evaluate the effectiveness of SAILS. More than 90% of the participants reported that they enjoyed the experience and were excited to use our system. The results indicated that SAILS was highly regarded by participants, providing a more immersive and enjoyable learning experience than video conferencing solutions. Participants also reported increased in-class interest and engagement, as well as a sense of "togetherness" while using SAILS.

2 Related Work

Synchronous distance education applications have gained significant interest in recent years, offering a virtual platform for real-time interaction between students and educators. However, many existing applications are not specifically designed for young children.

2.1 Online Conferencing Applications (2D)

Virtual conferencing applications, such as Zoom [15], Microsoft Teams [22], and Google Meet [14], have been widely adopted for synchronous distance education [8]. These tools allow participants to communicate in real-time through video and audio, share screens, and collaborate on virtual whiteboards. When having classes, students can ask questions through chat or virtual hand-raising. To facilitate group activities, breakout rooms are available where students can collaborate in smaller groups.

However, these applications, primarily designed for group meetings, may not provide the level of engagement and interaction necessary for effective online learning, especially for young students. Teaching with such tools typically centers around the teacher delivering lectures and sharing their screen, which often results in passive listening by students and a lack of a cohesive classroom experience. Students may choose not to turn on their webcams, further diminishing interaction and making it challenging for teachers to monitor participation.

Furthermore, assessing hands-on activities, such as drawing, becomes problematic. Teachers must rely on students holding their work up to the camera, which is time-consuming and often results in unclear visuals. This process can disrupt students' workflow and hinder effective assessment.

While virtual conferencing tools offer convenient and accessible solutions for online classes, they have limitations in creating immersive and engaging learning experiences and in allowing easy monitoring by teachers.

2.2 Immersive Applications (3D)

3D applications create a sense of depth and realism, making them more engaging than 2D applications. They can be used on virtual reality devices or computers.

2.2.1 VR-based. Virtual Reality (VR) has emerged as a technology that enables immersive and interactive experiences. It can serve as an advanced platform for synchronous online learning, offering a 3D environment where students and teachers can interact in real time, similar to a physical classroom [10, 25, 36, 38]. Some applications utilize VR headsets to enhance the immersive experience and offer educational features such as online classes [36, 38], lab simulations [25], and virtual field trips [25].

However, VR also comes with challenges such as the need for specialized hardware and potential discomfort or nausea for some users. Prolonged use of VR headsets, especially in applications that require significant attention and movement, can cause nausea or dizziness [37]. Moreover, manufacturers typically recommend that VR headsets be used by older children, generally 13 years and up, and under adult supervision to minimize risks [21]. They also emphasize that children under 10 cannot use VR headsets.

2.2.2 Computer-based. Computer-based applications allow users to navigate in a 3D virtual environment [12, 17, 23, 35]. Users can create rooms and invite others to join, enabling interaction in a shared virtual space. For online teaching, a teacher can create a virtual classroom that resembles a real one, sharing their screen and camera feed. This setup provides an intuitive overview of the class structure, allowing the teacher to see all virtual avatars in the classroom.

However, such applications are not specifically designed for a class of younger children. They offer a high degree of freedom, requiring each individual to control their own avatar. This can be distracting to the students. The teacher would have limited control over the class without the cooperation of the students.

Furthermore, these applications often use pixel-based [17] or cartoon-like [35] avatar representations, with some even lacking the lower body [12, 23]. Such designs lack crucial non-verbal information when users make facial expressions or perform movements, leading to less immersive and intuitive interactions. The absence of realistic avatar modeling, including detailed face modeling and accurate hand and body movement transfer, along with the inability to share real-time hands-on activities, makes these applications less suitable for young learners.

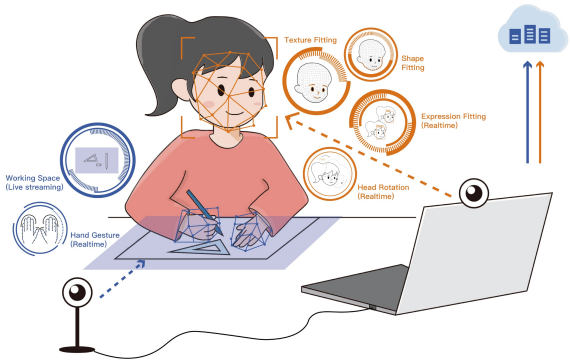


Figure 2: Device setup and client side illustration. Users need to equip their personal computers with two cameras: one front-facing camera to capture the user's face and another camera to capture the working area on the user's desk. This setup enables real-time facial expression, hand gesture, hands-on activities transfer to the virtual environment. All real-time data is processed by the client application and sent to the cloud server for distribution.

3 SAILS

3.1 System Overview

Our synchronous accessible immersive online learning system, SAILS, targets young learners and their educators, aiming to provide a sense of "togetherness" and lively engagement in classroom activities. The system is easy to set up on home devices and intuitive to use. We have designed two types of virtual classrooms in the system to cover most class types: a normal classroom and a PE classroom.

In the normal classroom, students are seated together around tables and have classes with their classmates. They can perform study tasks on their physical desks while sharing their work automatically in the virtual class. Their facial expressions and hand gestures are transferred in real-time to the virtual environment as well, allowing teachers to easily monitor students' learning progress and attentiveness. Teachers can present teaching materials or demonstrate hands-on activities on a virtual whiteboard.

In a PE classroom, the teacher demonstrates some physical exercises while students follow along in front of their respective cameras; the body movements of students and the teacher are transferred to their virtual avatars in the virtual classroom, creating an "exercising-together" scene in the system, as shown in Figure 1.

To use SAILS, users need to equip their personal computers with two cameras, as shown in Figure 2. The computer should have basic

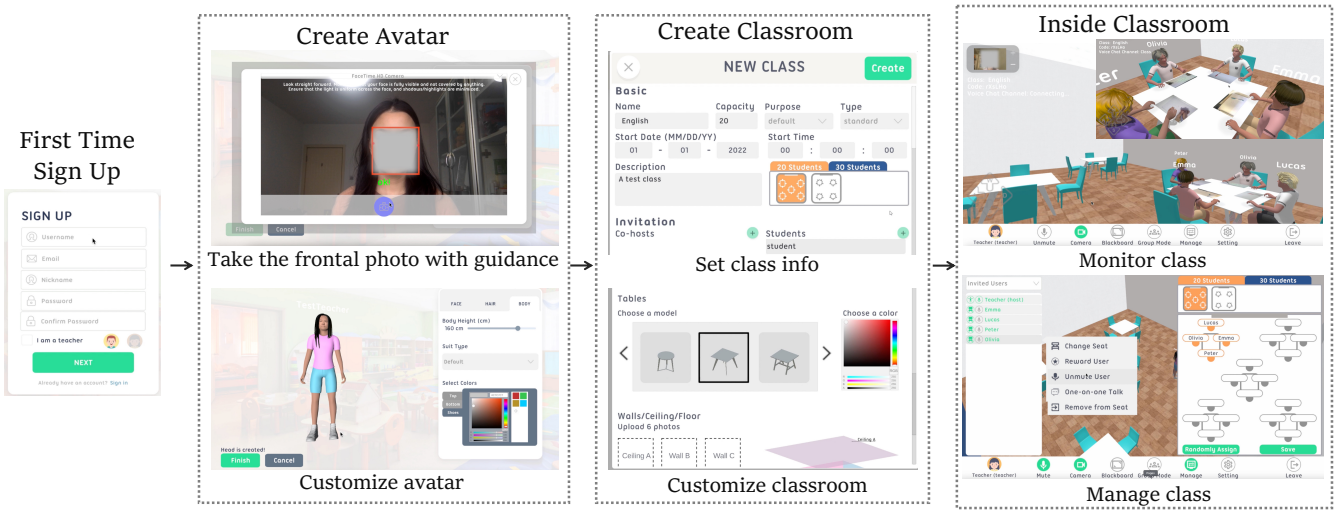


Figure 3: Application flow for a first-time teacher user in SAILS. Upon sign-up, users create and customize their avatars with guided instructions. Teachers can set up and personalize classrooms, and invite students to join. In the classroom, teachers can instruct, monitor, and interact with students.

graphic rendering capabilities. One camera should be front-facing to capture the user's face when they are sitting in front of the desk. This camera can be either the built-in camera or a webcam attached. The other camera should capture the working area on the user's desk. It can be an additional webcam connected to the computer. For the PE class, only one camera is required.

The application workflow for first-time users is illustrated in Figure 3. We use a teacher account as an example, as it includes more management functions. Upon sign-up, users follow the guidance to create and customize their avatars. Teachers can set up new classrooms by filling in basic class information and customizing the classroom to resemble a physical one. Students invited by teachers can then access the classroom. In the classroom, teachers can conduct lessons using a virtual whiteboard, manage the class in the management portal, and walk around to monitor and interact with students. Students can see their peers working together around a table, chat with them, raise hands to ask questions, send emojis to express their feelings, and receive rewards from teachers.

In the following subsections, we present the core components of SAILS: immersive modeling, interactive communication, and the "togetherness" design.

3.2 Immersive Modeling

Students would have a higher degree of classroom engagement if they were allowed a sense of presence with their teachers and classmates [1], possibly in a familiar classroom environment, and a visual realism with virtual avatars representing their teachers and peers. In addition, non-verbal cues such as body movements and facial expressions are vital for humans to interact and communicate with each other [2]. In a traditional classroom, teachers heavily rely on these cues for comprehending students' responses, especially when managing multiple students simultaneously.

To create an immersive virtual learning environment, we utilize 3D modeling to replicate the real classroom in a virtual space. We aim to ensure that the users' virtual representations (i.e., avatars), particularly their heads, are rendered as realistically as possible to promote engagement. Furthermore, we incorporate behavioral hints with real-time facial expressions and movement transfer techniques to enable a more genuine classroom learning experience.

3.2.1 Classroom Modeling. The SAILS platform provides a virtual 3D classroom environment that can be personalized to meet diverse teaching needs and create a sense of familiarity to students. To achieve this, our system offer a virtual reconstruction feature that allows teachers to recreate the physical classroom. This process involves collecting images of the walls, floor, and ceiling of the actual classroom, and applying them as textures in the virtual classroom. Teachers can also choose from various 3D table and chair model and customize colors based on their preferences. Moreover, they can select from different table/chair layouts, each with varying group division settings.

3.2.2 Virtual Avatar Modeling. Our technology allows users to create realistic and personalized avatars through advanced head modeling, hairstyle classification, and appearance customization.

Head modeling is a core component of the virtual avatar, as it enables users to recognize each other within the virtual environment. We adopt the FLAME [18, 41] morphable model structure, which generates a complete head model in one step from a single frontal face image. The process involves performing landmark detection to identify positions of facial features, then extracting the skin texture and constructing a 3D mesh that approximates the face in the 2D image. For facial texture generation, we segment the face area to create a facial mask and apply post-processing to obtain a natural skin color representation.

As the generated head model does not include hair, we implement a hairstyle classification function utilizing a Vision Transformer (ViT) [9] trained on a custom hairstyle dataset. It automatically suggests the closest matching hairstyle, represented by a realistic 3D hair model, based on the input image. This process reuses the single headshot photograph captured during the head modeling phase. Young learners can choose a hairstyle from our selection based on their preferences for a more enjoyable and diverse range of avatar appearances.

Beyond hairstyle, a user can adjust their hair color, set the height of their avatar body, select clothing styles, and customize clothes colors accordingly. This level of avatar customization not only enhances interaction within the system but also suits individual user preferences, establishing their individuality.

3.2.3 Facial Expression Transfer. The transfer of user expressions and movements to the virtual classroom heavily relies on camera capturing and image processing techniques. To detect facial, hand, and body landmarks and orientations from RGB video frames, we have chosen MediaPipe [20] and its provided models. The face landmark model, BlazeFace [4], provides real-time facial information that can be transferred to the virtual environment. To facilitate the representation of user expressions, we have developed new facial blend shapes that encompass a range of expressions, such as mouth opening, smiling, and blinking. By utilizing facial landmark data and calculating the vector difference between these blend shapes and the base model, we are able to dynamically display users' various expressions in real-time, enabling a more immersive virtual classroom experience.

3.2.4 Body Movement Transfer. Reproducing the user's body movements on the virtual avatar is an essential feature of our system, aimed at supporting students and teachers in PE courses. We utilize body pose landmarks obtained through Mediapipe's BlazePose [3] from the user's captured videos, which provides position information for each joint in image coordinates. We apply basic operations such as smoothing and normalization to the body joint landmark data as initial preparation for animation.

To animate the virtual avatar, we utilize Unity's Animation Rigging package [31], which uses Multi-Aim Constraints [32]. These constraints enable procedural motion by rotating the parent joint to face the target position specified by the child joint, at each parent-child joint level. However, the absence of ground-truth bone length measurements for users introduces variability in the scaling relationship between the user and their virtual avatar, particularly when the user is moving towards or away from the camera. Directly using the detected joint landmark position as the aim target can result in incorrect bone rotations. To address this issue, we introduce a vectorization process that generates a direction vector from the parent joint's landmark position to the child joint's landmark position. We set the aim target of the parent joint to its current position plus the direction vector. This process effectively maps rotation information from the detection results to the virtual avatar using position data only, without prior knowledge of bone length.

In conjunction with the vectorization process for joint mapping, we have integrated the Unity physics engine [33] to resolve differences between landmark position coordinates and the Unity game scene coordinates. Specifically, we apply gravitational forces to the avatars within the scene, utilizing the unity rigid body and colliders to simulate physical interactions between the avatar and the ground surface. We also incorporate velocity-based foot state detection to determine the vertical position of the virtual avatar. This integration ensures a more realistic representation of the virtual avatar's movements.

3.2.5 Hand Movement Transfer. In classes where students engage in hands-on activities, capturing and reconstructing the movement of their hands becomes crucial as it allows teachers to easily monitor students' activities and provide assistance if necessary.

The Mediapipe Hands model [40] is used to produce hand landmarks and predict the relative position between the hand and the camera. Considering that each person's palm size and finger length differ, and the virtual character's hands are predefined during the 3D modeling process, we implement a vectorization process, similar to the method used on pose detection data to ensure accurate alignment between the captured hand data and the virtual character's hands. This involves generating a direction vector based on the hand joint data and establishing an aiming target using Animation Rigging's Multi-Aim Constraint [32]. This guarantees precise local movements of the virtual character's hands.

In the real world, the joints of the fingers are limited by certain angles, typically ranging from 0 to 90 degrees. We incorporate this physical constraint into the virtual hands by calculating the angles between each joint vector derived from the hand landmarks. By imposing restrictions on the angles of the hand joints, we prevent excessive bending of the finger joints and ensure that the virtual hand movements are more natural and realistic.

Furthermore, we apply constraints on the virtual hand positions. The virtual hands closely follow the movements of real hands in the captured video shown on the virtual table. We also smooth the data to eliminate jittering.

3.3 Interactive Communication

We implement various communication means using images, videos, audio, and others for excellent interactivity in SAILS.

3.3.1 Virtual Whiteboard. Teachers can upload slides, deliver presentations, and perform live demonstrations on the virtual whiteboard from their local devices. The virtual whiteboard is displayed at the front, mirroring the physical classroom setup.

3.3.2 Desk Video. Each student's physical desk workspace is projected onto the virtual desk. Teachers can always have an unobstructed view of the students' work areas, regardless of the camera angle. Utilizing OpenCV [5], we have implemented customization of workspace calibration and selection, allowing users to specify specific areas they wish to be featured in the live video. This respects their privacy, as all processing related to this camera runs only on their local devices. Only the cropped image/video is uploaded to the cloud and shared with teachers.

3.3.3 Voice Chat. Beyond visual elements, we also implement voice chat functionalities using Vivox [34]. This feature allows teachers to communicate with the entire class, selected groups of students, or individual students using voice channels. By providing voice

communication, we promote effective audio information exchange and facilitate diverse modes of voice communication.

3.3.4 Others. We further design a variety of clicking-based means to enhance teacher-student interactions and simulate essential class features such as raising hands and giving rewards. Students can raise hands and express their sentiments in emojis by clicking dedicated buttons in the virtual classroom scene. Teachers can reward outstanding students in a similar way.

3.4 "Togetherness" Design

Video conferencing software provides essential tools for fulfilling basic teaching and learning requirements. Nevertheless, the absence of "togetherness" between teachers and students during online classes is a significant challenge. For instance, teachers often find themselves navigating through numerous small windows, monitoring individual students one by one. This fragmented view not only consumes valuable class time but also hampers meaningful interactions. The majority of classroom interactions tend to become one-way, leading to a shift towards passive learning.

Our system aims to infuse synchronous online teaching and learning with a sense of togetherness. Our design presented above has enabled the immersive and interactive experience in SAILS. We create a virtual environment that closely resembles physical classroom settings. This replication fosters a shared sense of belonging among students, allowing them to feel connected and engaged in online classes. Meanwhile, teachers can walk around, monitor student attentiveness and closely track learning progress, in a virtual classroom.

3.4.1 Gather Together. Students' engagement level increases when they experience a deep sense of "togetherness". Imagine a classroom where spatial orientation mirrors the physical setting — a space where teachers and classmates occupy familiar positions, akin to the arrangement in a physical room. This shared spatial context cultivates a feeling of unity, creating an environment where students feel connected and familiar.

By employing realistic classroom modeling techniques, teachers can design the virtual classroom to closely resemble real-life settings. The class is divided into groups, with each group sitting around a table within the same virtual space. Throughout the class, students are assigned specific seats and cannot move freely. Only the teacher has the authority to assign students to different seats and walk around. Although students cannot move around in the virtual classroom, they can visually explore their surroundings by looking around. They can observe the facial expressions and body gestures of their groupmates as in a physical classroom, mirroring the dynamics in a physical classroom.

3.4.2 Learn Together. SAILS utilizes both physical and virtual spaces to enhance online education for young learners. The students and teachers exist in a shared virtual space within the system, and aspects of their physical space are projected into this virtual space. This allows students to actively participate in class activities in a physical manner, while having an intuitive way to interact with their teacher and classmates through the virtual space.

During classes, teachers can utilize the virtual whiteboard to display and illustrate lecture materials. They also have the option to enable a full-screen mode, temporarily disabling the 3D scene and directing students' attention solely to the virtual whiteboard. This ensures students focus on the lecture material being presented.

In addition to one-way teaching, in-class activities are beneficial for students to understand the knowledge deeply and allow excellent opportunities for teachers to offer feedback and support to students based on their individual needs. For young learners, these activities often involve hands-on activities such as writing, drawing, or making handcrafts. To accommodate these activities, SAILS allows students to physically work on their desks, with the second camera capturing their activities and displaying them in real-time on the virtual table. Students view the 3D scene from a first-person perspective without seeing their own avatars.

With real-time hand movement transfer, behavioral cues are available inside the system. Teachers can monitor multiple students' engagement by observing their virtual hand movements simultaneously. Teachers can also thoroughly check each student's work by "walking" to their virtual avatar and viewing the live video displayed on the table. This enables teachers to assess student progress and provide personalized instructions and tailored education to individual students. Additionally, the system enables one-on-one chat between a teacher and a selected student to further enhance communication and address specific needs.

Peer-to-peer interactions play a crucial role in student development as well, which promote learning from peers, collaborative problem-solving, and the development of social skills [30]. Though not being able to move freely, students can look around and observe their peers' activities and facial expressions, creating an environment similar to a physical classroom. This setup helps keep the class organized. Additionally, the design of group-mode voice chat further enhances group exchange. Students seated together at a virtual table can engage in group discussions and collaborative conversations without disrupting other groups.

3.4.3 Exercise Together. SAILS extends the traditional classroom by offering virtual PE classes. The teacher can assign a standing point for each student. Students can do workouts in front of their respective device's camera. Their movements are captured in real time and transferred to their virtual avatars. All virtual avatars exercise in the same virtual classroom, forming an "exercise-together" scene. Similar to normal classes, students can only stay in place, while watching the live demonstration from the teacher or seeing all the classmates working out together.

3.5 Implementation and Capability

3.5.1 Client Application. The above sections mainly describe the client application of our SAILS system. Developed in Unity, our client application is designed to operate on both Windows (64-bit, Windows 10 or above) and Mac computers (MacOS 12 or above) with two cameras (360p or above). The home devices running our application do not have to be equipped with high computational power but should possess basic graphics rendering capabilities.

3.5.2 Server Implementation. The primary functions of our system's server backend include managing classrooms and users, distributing data to clients for individual 3D scene rendering, and executing high computational tasks (e.g., avatar head modeling).

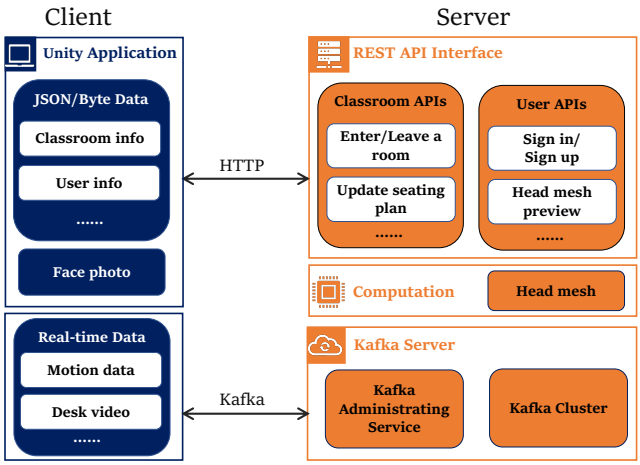


Figure 4: Client-server architecture of SAILS. The client Unity application sends JSON/byte format classroom and user information, as well as face photos, to the server via HTTP. The server processes the data (e.g., generating head meshes) and sends back the appropriate responses via REST APIs. Real-time data, such as motion data and desk video, are sent to the server via Kafka for distribution, ensuring efficient and prompt data exchange.

We employ Apache Kafka [11] as the backend framework, enabling efficient and effective handling of multimedia data sharing with client applications in real-time. Our system adaptively allocates processing and rendering tasks according to the processing capacity of client devices and the available network bandwidth between the client and the server. Computationally demanding offline tasks are delegated to the server. For instance, during the head modeling phase, the client application captures a frontal facial photo, which is then sent to the server for further processing and modeling. The server returns the result to the client once it has finished computation. To ensure smooth interactions, we strategically shift most of the real-time rendering and processing tasks from the server to the client. The client application handles real-time data detection and processing, including facial expression, and hand and body movement. The processed data is continuously transmitted to the server backend for distribution, ensuring synchronized experiences across users. A similar approach is adopted for desk video live streaming.

3.5.3 Client-Server Architecture. The server and client are connected through a combination of HTTP and Kafka protocols, as shown in Figure 4. We designed different communication protocols based on the data type.

For data that does not require continuous real-time synchronization, such as classroom and user information in JSON/byte format, we use the HTTP protocol for communication between the client and server REST API interface. When the client (Unity application) needs to send or request information (such as classroom info, user info), it makes HTTP requests to the server. The server processes these requests and sends back the appropriate responses. Specifically, the face photo is sent to the server via HTTP, where high computational head mesh computation is performed, and the generated head mesh result is returned to the client.

For data requiring real-time synchronization, such as motion data and desk video, we use the Kafka protocol. The client application processes motion data and desk video from webcam inputs and sends them to the server via Kafka. Data are queued and distributed by the server, which handles clustering and services for data transmission, maintaining seamless communication. Kafka serves as a message broker to manage real-time data streams between the client and server, ensuring efficient and timely data exchange.

3.5.4 Design Iteration and User Feedback. During development, we conducted iterative participatory design processes to effectively address users' needs and facilitate the adoption of technology in the classroom. To create user-friendly designs that meet the online teaching and learning requirements of young learners, we carried out periodic pilot trials with three young student teachers, four teenage students, and two young students under 10. By gathering their insights, preferences, and requirements on usability and in-class functions, we were able to refine and improve the design. We applied and enhanced existing technologies to address the specific challenges of this scenario, ensuring that the system is tailored to the needs and preferences of its users.

For instance, to ensure accurate body movement transfer for underage users with different skeletal proportions compared to adults, we developed a vectorization process to map their poses to 3D avatars accurately. Considering various user environments, such as suboptimal lighting conditions, we implemented a skin correction process for face modeling to obtain a more robust and natural skin texture.

In addition to addressing their specific needs, we made adjustments based on their feedback to ensure ease of use and interactivity. For example, we ensured that moving the avatar is intuitive and incorporated suggestions from students to refine interactive methods. These iterative cycles of gathering user feedback and making improvements were repeated several times until we finalized the design.

4 Experiments

To comprehensively evaluate the performance of SAILS, we conducted a user study involving 40 kindergarten and primary school students as well as their teachers. The study consisted of both subjective and objective evaluations. For the subjective user study, participants were asked to share their experiences of taking classes on the system, focusing on aspects such as system design, usability, and their feelings. For the objective evaluation, we established a server statistics monitoring stack to capture and analyze system data. Metrics such as latency, throughput and CPU utilization were measured to assess system efficiency and stability.

4.1 Subjective User Study

4.1.1 Participants. The user test involved 40 students ($M = 8.4, SD = 1.985$) and 6 teachers ($M = 34.3, SD = 13.426$). Participants included 10 kindergarten students ($M = 5.4, SD = 0.516$), 10 lower primary students ($M = 8.6, SD = 0.699$), and 20 upper primary students ($M = 9.8, SD = 0.951$). Half of the taught general subjects, while the

Table 1: Questionnaire. (x) indicates the question is excluded from the kindergarten questionnaire set.

Criteria	Questions
User Experience	
Enjoyment	I enjoy using the app.
Excitement	It is exciting to use the app.
Recommendation	I would like to recommend the app to a friend.
System Usability	
Simplicity	The app is easy to use. (x)
Frequency	I would like to use the app frequently.
System Design	
UI	The app UI is delightful.
Smoothness	The app runs without bugs or errors. (x)
Trustworthiness	I trust the app with my personal information. (x)
Special Features	
Attractiveness	I find the app to be attractive.
Functionality	The app meets my teaching/learning requirements.
Uniqueness	The app offers features that other online teaching tools don't. (e.g., Zoom) (x)

other half taught PE classes. All participants had prior experience with Zoom-like online tools.

4.1.2 Procedures. We provided on-site installation and comprehensive instructions to all participants on how to use the SAILS. Common personal computers, such as MacBook Air, were used. Each participant received a brief introduction to navigate and operate within the system. Additionally, teachers were given specific instructions on how to conduct and manage classes efficiently within the application.

All participants were required to create their own virtual avatars with in-app guidance. They were instructed to take a frontal facial photo and personalize their avatars by adjusting hair color, clothing style, and clothing color. Teachers were also responsible for creating a virtual classroom specifically for testing purposes.

The student participants were divided into batches of ten. They engaged in the same normal and PE classes led by their respective teachers within SAILS. In normal classes, teachers used slides for teaching and included hands-on activities such as writing and drawing related to the learning materials. Teachers were also encouraged to move around within the virtual environment to monitor students' progress during in-class exercises. In PE classes, students actively participated in a series of exercises guided by their PE teachers.

The testing session with each participant group (10 students and 2 teachers) lasted approximately 30 minutes. After experiencing the application, each participant was asked to complete a questionnaire to gather their feedback and insights. For kindergarten participants, instead of having them write their feedback, we conducted short interviews to ask them to rate their experience.

4.1.3 Measurements. We designed a questionnaire to assess the user experience of SAILS, which mostly consisted of rating-based questions. Users were asked to provide ratings on a 5-point Likert scale [19], ranging from 1 (strongly disagree) to 5 (strongly agree). The questions were selected from two established questionnaires, the System Usability Scale (SUS) [6] and the Standardized User Experience Percentile Rank Questionnaire (SUPR-Qm) [28].

We tailored the questionnaire to different participant groups. Kindergarten participants were given a subset of the questionnaire (7 in total) related to their user experience, as they were expected to use the application under parental supervision. Other participants, including teachers, were given a broader set of 11 questions, which included questions about app design in addition to user experience. The questions were organized into four categories, with each category containing two to three questions, as listed in Table 1.

Alongside rating-based questions, we included an optional open-ended question inviting participants to share what they liked most about SAILS. This qualitative feedback provided valuable context and complemented the quantitative ratings.

4.1.4 Results from Students. We utilized statistical inferences to evaluate the rating responses from students and gain insights. Our analysis focused on two key aspects for each question: the mean rating, which represents the general group opinion, and the frequency distribution, which reveals the degree of variability in the groups' opinions.

In analyzing the mean value of 5-point Likert scale questions, the scale interval of each agreement level is defined at 0.8 [29]. If the mean rating falls into the interval $(4.2, 5]$, it indicates that the group "strongly agrees" with the statement. If the mean rating is within the interval of $(3.4, 4.2]$, it signifies the group generally "agrees" with the statement and so on. The results are as follows:

- User Experience: All student participants reported a positive experience using the app, finding it enjoyable ($M = 4.55, SD = 0.783$) and exciting($M = 4.7, SD = 0.648$), and would likely recommend the app for distance learning ($M = 4.25, SD = 1.149$).
- App Usability: The participants agreed that the application was easy to use ($M = 3.87, SD = 1.279$) and that they would use it frequently($M = 4.25, SD = 1.171$).
- App Design: The student group strongly agreed that the user interface was well-designed and delightful ($M = 4.4, SD = 0.810$). The group also generally agreed that the application ran smoothly ($M = 3.7, SD = 1.291$) and could protect privacy ($M = 3.97, SD = 1.272$).
- Special Features: Participants strongly agreed that the new features were attractive ($M = 4.4, SD = 0.900$) and unique compared with other online conference tools ($M = 4.25, SD = 1.080$). The group also generally agreed that SAILS can satisfy their learning requirements ($M = 4.07, SD = 1.388$).

Overall, the mean rating of every question was above 3.7 ("agree" level), indicating that participants had a positive perception of our system on average. Our system was highly rated by the students, particularly in terms of user experience. Students highly rated our system in terms of user experience, with all criteria under this category receiving a rating above 4.2 ("strongly agree" level). This suggests that our system can provide a positive experience. The system design, usability, and special features also received positive feedback from the participants.

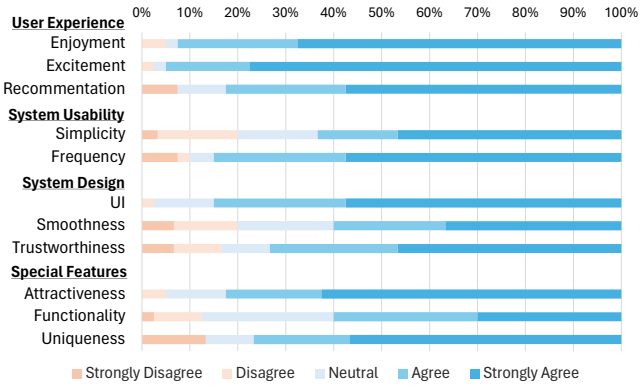


Figure 5: Frequency distribution of students' responses.

To analyze the frequency distribution of individual questions, we counted the number of responses for each rating and represented it in a stacked bar chart, as shown in Figure 5. The results revealed that all distributions leaned towards the positive side. We further conducted an analysis of interrater agreement among all student participants by calculating the Finn coefficient (two-way) as recommended in [24]. The computed value was 0.708, indicating a substantial level of agreement among the participants.

Specifically, less than 20% of respondents expressed disagreement with our system features across all evaluation criteria, indicating that our system was generally accepted by the majority. Notably, more than 90% of participants expressed enjoyment and excitement when using the system. Over 80% of participants were drawn to the special features of our application and expressed a liking for our UI design. The majority of participants expressed a desire to frequently use SAILS as an online learning tool and were willing to recommend it to others. They also highly approved of the unique design. However, we identified areas for improvement. Around one-third of the students agreed that our app had bugs and encountered difficulties in operations. This feedback has motivated us to enhance the stability and user-friendliness of the system.

As for the open question, several young participants expressed their enthusiasm for the customized avatar creation feature. Some also enjoyed using emojis in the normal class and found the PE class fun and interactive. Some mentioned that they could easily know what other classmates were working on, and the virtual classroom environment made them feel like staying in the physical classroom. Their feedback proved that SAILS could raise their interest and improve their engagement in classroom activities. Below are some representative comments from the students, with their participant number and age in parentheses:

- *"I'm excited about creating my avatar."* (Student1, 5)
- *"I enjoy using emojis and participating in the PE class. It's fun and interactive."* (Student2, 6)
- *"I love the pink hair and clothes on my avatar."* (Student3, 5)
- *""In normal class, I can see my friends at the same table and the teacher walking around. It's exciting."* (Student10, 5)
- *"I like the avatar design. I can customize it based on my preferences."* (Student11, 10)
- *"It is convenient and interesting."* (Student29, 12)
- *"I love PE classes the most. They are fun."* (Student35, 8)

4.1.5 Comparison among Different Student Age Groups. We observed that some questions had a high standard deviation, suggesting that some student participants had varying opinions. We then calculated the mean values and standard deviation for the kindergarten, lower primary, and upper primary groups. We found that the kindergarten and lower primary groups had similar results.

For comparison, we categorized participants into two distinct age groups: kindergarten and the lower primary section as the younger group ($M = 7.0, SD = 1.747$) and the upper primary section as the older group ($M = 9.8, SD = 0.951$). The aim was to identify any significant differences between their responses to rating-based questions with large variances.

We conducted a t-test and discovered that the aspect related to system functionality (i.e., "*The app's capabilities meet my learning requirements*") had statistically significant differences ($p < 0.05$) between the two groups. The average rating of the younger group was 4.05 ($SD = 1.099$), and the older group rated the same aspect with an average of 3.45 ($SD = 0.999$).

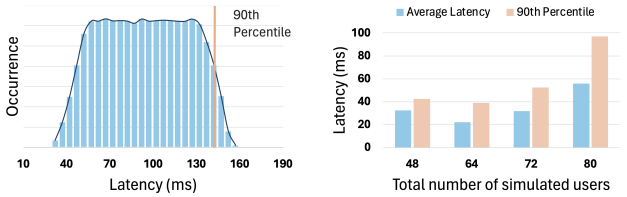
The results suggested that older students perceived our app as less aligned with their learning requirements compared to their younger counterparts. We hypothesize that this discrepancy arises due to differences in classroom activities: Upper primary students are less engaged in hands-on activities such as drawing or crafting; young learners benefit more from our system, especially when engaged in hands-on activities.

4.1.6 Feedback from Teachers. Teacher participants praised SAILS for providing several advantages that enhanced the overall teaching experience. Below we present a summary of their feedback from both normal classes and PE classes.

Teacher participants in normal classes appreciated the ability to create and customize virtual avatars based on individual appearances and preferences. This feature attracted the interest and curiosity of children, making the classroom environment more engaging. Additionally, the classroom environment settings met fundamental teaching needs, creating a learning space that felt like a real classroom. Students could concentrate on their hands-on activities, and teachers could easily check progress without interrupting students.

- *"SAILS allows users to create virtual avatars based on individual appearances and customize according to preferences. This feature is interesting and attracts children's interest and curiosity. I can see that students are engaged and excited. The classroom environment settings are also very complete, meeting basic teaching needs."* (Teacher1)
- *"Students can attend classes in a virtual environment that closely resembles a physical classroom. Compared to Zoom, using SAILS feels more like participating in an actual classroom setting."* (Teacher2)
- *"It is beneficial to see several students' work displayed on the same virtual table, making it easy to quickly check their progress."* (Teacher3)

Teachers leading PE classes said that the system provided an innovative way to deliver PE classes by capturing movements and transferring them to avatars in real-time. This feature allowed students to engage in physical activities while enjoying the virtual environment. They observed that students were more engaged in exercise when using SAILS compared to Zoom-like software.



(a) (b)

Figure 6: Server result. (a) Server latency distribution in the user study. (b) Average server latency in the stress test.

- *"The idea of capturing movements and applying them to avatars is very appealing."* (Teacher4)

- *"The user interface is interesting. sers can customize avatars, including hair, clothes, and more."* (Teacher5)

- *"Conducting classes in SAILS is more engaging than using Zoom. I noticed that students are more involved and attentive."* (Teacher6)

Due to page limits, rating-based questionnaire results from teachers are included in the supplementary material.

4.1.7 Observations. SAILS is primarily designed for online synchronous learning, where students participate from separate locations. However, during the user study, participants were in the same room and connected to the same Wi-Fi network. This sometimes led to slower rendering and occasional echoes when they simultaneously enabled voice chat. This setting may not have fully showcased the features of SAILS. Nevertheless, we highly value the insightful feedback provided by the participants, as it serves as a driving force for us to further enhance the robustness and usability of the system.

4.2 Objective Evaluation

4.2.1 Procedure. We extracted latency data from server logs and used JMX Exporter [26] and Prometheus [27] to gather throughput data and monitor CPU usage during user study. To evaluate the scalability of the server for real-time applications, we conducted a stress test. Three cloud servers were utilized to simulate several tens of clients in four virtual classrooms. Each server ran multiple instances of Python clients to simulate real-time data transmission. The server utilized was a virtual machine equipped with a 4-core CPU (64-bit Intel Core Broadwell IBRS) and 16 GB of memory, operating on Ubuntu 20.04.6 LTS. It had a maximum disk speed of 150 Mbps and a bandwidth capacity of 10 Gbps.

4.2.2 Measurements. We measured the system's performance using three evaluation metrics: latency, data throughput, and CPU usage. These metrics helped us determine the system's responsiveness in handling high-volume data transmission and complex processing. Latency was measured as the time difference between the server's message distribution timestamp and the client's acknowledgment receipt. The data throughput was the total amount of data output from the server.

4.2.3 Results. We analyzed the distribution of latency obtained during the user study, as shown in Figure 6 (a). 90% of the recorded latency values were below 135ms, indicating SAILS efficiently handled the user load during the study. Addtionally, the average data throughput was 78.96 Mbps, within the network capacity and the maximum disk speed. However, the CPU usage exceeded 250% during the avatar creation process, because of the significant computational overhead associated with head modeling, which also contributed to latency.

The latency results of stress tests under different settings are presented in Figure 6 (b). During the test, the network bandwidth was maintained at 80 Mbps. In our setting, the size of the data transmitted was slightly reduced with an increasing number of simulated users. The 90th percentile latency of SAILS remained below 100ms with 80 users. However, the average latency increased significantly to 1418.3ms with 96 users. SAILS was capable of supporting around 80 users for real-time streaming, with CPU usage remaining below 55%. Detailed experiments are available in the supplementary material.

4.2.4 Observations. Due to the user study setup, users may experience a less stable network connection, resulting in higher latency. Additionally, the high-computational head modeling process overloaded the CPU when multiple users were creating avatars simultaneously. This led to system hangs, where users experienced substantial delays in head modeling. Offloading the head modeling process to a GPU backbone would alleviate this system bottleneck and further enhance the user experience.

4.3 Future Work

4.3.1 User Study. The current evaluation was limited to a single session; future work will include longitudinal studies to assess the system's long-term capabilities and evolving user behaviors. To improve feedback quality, we will integrate structured interviews with post-study questionnaires, as younger participants may express their perspectives more effectively verbally. We will also refine question phrasing to reduce ambiguities. Quantitative metrics (e.g., immersion, engagement, learnability) will be introduced to supplementing subjective feedback.

4.3.2 App Design. To improve accessibility and reliability, we will enhance the app design for diverse age groups, optimize stability under network congestion, and ensure cross-device compatibility.

5 Conclusion

In this paper, we propose SAILS, a Synchronous Accessible Immersive Online Learning System. We incorporate immersive modeling, interactive communication, and "togetherness" design to provide an engaging virtual learning experience in a realistic setting. SAILS can support hands-on activities and in-class interactions for students while enabling easy monitoring for teachers. We discovered a positive preference for SAILS through a user study involving young learners of various ages and their instructors, showcasing the system's advantages in enhancing remote classroom engagement and effectiveness for young learners.

Acknowledgments

This work was supported by Hong Kong Innovation and Technology Commission's Innovation and Technology Fund (Award No. ITS/332/21FP) and Hong Kong RGC grants CRS_PolyU50123 and T43-51323-N (TRS).

References

[1] Leonard A. Annetta. 2006. Creating Presence and Community in a Synchronous Virtual Learning Environment Using Avatars. *International Journal of Instructional Technology and Distance Learning* 3 (2006), 27–41. https://api.semanticscholar.org/CorpusID:208126489

[2] German Barquero, Johnny Núñez, Sergio Escalera, Zhen Xu, Wei-Wei Tu, Isabelle Guyon, and Cristina Palmero. 2022. Didn't see that coming: a survey on non-verbal social human behavior forecasting. In *Understanding Social Behavior in Dyadic and Small Group Interactions (Proceedings of Machine Learning Research, Vol. 173)*, Cristina Palmero, Julio C. S. Jacques Junior, Albert Clapés, Isabelle Guyon, Wei-Wei Tu, Thomas B. Moeslund, and Sergio Escalera (Eds.). PMLR, 139–178. https://proceedings.mlr.press/v173/barquero22b.html

[3] Valentin Bazarevsky, Ivan Grishchenko, Karthik Raveendran, Tyler Zhu, Fan Zhang, and Matthias Grundmann. 2020. Blazepose: On-device real-time body pose tracking. *arXiv preprint arXiv:2006.10204* (2020).

[4] Valentin Bazarevsky, Yury Kartynnik, Andrey Vakunov, Karthik Raveendran, and Matthias Grundmann. 2019. Blazeface: Sub-millisecond neural face detection on mobile gpus. *arXiv preprint arXiv:1907.05047* (2019).

[5] Gary Bradski. 2000. The opencv library. *Dr. Dobb's Journal: Software Tools for the Professional Programmer* 25, 11 (2000), 120–123.

[6] John Brooke. 1995. SUS: A quick and dirty usability scale. *Usability Evaluation In Industry* 189 (11 1995).

[7] Zhilong Chen, Hancheng Cao, Yuting Deng, Xuan Gao, Jinghua Piao, Fengli Xu, Yu Zhang, and Yong Li. 2021. Learning from Home: A Mixed-Methods Analysis of Live Streaming Based Remote Education Experience in Chinese Colleges during the COVID-19 Pandemic. In *Proceedings of the 2021 CHI Conference on Human Factors in Computing Systems* (Yokohama, Japan) *(CHI '21)*. Association for Computing Machinery, New York, NY, USA, Article 348, 16 pages. https://doi.org/10.1145/3411764.3445428

[8] Ceren Çubukçu and Cemal Aktürk. 2020. The rise of distance education during Covid-19 Pandemic and the related data threats: A study about Zoom. *Iğdır Üniversitesi Sosyal Bilimler Dergisi* EK SAYI (2020) (2020), 127–144.

[9] Alexey Dosovitskiy, Lucas Beyer, Alexander Kolesnikov, Dirk Weissenborn, Xiaohua Zhai, Thomas Unterthiner, Mostafa Dehghani, Matthias Minderer, Georg Heigold, Sylvain Gelly, Jakob Uszkoreit, and Neil Houlsby. 2021. An Image is Worth 16x16 Words: Transformers for Image Recognition at Scale. In *International Conference on Learning Representations*. https://openreview.net/forum?id=YicbFdNTTy

[10] Haihan Duan, Jiaye Li, Sizheng Fan, Zhonghao Lin, Xiao Wu, and Wei Cai. 2021. Metaverse for Social Good: A University Campus Prototype. In *Proceedings of the 29th ACM International Conference on Multimedia* (Virtual Event, China) *(MM '21)*. Association for Computing Machinery, New York, NY, USA, 153–161. https://doi.org/10.1145/3474085.3479238

[11] Apache Software Foundation. 2023. Apache Kafka. https://kafka.apache.org.

[12] FRAME. 2024. FRAME : Virtual Learning in a browser based campus. https://learn.framevr.io/education-campus.

[13] Magdalena Garlinska, Magdalena Osial, Klaudia Proniewska, and Agnieszka Pregowska. 2023. The Influence of Emerging Technologies on Distance Education. *Electronics* 12, 7 (2023). https://doi.org/10.3390/electronics12071550

[14] Google. 2024. Google Meet. https://meet.google.com/.

[15] Zoom Video Communications Inc. 2024. Zoom. https://zoom.us/.

[16] Jinyoung Kim. 2020. Learning and teaching online during Covid-19: Experiences of student teachers in an early childhood education practicum. *International Journal of Early Childhood* 52, 2 (2020), 145–158.

[17] FUTURE WORK LAND. 2024. Virtual office and class software: Remote team and students management. https://futurework.land/classroom.

[18] Tianye Li, Timo Bolkart, Michael. J. Black, Hao Li, and Javier Romero. 2017. Learning a model of facial shape and expression from 4D scans. *ACM Transactions on Graphics, (Proc. SIGGRAPH Asia)* 36, 6 (2017), 194:1–194:17. https://doi.org/10.1145/3130800.3130813

[19] Rensis Likert. 1932. A technique for the measurement of attitudes. *Archives of Psychology* (1932).

[20] Camillo Lugaresi, Jiuqiang Tang, Hadon Nash, Chris McClanahan, Esha Uboweja, Michael Hays, Fan Zhang, Chuo-Ling Chang, Ming Guang Yong, Juhyun Lee, et al. 2019. Mediapipe: A framework for building perception pipelines. *arXiv preprint arXiv:1906.08172* (2019).

[21] Meta. 2024. Meta Quest Safety Information for Parents and Pre-teens. https://www.meta.com/quest/parent-info/.

[22] Microsoft. 2024. Microsoft Teams. https://www.microsoft.com/en-us/microsoft-teams/group-chat-software.

[23] Mozilla. 2024. Hubs - Private, virtual 3D worlds in your browse. https://hubs.mozilla.com/.

[24] Thomas A. O'Neill. 2017. An Overview of Interrater Agreement on Likert Scales for Researchers and Practitioners. *Frontiers in psychology* 8 (2017), 777.

[25] Optima Academy Online. 2024. Optima Academy Online - Online School Reimagined. https://www.optimaacademy.online.

[26] Prometheus. 2023. JMX Exporter. https://github.com/prometheus/jmx_exporter.

[27] Prometheus. 2023. Prometheus. https://prometheus.io.

[28] Jeff Sauro and Pareezad Zarolia. 2017. SUPR-Qm: a questionnaire to measure the mobile app user experience. *Journal of Usability Studies* 13, 1 (nov 2017), 17–37.

[29] Erol Sözen and Ufuk Guven. 2019. The Effect of Online Assessments on Students' Attitudes Towards Undergraduate-Level Geography Courses. *International Education Studies* 12 (09 2019), 1. https://doi.org/10.5539/ies.v12n10p1

[30] Harriet R Tenenbaum, Naomi E Winstone, Patrick J Leman, and Rachel E Avery. 2020. How effective is peer interaction in facilitating learning? A meta-analysis. *Journal of Educational Psychology* 112, 7 (2020), 1303.

[31] Unity. 2024. Animation Rigging. https://docs.unity3d.com/Packages/com.unity.animation.rigging@1.0/manual/index.html.

[32] Unity. 2024. Multi-Aim Constraint. https://docs.unity3d.com/Packages/com.unity.animation.rigging@1.0/manual/constraints/MultiAimConstraint.html.

[33] Unity. 2024. Physics Engine. https://docs.unity3d.com/Manual/PhysicsSection.html.

[34] Unity. 2024. Vivox voice chat. https://unity.com/cn/products/vivox-voice-chat.

[35] ViewSonic. 2024. UNIVERSE by ViewSonic. https://universe.viewsonic.io/.

[36] Virbela. 2024. Virbela: A Virtual World for Work, Education and Events. https://www.virbela.com.

[37] Yuyang Wang, Jean-Rémy Chardonnet, and Frédéric Merienne. 2021. Development of a speed protector to optimize user experience in 3D virtual environments. *International Journal of Human-Computer Studies* 147 (2021), 102578.

[38] Yuyang Wang, Lik-Hang Lee, Tristan Braud, and Pan Hui. 2022. Re-shaping Post-COVID-19 Teaching and Learning: A Blueprint of Virtual-Physical Blended Classrooms in the Metaverse Era. In *2022 IEEE 42nd International Conference on Distributed Computing Systems Workshops (ICDCSW)*. IEEE, 241–247. https://doi.org/10.1109/ICDCSW56584.2022.00053

[39] Eva Lau Yi-hung. 2022. Survey: Nearly 70% of Parents Find Their Children Have Difficulty Learning at Home. https://www.eduhk.hk/en/press-releases/survey-nearly-70-of-parents-find-their-children-have-difficulty-learning-at-home.

[40] Fan Zhang, Valentin Bazarevsky, Andrey Vakunov, Andrei Tkachenka, George Sung, Chuo-Ling Chang, and Matthias Grundmann. 2020. Mediapipe hands: On-device real-time hand tracking. *arXiv preprint arXiv:2006.10214* (2020).

[41] Wojciech Zielonka, Timo Bolkart, and Justus Thies. 2022. Towards Metrical Reconstruction of Human Faces. In *European Conference on Computer Vision*. https://api.semanticscholar.org/CorpusID:248177832

SAMPL: Self-Attention Modelled Patch Learning for Efficient Visual Understanding

Zhiming Hu*
zhiming.hu@samsung.com
Samsung AI Center
Toronto, Ontario, Canada

Salar Hosseini Khorasgani*†
salar.hosseinikhorasgani@mail.utoronto.ca
Tenstorrent
Toronto, Ontario, Canada

Weiming Ren
w2ren@uwaterloo.ca
Samsung AI Center
Toronto, Ontario, Canada

Iqbal Mohomed
i.mohomed@samsung.com
Samsung AI Center
Toronto, Ontario, Canada

Abstract

We study the patch selection problem for efficient transformer-based visual understanding wherein a sampler can be used to drop less informative patches *at inference* in order to speed up model execution on resource-constrained devices. As no labels are available on the saliency of the patches, existing works either try to solve an auxiliary task of locating distinctive patches or learn a policy network through the global image/video-level supervision. The former approach could drop redundant but important patches while the latter suffers from the weak supervision of *a single class label per image/video*. In this work, we observe that the attention weights in trained transformer-based models clearly highlight the salient regions in images and videos. Therefore, we propose a learned patch sampling framework called SAMPL that utilizes the attention weights as *fine-grained patch-level supervision* to learn a lightweight policy network for patch selection. To train SAMPL end-to-end with the transformer-based models, we introduce a new loss function based on the REINFORCE algorithm to match the distribution of patch selection probabilities and the attention scores. Experimental results on ImageNet, UCF101, Something-Something v2 and Kinetics-400 show that SAMPL can effectively increase the throughput by at least 1.5× while achieving competitive classification accuracy.

CCS Concepts

• **Computing methodologies** → **Classification and regression trees**; **Neural networks**.

Keywords

Vision Transformer, Patch Sampling, Self-Attention

*Both authors contributed equally to this research.
†Work done during internship at Samsung AI Center, Toronto.

Permission to make digital or hard copies of all or part of this work for personal or classroom use is granted without fee provided that copies are not made or distributed for profit or commercial advantage and that copies bear this notice and the full citation on the first page. Copyrights for third-party components of this work must be honored. For all other uses, contact the owner/author(s).
MMSys '25, March 31-April 4, 2025, Stellenbosch, South Africa
© 2025 Copyright held by the owner/author(s).
ACM ISBN 979-8-4007-1467-2/2025/03
https://doi.org/10.1145/3712676.3714447

ACM Reference Format:
Zhiming Hu, Salar Hosseini Khorasgani, Weiming Ren, and Iqbal Mohomed. 2025. SAMPL: Self-Attention Modelled Patch Learning for Efficient Visual Understanding. In *ACM Multimedia Systems Conference 2025 (MMSys '25), March 31-April 4, 2025, Stellenbosch, South Africa.* ACM, New York, NY, USA, 11 pages. https://doi.org/10.1145/3712676.3714447

1 Introduction

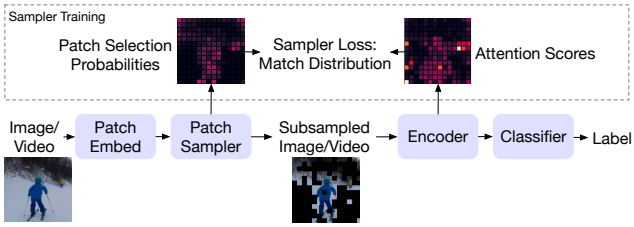
Vision transformers (ViTs) have shown great success on various tasks [2, 8, 10, 17, 34, 35]. However, when deployed, they are usually computationally expensive due to the quadratic complexity of self-attention [37] with respect to the number of input patches. Fortunately, as shown in the literature [5, 24, 27], not all patches are equally important and dropping unimportant patches has limited impact on classification accuracy. Therefore, dropping less informative patches and only processing the remaining patches has the potential to greatly reduce the computational overhead at test time while maintaining similar performance.

However, the selection of relevant patches is non-trivial for the following reasons. First, there are no labels available pertaining to whether a patch is relevant to a class label. Such a labelling process would be tedious and expensive if not impossible. Second, if we train a policy network for this task, it needs to make discrete actions, which makes it challenging to be trained end-to-end.

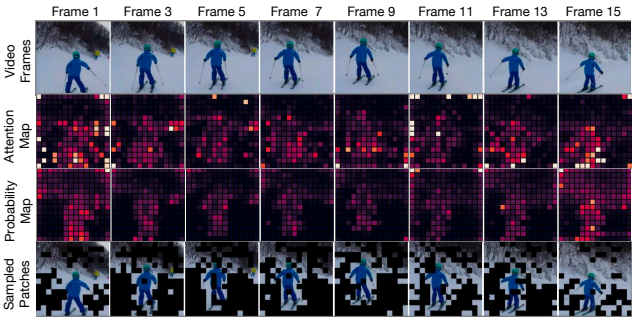
There are two main lines of work that could potentially be applied to address the challenges described above. The first approach selects distinctive patches using the K-Center search algorithm [27]. This approach can effectively reduce the redundancy in the input patches. However, distinctive patches may not always be relevant or informative (*e.g.*, random objects/patches in the background). The second line of work [6, 38] proposes to train a policy network[1] using a differentiable top-k operator (*i.e.*, Perturbed Top-K [6]) end-to-end with the original classification model. However, the policy network is hard to be trained and yields limited performance gains as it only relies on the weak supervision of a single class label per image/video, as we later show in our experiments.

In this work, we propose SAMPL, a self-attention based input patch sampling framework to improve the inference efficiency of ViTs. A simplified diagram of SAMPL is shown in Figure 1a. Unlike existing works [6, 38] where the policy network is trained

[1]A policy network is a neural network that learns the mapping from states of the environment (*e.g.*, patch features) to actions (*e.g.*, keep or drop a patch).



(a) Overview of SAMPL at a glance.



(b) A running example of SAMPL.

Figure 1: Overview of SAMPL with a running example on video action recognition. In Figure 1a, there is a patch sampler in the classification pipeline that drops unimportant patches and passes only the remaining patches to the encoder to save computation during inference time. The patch sampler is trained with the supervision of the attention weights in the transformer encoder. In Figure 1b, we show an example video, the aggregated attention weights from the last four layers of the ViT model (lighter colors represent higher attention weights), the patch sampling probabilities learned by SAMPL, and the masked video. We can clearly see that SAMPL has learned to select the patches with higher attention weights. Images are from the video [20170101 Skier JA] by [S&B GV] in Kinetics-400 [19] licensed under CC BY 3.0.

with the weak supervision of one class label per image/video, we observe that the self-attention weights in the transformer-based encoder can naturally provide a much richer signal to guide the patch selection. This is because self-attention weights directly indicate the importance of the patches/tokens across the transformer network [10, 24]. This is depicted in the example results of SAMPL shown in Figure 1b, where the more relevant patches correspond to higher attention weights (lighter colors) while the less relevant patches correspond to lower attention weights (darker colors).

To this end, we propose to train a policy network with the fine-grained patch-level attention-based supervision. The policy network is a one-layer transformer that takes patch embeddings as inputs and learns a patch selection probability distribution. To address the differentiability issue of sampling from the learned distribution, the policy network is learned with a separate loss function motivated by the REINFORCE algorithm [40]. In this setting, patch selection is the action and attention weights corresponding to each selected patch are the rewards. Therefore, SAMPL is trained to select patches with higher attention weights in the last few layers of the transformer-based encoder. For both training and inference, only patches sampled from the learned probability distribution will be fed to the transformer-based encoder to save computation.

Highlights of our main contributions in this paper are as follows:

- To the best of our knowledge, we are the first to leverage attention weights in ViT encoders as direct patch-level supervision to train a policy network for patch selection, which greatly improves the performance over other patch sampling approaches.
- We introduce a loss function based on REINFORCE [40] algorithm that encourages the patch sampler to select patches with higher attention weights.
- We conduct extensive evaluations on ImageNet, UCF101, Something-Something v2 and Kinetics-400. SAMPL can drop around 70% of patches in videos and 50% of patches in images while maintaining decent performance.

2 Related Work

Patch Selection for Convolutional Neural Networks (CNNs) Patch selection [18, 30, 36] is an important step to efficiently process high resolution images, especially in the case where only a small portion of the images contains relevant information (*e.g.*, a small speed sign in an image taken on a highway for a traffic sign classification task). To address the patch selection problem, [18, 30] propose to extract an attention map from a low resolution image and use the attention map to guide the sampling of patches on the high resolution version of the same image. In these works, attention is defined using an approximate expectation of the intermediate features in a CNN, whereas we leverage the attention weights from the self-attention mechanism [37] in transformers. Addressing the same problem, [6] proposes using a differentiable top-k operator (Perturbed Top-K [6]) to learn to select the most relevant patches for processing high resolution inputs. This approach relies solely on the gradients propagated from the classification loss, which provides a weak image-level supervision since it encapsulates all the patch selection actions on an entire image using a single label. Instead, SAMPL adopts fine-grained patch-level supervision based on the self-attention weights within transformer blocks to learn a stronger policy.

Patch Selection for Transformers (Before Transformer Blocks) The most closely related work to ours is that of Park et al. [27] as they also perform patch sampling for efficient video recognition with transformer-based models. They adopt the K-Center search [14, 16] algorithm to iteratively select the farthest patch (in terms of color and position) to what has been chosen so far, which may result in a diverse set of patches. However, since this approach does not consider the relevance of each patch to the semantic label of a video, it may drop informative patches while keeping diverse but uninformative patches.

Token Reduction Within Transformer Encoder While patch selection methods drop patches before any transformer blocks, token reduction based approaches typically reduce the number of tokens within the transformer blocks [3, 10, 21, 29, 44] or between the transformer blocks [23, 25, 26, 31, 33, 38, 41–43]. Comparing with token reduction approaches, SAMPL can drop patches in an earlier stage of the forward pass and therefore has the potential to

save more computation. For instance, many of the aforementioned approaches can only reduce the computation by around 30% while SAMPL can reduce the computation by as much as 90% or more. In addition, while most approaches are only designed for image classification, SAMPL supports both image and video inputs. Notably, SAMPL can be seamlessly integrated into an edge-cloud architecture where the patch sampler and the image/video classification network could be deployed at the edge and cloud, respectively. In this scenario, SAMPL can optimize the overall resource utilization by filtering out non-essential patches at the edge node and transmitting only the most informative patches to the server, thereby reducing both network bandwidth consumption and computational overhead on the cloud.

MAEs for Images and Videos Our work is also also related to masked autoencoders (MAE) [1, 12, 17, 28, 34] since they can greatly improve the efficiency of unsupervised pre-training by randomly masking a large portion of patches before feeding the remaining ones to an encoder. In particular, AdaMAE [1] is related to our approach since it learns a policy for sampling patches. However, that method is applied to the reconstruction task during pre-training and uses reconstruction error, which is only available during pre-training, to train the sampler, whereas we focus on classification and train our sampler using attention weights for efficient model inference. While being efficient during pre-training, MAE-based models still use all the patches for finetuning and inference. We address this by proposing a learning-based patch sampler that can improve the efficiency of transformer-based encoder during inference while achieving competitive classification accuracy.

3 Method

Our goal is to learn which patches of a given image or video are the most important to keep as inputs to a transformer-based encoder to maintain high performance on classification tasks while reducing computation. To achieve this, we propose a framework called SAMPL which learns to model a distribution for how informative each patch is by using the self-attention weights of the transformer-based encoder as the fine-grained supervision signal. An overview of SAMPL is shown in Figure 2. In the following subsections, we will describe the core elements of our proposed framework: (i) a lightweight policy network which models the probability distribution for selecting each patch (Section 3.1), (ii) the attention-based supervision signal for patch selection (Section 3.2), and (iii) the sampling loss function used to optimize the policy network during end-to-end training (Section 3.3).

3.1 Policy Network for Patch Sampling

To model the probability distribution for selecting patches, we seek a policy network that is differentiable and lightweight. Here, a policy network is a neural network that learns a policy to choose actions (*i.e.*, keep or drop patches) given the patch features. Having a differentiable policy network enables us to learn the patch selection distribution directly through gradient descent using our sampling loss function, which will be discussed in Section 3.3. Furthermore, it should be lightweight since our goal is to achieve high performance with minimal extra overhead.

Patch Embedding Layer Given an image/video that has been tokenized using the patch embedding layer of a standard ViT encoder [8] (a 2D convolution layer for images or a 3D convolution layer for videos), the input to the policy network is a set of patch tokens $\mathbf{X} \in \mathbb{R}^{N \times d}$ where $N = \frac{T}{t} \times \frac{H}{h} \times \frac{W}{w}$ is the number of patches, and d is the embedding dimension. (T, H, W) and (t, h, w) denote the temporal dimension, height, and width of the input image and the patch where $T = t = 1$ for images. Prior to passing the patch tokens to the policy network, we add the fixed periodic positional encoding [37]. This allows the policy network to consider positional information when modelling the patch selection distribution.

Patch Sampler As in Figure 2, the policy network is a standard one-layer transformer block [37], which consists of a multi-head attention (MHA) sub-layer and a fully connected feed-forward network sub-layer [37], followed by a linear layer and a softmax activation. The outputs of the policy network are $\mathbf{P} \in \mathbb{R}^N$, representing the probabilities of selecting each patch. Using the predicted probabilities, we perform multinomial sampling to draw $K < N$ patch indices $\mathbf{I_s}$ without replacement, where K is the number of sampled patches. Finally, indexing is applied using these indices to retrieve the sampled patch tokens $\mathbf{X_s} \in \mathbb{R}^{K \times d}$ that are inputted to the encoder. We note that multinomial sampling and indexing are non-differentiable operations, thereby requiring the sampler weights to be updated using the auxiliary sampling loss discussed in Section 3.3.

3.2 Attention-based Patch-Level Supervision

To train the policy network described in Section 3.1, we require a supervision signal that can describe how informative each patch is for the classification task. One option is to use the classification loss as supervision [6], but that would provide a weak signal since it describes the importance of *all selected patches* using a joint supervision signal. For example, a high classification loss value will cause the patch sampler to learn to decrease the sampling probability of *all selected patches*, regardless of the saliency of each individual patch. Instead, we propose using attention weights as the supervision signal since they provide patch-level granularity (i.e., each patch gets its own supervision signal) and attention weights have been shown to be an effective indicator of the saliency of patches [4, 24]. Moreover, as we extract the attention weights from the same model that is being subjected to patch dropping, we incur no extra computation to obtain the attention weights. We highlight this in Figure 3, where we visualize the aggregated attention weights from different layers in the transformer. Clearly, the attention weights, especially those aggregated from the last few layers, can capture the salient regions in the video.

For reasons that will be made clear in Section 3.3, we will denote our supervision signal for the K sampled patch tokens as the attention reward $\mathbf{R_{attn}} \in \mathbb{R}^K$. In the equations that follow, we will omit the head dimension in MHA for simplicity. To obtain the attention reward, we first extract the attention weights $\mathbf{A}$ from the MHA calculation in a transformer layer:

$$\text{Attention}(\mathbf{Q}, \mathbf{K}, \mathbf{V}) = \mathbf{A}\mathbf{V} \tag{1}$$

$$\mathbf{A} = \text{softmax}(\frac{\mathbf{Q}\mathbf{K}^\mathbf{T}}{\sqrt{d_k}}) \tag{2}$$

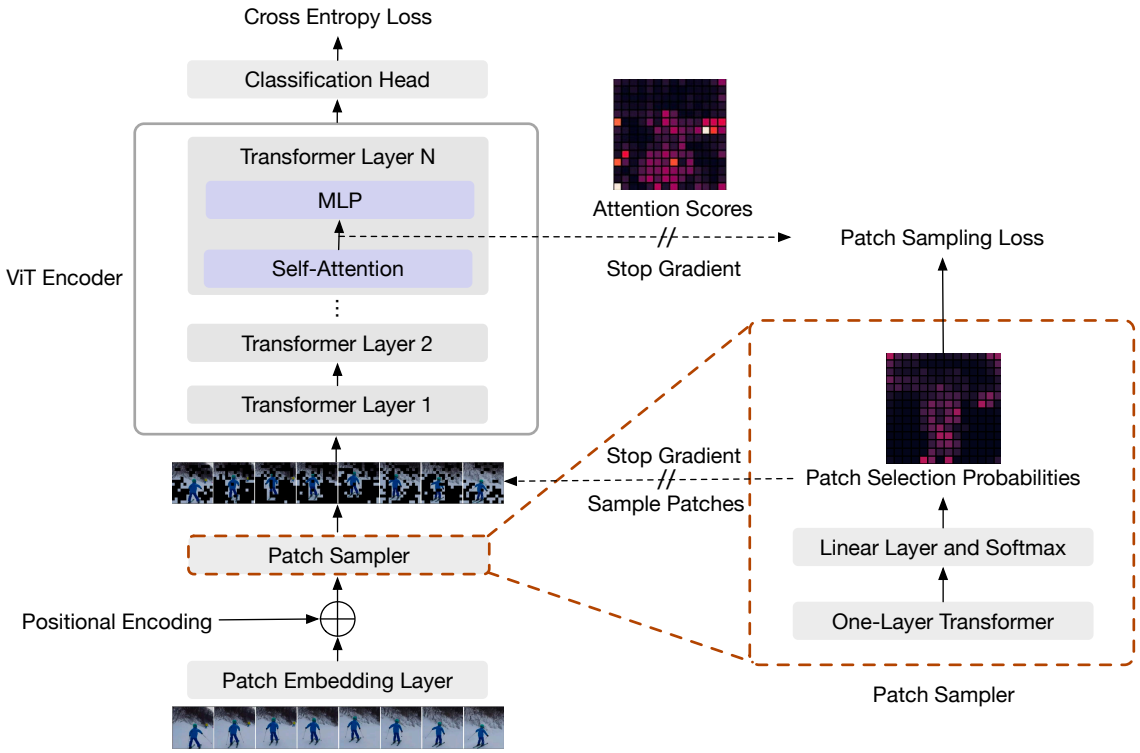


Figure 2: An overview of our proposed method SAMPL applied to video action recognition. After computing the patch embeddings of the video and adding the positional encoding, we (i) pass the patch embeddings to an adaptive patch sampler consisting of a policy network and a multinomial sampling step, (ii) pass only the sampled patches to the transformer-based encoder and compute attention scores using the intermediate attention weights, and (iii) compute a patch sampling loss for updating the policy network. The same method can be applied to image classification without modifications. Images are from the video [20170101 Skier JA] by [S&B GV] in Kinetics-400 [19] licensed under CC BY 3.0.

where $\mathbf{Q}, \mathbf{K}, \mathbf{V}$ are query, key, and value matrices, and d_k is the dimension of the keys. We then extract the attention weights of the output tokens. In the case of models that perform global pooling (averaging) on all tokens to compute the output and have no `cls` token ($\mathbf{A} \in \mathbb{R}^{K \times K}$), this is $\frac{1}{K}\sum_{k=1}^{K} \mathbf{A}_{k,j}$, which represents the importance of patch j to the output. In addition, we scale this value by the norm of $\mathbf{V}_j$ since that also affects the output of Equation 1 (this scaling technique has also been applied for token reduction in [10]). Overall, in the case of global pooling, we calculate the attention reward/score (from layer i) of patch j as:

$$\mathbf{R}_{\text{attn}}{}_j^{(i)} = \left(\frac{1}{K}\sum_{k=1}^{K} \mathbf{A}_{k,j}^{(i)}\right) \cdot \|\mathbf{V}_j^{(i)}\| \tag{3}$$

The attention reward can similarly be computed for models that use a `cls` token ($\mathbf{A} \in \mathbb{R}^{(K+1)\times(K+1)}$) as the output by replacing the sum term in Equation 3 with $\mathbf{A}_{\texttt{cls},j}^{(i)}$ (i.e. the attention weight from the `cls` token to the j^{th} patch in layer i) instead to represent the importance of patch j to the output.

Finally, we aggregate the attention rewards of the chosen layers $l_m, \ldots, l_n$:

$$\mathbf{R}_{\text{attn}} = \sum_{i=l_m}^{l_n} \mathbf{R}_{\text{attn}}{}^{(i)} \tag{4}$$

To account for the fact that different heads in MHA might attend to different regions of importance, we sum the attention rewards over all heads.

3.3 Patch Sampling Loss

To optimize the policy network containing parameters θ, we propose to use a patch sampling loss function which follows the expected reward maximization in the REINFORCE algorithm [1, 40]. Moreover, considering patch sampling (using the probabilities $\mathbf{P}$ predicted by the policy network in Section 3.1) as an action and the attention rewards $\mathbf{R}_{\text{attn}}$ (defined in Section 3.2) as rewards, we aim to maximize the expected attention reward $\mathbb{E}_\theta[\mathbf{R}_{\text{attn}}]$. Our patch sampling loss function is therefore defined as:

$$\mathcal{L}_{PS} = -\sum_{j=1}^{K} \log(\mathbf{P}_{(\mathbf{I}_s)_j}) \cdot (\mathbf{R}_{\text{attn}})_j \tag{5}$$

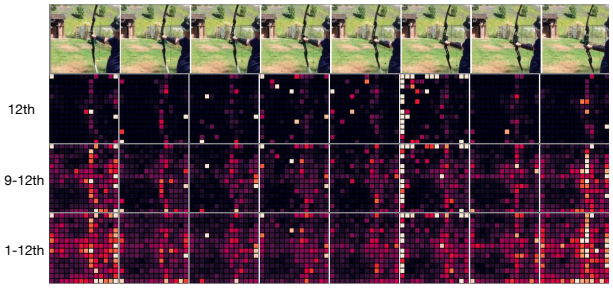


Figure 3: The aggregated attention weights from different layers in the transformer. From top to bottom, we visualize the aggregated attention weights of the VideoMAE [34] ViT-B model (after SAMPL training) obtained from (i) only the 12^{th} (last) layer, (ii) the 9^{th} to 12^{th} (last 4) layers, and (iii) the 1^{st} to 12^{th} (all) layers. The attention weights clearly indicate the salient regions in the video, with deeper layers showing more fine-grained attention. Images are from the video [Santiago Park Archers - Swinging Ball] by [Sam Sanchez] in Kinetics-400 [19] licensed under CC BY 3.0.

where $\mathbf{I_s}$ are the indices of the sampled patches as defined in Section 3.1. Here, $\mathbf{P}$ is indexed using $(\mathbf{I_s})_j$ since it contains the selection probabilities of all patches. We note that the logarithm of the probabilities is taken rather than the probabilities themselves to avoid precision errors. In addition, we detach the attention rewards from the computation graph to prevent this loss from affecting the encoder weights. Intuitively, this loss function will result in the policy network to align the patch selection distribution with the distribution of the attention weights since the probabilities of the patches with the highest attention rewards will be maximized. This intuition is supported by the fact that this loss function can be interpreted as the cross entropy between the patch selection probabilities and the attention rewards. As the goal is to align the patch selection distribution with the distribution of attention weights, Kullback-Leibler (KL) Divergence loss [20] could potentially be applied for our problem as well, which is left for future work.

4 Experiments

4.1 Experimental Setup

Datasets. We evaluate SAMPL on one image dataset and three video datasets, which are ImageNet-1K [7], Kinetics-400 [19], Something-Something v2 (SSv2) [15] and UCF101 [32]. In ImageNet-1K, there are around 1.3M training images and 50K validation images, categorized into 1000 object classes. Kinetics-400 is a large scale video dataset on human actions. It contains around 220K training videos, 20K validation videos, and 35K testing videos for a total of 400 action classes. SSv2 has approximately 170K training videos and 20K validation videos with 174 classes. Lastly, for UCF101, there are 9537 training videos and 3783 validation videos in 101 classes.

Baselines. **SAMPL**(Attn) denotes our attention-based patch sampling approach. Our baselines are as follows:

- **The state-of-the-art (SOTA) Patch Selection Methods** Patch selection is the main task for this paper. We compare SAMPL against SOTA patch selection approaches, which are K-Center [27] and Perturbed Top-K [6]. K-Center was initially proposed for video action recognition. We re-implement and integrate it into our framework to support both image and video inputs. Moreover, Perturbed Top-K was originally designed for selecting relevant patches from high resolution input images. We also re-implement it and compare it with SAMPL on ImageNet.
- **Token Reduction Inside Transformer Encoder** Due to the scope of the paper, we select two SOTA token reduction approaches, which are ToMe [3] and ATS [10], as the representative methods for token reduction.
- **SAMPL(Random)** Finally, to verify the effectiveness of our adaptive patch sampling method, we consider a trivial version of our method where we randomly sample patches for the ViT encoder. The rest of the configurations are kept the same as **SAMPL**(Attn).

Backbone Models. SAMPL can accelerate off-the-shelf classification models without needing to change the model architectures. To evaluate the performance of SAMPL on images, we take the pre-trained ViT-B classification model from MAE [17] and DINO [4]. DINO (ViT-B) has the same ViT-B architecture as MAE, except that it has a `cls` token and uses that to compute the attention weights and final output instead of performing global pooling. For videos, we take the pre-trained ViT-B classification models from VideoMAE [34] for the UCF101, SSv2 and Kinetics-400 datasets.

Training and Inference Details. Unless stated otherwise, we use the same training hyperparameters and augmentations as MAE [17] for all image experiments and VideoMAE [34] for all video experiments. When performing end-to-end training with our policy network and the encoder (that was already trained for classification), we multiply the base learning rate by a factor of 1.0 for the policy network and 0.1 for all other layers in the ViT encoder. This is done to reduce the changes to the encoder weights while focusing primarily on learning the sampling actions. On UCF101 and ImageNet, the base learning rate is set to 5e-4, while on SSv2 and Kinetics-400 it is set to 1e-5. For experiments on ImageNet, UCF101, SSv2, and Kinetics-400, training lasts for 30, 100, 50, and 50 epochs respectively. Training on ImageNet is done on 4 GPUs (with an effective batch size of 1024 on 70% and 90% masking experiments, and 512 on experiments with other masking ratios) while training on the video datasets is done on 4 GPUs with an effective batch size of 64. The spatial input dimension is 224 x 224 and for video experiments, the number of input frames is 16.

We do not use the cutmix and mixup augmentations for any experiments, and we do not use random erase for video experiments. This is because these augmentations perform patch-level modifications and can therefore bias the policy (or the baseline methods) to select patches that would otherwise be uninformative. Specific to our method, we set $l_m = 9, l_n = 12$ when producing final results. In other words, we aggregate the attention weights from the 9^{th} to 12^{th} layers (last 4 layers in ViT-B). We perform additional ablation experiments in Section 4.2.2 to justify this choice. In the implementation of Perturbed Top-K [6], we set the number of Gaussian samples as $n = 100$ and set the noise variance as $\sigma = 0.1$.

4.2 Experimental Results

4.2.1 Main Results.

Image Classification with MAE and DINO. We take the ViT-B classification model from MAE [17] and DINO [4]. We further finetune them using the settings described in Section 4.1. We also choose a variety of patch dropping ratios from 30% to 90% for the experiments. The throughput is evaluated on one A100 GPU[2] with a batch size of 128 for all cases.

In Table 1(a), we can see that with 70% and 90% dropping, all of the sampling methods can greatly improve the throughput by around 3× and 5× compared to the case of using all the patches. At the same time, **SAMPL**(Attn) is able to achieve the best Top-1 and Top-5 accuracy among the sampling methods, and only has a small performance drop of 1.2% in Top-1 with 30% of patch dropping when compared with using all patches.

When comparing with ToMe, **SAMPL**(Attn) with 30% masking[3] is slightly worse than ToMe (r=7). Note that due to gradual token merging within transformer, ToMe has higher memory cost when compared with our method, which drops unimportant patches at once before the transformer encoder. Moreover, **SAMPL**(Attn) with 50% masking outperforms ToMe for both throughput and classification accuracy. This reaffirms the superior performance of SAMPL when there is a significant need to minimize the computational requirements.

Among the sampling-based approaches, Perturbed-Top-K has the lowest classification accuracy and throughput. One hypothesis for this is that it only relies on the weak supervision of a single label for all the patch selection actions on one image, which makes it challenging to be trained. Regarding throughput, Perturbed-Top-K is more computationally expensive because it needs to run many simulations to obtain the patch selection probabilities.

Moreover, **SAMPL**(Attn) improves the Top-1 accuracy by 1.09% for 70% masking and 4.13% for 90% masking when compared with K-Center. The gap is larger with 90% masking because patch selection plays a more important role in the final performance when using higher masking ratios. Interestingly, we also observe that **SAMPL**(Random) shows descent performance overall with slightly lower accuracy. We hypothesize that this is due to the redundancy in patches as well as the resiliency of ViTs to patch dropping and occlusion stemming from the self-attention mechanism [24].

To show how SAMPL generalizes over different backbones, we show the results with the DINO model in Table 1(b). In general, we see similar trends where SAMPL performs the best, followed by K-Center, Random and Perturbed-Top-K on high masking ratios while ToMe works well on low masking ratios such as 30% and 50%. The other interesting observation is that the relative accuracy drop for MAE models are generally smaller than with DINO models under the same masking ratios. Our hypothesis is that MAE models are more resilient to patch drops because of their pre-training process where the encoder only processes a small portion of patches. To validate our hypothesis, we also run extensive experiments with SAMPL on VideoMAE models.

[2]When we say A100 GPU, we refer to the 40GB variant.
[3]We use patch dropping and patch masking interchangeably in this paper.

Video Action Recognition. We show the results on UCF101, SSv2 and Kinetics-400 in Table 2. We have four main baselines, which include using all patches, patch sampling with random selection (**SAMPL**(Random)), patch sampling with K-Center [27] and the SOTA token reduction approach ToMe [3]. For all the approaches, we measure the throughput as the number of items (videos) processed within one second on one A100 GPU with a batch size of 8. We further finetune the ViT-B classification model from VideoMAE for each sampling method using the settings described in Section 4.1. We consider masking ratios of 40%, 50%, 70% and 90% (percentage of patches dropped).

As shown in the table, when compared with ToMe [3], **SAMPL**(Attn) performs similarly with ToMe [3] when we merge 65 tokens in every transformer block (*i.e.*, r=65) for both the throughput and the accuracy on all the three datasets. However, when we reduce the number of patches more aggressively, **SAMPL**(Attn) greatly outperforms ToMe (r=130) in accuracy where 130 tokens are merged for every transformer block, while achieving similar throughput. More specifically, SAMPL improves the top-1 accuracy on Kinetics-400 by around 4.7% compared to ToMe (r=130). Finally, we would like to emphasize that, for ToMe, we cannot further increase the number of tokens to be merged to achieve even higher throughputs because it will run out of tokens. However, SAMPL can push patch dropping ratio to 70% and 90% to improve the throughput further.

Comparing with patch sampling methods, with the same patch masking ratio, **SAMPL**(Attn) achieves higher accuracy than Random and K-Center in most cases. Moreover, as in the results on images, the performance gap is larger with higher masking ratios because patch selection will be even more important in those cases. For instance, with 90% masking, **SAMPL**(Attn) improves the top-1 accuracy over K-Center by 2.26% and 1.8% on SSv2 and Kinetics-400, respectively. Notably, even with 70% masking, **SAMPL**(Attn) still shows strong action recognition results with marginal performance drop when compared with using all the patches.

For throughput, K-Center and **SAMPL**(Attn) perform similarly while Random has the highest throughput because there is almost no computation involved for random sampling. Moreover, all the sampling approaches show significant improvements over the baseline of using all patches. For instance, the throughputs of the sampling based approaches are more than 2× and 3× the case of using all patches when the masking ratio is 50% and 70%, respectively.

Furthermore, we show the results comparing with SOTA approaches for action recognition on Kinetics-400 in Table 3. As in the table, **SAMPL**(Attn) with 40% mask ratio reduces the GFLOPS by 31% without sacrificing the classification accuracy when compared with VideoMAE (ViT-B) [34]. Overall, **SAMPL**(Attn) achieves strong model performance with reduced computation.

In summary, the results on both images and videos reaffirm there is a considerable amount of redundancy in images and videos as discussed in prior works [12, 34]. Moreover, the attention weights in ViTs are informative supervising signals for patch selection. Finally, SAMPL can leverage those facts during inference time by dropping uninformative patches while maintaining competitive classification accuracy on both images and videos.

Adapting to Various Masking Ratios. To show whether SAMPL works in a zero-shot fashion with various masking ratios, we train

Table 1: Results for image classification when finetuning the pre-trained MAE ViT-B classification model (on the left) and DINO model (on the right) on ImageNet-1K. No dropping refers to the performance from using all patches without additional training. * indicates that K-Center [27] and Perturbed Top-K [27] results are based on our re-implementation with all the same training settings. Note that, for both models, the throughput will be the same for each patch dropping ratio as they share the same ViT-B architecture.

Method	% Patches Dropped	Throughput (images/s)	ImageNet-1K Top-1	ImageNet-1K Top-5
MAE (ViT-B) [17]	0	430.76	83.72	96.53
ToMe (r=7) [3]	0	529.18	**83.02**	**96.24**
SAMPL (Attn)	30	**536.16**	82.52	95.94
ToMe (r=13) [3]	0	694.63	81.50	95.51
SAMPL (Attn)	50	**760.78**	**81.86**	**95.59**
Perturbed Top-K* [6]	70	1088.63	60.80	82.50
K-Center* [27]	70	1382.15	78.60	93.97
SAMPL (Random)	70	**1428.89**	77.26	93.28
SAMPL (Attn)	70	1128.39	**79.69**	**94.42**
Perturbed Top-K* [6]	90	2094.11	43.96	67.45
K-Center* [27]	90	3385.35	66.16	85.74
SAMPL (Random)	90	**3597.63**	63.35	84.50
SAMPL (Attn)	90	2147.86	**70.29**	**88.70**

(a) MAE

Method	% Patches Dropped	ImageNet-1K Top-1	ImageNet-1K Top-5
DINO (ViT-B) [4]	0	81.73	95.63
ToMe (r=7) [3]	0	**81.31**	**95.32**
SAMPL (Attn)	30	80.89	95.17
ToMe (r=13) [3]	0	**79.72**	94.50
SAMPL (Attn)	50	79.56	**94.61**
Perturbed Top-K* [6]	70	56.90	79.11
K-Center* [27]	70	76.63	92.91
SAMPL (Random)	70	74.37	91.96
SAMPL (Attn)	70	**76.73**	**92.92**
Perturbed Top-K* [6]	90	38.95	62.37
K-Center* [27]	90	62.32	82.70
SAMPL (Random)	90	57.34	79.92
SAMPL (Attn)	90	**63.13**	**83.96**

(b) DINO

Table 2: Results for video action recognition when finetuning the VideoMAE ViT-B classification model on UCF101, SSv2, and Kinetics-400. No patch dropping refers to the performance from using all patches without additional training. Throughput is measured on a single A100 GPU. * indicates that K-Center [27] results are based on our re-implementation with all the same training settings as SAMPL.

Method	% Patches Dropped	Throughput (videos/s)	UCF101 Top-1	UCF101 Top-5	SSv2 Top-1	SSv2 Top-5	Kinetics-400 Top-1	Kinetics-400 Top-5
VideoMAE (ViT-B) [34]	0	33.95	91.25	98.55	69.56	91.86	80.14	94.68
ToMe (r=65) [3]	0	47.45	91.41	98.41	**69.42**	91.74	**80.12**	**94.58**
SAMPL (Attn)	40	50.41	**91.81**	**98.65**	69.23	**91.77**	80.07	94.55
ToMe (r=130) [3]	0	69.90	90.85	98.23	66.85	89.41	75.18	89.84
K-Center* [27]	50	63.60	91.44	98.57	68.96	91.61	**80.05**	94.46
SAMPL (Random)	50	**79.34**	91.22	98.60	68.86	91.48	79.65	94.21
SAMPL (Attn)	50	67.93	**91.65**	**98.70**	**69.23**	**91.64**	79.89	**94.49**
K-Center* [27]	70	104.87	90.64	98.23	68.09	91.03	79.20	93.86
SAMPL (Random)	70	**133.05**	90.25	**98.33**	67.76	90.81	78.61	93.71
SAMPL (Attn)	70	114.81	**90.99**	98.31	**68.60**	**91.27**	**79.20**	**94.03**
K-Center* [27]	90	234.71	88.00	97.81	62.99	87.63	74.24	91.16
SAMPL (Random)	90	**291.53**	87.63	97.67	63.03	87.49	74.40	91.45
SAMPL (Attn)	90	228.21	**89.61**	**98.02**	**65.25**	**89.10**	**76.04**	**92.15**

SAMPL(Attn) with the supervision of the attention weights from the DINO backbone with zero patch masking. In other words, we still learn the patch selection probabilities, but feed all the patches (without sampling) to the DINO backbone. In this training setting, we will obtain a DINO-based classifier that takes all patches as inputs where only the classifier head was trained ("trained classifier") and a trained patch sampler ("trained sampler"). Note that the DINO backbone is frozen in this experiment.

Table 3: Comparison with SOTA approaches for action recognition on Kinetics-400. We calculate total GFLOPS with the multiplication of the inference cost for a single view and the number of views (# of segments × # of crops). The image size is 224. The masking ratio for SAMPL is 40%.

Method	GFLOPS	Views	Top-1	Top-5
TimeSformer[2]	7140	1×3	80.7	94.7
TimeSformer + ATS [10]	3510	1×3	80.5	94.6
TSM R50 [22]	650	1 ×10	74.7	91.2
I3D [39]	10800	10× 3	77.7	93.3
SlowFast (R101) [13]	7020	10×3	79.8	93.9
MViT-B [9]	4095	3×3	81.2	95.1
X3D-XXL [11]	5823	10×3	80.4	94.6
TokenLearner (L/16) [31]	4596	4 × 3	82.1	-
VideoMAE (ViT-B) [34]	2025	5×3	80.1	94.6
VideoMAE + **SAMPL** (Attn)	1395	5×3	80.1	94.6

Table 4: Comparison of patch sampling methods on image classification when freezing the DINO backbone and only finetuning the classifier head (same architecture as ViT-B) on ImageNet-1K. Zero-Shot means that the patch sampler is not finetuned for a specific masking ratio. No patch dropping refers to the performance from using all patches without additional training. * indicates that K-Center [27] results are based on our re-implementation with all the same training settings as SAMPL.

Method	% Patches Dropped	Zero-Shot	ImageNet-1K (Top-1)
DINO [4]	0	-	77.3
K-Center* [27]	30	✓	76.3
SAMPL (Random)	30	✓	75.7
SAMPL (Attn)	30	✓	**76.6**
SAMPL (Attn)	30	✗	76.5
K-Center* [27]	50	✓	73.6
SAMPL (Random)	50	✓	72.7
SAMPL (Attn)	50	✓	74.5
SAMPL (Attn)	50	✗	**74.8**
K-Center* [27]	70	✓	66.0
SAMPL (Random)	70	✓	64.5
SAMPL (Attn)	70	✓	68.6
SAMPL (Attn)	70	✗	**70.1**
K-Center* [27]	90	✓	41.1
SAMPL (Random)	90	✓	39.1
SAMPL (Attn)	90	✓	46.4
SAMPL (Attn)	90	✗	**51.1**

In Table 4, we show the performance of K-Center, **SAMPL**(Random) and **SAMPL**(Attn) with several masking ratios in the zero-shot setting where we select the patches based on the "trained sampler" for the "trained classifier". As in the results, with the zero-shot setting, **SAMPL**(Attn) outperforms K-Center and **SAMPL**(Random) on all the masking ratios with more performance gains at higher masking ratios. For instance, with 90% masking, **SAMPL**(Attn) improves the top-1 accuracy of K-Center by more than 5%. Moreover, we also show the performance of **SAMPL**(Attn) when we train the sampler along with the classifier head for each specific masking ratio. We can see that the zero-shot performance of **SAMPL**(Attn) is comparable to the finetuned cases on 30% and 50% masking. With 70% and 90% masking, finetuning the sampler together with the classifier head achieves the best performance with improved top-1 of 1.5% and 4.6%, respectively.

Table 5: Ablation study on computing the attention reward on UCF101, SSv2 and Kinetics-400 video action recognition. 'Layers' refers to which layers' attention weights are aggregated. 'Scale by |V|' indicates multiplying the attention weights by the norm of the the values. The patch masking ratio is 50% for UCF101 and Kinetics-400, and 70% for SSv2.

		UCF101		SSv2		Kinetics-400	
Layers	Scale by |V|	Top-1	Top-5	Top-1	Top-5	Top-1	Top-5
Only 12^{th}	✗	91.33	98.47	68.31	91.14	79.84	94.36
9^{th} to 12^{th}	✗	**91.78**	**98.73**	68.47	91.18	79.88	94.36
9^{th} to 12^{th}	✓	91.65	98.70	**68.60**	**91.27**	**79.89**	**94.49**

4.2.2 *Ablation Studies.*

Attention-based Rewards. In Table 5, we show the results for additional ablation studies on the computation of the attention reward. The two design options we ablate are (i) the layers used to extract the attention weights and (ii) whether we should scale the attention weights by the norm of the values (see Section 3.2 for more details). In the results, we can see that, for UCF101, SSv2 and Kinetics-400, aggregating the attention weights from the last four layers works better than just using the attention weights from the last layer. This aligns with the observations reported in prior work [24]. When scaling the attention weights with the norm of the values, there is a slight degradation in accuracy on UCF101. However, this feature is able to improve the classification accuracy on larger datasets such as SSv2 and Kinetics-400. This is because both the attention weights and the scale of the values indicate the importance of the patch tokens. Since using the last four layers and scaling the attention weights by the value norms provides the best results on SSv2 and Kinetics-400, we use that as our default setting for all experiments unless otherwise specified.

Table 6: Computational costs for different components in the lightweight patch sampler. Metrics are collected on one 40GB A100 GPU with a batch size of 1. Note that "Total" is for all the modules in the one-layer transformer, which is not limited to the three key components listed in this table.

Component	GFLOPS	Parameters (M)	Latency (ms)
Total	11.1	7.09	3.31
Attention Layer	3.7	2.36	1.52
MLP	7.4	4.72	1.21
Linear Layer	0.0012	769 x 1e-6	0.12565

Runtime Efficiency of SAMPL. The patch sampler needs to be lightweight so that we can reduce the overall computational overhead through patch sampling. As we can see in Table 1 and Table 2, SAMPL shows similar throughputs when compared with

Table 7: Ablation study on the MLP ratio in the token sampler. A MLP ratio of 1 is more lightweight and results in higher throughputs. However, a MLP ratio of 4 has better performance in most cases, especially for larger datasets.

MLP Ratio	Dataset	% Dropped	Items/s	Top-1	Top-5
1.0	UCF101	50%	71.00	**91.86**	**98.76**
4.0	UCF101	50%	67.93	91.78	98.73
1.0	ImageNet	70%	1239.52	**79.71**	**94.48**
4.0	ImageNet	70%	1128.39	79.69	94.42
1.0	SSv2	70%	110.96	68.39	91.17
4.0	SSv2	70%	103.82	**68.60**	**91.27**
1.0	Kinetics-400	70%	110.96	79.16	94.01
4.0	Kinetics-400	70%	103.82	**79.20**	**94.03**

random sampling and K-Center. This is because the patch sampler in SAMPL has a simple architecture which is only comprised of a one-layer transformer encoder to process the patch embeddings and a linear layer after that to generate the logits.

In Table 6, we show the GFLOPS, number of parameters, and latency for the key components in the patch sampler. We can see that the attention layer and the MLP layer contribute the most for all three metrics. Notably, the MLP has around two times the GFLOPS and parameters of the attention layer. Therefore, optimizing the MLP layer can potentially further improve the efficiency of SAMPL.

As discussed, the MLP layers in the patch sampler (*i.e.*, one-layer transformer) contributes the most to the GFLOPS and parameters of the sampler. Therefore, we explore the performance of **SAMPL**(Attn) when using different MLP ratios in the patch sampler in Table 7. Here, if the MLP ratio is M, the input and output dimensions of the two linear layers in the MLP will be (D, $M \times D$) and ($M \times D$, D) where D is the feature dimension. Therefore, reducing the MLP ratio can reduce the network complexity of the patch sampler and increase the throughput. This is confirmed in Table 7, where we can see that a smaller MLP ratio does improve the throughput in all cases. Regarding classification accuracy, in some of the settings such as UCF101 with 50% masking and ImageNet with 70% masking, the simpler architecture actually improves accuracy as well. However, for larger video datasets, a MLP ratio of 4 still works the best. We hypothesize that this is due to the larger datasets containing a greater diversity of videos and thus benefiting from increased capacity in the sampler network. Thus, the MLP ratio is set to 4 by default, unless otherwise specified.

Effect of Batch Size and MLP Ratio. In Table 8, we show how throughput changes with different MLP ratios and batch sizes. Throughput is measured on one 40GB A100 GPU for 40 runs, 10 of which are warm-up runs. First, we observe that for all approaches the throughput increases with the batch size. Second, when compared with K-Center, **SAMPL**(Attn) has higher throughput for smaller batch sizes and slightly lower throughput for a batch size of 8. Lastly, regarding the MLP ratio, we can see that the throughput is higher with a smaller MLP ratio for SAMPL in most cases because the policy network will be less computationally expensive.

Softmax Temperature in the Patch Sampler. In the patch sampler (as in Figure 2), we have a softmax function to generate the probabilistic distribution of the sampling actions. To show how the temperature parameter in the softmax function affects the performance, we run an ablation study on SSv2 dataset with 90% masking. The results are shown in Table 9. As in the table, we show the top-1 and top-5 accuracy with the temperature ranging from 1.0 to 0.001. Moreover, we also evaluate the performance when the temperature decreases from 5.0 to 0.5 through an exponential decay scheduler. In the results, **SAMPL**(Attn) works well for the first two cases and the last case. For simplicity, the temperature was set to be 1.0 by default, unless stated otherwise.

Table 8: Ablation study on the effect of batch size and MLP ratio on throughput for different sampling methods. The results are collected on UCF101 with a masking ratio of 90%. * indicates that K-Center [27] results are based on our re-implementation with all the same training settings as SAMPL.

Method	Batch Size	MLP Ratio	Throughput (videos/s)
K-Center* [27]	1	-	88.22
SAMPL (Attn)	1	1	116.49
SAMPL (Attn)	1	4	**118.71**
K-Center* [27]	2	-	141.16
SAMPL (Attn)	2	1	**173.36**
SAMPL (Attn)	2	4	150.06
K-Center* [27]	4	-	186.30
SAMPL (Attn)	4	1	**196.01**
SAMPL (Attn)	4	4	174.00
K-Center* [27]	8	-	**234.71**
SAMPL (Attn)	8	1	213.41
SAMPL (Attn)	8	4	188.37

Table 9: Ablation study on the temperature parameter in the softmax function from the patch sampler. The dataset is SSv2 and the masking ratio is 90%. For the last row, the temperature changed from 5.0 to 0.5 through an exponential decay scheduler.

Temperature	Top-1	Top-5
1.0	65.25	**89.10**
0.1	**65.31**	89.07
0.01	62.38	86.77
0.001	63.17	87.51
$5.0 \mapsto 0.5$	65.05	88.95

Number of Heads in the Patch Sampler. In the one-layer transformer from the patch sampler, besides the MLP ratio, the number of attention heads will also affect the computational overhead and the model performance. We conduct the ablation studies on SSv2 dataset with 90% masking and the results are shown in Table 10. In this table, we can see that, the performance generally improves with the number of heads, with the best performance achieved with eight attention heads. This is because more attention heads will

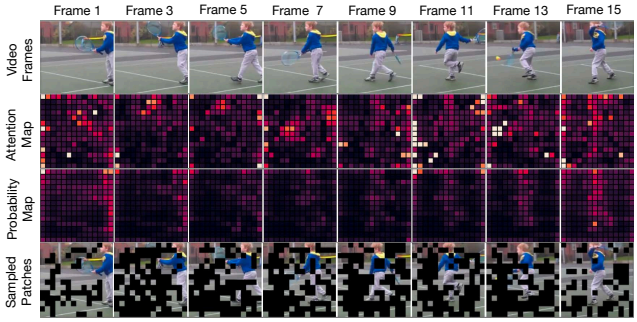
increase the network capacity in the patch sampler, thus resulting in better performance overall. The number of heads is set to be eight by default in all the experiments, unless stated otherwise.

Table 10: Ablation study on the number of heads in the patch sampler. The dataset is SSv2 and the masking ratio is 90%.

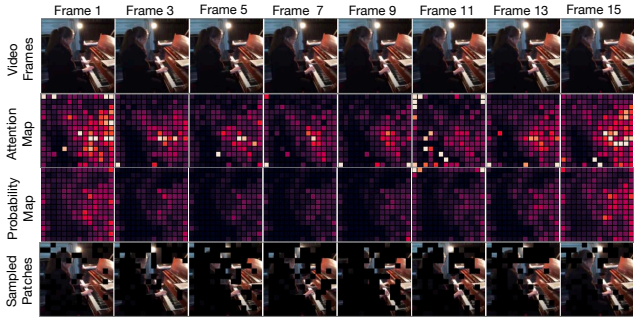
# of heads	Top-1	Top-5
1.0	64.92	88.83
2.0	65.04	89.10
4.0	65.04	89.08
8.0	**65.25**	**89.10**

4.2.3 Qualitative Results. In this section, we show more examples of SAMPL on a few videos from Kinetics-400 in Figure 4. For all the examples, we show the aggregated attention weights from the last four layers of the ViT encoder, patch sampling probabilities, and the sampled patches. More specifically, in Figure 4a, we show an example video of a person playing tennis. The attention map highlights both the person and tennis racket. From the probability map learned by the sampler, we can see that the patch sampler also learns to attend to the person. Based on the sampled patches in the last row, important regions containing both the person and the tennis racket are kept for further processing.

Similarly in Figure 4b, the attention map can accurately locate the important regions for the action, which are the hand and the keyboard of the piano. Moreover, the patch sampler has learned to focus on those regions as shown in the probability map. Lastly, the sampled patches in most frames are mainly comprised of the patches relating to the hand and the keyboard.



(a) Action Label: Playing Tennis



(b) Action Label: Playing Piano

Figure 4: Additional example videos with their corresponding aggregated attention weights (from the last four layers of the ViT model), patch sampling probabilities, and applied masking. Images are from the video [6 year old playing tennis - Forehands and backhands off the correct foot] by [Tom Barrett] and the video [Chatsworth House - Jasmine Cygan Plays Piano] by [Da Vinci Academy] in Kinetics-400 [19] licensed under CC BY 3.0.

5 Conclusion

We study the problem of patch selection to improve the inference efficiency for transformer-based image and video classification tasks. We observe that attention weights in trained transformer models can serve as fine-grained patch-level supervision for patch selection. To this end, we propose SAMPL, a policy network based patch sampler that can be learned through the guidance of patch-level supervision in an end-to-end fashion. In our results, we show that transformer-based models are resilient to patch drops and SAMPL can leverage this fact to greatly improve inference efficiency. SAMPL can potentially be extended to other tasks such as edge-cloud symbiosis and multimodal large language models where an efficient visual encoder is greatly needed. We believe our work can enable transformer-based models to become more popular on consumer electronics hardware, which are generally resource constrained.

Acknowledgement

We would like to thank anonymous reviewers and our shepherd Arvind Narayanan for their constructive comments and suggestions, which greatly improved this work. We are grateful to Samrudhdhi Rangrej and Tristan Aumentado-Armstrong, whose insights and discussions shaped this research.

References

[1] Wele Gedara Chaminda Bandara, Naman Patel, Ali Gholami, Mehdi Nikkhah, Motilal Agrawal, and Vishal M Patel. 2023. AdaMAE: Adaptive Masking for Efficient Spatiotemporal Learning with Masked Autoencoders. In *CVPR*. 14507–14517.

[2] Gedas Bertasius, Heng Wang, and Lorenzo Torresani. 2021. Is Space-Time Attention All You Need for Video Understanding?. In *ICML*.

[3] Daniel Bolya, Cheng-Yang Fu, Xiaoliang Dai, Peizhao Zhang, Christoph Feichtenhofer, and Judy Hoffman. 2022. Token Merging: Your ViT but Faster. *arXiv preprint arXiv:2210.09461* (2022).

[4] Mathilde Caron, Hugo Touvron, Ishan Misra, Hervé Jégou, Julien Mairal, Piotr Bojanowski, and Armand Joulin. 2021. Emerging Properties in Self-Supervised Vision Transformers. In *ICCV*. 9650–9660.

[5] Rohan Choudhury, Guanglei Zhu, Sihan Liu, Koichiro Niinuma, Kris M Kitani, and László Jeni. 2024. Don't Look Twice: Faster Video Transformers with Run-Length Tokenization.

[6] Jean-Baptiste Cordonnier, Aravindh Mahendran, Alexey Dosovitskiy, Dirk Weissenborn, Jakob Uszkoreit, and Thomas Unterthiner. 2021. Differentiable Patch Selection for Image Recognition. In *CVPR*. 2351–2360.

[7] Jia Deng, Wei Dong, Richard Socher, Li-Jia Li, Kai Li, and Li Fei-Fei. 2009. ImageNet: A Large-Scale Hierarchical Image Database. In *CVPR*. IEEE, 248–255.

[8] Alexey Dosovitskiy, Lucas Beyer, Alexander Kolesnikov, Dirk Weissenborn, Xiaohua Zhai, Thomas Unterthiner, Mostafa Dehghani, Matthias Minderer, Georg Heigold, Sylvain Gelly, et al. 2020. An Image is Worth 16x16 Words: Transformers for Image Recognition at Scale. *arXiv preprint arXiv:2010.11929* (2020).

[9] Haoqi Fan, Bo Xiong, Karttikeya Mangalam, Yanghao Li, Zhicheng Yan, Jitendra Malik, and Christoph Feichtenhofer. 2021. Multiscale Vision Transformers. In *ICCV*. 6824–6835.

[10] Mohsen Fayyaz, Soroush Abbasi Koohpayegani, Farnoush Rezaei Jafari, Sunando Sengupta, Hamid Reza Vaezi Joze, Eric Sommerlade, Hamed Pirsiavash, and Jürgen Gall. 2022. Adaptive Token Sampling for Efficient Vision Transformers. In *ECCV*. Springer, 396–414.

[11] Christoph Feichtenhofer. 2020. X3D: Expanding Architectures for Efficient Video Recognition. In *CVPR*. 203–213.

[12] Christoph Feichtenhofer, Haoqi Fan, Yanghao Li, and Kaiming He. 2022. Masked Autoencoders as Spatiotemporal Learners. *NeurIPS* 35 (2022), 35946–35958.

[13] Christoph Feichtenhofer, Haoqi Fan, Jitendra Malik, and Kaiming He. 2019. Slowfast Networks for Video Recognition. In *ICCV*. 6202–6211.

[14] Teofilo F Gonzalez. 1985. Clustering to Minimize the Maximum Intercluster Distance. *Theoretical Computer Science* 38 (1985), 293–306.

[15] Raghav Goyal, Samira Ebrahimi Kahou, Vincent Michalski, Joanna Materzynska, Susanne Westphal, Heuna Kim, Valentin Haenel, Ingo Fruend, Peter Yianilos, Moritz Mueller-Freitag, et al. 2017. The "Something Something" Video Database for Learning and Evaluating Visual Common Sense. In *CVPR*. 5842–5850.

[16] Sariel Har-Peled. 2011. *Geometric Approximation Algorithms*. Number 173. American Mathematical Soc.

[17] Kaiming He, Xinlei Chen, Saining Xie, Yanghao Li, Piotr Dollár, and Ross Girshick. 2022. Masked Autoencoders are Scalable Vision Learners. In *CVPR*. 16000–16009.

[18] Angelos Katharopoulos and François Fleuret. 2019. Processing Megapixel Images with Deep Attention-Sampling Models. In *ICML*. 3282–3291.

[19] Will Kay, Joao Carreira, Karen Simonyan, Brian Zhang, Chloe Hillier, Sudheendra Vijayanarasimhan, Fabio Viola, Tim Green, Trevor Back, Paul Natsev, et al. 2017. The Kinetics Human Action Video Dataset. *arXiv preprint arXiv:1705.06950* (2017).

[20] Solomon Kullback and Richard A Leibler. 1951. On Information and Sufficiency. *The annals of mathematical statistics* 22, 1 (1951), 79–86.

[21] Youwei Liang, GE Chongjian, Zhan Tong, Yibing Song, Jue Wang, and Pengtao Xie. 2021. EViT: Expediting Vision Transformers via Token Reorganizations. In *ICLR*.

[22] Ji Lin, Chuang Gan, and Song Han. 2019. TSM: Temporal Shift Module for Efficient Video Understanding. In *ICCV*. 7083–7093.

[23] Dmitrii Marin, Jen-Hao Rick Chang, Anurag Ranjan, Anish Prabhu, Mohammad Rastegari, and Oncel Tuzel. 2021. Token Pooling in Vision Transformers. *arXiv preprint arXiv:2110.03860* (2021).

[24] Muhammad Muzammal Naseer, Kanchana Ranasinghe, Salman H Khan, Munawar Hayat, Fahad Shahbaz Khan, and Ming-Hsuan Yang. 2021. Intriguing Properties of Vision Transformers. *NeurIPS* 34 (2021), 23296–23308.

[25] Bowen Pan, Rameswar Panda, Yifan Jiang, Zhangyang Wang, Rogerio Feris, and Aude Oliva. 2021. IA-RED^2: Interpretability-Aware Redundancy Reduction for Vision Transformers. *NeurIPS* 34 (2021), 24898–24911.

[26] Zizheng Pan, Bohan Zhuang, Jing Liu, Haoyu He, and Jianfei Cai. 2021. Scalable Vision Transformers with Hierarchical Pooling. In *ICCV*. 377–386.

[27] Seong Hyeon Park, Jihoon Tack, Byeongho Heo, Jung-Woo Ha, and Jinwoo Shin. 2022. K-Centered Patch Sampling for Efficient Video Recognition. In *ECCV*. Springer, 160–176.

[28] Zhiwu Qing, Shiwei Zhang, Ziyuan Huang, Xiang Wang, Yuehuan Wang, Yiliang Lv, Changxin Gao, and Nong Sang. 2023. Mar: Masked Autoencoders for Efficient Action Recognition. *IEEE Transactions on Multimedia* (2023).

[29] Yongming Rao, Wenliang Zhao, Benlin Liu, Jiwen Lu, Jie Zhou, and Cho-Jui Hsieh. 2021. DynamicViT: Efficient Vision Transformers with Dynamic Token Sparsification. *NeurIPS* 34 (2021), 13937–13949.

[30] Adria Recasens, Petr Kellnhofer, Simon Stent, Wojciech Matusik, and Antonio Torralba. 2018. Learning to Zoom: a Saliency-Based Sampling Layer for Neural Networks. In *ECCV*. 51–66.

[31] Michael S Ryoo, AJ Piergiovanni, Anurag Arnab, Mostafa Dehghani, and Anelia Angelova. 2021. Tokenlearner: What Can 8 Learned Tokens Do for Images and Videos? *arXiv preprint arXiv:2106.11297* (2021).

[32] Khurram Soomro, Amir Roshan Zamir, and Mubarak Shah. 2012. UCF101: A Dataset of 101 Human Actions Classes from Videos in the Wild. *arXiv preprint arXiv:1212.0402* (2012).

[33] Yehui Tang, Kai Han, Yunhe Wang, Chang Xu, Jianyuan Guo, Chao Xu, and Dacheng Tao. 2022. Patch Slimming for Efficient Vision Transformers. In *CVPR*. 12165–12174.

[34] Zhan Tong, Yibing Song, Jue Wang, and Limin Wang. 2022. VideoMAE: Masked Autoencoders are Data-Efficient Learners for Self-Supervised Video Pre-training. *NeurIPS* 35 (2022), 10078–10093.

[35] Hugo Touvron, Matthieu Cord, Matthijs Douze, Francisco Massa, Alexandre Sablayrolles, and Herve Jegou. 2021. Training Data-Efficient Image Transformers Distillation Through Attention. In *ICML*, Vol. 139. 10347–10357.

[36] Burak Uzkent and Stefano Ermon. 2020. Learning When and Where to Zoom with Deep Reinforcement Learning. In *CVPR*. 12345–12354.

[37] Ashish Vaswani, Noam Shazeer, Niki Parmar, Jakob Uszkoreit, Llion Jones, Aidan N Gomez, Łukasz Kaiser, and Illia Polosukhin. 2017. Attention is All You Need. *NeurIPS* 30 (2017).

[38] Junke Wang, Xitong Yang, Hengduo Li, Li Liu, Zuxuan Wu, and Yu-Gang Jiang. 2022. Efficient Video Transformers with Spatial-Temporal Token Selection. In *ECCV*. Springer, 69–86.

[39] Xiaolong Wang, Ross Girshick, Abhinav Gupta, and Kaiming He. 2018. Non-Local Neural Networks. In *CVPR*. 7794–7803.

[40] Ronald J Williams. 1992. Simple Statistical Gradient-Following Algorithms for Connectionist Reinforcement Learning. *Machine learning* 8 (1992), 229–256.

[41] Yifan Xu, Zhijie Zhang, Mengdan Zhang, Kekai Sheng, Ke Li, Weiming Dong, Liqing Zhang, Changsheng Xu, and Xing Sun. 2022. Evo-ViT: Slow-Fast Token Evolution for Dynamic Vision transformer. In *AAAI*, Vol. 36. 2964–2972.

[42] Hongxu Yin, Arash Vahdat, Jose M Alvarez, Arun Mallya, Jan Kautz, and Pavlo Molchanov. 2022. A-ViT: Adaptive Tokens for Efficient Vision Transformer. In *CVPR*. 10809–10818.

[43] Xiaoyu Yue, Shuyang Sun, Zhanghui Kuang, Meng Wei, Philip HS Torr, Wayne Zhang, and Dahua Lin. 2021. Vision Transformer with Progressive Sampling. In *ICCV*. 387–396.

[44] Zhuofan Zong, Kunchang Li, Guanglu Song, Yali Wang, Yu Qiao, Biao Leng, and Yu Liu. 2022. Self-Slimmed Vision Transformer. In *ECCV*. Springer, 432–448.

Enablers of low-latency immersive interaction in future remote-rendered Mixed Reality applications

János Dóka[1,2], Bálint György Nagy[1,2], Dávid Jocha[3], Bence Formanek[3], Iván Viciedo[4], Adrian Rodrigo[4], David Gomez-Barquero[4], Balázs Sonkoly[1,2*]
[1]HSN Lab, Department of Telecommunications and Artificial Intelligence, Faculty of Electrical Engineering and Informatics, Budapest University of Technology and Economics, Hungary,
[2]HUN-REN-BME Cloud Applications Research Group, Hungary, [3]Ericsson Research, Hungary,
[4]Institute of Telecommunications and Multimedia Applications, Universitat Politecnica de Valencia, Spain

Abstract

Mixed Reality (MR) has launched from science fiction but its entrance in reality can reshape our society. By combining the physical and virtual worlds, it provides novel ways of immersive interactions and experiences and by these means enables a new generation of applications. The most exciting and challenging ones support collaborative, multi-user operation in large geographical scale, require real-time environment comprehension and high visual fidelity. The success or failure is definitely impacted by the capabilities and performance limits of edge cloud platforms and 5G/6G networks providing the offloading features for CPU/GPU intensive MR functions. In addition, the desired quality of user experience calls for further mechanisms at the application level hiding the consequences of varying network characteristics. In this paper, we propose a novel edge cloud based architecture for future remote-rendered MR applications supporting low-latency immersive interactions. Our contribution is threefold. First, the system architecture is presented focusing on the remote rendering, 3D simulation and environment detection control loops. Second, we highlight the main features of our proof-of-concept prototype and our dedicated application, namely the Mixed Reality version of a Rocket League inspired game. Third, the concepts are validated via experiments in a Beyond 5G infrastructure where we analyze the operation and latency characteristics of the overall system. In addition, the quality of the user experience is also evaluated via real-life experiments conducted as part of a student competition. The results show that the latency and jitter characteristics of the most sensitive render loop can be managed efficiently together by a network-level control (slice priorities) and an application-level (dynamic jitter buffer) mechanism.

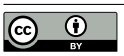
CCS Concepts

• **Human-centered computing → Mixed / augmented reality**.

Keywords

Human computer interaction (HCI), Mixed Reality, Augmented Reality, immersive applications, edge computing, environment understanding, 5G/6G

*János Dóka and Bálint György Nagy contributed equally to this research.



This work is licensed under a Creative Commons Attribution 4.0 International License.
MMSys '25, Stellenbosch, South Africa
© 2025 Copyright held by the owner/author(s).
ACM ISBN 979-8-4007-1467-2/25/03
https://doi.org/10.1145/3712676.3714448

ACM Reference Format:
János Dóka, Bálint György Nagy, Dávid Jocha, Bence Formanek, Iván Viciedo, Adrian Rodrigo, David Gomez-Barquero, Balázs Sonkoly. 2025. Enablers of low-latency immersive interaction in future remote-rendered Mixed Reality applications. In *ACM Multimedia Systems Conference 2025 (MMSys '25), March 31-April 4, 2025, Stellenbosch, South Africa.* ACM, New York, NY, USA, 11 pages. https://doi.org/10.1145/3712676.3714448

1 Introduction

Mixed Reality (MR) is the youngest member of the Extended Reality (XR) application family. It is a blend of the physical and virtual worlds, creating immersive environments through the use of XR glasses where virtual elements interact with the real world in real-time. These capabilities open the door to a new generation of XR applications in many fields with special societal importance and business potential. Medical/healthcare [1], Industry 4.0/5.0 [9], entertainment and online gaming [22], construction/architecture [5] are just highlighted areas where only low-latency immersive interaction and high visual fidelity can ensure the correct and acceptable operation.

For example, a live, remotely assisted surgery via a head-mounted MR display or a dynamic, personalized billboard attached to a moving vehicle are closer to science fiction today, but might be part of our life tomorrow thanks to the revolutionary MR technologies. The main challenges of these future applications stem from the collaborative, multi-user operation, the resource demand of high visual fidelity and the large-scale environment detection, understanding and comprehension. The latter capability is the key enabler of future MR features. Offloading the CPU/GPU intensive MR functions to the edge cloud infrastructure is a straightforward and inevitable step. For example, by applying remote rendering techniques [11, 13], we can render objects with a much higher polygon count, significantly enhancing visual fidelity. However, the challenging latency, jitter and bandwidth requirements call for novel 5G/6G networking technologies as well in order to keep the different control loops in the appropriate operation regime. But in spite of the promises of future 6G systems, additional techniques are also necessary to overcome the impact of increased latency. For example, recently proposed warp methods are able to distort/adjust the remotely rendered, slightly delayed images in order to follow the changes in headpose or user position (happened after the creation of the frame) just before displaying them. Moreover, sophisticated movement prediction algorithms can mitigate lagging of object tracking functions.

In this paper, we propose a novel edge cloud based architecture for future MR applications enabling low-latency immersive interaction in remote-rendered virtual spaces combined with physical environments. Our contribution is threefold. First, the system architecture is presented focusing on the main control loops, namely the remote rendering, the 3D simulation and the environment detection loops. These control mechanisms show different operation characteristics and require distinct capabilities and performance from the underlying network and edge cloud infrastructure. In addition, they directly affect the user experience. Second, we highlight our proof-of-concept prototype implementing the key elements of the proposed architecture. A special application is also delivered that poses substantial challenges to the system and the underlying infrastructure. It is the Mixed Reality version of a Rocket League [15] (car soccer) inspired game. Third, we analyze the operation and latency characteristics of the overall system in a Beyond 5G (B5G) infrastructure, including network and edge resources. Besides the quantitative validation, the concepts are also evaluated via real-life experiments conducted as part of a student competition where the quality of the user experience is analyzed.

The rest of the paper is organized as follows. In Sec. 2, the main challenges of immersive interaction in MR applications are highlighted and the related state-of-the-art solutions are summarized. Sec. 3 is devoted to our proposed architecture and its key features. A proof-of-concept embodiment of this architecture and a test application are presented in Sec. 4. The measurement environment and the main results of the evaluation are presented in Sec. 5 and Sec. 6, respectively. And finally, Sec. 7 concludes the paper.

2 Immersive interactions: challenges

In this section, we identify and analyze the main challenges of immersive interactions in MR applications and also highlight related solutions which partially address these problems.

2.1 Low-latency interaction

It is evident that the sensitivity to latency varies for different types of interactions in MR applications [2]. Anyone who used e.g. HoloLens 2 noticed that grabbed objects follow the user's hand with a slight delay, but this constant delay does not degrade the user experience. If the interaction is driven by e.g. the number of people in a store, the latency requirement of that detection function is typically not so strict. In contrast, if a virtual ball bounces back with a delay from a physical wall, it distorts the image and impairs the user experience. Similarly, in a ping-pong application, where we hold a controller, our virtual racket must be precisely in the position of that controller. Obviously, in the latter examples, latency is critical and cannot be tolerated. As a consequence, we can distinguish between moderate and the much more challenging low-latency interactions based on how their processing time affects the user experience.

Mixed reality systems are still far from perfectly integrating life-like virtual objects into physical space. There are several reasons for this. One of them is the quality of the rendered image, as current headsets are unable to render high-polygon-count objects in real-time due to lack of sufficient computing capacity, making the appearance of virtual objects less realistic. To address this issue, remote rendering systems have been developed, where the rendering task is offloaded from the devices to powerful servers running in the cloud or at the edge. Such solutions are Nvidia CloudXR [13] and Microsoft Azure Remote Rendering [11]. Although the image quality improves with remote rendering, a delay is added to the system, which raises extra challenges in terms of immersive interactions. While the image is rendered for the current frame, the user can move (even in case of local rendering). This very slight movements can cause that our brain perceives we are not seeing the intended image. Over time, this discrepancy can lead to motion sickness. To mitigate these issues, remote rendering systems often use warp techniques, primarily time warp [24] and space warp [28], to ensure that the streamed image stays in sync with the user's movements and to prevent potential image loss from degrading the user experience. During time warp algorithms, the incoming image is slightly adjusted based on the user's position changes that occur during the operation time of the render loop. However, time warp cannot function effectively if the latency is too high. The utilization of space warp eliminates the need for rendering every frame. Instead, every other frame is generated on the client side by leveraging the previous frame and motion vector data. This technology is particularly beneficial in remote rendering systems, as it optimizes bandwidth usage by allowing interpolated frames to be locally generated on the client, thereby reducing the volume of transmitted data. Furthermore, it enhances the user experience in scenarios with unstable network connections by compensating for missing frames, ensuring smoother visual continuity.

2.2 Static vs. dynamic environments

The level of challenge is greatly impacted by the characteristics of the surrounding physical environment, i.e., whether there are static or dynamically changing objects within the application. If we only want to display a virtual object at a specific point, its placement and rendering depend solely on the device, a task easily accomplished by modern headsets equipped with multiple sensors and cameras. However, when these objects are in motion, perhaps bouncing off physical objects or following them, we are dealing with a considerably more challenging task that significantly burdens the devices. Moreover, if the semantic understanding of the environment is also intended, the task's complexity and its resource demand (e.g. GPU and memory for handling large AI models) will be much higher, making it evident that environment detection and tracking can easily be a bottleneck in MR applications.

Object detection in static environments and combined object detection and tracking are (will be) crucial in modern (future) MR services. However, the devices can only recognize simple surfaces by default. It is possible to run object detection on them, for example, by using the Unity Barracuda solution [23] or by incorporating YOLO [12]. The advantage of these solutions is that they are fast when we only want to detect a few objects, but their disadvantage is that they take a significant amount of resources from devices that are usually resource-poor anyway, and they cannot provide adequate performance when running complex models [8].

Offloading these tasks to cloud/edge servers is a solution that allows complex models as well. However, the introduced extra

latency is a serious drawback to be mitigated with low-latency network protocol [6], or the use of dedicated hardware acceleration in the cloud/edge (e.g.: GPU), which in many cases provides a significant advantage. Another benefit is that we can extract further information from the dispatched image. With the help of innovative AI technologies, we can even determine material properties, which can further improve visual fidelity. For instance, a virtual ball would bounce differently off concrete than off sand. Furthermore, device (which is usually a mobile device with a finite energy supply) energy usage can be reduced by outsourcing compute-heavy tasks [29].

In a dynamic environment, object detection alone is not sufficient, as objects can move, so we also need to apply object tracking. ARKit [3] has developed a solution for this, but currently, it updates the object's position very slowly, more like sequential detections. VisionLib's solution [25], however, is much faster and more dynamic. The disadvantage is that it can only be used with pre-trained objects, and the trained model must be stored on the device. Therefore, it is currently not suitable for automatically recognizing a moving object if it is not included in the installed model.

The situation is further complicated if several objects have to be tracked at the same time. Multiple Object Tracking (MOT) has gained significant attention in recent years, driven by advancements in computer vision and deep learning. Many solutions implement tracking using ORB key points [4, 20, 26, 27]. For example, in [4] a multi-2D object tracking solution is presented and ORB is used to match and associate key points extracted from consecutive RGB frames. The association of the ORB descriptors between the frames was improved with an algorithm that calculates the distance of the key point of the movement speed of the features, which makes it possible to filter out unrealistic deviations, thus verifying the accuracy of the association.

2.3 Multi-user and large-scale applications

In addition, if we assume multi-user systems, where the inputs of environment detection and comprehension come from different users and devices, additional delays are introduced which affect the user experience and require more complex synchronization methods. However, large-scale MR applications need to apply the "more eyes see more" rule to get much more information about the environment in order to enable e.g. global scenes and immersive interactions in that. Multi-user MR applications pose further challenges and it is worth noting that creating them is far from easy. One of the key problems to be solved is the synchronization of coordinate systems. In contrast to multiplayer computer games, where each player roams within a completely virtual world with known origin and clear coordinates of positions, in MR applications, the users move within the real physical world and each of them designates the starting point as their own origin when launching the application. Therefore, if we instruct the program to display an object at specific 3D coordinates, everyone will see it in a different location. Of course, there are several methods for synchronizing the coordinate systems, but none of them are perfect. One of these common techniques is the QR code or image recognition technique, when we place an image or QR code in the physical space that every user sees in the same location, allowing us to use its pose as the origin. The main issue of this solution is that devices may not perfectly recognize the rotation of the images, and even just one degree of deviation away from the origin can cause significant differences in the system. The other common method is based on the matching of feature points, which can be used relatively well in small spaces, for example in a small room. However, since the problem can be traced back to the point cloud registration, if we want to use this technique in a larger space, the synchronization will either be inaccurate or take several seconds [17].

Another option invokes offloading and runs the SLAM (Simultaneous Localization and Mapping) function in the edge cloud. It can be combined with jointly built and managed global maps, which can address the issue of coordinate system synchronization in multiplayer systems [19]. If several devices do this in a cooperative manner, this can also enable an environment interpretation where we get a detailed picture of the part of our environment that we do not have a direct view of. Thus, we can create a model of an object as a whole, even if we only have a view of the front side of it.

In multiplayer Extended Reality (XR) environments, network requirements play a crucial role in maintaining an immersive and synchronized experience. Achieving low-latency, high-bandwidth (in some cases), and reliable connections are essential to avoid lag and desynchronization, which would lead to a drastic decrease in the user experience. Key requirements for multiplayer XR is the low latency (typically below 20 ms) with the low jitter, but in case of remote rendering the high bandwidth (couple of tens of Mbps bitrate) is also important [2, 10]. Meeting these requirements is a huge challenge with today's mobile networks, so in many cases we (will) have to rely on B5G or 6G solutions [7, 21, 30].

3 Proposed architecture

Addressing the highlighted challenges, we have designed an edge cloud based system architecture for future MR applications supporting low-latency immersive interactions together with high visual quality. The proposed architecture is shown in Fig. 1. Central to the concept is the offloading of resource-intensive tasks to the edge cloud while the device executes lighter or delay-critical functions. By the end of the day, three different functionalities and related control loops shall be harmonized, which are denoted by different colors in the figure. The lower part (green) is responsible for object detection and tracking, the center elements (orange) correspond to the 3D simulation, while the upper section (blue) denotes components realizing remote rendering solutions. The main steps of the operation are the following: *i*) 3D environment in the simulation server is updated based on object detection/tracking information, *ii*) pose (position and orientation) of the HMD device is sent to the server, *iii*) a 2D image is rendered by the server based on its 3D view and sent to the device, and finally *iv*) the 2D image is warped and displayed. In this section, the main concepts and components of the architecture are detailed.

3.1 Object centric rendering

In the industry, there are already several remote rendering platforms. However, it can be observed that in each of these, the server sends an image and possibly its depth information to users, including all virtual objects. However, this solution has its drawbacks, particularly in managing movements. For example, if two objects

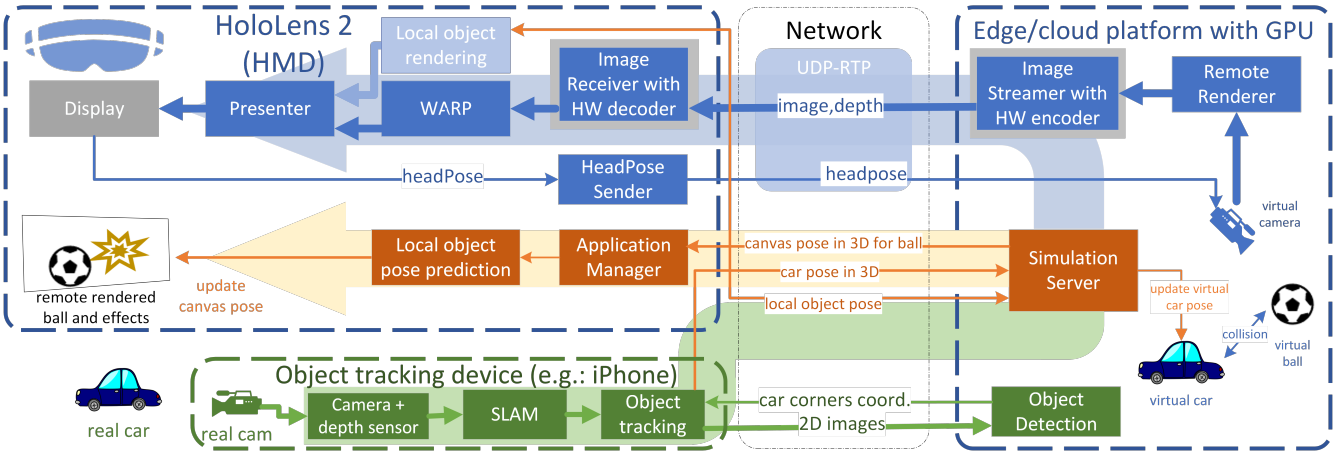


Figure 1: Our multi-user edge/cloud-based Mixed Reality platform

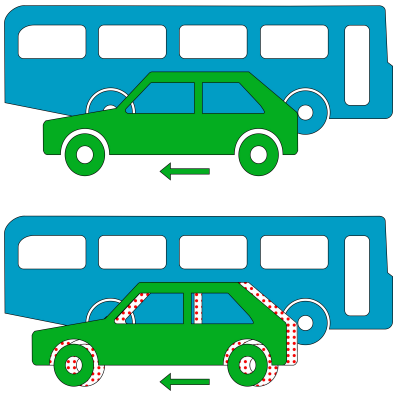


Figure 2: Lack of pixels in case of object movement

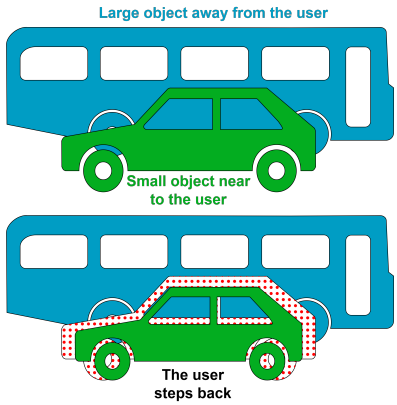


Figure 3: Lack of pixels in case of user movement

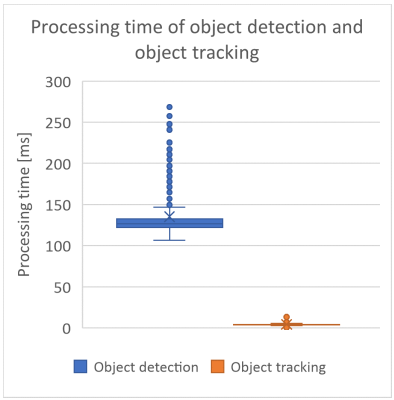


Figure 4: Processing time statistics of object detection and object tracking

move in opposite directions, and for some reason a frame does not arrive in time or is discarded, the system attempts to track what should be displayed to the user based on the information from the previous frames. However, as shown in Fig. 2, in such cases, not all pixels are available. Information is missing from the area marked with red dots in the system. Similarly, in cases where objects are stationary but overlap, a similar phenomenon can occur. For instance, if the user starts moving away from a virtual object in front of them, they should progressively see more parts of the objects behind the first one, but there is no information about these. Fig. 3 illustrates this scenario. These pixel losses not only degrade the user experience but also contribute to motion sickness.

Object centric rendering as defined by us, can provide a solution to these problems. In this case, we don't send only one image to the user about the scene, but we render an image for each object separately and display them on the client in the position of the object on a canvas. Thus, if an image is lost, the previous images contain the entire object, thereby avoiding pixel loss in the system.

In our prototype, only two objects are rendered remotely, with each client receiving two separate streaming channels. Of course, the question arises, if we render a separate image for each object and send the image to the clients separately, won't our traffic increase enormously? Of course, our traffic may increase, but not to a large extent. We have several optimization options. For example, it is not necessary to render all images in high resolution, since distant objects will be placed far away on a canvas anyway. We do not need to start a separate stream for each object, as this would put a significant load on the devices. We can copy several smaller images created by us into the memory area of a large image one after the other, so only one image needs to be encoded on the server and decoded on the client.

In addition, in many cases it is not necessary to render all objects remotely, since several applications use objects with a low polygon count, which the device can also render. Hence, a hybrid or split rendering approach can be applied [16, 18].

3.2 Latency hiding

Since we outsource several components, it's evident that there will be latency in the system, which we must try to conceal to avoid degrading the user experience. The warp itself is a well-established concept in virtual reality, aiming to ensure that the user sees the image corresponding to their current head pose. It is also crucial in MR applications supporting low-latency interactions.

Another important mechanism for hiding latency is prediction. In our system, we assume various movements, such as head motion and the motion of tracked objects, but we need to transmit these to the server and then to the clients. Consequently, the data would be outdated by the time it reaches the users. For instance, a virtual object attached to a moving object would always lag behind its actual motion. To solve this, we apply prediction algorithms in the system at the client and the server side to forecast the position of the moving object at the time of display, based on our current knowledge.

Offloading not only introduces latency into the system, but also jitter. In many cases, this causes a bigger problem than the delay itself, since if we experience very large jumps in the delay on the network, we will also see jumps and outages in the remote rendered stream and in updating object positions. A jitter buffer (which is a temporary storage to compensate for jitter in arrival times caused by network delays) can reduce it at the cost of extra delay. We can achieve a significantly better user experience if there are no frame drops in the system. However, too large jitter buffer can introduce a disturbingly high delay into the system, which also leads to a deterioration of the user experience. Therefore, finding the golden mean in choosing the jitter buffer size in a remote render system is a particular challenge and depends on the characteristics of the network and the specific use-case.

Object-centric rendering enables another latency-hiding technique when each object is handled separately, and the image and position associated with each object are sent via separate streams. It is evident that the rendering loop (blue arrow in Fig. 1), including rendering, encoding, sending, decoding, and displaying the image, is "slower" than the sending of position updates (orange arrow). By leveraging this, we can respond to movements faster than if we were using a fully remote rendering system. We can move the image associated with the object before the updated frames actually arrive, which provides roughly a 20-30 ms advantage, but this number also depends on the size of the jitter buffer. In practical terms, if a real toy car collides with a virtually remote-rendered ball, we can move the ball sooner than the rendered image showing the effect of the collision reaches the device. By these means, we can avoid visual inconsistencies in our use-case, where the car would be partially inside the ball without the ball moving yet.

3.3 Fast and slow paths

As Fig. 1 shows, there are several communication paths in our system, which usually differ in update frequency and message sending speed. In the detection and tracking part (marked by green color), the detection has been separated and only tracking runs locally. There are several reasons for this. First, the two components do not necessarily have to run together, since tracking does not require to know exactly what we are tracking, only the points to be tracked are needed. Second, object detection is a much more resource-intensive and slow process, thus outsourcing of that is quite straightforward. Fig. 4 presents a baseline measurement indicating the significant difference in runtime between the two components. Finally, due to the separation, the training data set of the detection algorithm or the object models can easily be replaced, extended or refined.



Figure 5: The remotely rendered MR car soccer game

As tracking runs separately and it just needs to know which points to track, it is not necessary to continuously send an image to the object detection component. In our case, since we want to track a slowly moving toy car, we have found that sending one or a couple of images every second when the tracker loses the points to be tracked is enough. This route is therefore a slow path. In contrast, the object tracking result directly affects the user experience, so we have to run it with the highest possible frequency and forward the results to the simulation server as quickly as possible (which is 30 Hz in our system).

4 Our proof-of-concept prototype

To test the system components and validate our concept, we developed the outlined system and created a Mixed Reality game. A snapshot of the game is shown in Fig. 5. In this game, which resembles soccer, two users control real remote-controlled cars and use HoloLens 2 headsets to drive virtual balls into virtual goals. The developed multiplayer game supports two players who can compete against each other either in the same physical location or from two different remote locations. The game features pickup objects that, when collected, provide users with goalkeeper monsters as temporary assistance. The positions of the cars are continuously tracked by the environment detection and tracking system, and the data is sent to the game server. The application uses split rendering, so objects that need to appear as realistic as possible with high quality, like the soccer ball and monsters, are rendered remotely, while less important static objects like the goals and field elements are rendered locally. The implementation details of each component are explained in the following subsections.

4.1 Environment detection and tracking

The lower control loop with green boxes in Fig. 1 is the real-time environment detection service, which is split between the tracking

device and the cloud platform. On the one hand, the AI-based object detection is offloaded to the GPU-powered edge cloud and it performs the image classification on the received 2D images streamed from the client devices. On the other hand, the delay-sensitive tracking function is run locally e.g. on an iPhone (mobile AR) device. In our prototype, the tracking device receives updates from the object detector every second, while the tracking runs continuously (at 30 FPS in our setup). During the prototype development, an iPhone was utilized to determine its position using the ARKit library and to synchronize its coordinate system with the HoloLens by recognizing the QR code placed on the track. Subsequently, the iPhone was mounted on a fixed stand, eliminating any movement. This allowed for the deactivation of the SLAM (Simultaneous Localization and Mapping) component, thereby reducing computational load and optimizing the system's runtime efficiency.

We send 1920x1440 resolution images to the object detection server at a rate of one frame per second using UDP streaming. On the server, we use deep learning image classification to determine the positions of the four corners of the toy car in the image. This information is then sent back to the object tracking device, which is an iPhone 13 Pro with LiDAR scanner in our case. The tracker device uses this data along with depth information collected by the LiDAR to calculate the 3D coordinates of the four points and tracks them with OpenCV. In each cycle, the tracker forms a pose data of the car from the updated four positions and transmits this to the simulation server.

4.2 Unity-based object-centric rendering

The server-side part of the upper two control loops was implemented in Unity 3D engine. The first part is the Simulation Server. It performs the physical calculation (such as collision), records the position of the players and also has a picture of the real environment by the green environment detection service. To handle a rapidly changing dynamic environment, the simulation server (regardless of the rendering loop) can process the environment detection information and send updates about the pose of virtual objects based on this, or predict the movements of objects in the game space based on past data. Thus, the simulation is as close to the actual state of reality as possible, and the image displayed on the HMD reacts more quickly to environment changes, which is of paramount importance when the virtual content is in close interaction with the environment (e.g.: a real moving object collides with a virtual ball).

The second loop is responsible for remote rendering (blue components in Fig. 1). This starts with the client device (HoloLens 2 headset in our prototype), which sends the head pose of the user to the server . HoloLens 2 also provides the user's hand-gesture and all the coordinates of their hand, which may be useful for another use-cases, but we are currently only using the head pose. Based on this, we update the position of the virtual cameras associated with the user and does the rendering. The images are delivered to the clients by a native C++ plugin consisting of a hardware based encoder (making use of Nvidia's NVCodec library) and our custom RTP streamer which transports the video streams. Note that, we update the position of the virtual cameras associated with the user, but not their rotation. According to object-centric guidelines the camera on the server side always aims at the virtual object and the content is displayed on the client's canvas even if the user is not looking there. Hence, quick movements and viewport updates can also be handled.

At the client side, we have prepared a HoloLens 2 application with Unity that receives and displays the remote rendered images. A C++ plugin was created to receive and decode the image making use of AVCodec library with hardware support. It decodes the images into YUV format, so we create the final RGB images from the Y and UV layers using a Unity shader. In the application (Presenter module in Fig. 1) , we display the decoded images on dedicated canvas where the effects of latency caused by the remote rendering is mitigated by applying our own warp algorithm, which is an image distortion mechanism before displaying the image and makes it more realistic (for further details, see Sec. 4.3).

4.3 Latency and jitter hiding

As we mentioned before, we can apply a special warp mechanism for latency hiding building on the object-centric rendering concept. When the image of the given object arrives on the clients, we display it to the user on a canvas. However, we do not turn this canvas towards the user's actual head position, but to the position to which the currently displayed image belongs. So we use a head position from the past. Thus, the user's viewing angle differs by a few degree and the user looks at the canvas at a bit flatter angle. The image seen by the user is therefore minimally distorted, however, sudden motions can be followed more smoothly.

Furthermore, the pose data sent to the simulation server is of course outdated data, because no matter how fast the tracker is, it works from past data. For this reason, we use a pose prediction to estimate where the object will be when our data reaches the client devices. This algorithm is currently based on simple curve fitting to historical data and its accuracy depends on the velocity and acceleration of the moving object. This simple method is suitable in context of the prototype application, but more sophisticated (e.g. AI-based) methods can be added later to handle other use-cases as well.

Optimally, the prediction time on the server will be very close to the users display time. Although this is a short amount of time, a fast-moving object can move significantly during this time. Normally this does not cause serious visible errors in the display, but when a virtual object is attached to the moving object, the display of the virtual object may show a visible lag. To mitigate this problem, we also implemented a local prediction algorithm on the clients, which creates a predicated pose for the planned display time based on the received data (Local object pose prediction component in Fig. 1).

If the network conditions require, for example due to background traffic, then it should be necessary to use a dejitter buffer on the client side in order to ensure a continuous stream. We will discuss the importance of this in Sec. 6.1.

5 Measurement environment

This section is devoted to our measurement environment, including the description of the academic Beyond 5G testbed and the deployment of our system.

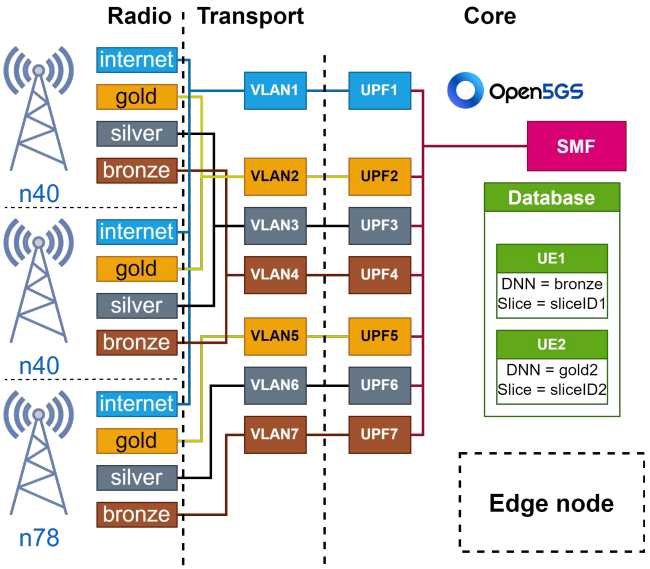


Figure 6: The academic Beyond 5G network infrastructure

5.1 Beyond 5G test infrastructure

The Beyond 5G network (IMAGINE B5G @UPV), shown by Fig. 6, consists of three Radio Access Network (RAN) nodes, an Open5GS core [14], and a router interconnecting them. Two nodes operate in the n40 frequency band, while the third operates in the n78 frequency band. The core runs seven distinct User Plane Functions (UPFs) to serve User Equipment (UE). All UPFs are deployed on the same virtual machine (VM) with 16 GB of RAM and six CPUs. Three UPFs are configured to serve UEs connected to the n40 frequency band, another three serve the n78 frequency band, and the last one is shared among the two bands. The selection of the UPF is determined by the Data Network Name (DNN) chosen by the UE.

Network slicing is an essential feature of 5G/B5G systems where the physical network infrastructure is shared among logical networks comprising all of the required network resources from radio via transport to the core network in order to provide the defined end-to-end service quality for a given set of users/devices. Mapping DNNs to slices (and UPFs) is described in the core database. Each UPF uses a separate network interface (subinterface) to connect to the RAN data plane (N3 interface of the reference architecture), and the traffic separation is provided by VLAN tagging. This network setup enables the implementation of Quality of Service (QoS) through different mechanisms.

Some relevant QoS mechanisms are associated with the 5QI (5G QoS Identifier) parameter. 5QI is configured for each UE at the core level and is linked at the RAN level to the Scheduling Weight and Differentiated Services Code Point (DSCP) parameters, which can directly control the QoS of the UE. On the one hand, the Scheduling Weight influences the gNB (base station) MAC scheduler for uplink and downlink traffic, as well as the multiplexing of Data Radio Bearers (DRBs) for downlink traffic. This ensures that more radio resources are allocated for UEs with a higher scheduling weight. On the other hand, DSCP is an IP header field for traffic labeling. This

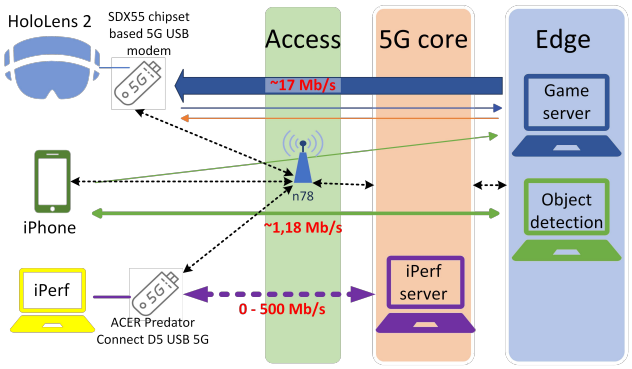


Figure 7: B5G deployment of our MR platform

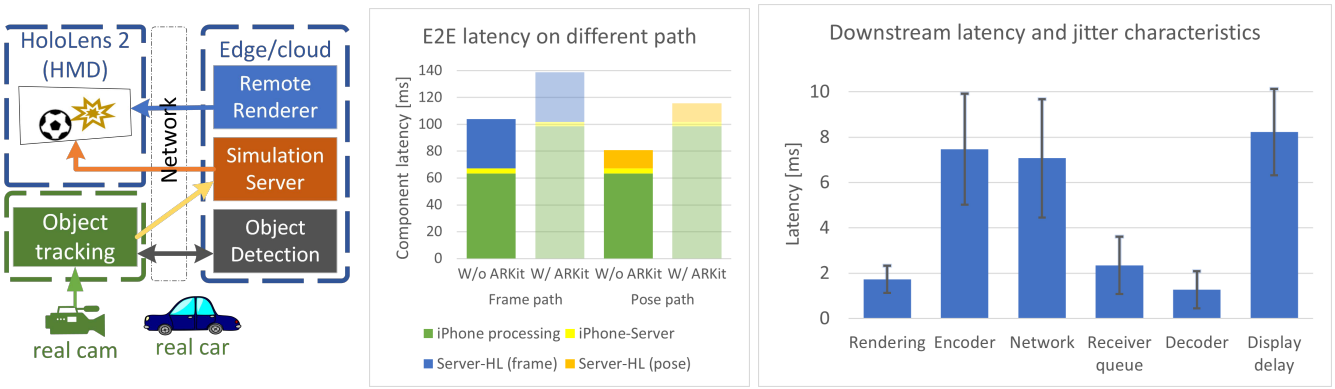
label is tagged at the RAN based on the 5QI parameter. The router between the RAN and the core can read DSCP labels and apply policies accordingly. For instance, the router may set a maximum throughput value for a specific DSCP label, discarding excess traffic. In addition, the latency-sensitive components can be deployed to edge nodes close to the users. In order to provide an interface for configuring these network capabilities, CAPIF (Common API Framework for 3GPP Northbound APIs) was introduced by 3GPP Release 15 and was further evolved in subsequent releases (Beyond 5G). It is a key component providing a standardized way for applications to interact with the network to foster a wide range of innovative applications.

5.2 B5G deployment of the MR platform

Our system has been deployed to this Beyond 5G (B5G) network infrastructure. The integration and the mapping of the components are illustrated by Fig. 7. In this network, components associated with the edge/cloud platform are directly connected to the core network, while the HoloLens HMDs are connected to the access network using a Qualcomm Snapdragon X55 chipset (SDX55) based USB dongle. The iPhone 13 Pro, which is responsible for tracking, is equipped with specific 5G SIM card and connected via given B5G radio channels. Additionally, several traffic generator clients are deployed making use of Acer Predator 5G dongles. By these means, we can generate traffic and congestion in the radio access network, making the measurement scenarios more realistic. During the measurements, the average total traffic generated by the clients was 500 Mbps.

Connecting the HoloLens to the B5G network turned to be challenging. On the one hand, several dongles are not supported by HoloLens and the 5G connection cannot be established. On the other hand, we can opt for combined modem/router devices, such as Advantech ICR-4453 5G router or Siemens Scalance 5G modem. However, we can only use these modems with the HoloLens via WiFi, which introduces significant jitter to the system, thereby skewing the measurement results, or we have to connect the HoloLens through an Ethernet cable, requiring an additional conversion (Ethernet-to-USB-C adapter).

As a consequence, only the SDX55 chipset based 5G modem has allowed us to connect HoloLens HMDs directly to the B5G



(a) High-level measurement setup **(b) E2E latency of the control loops** **(c) Latency & jitter characteristics of the rendering stream**

Figure 8: Latency measurements

network via the USB-C port. Therefore, in the paper, we present the measurement results obtained with this 5G modem.

6 Evaluation

In this section, we evaluate the performance of our system in different scenarios. First, we examine the runtime of each component. Second, the results of the B5G experiments are summarized, including quantitative and qualitative (subjective) aspects as well.

6.1 Latency analysis of the components

We have developed a detailed measurement method for measuring the delays of each relevant software component in our system. The high-level setup is shown in Fig. 8a (which is a simplified version of the complex architecture plot presented earlier by Fig. 1). In order to be able to measure the latency on the paths through different devices, it is necessary to synchronize the clocks. For this, we use our own NTP client on all devices (HoloLens, iPhone) and they synchronize to the gameserver's clock.

Fig. 8b summarizes the average The different types of information (pose or frame) reach the user's display via different routes. The 1st bar (Frame path) describes the average latency until the frame is displayed on the HoloLens, which was made based on the environment detection's pose information. The 3rd bar (Pose path) shows the average latency until the pose information reaches the HoloLens from the environment detection. The components in Fig. 8a and Fig. 8b are marked with similar colors and represent the following elements: 1) the green part is the processing time of environment detection (including image retrieval from iPhone and object tracking based on the low-FPS detection feed); 2) the yellow part is the network transmission time between the iPhone and the gameserver; 3) the orange part is the server processing time (simulation time, object pose prediction, pose updates of the virtual objects); and here, the route splits. 4a) The gameserver sends the 3D car pose information directly to the HoloLens (orange part in Fig. 8b and orange arrow in Fig. 8a) and 4b) based on this information, the server updates the position of the virtual objects, renders the given frame, sends it to the client and the HoloLens present it to the user (blue part in Fig. 8b and blue arrow in Fig. 8a). Here, we do not use a jitter buffer.

As one can see, the processing time of the object tracking is about 64 ms. This is mainly determined by the image retrieval from the iPhone. Unfortunately, the situation is even worse if SLAM is also enabled on the device (this is about an extra 40 ms), since we can request the image only after the local SLAM has processed it (part of ARKit in iPhone). This case is indicated by the faded bars (2nd bar for frame path and 4th bar for pose path) in Fig. 8b. However, if we turn off ARKit, we cannot move the phone (or any other tracker device) because the current pose of the tracking device is not known without the SLAM. The iPhone-Server transmission time is negligible (about 3 ms), and the server processing time is not high (about 10 ms). However, it is clearly visible that sending the image (about 37 ms) takes much more time than sending the pose to the HoloLens (about 14 ms). Here, the time until the main Unity thread processes the asynchronous messages received from environment tracking is also included.

As mentioned, the position data reaches client devices before the image. Thus, we can respond to movement-induced collisions faster using the object-centric approach compared to fully remote rendering systems. The approximately 20-25 ms response time difference is noticeable even to the naked eye.

The blue part shown in Fig. 8b can be futher broken down as shown in Fig. 8c. This represents the average delay and the standard deviation of the rendering, encoder, network transmission, receiver side queue, decoder components and the display delay (reported by Unity). Rendering takes less than 2 ms, the hardware encoder (NVidia NVEncode) and the network transmission time (RTP streamer) are about 8 ms and the hardware decoder on HoloLens side (AVCodec) finishes within 1-2 ms. The other two components (receiver queue and display delay) are primarily determined by the difference between the async functions (frame reception, decoding, display) and the main thread of Unity. This is greatly influenced by the frame rate of the Unity app running on HoloLens.

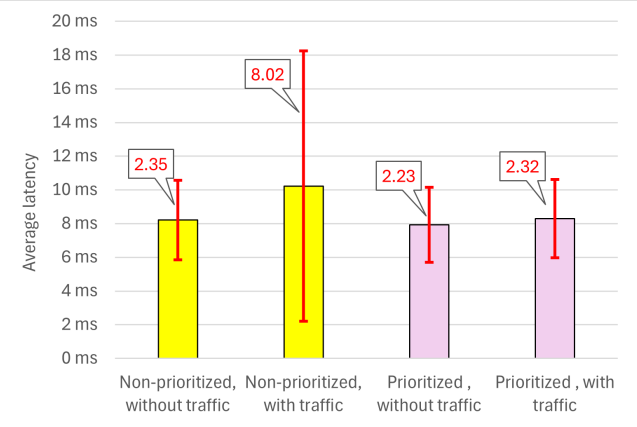


Figure 9: Average latency & jitter of remote rendered frames

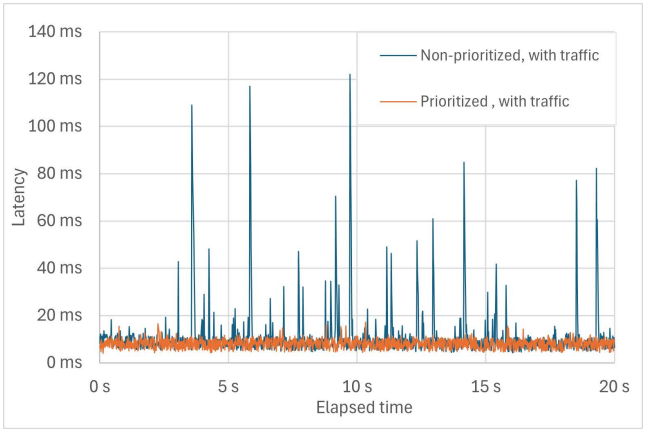


Figure 10: Latency characteristic of received frames

6.2 Beyond 5G network experiments

Our B5G environment, providing extended capabilities and features for available 5G network, allows different scenarios to be measured. Here, we analyze the operation of our XR platform and the application in the B5G setup, reveal the impact of congestion in the network (radio access network) and evaluate mitigation techniques at the network level (slice prioritization) and at the application level (jitter buffer). All components were monitored but here we primarily focus on the remote-rendered video stream, because we found that it has the greatest impact on the user experience. Latency and jitter characteristics are analyzed in idle network and under high-traffic conditions, respectively. Then the impact of the prioritization of the video stream to HoloLens and the dynamic setting of the jitter buffer are investigated.

Fig. 9 summarizes the results of our measurements. The two yellow bars on the left show the basic setup (without priority) in idle and congested network, respectively. The average latency jumps by 2 ms but the jitter exhibits much more concerning characteristics and its value nearly quadruples. The purple bars represent scenarios with priority enabled (on slice including HoloLens). The increase in latency and jitter due to the network congestion is minimal, which means a significant improvement.

The detailed characteristics of the low-level data series are further analyzed. This is crucial, as we need to apply a completely different jitter buffer if the 8 ms jitter comes from many smaller spikes rather than fewer, larger latency spikes. Fig. 10 shows the detailed latency characteristics of the time series of the two different streams. It can be seen that, in the case without priority (blue line), there are 100-140 ms jumps, while with priority (orange line), the latency does not exceed 20 ms.

These large spikes can cause frame drops, significantly reducing the user experience. To counteract this, we apply a jitter buffer (discussed in Sec. 3.2), which transforms jitter (and frame drops) into additional latency. Our goal is to find the optimal trade-off that provides the best image quality (minimal drops) with the lowest possible latency. To achieve this, we can look at the time gaps between frames received on the HoloLens to figure out how often they were delayed. From this, we can calculate the percentage

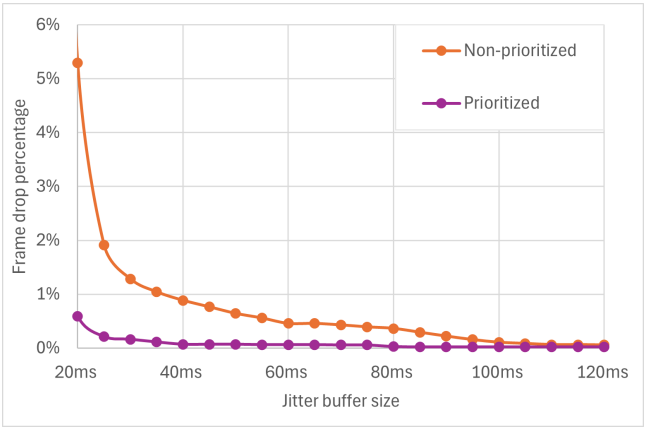


Figure 11: Impact of jitter buffer size on frame drop rate

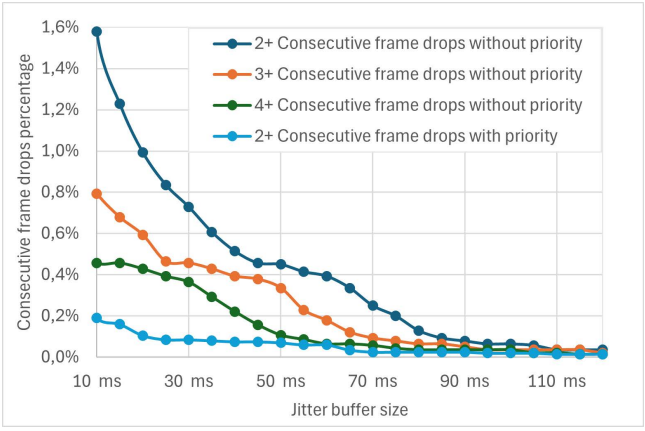


Figure 12: Rate of consecutive frame drops in terms of jitter buffer size w/ and w/o priority

of frame drops in the system, meaning the cases when the next frame does not arrive in time. In Fig. 11, we demonstrate how the

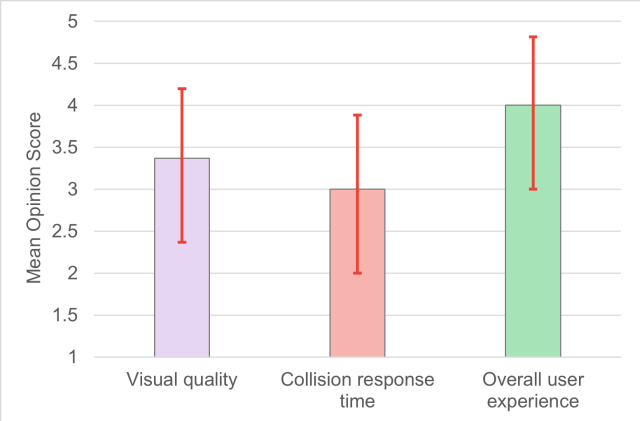


Figure 13: Subjective evaluation

percentage of dropped frames varies in terms of the size of the jitter buffer under heavy network traffic. This shows that, in prioritized scenarios, a jitter buffer of 20-30 ms is typically sufficient, causing only a 1-2 frame delay at a 60 Hz display rate. In non-prioritized scenarios, however, this buffer size requirement increases to 100-120 ms, resulting in a delay of up to 8 frames!

A single frame drop might slightly reduce video quality, but users probably will not notice if they are seeing 59 frames per second instead of 60. However, if multiple frame drops occur in a row, the resulting stutter in the video stream is more noticeable to the user. Hence, it is important to reveal how likely consecutive frame drops occur in the system. This is illustrated in Fig. 12. As shown, the probability of even two consecutive frame drops is lower in prioritized cases than the chance of four consecutive drops in non-prioritized setups.

Jitter differences are significant not only for frame transmission but also play a crucial role in position updates. Large jumps in position updates can lead to choppy object movement, which similarly degrades the user experience.

6.3 Subjective evaluation

We conducted a subjective test with 19 users (familiar with XR technologies) to evaluate the visual quality of the application, the response time of the car-ball collision, and the overall user experience. During the measurements, we prioritized the traffic of the HoloLenses over other network traffic without generating additional traffic on the network. During testing, two users played a five-minute match against each other at the same time, after which they evaluated the application. The ratings were on a scale from 1 to 5 (1 is lowest, 5 is highest quality). The test results are illustrated in Fig. 13. In terms of visual quality, the app received an average rating of 3.36, with scores ranging from 2 to 5. While some users gave higher ratings, several participants noted deficiencies in the graphical presentation. This could be due to the fact that, as revealed in the test, the vast majority of the testers regularly play video games, which obviously provide a higher visual experience than the app we developed.

The response time of the car-ball collision was also evaluated by users, receiving an average score of 3.0, with values ranging between 1 and 4. Based on the relatively low average score, many users found the delay in the ball's movement unsatisfactory. This was unfortunately expected, as we measured that the car's movement becomes noticeable on the HoloLens after approximately 100 ms, which is a delay that all users can perceive. In the future, we plan to replace the iPhone with a faster device with lower image access times to significantly improve this metric.

The overall user experience had an average rating of 4.0, with scores ranging from 2 to 5, indicating that, on average, we were able to provide a good user experience to the participants.

7 Conclusion

In this paper, we have proposed a novel edge cloud based architecture answering different challenges of future remote-rendered MR applications. The most critical feature/aspect we have addressed is the low-latency immersive interaction. This requires several capabilities and operation characteristics from the underlying infrastructure, including edge cloud, networks and devices. In order to meet the visual fidelity goal, offloading to edge cloud infrastructure is crucial, but as a result, the introduced additional latency has to be managed or mitigated by other novel techniques. By the end of the day, the remote rendering, 3D simulation and environment detection control loops have to be harmonized in an efficient way.

To validate our concepts, a proof-of-concept prototype together with a challenging MR application have been delivered and experiments in a Beyond 5G network infrastructure were conducted. We have analyzed the operation of our XR platform and the application in different scenarios, revealed the impact of congestion in the radio access network and evaluated mitigation techniques at the network level (slice prioritization) and at the application level (dimensioning of a jitter buffer).

We found that the remote-rendered video stream has the greatest impact on the user experience. The results show that the latency and jitter characteristics can be managed efficiently together by a network-level control mechanism (prioritizing network slices including the HMDs) and a dynamic jitter buffer at the application-level. Furthermore, we plan to supplement the system with dynamic bit-rate control based on network congestion prediction.

Besides, the quality of the user experience has also been assessed via real-life experiments conducted as part of a student competition. The overall user experience can be summarized as good, however, the visual quality and the response time of physical-virtual interactions need further improvement which is part of our future work.

Acknowledgments

This work has been supported in part by the European Commission under the SNS project IMAGINE-B5G (Advanced 5G Open Platform for Large Scale Trials and Pilots across Europe), grant number 101096452. This project has received funding from the CHIPS Joint Undertaken as part of the European Union's Horizon Europe research and innovation programme, SMARTY Project, grant agreement No. 101140087. B. Sonkoly was supported by the János Bolyai Research Scholarship of the Hungarian Academy of Sciences.

References

[1] Narmeen N. Al-Hiyari and Shaidah S. Jusoh. 2021. Healthcare Training Application: 3D First Aid Virtual Reality. In *International Conference on Data Science, E-Learning and Information Systems 2021 (DATA'21)*. Association for Computing Machinery, 107–116. doi:10.1145/3460620.3460741

[2] Fredrik Alriksson, Oskar Drugge, Anders Furuskär, Jonas Kronander, Jose Luis Pradas, Ying Sun, et al. 2023. Future network requirements for extended reality applications. *Ericsson Technology Review* 2023, 4 (2023), 2–12.

[3] Apple. 2025. ARKit. https://developer.apple.com/augmented-reality/arkit/.

[4] Jieyu Chen, Zhenghao Xi, Junxin Lu, and Jingjing Ji. 2022. Multi-object tracking based on network flow model and ORB feature. *Applied Intelligence* 52, 11 (Sept. 2022), 12282–12300. doi:10.1007/s10489-021-03042-6

[5] Hung-Lin Chi, Shih-Chung Kang, and Xiangyu Wang. 2013. Research trends and opportunities of augmented reality applications in architecture, engineering, and construction. *Automation in Construction* 33 (Aug. 2013), 116–122. doi:10.1016/j.autcon.2012.12.017

[6] Archi Dasgupta, Mark Manuel, Rifat Sabbir Mansur, Nabil Nowak, and Denis Gračanin. 2020. Towards Real Time Object Recognition For Context Awareness in Mixed Reality: A Machine Learning Approach. In *2020 IEEE Conference on Virtual Reality and 3D User Interfaces Abstracts and Workshops (VRW)*. 262–268. doi:10.1109/VRW50115.2020.00054

[7] Mohammed S. Elbamby, Cristina Perfecto, Mehdi Bennis, and Klaus Doppler. 2018. Toward Low-Latency and Ultra-Reliable Virtual Reality. *IEEE Network* 32, 2 (2018), 78–84. doi:10.1109/MNET.2018.1700268

[8] Yalda Ghasemi, Heejin Jeong, Sung Ho Choi, Kyeong-Beom Park, and Jae Yeol Lee. 2022. Deep learning-based object detection in augmented reality: A systematic review. *Computers in Industry* 139 (2022), 103661. doi:10.1016/j.compind.2022.103661

[9] Tariq Masood and Johannes Egger. 2019. Augmented reality in support of Industry 4.0—Implementation challenges and success factors. *Robotics and Computer-Integrated Manufacturing* 58 (2019), 181–195. doi:10.1016/j.rcim.2019.02.003

[10] Microsoft. 2024. Network requirements. https://learn.microsoft.com/en-us/azure/remote-rendering/reference/network-requirements.

[11] Microsoft. 2025. Azure Remote Rendering. https://azure.microsoft.com/en-us/products/remote-rendering.

[12] Reza Moezzi, David Krcmarik, Jindřich Cýrus, Haythem Bahri, and Jan Koci. 2022. Object Detection Using Microsoft HoloLens by a Single Forward Propagation CNN. In *Proceedings of the International Conference on Intelligent Vision and Computing (ICIVC 2021)*. Springer International Publishing, Cham, 507–517.

[13] Nvidia. 2025. CloudXR. https://www.nvidia.com/en-us/design-visualization/solutions/cloud-xr/.

[14] Open5GS. 2024. Open5GS. https://github.com/open5gs.

[15] Psyonix. 2024. RocketLeague. https://www.rocketleague.com/en.

[16] Qualcomm. 2018. Boundless XR in a new era of distributed computing. https://www.qualcomm.com/news/onq/2018/09/boundless-xr-new-era-distributed-computing.

[17] Shu Gan Raobo Li, Xiping Yuan and Rui Bi. 2024. A robust correspondence-based registration method for large-scale outdoor point cloud. *International Journal of Digital Earth* 17, 1 (2024), 2407943. doi:10.1080/17538947.2024.2407943 arXiv:https://doi.org/10.1080/17538947.2024.2407943

[18] Jangwoo Son, Serhan Gül, Gurdeep Singh Bhullar, et al. 2020. Split Rendering for Mixed Reality: Interactive Volumetric Video in Action. In *SIGGRAPH Asia 2020 XR*. New York, NY, USA. doi:10.1145/3415256.3421491

[19] Balázs Sonkoly, Bálint György Nagy, János Dóka, Zsófia Kecskés-Solymosi, János Czentye, Bence Formanek, Dávid Jocha, and Balázs Péter Gerő. 2023. Towards an Edge Cloud Based Coordination Platform for Multi-User AR Applications Built on Open-Source SLAMs. In *NOMS 2023-2023 IEEE/IFIP Network Operations and Management Symposium*. 1–6. doi:10.1109/NOMS56928.2023.10154295

[20] Daniel Stadler and Jürgen Beyerer. 2022. BYTEv2: Associating more detection boxes under occlusion for improved multi-person tracking. In *International Conference on Pattern Recognition*. Springer, 79–94.

[21] Jakob Struye, Sam Van Damme, Nabeel Nisar Bhat, Arno Troch, Barend Van Liempd, Hany Assasa, Filip Lemic, Jeroen Famaey, and Maria Torres Vega. 2024. Toward Interactive Multi-User Extended Reality Using Millimeter-Wave Networking. *IEEE Communications Magazine* 62, 8 (2024), 54–60. doi:10.1109/MCOM.001.2300804

[22] Bruce H. Thomas. 2012. A survey of visual, mixed, and augmented reality gaming. *Computers in Entertainment* 10, 1 (Oct. 2012), 1–33. doi:10.1145/2381876.2381879

[23] Unity. 2025. Barracuda. https://docs.unity3d.com/Packages/com.unity.barracuda@3.0/manual/.

[24] J. M. P. van Waveren. 2016. The asynchronous time warp for virtual reality on consumer hardware. In *Proceedings of the 22nd ACM Conference on Virtual Reality Software and Technology* (Munich, Germany) *(VRST '16)*. Association for Computing Machinery, New York, NY, USA, 37–46. doi:10.1145/2993369.2993375

[25] Visiometry. 2024. VisionLib. https://visionlib.com/.

[26] Shuang Wu, Yawen Fan, Shibao Zheng, and Hua Yang. 2012. Object tracking based on ORB and temporal-spacial constraint. In *2012 IEEE Fifth International Conference on Advanced Computational Intelligence (ICACI)*. IEEE, 597–600.

[27] Xuefeng Xie and Hefeng Wu. 2012. ORB tracking via random model and sample consensus. In *2012 5th International Congress on Image and Signal Processing*. IEEE, 113–117.

[28] Yingen Xiong and Christopher Peri. 2021. Space-Warp with Depth Propagation in XR Applications. In *2021 IEEE International Symposium on Multimedia (ISM)*. 50–57. doi:10.1109/ISM52913.2021.00017

[29] Vinay Yadhav, Andrew Williams, Ondrej Smid, Jimmy Kjällman, Raihan Islam, Joacim Halén, and Wolfgang John. 2024. Benefits of Dynamic Computational Offloading for Mobile Devices. In *Proceedings of the 14th International Conference on Cloud Computing and Services Science - Volume 1: CLOSER*. INSTICC, SciTePress, 265–276. doi:10.5220/0012719800003711

[30] Zhehui Zhang, Shu Shi, Varun Gupta, and Rittwik Jana. 2019. Analysis of cellular network latency for edge-based remote rendering streaming applications. In *Proceedings of the ACM SIGCOMM 2019 Workshop on Networking for Emerging Applications and Technologies*. 8–14.

Accelerating Video Segment Access via Quality-Aware Multi-Source Selection

Dominik Winecki
winecki.1@osu.edu
The Ohio State University
Columbus, Ohio, USA

Arnab Nandi
nandi.9@osu.edu
The Ohio State University
Columbus, Ohio, USA

Abstract

Video data can be slow to process due to the size of video streams and the computational complexity needed to decode, transform, and encode them. These challenges are particularly significant in interactive applications, such as quickly generating compilation videos from a user search. We look at optimizing access to source video segments in multimedia systems where multiple separately encoded copies of video sources are available, such as proxy/optimized media in conventional non-linear video editors or VOD streams in content distribution networks. Rather than selecting a single source to use (e.g., "*use the lowest-bitrate 720p source*"), we specify a minimum visual quality (e.g., "*use any frames with VMAF ≥ 85*"). This quality constraint and the needed segment bounds are used to find the lowest-latency operations to decode a segment from multiple available sources with diverse bitrates, resolutions, and codecs. This uses higher-quality/slower-to-decode sources if the encoding is better aligned for the specific segment bounds, which can provide faster access than using just one lower-quality source. We provide a general solution to this *Quality-Aware Multi-Source Selection* problem with optimal computational complexity. We create a dataset using adaptive-bitrate streaming Video on Demand sources from YouTube's CDN. We evaluate our algorithm on simple segment decoding as well as embedded into a larger editing system—a declarative video editor. Our evaluation shows up to 23% lower latency access, depending on segment length, at identical visual quality levels.

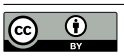
CCS Concepts

• **Information systems** → **Multimedia information systems**; **Multimedia databases**.

Keywords

multimedia databases, declarative video editing

ACM Reference Format:
Dominik Winecki and Arnab Nandi. 2025. Accelerating Video Segment Access via Quality-Aware Multi-Source Selection. In *ACM Multimedia Systems Conference 2025 (MMSys '25), March 31-April 4, 2025, Stellenbosch, South Africa.* ACM, New York, NY, USA, 9 pages. https://doi.org/10.1145/3712676.3714449



This work is licensed under a Creative Commons Attribution 4.0 International License.
MMSys '25, Stellenbosch, South Africa
© 2025 Copyright held by the owner/author(s).
ACM ISBN 979-8-4007-1467-2/25/03
https://doi.org/10.1145/3712676.3714449

1 Introduction

This paper proposes an approach for accelerating access to video segments when multiple source videos are available. Accessing encoded video is a fundamental operation in multimedia systems. Video editing tasks, for example, are particularly sensitive to seeking latency when accessing their underlying source videos. Frame sequences are compressed together into Groups of Pictures (GOPs), and cameras often use long-GOP encoding to minimize file sizes and the required storage write speed, which can increase the burden when seeking. The standard solution is to use *optimized media*, where a high-bitrate, usually All-I, stream is created from the raw source media, which serves as the main editing stream. Sometimes, even optimized media cannot meet editing needs, usually when the source resolution is high or when the size of optimized media is too great, particularly when editing on a laptop. In these cases, *proxy media* is used instead, which is a lower quality version, usually with a lower resolution, stored in easy-to-edit codecs. Proxy media provides lower storage and compute requirements while editing, while the raw source or optimized media is used to render a final output. Both use a separately encoded source to target a particular set of encoding tradeoffs.

Video content has rapidly become a mainstream communication medium, accounting for more than 65% of all internet traffic [4, 19], served primarily by modern Video on Demand (VOD) streaming services. Adaptive bitrate streaming underpins these systems, allowing clients to navigate the curve of visual quality, compute, network bandwidth, and memory tradeoffs. Modern VOD content distribution networks excel at providing variety in encoded multimedia for efficient delivery and playback.

Consider a video search engine in a streaming service responding with a compilation/supercut of video segments matching the search query. The underlying query can often run in negligible time, searching subtitles or cached ML model results in milliseconds. Materializing this into an encoded result is far slower: many short clips must be quickly accessed to create the compilation. The naïve solution involves using a preferred source to access. For example, we can set a business rule such as "*use the highest-bitrate 720p source, else the highest resolution source under 720p*". We can then use this source and do the same for every needed video segment in the compilation. This naïve approach has several drawbacks. First, we only use the preferred source selection as an indirect method to minimize latency. While the preferred source will tend to be faster, we do not guarantee it is the fastest source capable of meeting our needs. Second, we only use preferred source selection as a proxy for visual quality. Different content will have different bitrate requirements to match the same quality level, including at different

times within the same stream. These two proxies for desired behavior are a crude method of controlling the quality/latency tradeoff curve. Additionally, preferred source selection misses exploitable opportunities for shaving off latency on the *specific segment needed* compared to considering multiple sources. For example, selecting a slightly harder-to-decode source may be preferable if the needed segment begins on the last frame of a GOP from one source but on the first frame of another harder-to-decode GOP.

We believe editing systems, especially those built on top of VOD streaming infrastructure, can accelerate segment access by using multiple sources in place of optimized or proxy media, considering all sources simultaneously, and explicitly optimizing for latency and visual quality. These sources may vary in resolution, codec, and bitrate. We replace the preferred source selection rule with a direct quality constraint and only consider segments of available sources that meet the quality constraint. From the segments with sufficiently high quality, we match each needed frame to a segment that will provide it. This is done specifically for the needed segment, not the entire video. We show how to optimize this selection to minimize the latency of decoding each needed segment. The result is a plan to decode a video segment where each frame meets the provided quality constraint. We call this task *Quality-Aware Multi-Source Selection*. By providing explicit control over *actual* visual quality, upstream users of the decoded application can rely on receiving frames in the required format and level of quality without concern about how it was retrieved. It functions as the general solution for proxy editing on existing media. Additionally, this approach can deliver substantial reductions in latency when reducing visual quality, which is paramount for interactive applications.

Contributions:

- We propose opportunistically using *all* available sources of a video to quickly and efficiently decode video segments, such as from adaptive-bitrate VOD streaming systems.
- We propose `MssOpt`, an algorithm for multi-source selection providing control over the latency/quality tradeoff and prove its optimality and analyze its runtime complexity.
- We empirically evaluate how our approach efficiently controls latency and visual quality tradeoffs for standalone decoding and when integrated into a video editor.

2 Related Work

Video Processing Pipelines: Several techniques have been explored to optimize video processing performance, such as reducing decode overhead in TASM [5, 17] and pushing down analytics operations into media manipulation tools like FFmpeg [22]. Accelerating such pipelines in hardware has also been studied [17, 18].

Encoder Optimization: Controlling how videos are encoded is a massive area in research and practice. Optimization techniques are often used to tune encoder parameters; this is a standard operation [14, 16] and the primary use case for VMAF [6, 15], the visual quality metric we use in this paper. Because of this, we consider encoding a largely solved problem. This research has minimal encode-side optimizations and rather focuses primarily on decode-side optimizations.

Automatic Video Editing: Automatically editing videos is an established problem [3]. Many solutions are domain-specific, such as automatically editing home videos [10] and meetings [7]. *Supercuts* [20] are an artistic style of compiling montages of short clips with the same theme, often seen in the cinema community. Automating the supercut process has been explored from a creative perspective [9] through manual annotation and video stitching. This related work highlights the importance of automatic video editing. While there is a large body of work on creating such edits, this focus on creating high-level edits, not low-level execution.

Queries Returning Video Results: Many VDBMSs can display resulting frames and videos to the user, a form of automatic editing [21], but these are often purpose-built scripts to perform clips or a single-purpose visualization. For instance, VIVA [12] allows basic clip selection, but generating these clips remains open-ended. On the other hand, EVA [11] provides post-processing scripts to generate annotated videos for specific tasks, such as drawing bounding boxes around detected objects.

Declarative Video Editing Similar to how a database optimizes a declarative query, we believe materializing automated edits can be done with an optimizing declarative video editing system. While there are DSLs [1, 2] for expressing imperative/declarative video edits, they do not perform optimizations. V2V [21] is the only system we know, and we use it for part of our evaluation.

3 Problem Statement

3.1 Video Data Model

A *video* is the high-level representation of a piece of media content. Each video is comprised of a set of *sources*. While sources may vary in resolution, bitrate, codec, and other stream-level differences, they must all represent the same content. All source videos must have the same number of frames, the same frame times/presentation time stamps, and represent the same *content* of all other frames with the same position. We define a *root source* as the "reference" copy of the video. Whenever comparing quality, we use the *root source* as the baseline for lossless/perfect quality. There are also *derived sources*, which are re-encoded copies of the root source. Some quality metrics, including VMAF, do not report lossless sources as perfect quality, so to support lossless pipelines correctly, we explicitly set root sources to the maximum quality of 100.

Source videos are split into GOPs, each starting on a keyframe. Each is followed by dependent frames, such as P and B (progressive and bi-directional frames), requiring other referenced frames to be decoded first. Following a GOP's internal frame dependency is highly granular and codec-specific, requiring a degree of control in decoding that we have not found in any tooling. Thus, we treat a GOP decode as a single atomic operation, excluding keyframe-only decodes as discussed in Sec. 4.3.2, and assume that decoding any GOP can be done independently of others. This paper uses GOP to refer to independently encoded blocks of frames, as is the case in a *closed GOP* design. Newer codecs allow for *open GOPs* with dependencies between GOPs, with IDR frames denoting that no prior GOPs will be referenced. We define GOPs as intervals between keyframes, the IDR frames in such codecs, allowing for open GOPs despite the semantic differences.

An example of GOP distribution across sources is shown in Fig. 1. It is a segment from the start of a video, following the title screen. Fig. 1a shows keyframes at consistent intervals for segment boundaries. In addition to these regular keyframes, sometimes segments are split into two GOPs during cuts in the video. However, it is worth noting some boundaries are off by a single frame. We believe this slightly increases access latency over these regions by reducing the proportion of perfect source-to-source jump points.

3.2 Visual Quality Model

Visual Quality. Visual quality is comprised of visual frame quality and temporal quality. Frame quality is the perceptual similarity between a frame and the reference raw frame. Temporal quality is the perceptual similarity of videos with different frame rates. Since frame quality is dependent on human perception, proxy measures are used. We use VMAF [6, 15] (Video Multimethod Assessment Fusion), an Emmy-winning perceptual video quality assessment algorithm developed by Netflix, which is an industry-standard approach to defining and managing visual quality. We chose to use VMAF in this paper for its ubiquity, however our approach is not specific to VMAF. Temporal quality is how the video is perceived as a sequence, which is highly dependent on the frame rate and the specific content of the video. There is no equivalent metric for measuring temporal quality between two sources, so we do not consider temporal quality in this research. Due to this, we exclusively use '*visual quality*' to refer to visual frame quality outside of this section.

3.3 Incorporating Quality & Latency Metrics via Zone Maps

Verifying that a source GOP meets a certain visual quality threshold requires advanced knowledge of the visual quality of source frames. This can be either predicted with heuristics or by measuring the actual per-frame quality of sources. In this paper, we measured in advance as building heuristics for quality is out of scope. As an implementation note, some VOD services with high watch/video ratio reduce bandwidth by optimizing source resolution, or other parameters, on specific videos to maximize VMAF for each source on the specific content; in such systems exact quality measures are available. Quality can also be approximated by observing the encoder, such as the first-pass profile of a two-pass variable-bitrate (VBR) encode, or simply using resolution or bitrate.

Similar to visual quality, we need prior knowledge of how long a GOP will take to decode, and we measure this value, too. Specifically, we measure the time to decode a GOP, including I/O overhead, and rescaling it, if needed. In practice the latency prediction is purely an arbitrary cost value, so additional considerations may be incorporated as well. For example, sources may exist in multiple points of presence or in different cache levels. In such cases, multiple virtual sources can be created to estimate the access time for each of these locations.

Once determined these values for quality and latency must be stored. A per-frame precomputed quality measurement is too granular and would incur unnecessary storage and compute costs. Additionally, only I-frames support independent decoding. Rather than storing quality per frame, we apply zone maps [24] over zones/segments of frames. Each zone corresponds to a single GOP, and we store the minimum quality of any frame within the GOP. This allows rapid lookups by minimum quality using negligible stored data. An encoder's selection of GOP boundaries may be content-aware, so GOPs are likely to have similar visual quality throughout. Because of this, GOP-level zone maps for quality will enforce minimum constraints while only rarely pruning valid frames. We take a similar approach for latency by storing the predicted latency for each GOP. In our example clip, the visual quality of each GOP, shown in Fig. 1b, and their costs, in Fig. 1c, show that different sources represent segments with varying efficacy, as they are not exact inverses, which can be exploited by multi-source selection.

3.4 Problem Definition

When a segment is needed, we are given a video from which we need to decode a certain time span that meets a particular quality threshold. This video will have one or more sources, each a separately encoded copy of the video. Consider an example video with five frames and two sources, A & B, as shown in Fig. 2. Each GOP has attributes for its time range, latency to decode, and quality. The range is represented as a first and last frame time, g.first_t and g.last_t. We measure the decode time in milliseconds as the latency value in our implementation. This avoids errors from latency prediction models. Lastly, each GOP has a quality g.quality. We define this as the minimum quality of any frame within the GOP.

Our task is to select the lowest latency set of operations that covers a desired frame range with frames meeting a quality constraint. For example, we can construct a plan to decode the first three frames from the GOP structure in Fig. 2. Assuming all the GOPs have sufficiently high quality, the optimal plan depends on their latencies. If decoding A1 is faster than decoding both B1 and B2 combined then the optimal plan only decodes A1. However, if decoding both B1 and B2 is faster than decoding A1, then the optimal plan uses B1 and B2.

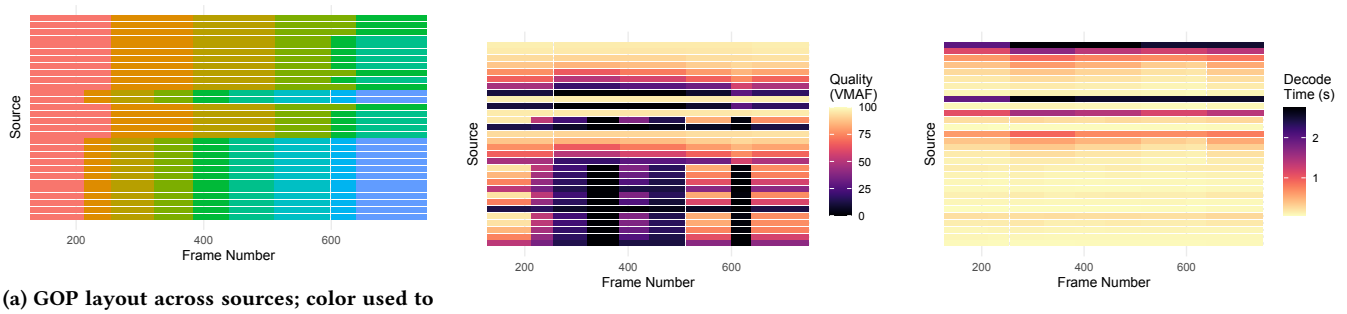
4 Multi-Source Selection

We present the MssOpt algorithm, which quickly and optimally solves the Quality-Aware Multi-Source Selection problem. In the following sections, we present the algorithm, analysis, and further extensions we have made to MssOpt.

4.1 Algorithm

MssOpt works in three parts: transforming the task into a directed weighted graph, finding the shortest path, and using the shortest path to construct an optimal plan. The pseudocode for this is shown in Algorithm 1. We have found graph-based optimization ideal, as graph algorithms allow an intuitive understanding of the task. Dynamic programming is another common approach, but dynamic programming and graph solutions are identical for such DAG processing; sufficiently optimizing both will yield identical code. Additionally, our solution produces optimal selections within an optimal computational asymptotic bound, making more expressive methods such as constraint solving/ILP are unnecessary.

Our example GOP set, in Fig. 2, corresponds to this graph shown in Fig. 3. Each edge is labeled with its cost, where "X2" refers to



(a) GOP layout across sources; color used to show the GOP number within the segment.

(b) GOP visual quality heatmap.

(c) GOP decode time heatmap.

Figure 1: GOP layout and properties of the sources representing a single YouTube video. Each block represents one GOP. Each row represents a different adaptive-bitrate streaming source, sorted by YouTube source ID, which are sorted by codec, then protocol (MP4/WebM or HLS/DASH), then resolution.

	Frames				
Source	0	1	2	3	4
A	K	·	·	·	K
B	K	K	·	·	·

GOP	first_t	last_t
A1	0	3
A2	4	4
B1	0	0
B2	1	4

Figure 2: An example video with two sources, each containing two GOPs; A 'K' denotes a keyframe, whereas a '·' denotes a dependent non-key frame

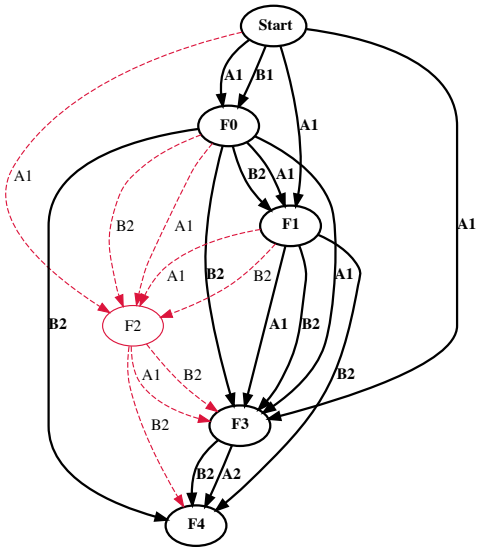


Figure 3: Frame transition graph, as created by MssOpt in Algorithm 1 from the GOP layouts in Fig. 2 to access all 5 frames. The redundant subset of operations (in dashed red) may be pruned, as detailed in Sec. 4.3.1

Algorithm 1 MssOpt Algorithm

```
1:  procedure MssOpt(S, first_t, last_t, min_quality)
2:      Let G be a directed weighted empty graph
3:      for source ∈ S do
4:          candidate_gops ← gops_intersecting_t(s, first_t, last_t)
5:          for g ∈ candidate_gops do
6:              for i ← (g.first_t − 1) to g.last_t do
7:                  if ¬(first_t ≤ i + 1 ≤ last_t) then
8:                      continue
9:                  if g.quality < min_quality then
10:                     continue
11:                 for j ∈ i + 1 to min{g.last_t, last_t} do
12:                     if i + 1 = first_t then
13:                         n ← "Start"
14:                     else
15:                         n ← i
16:                     if (n, j) ∉ G ∨ g.latency < G[(n, j)].cost then
17:                         G.set_edge(from = n, to = j, cost = g.latency, gop = g)
18:     P ← shortest_path(G, from="Start", to=last_t)
19:     O ← []
20:     for edge ∈ P do
21:         start ← edge.source
22:         if start = "Start" then
23:             start ← first_t
24:         end ← edge.target
25:         O.append((edge.gop, start, end))
26:     return O
```

the second GOP's latency in source X. Note that, in this graph, "Fn" means that we have already decoded the frame at time n. The path from Start to F4 corresponds to the optimal cost plan. A subset of this graph is shown in dashed red; this subset is pruned early for optimization as detailed in Sec. 4.3.1.

4.2 Runtime Complexity

The runtime complexity is an important consideration for an optimizer. Source videos may be multiple hours long, including some

with high frame rates. This can create situations where an MssOpt may consider millions of frames. We prove that the MssOpt algorithm, as presented, runs proportionally to the product of the number of sources and the length of the needed segment. This is important as we expect optimized plans to use more efficient sources, producing a constant speedup for each GOP. Since the number of sources is generally low, a runtime proportional to the segment length ensures that the optimizer's computation time remains negligible compared to the execution time of the optimized plan, thus providing a consistent speedup.

Definitions. We have a set of sources S, and are decoding the time range [first_t, last_t]. Each source is a separately encoded copy of the same video. The length of the video and each of its sources is N frames. Similarly, we define R as the number of frames in the decoded range: $R = \text{last_t} - \text{first_t} + 1$.

Assumptions. We assume that a finite value, M, is the maximum number of frames in a GOP. Video codecs and multimedia standards limit the length of GOPs to enable efficient random seeking of videos and to bound the memory needed by a decoder. Practically, most GOPs are within an order of magnitude of $\sim$ 100 frames or $\sim$ 3 seconds. Therefore it is safe to assume all GOPs are finite.

$$\forall g : \text{GOP}, g.\text{last_t} - g.\text{first_t} + 1 \leq M$$

4.2.1 Proof. Taken together, Lemmas 4.1, 4.2, and 4.3 show that MssOpt runs in $O(|S| \cdot [R + \log N])$.

Lemma 4.1. *The construction of the frame transition graph (Algorithm 1 line 3) runs in $O(|S| \cdot [R + \log N])$ time.*

Proof. The outer loop over s iterates through each of the $|S|$ sources. On each of these sources, we execute gops_intersecting_t(S, first_t, last_t) to find the candidate GOPs on that source, then it adds each candidate GOP to the frame transition graph. Executing gops_intersecting_t(S, first_t, last_t) is $O(\log N)$, as it is a simple range search. A binary search runs in an array in $O(\log N)$ time, as would a range search within an appropriate database index.

The number of GOPs in the candidate set is between R (if all are of length 1) and $\frac{R}{M}$ (if all GOPs are the maximum length) by our assumption on the maximum size of a GOP as M. We assume a worst case of the loop at line 5 executing R times.

The inner loops at 6 consider each transition from one frame in a GOP to another, which is at worst $\frac{M^2}{2}$, which is constant.

These, together, result in the creation of the frame transition graph in $O(|S| \cdot [R + \log N])$. □

Lemma 4.2. *The shortest path search executes in $O(|S| \cdot |R|)$*

Proof. The shortest path of a weighted directed acyclic graph can be found in $\Theta(E+V)$. It suffices to show that the graph is acyclic, $V \in O(R)$, and $E \in O(|S| \cdot |R|)$.

First, we show that G is acyclic. Each edge in G is from i to j from the loops at lines 6 and 11. From their ranges, we can see that $i < j$ forms an ordering over the edges. The special "Start" vertex precedes all frame vertices in this ordering. The existence of this total ordering proves the graph is acyclic.

Since each read frame corresponds to a single vertex plus one global "Start" vertex, $V \in O(R)$ is trivial.

As detailed in Lemma 4.1, the outer loops at lines 3 and 5 execute their body $O(|S| \cdot R)$ times. Since the inner loops may insert at most $\frac{M^2}{2}$ edges, a constant, therefore $E \in O(|S| \cdot |R|)$. □

Lemma 4.3. *Constructing a plan from the shortest path runs in $O(|R|)$ time.*

Proof. The length of the shortest path is, at most, the number of nodes in the graph. Lemma 4.2 shows that $V \in O(|R|)$. The plan construction is proportional to the length of the shortest path (as a single iteration each in the loop at line 20). □

4.3 MssOpt Extensions

Here, we describe extensions to MssOpt, further increasing its performance and capabilities. For brevity, we describe only the function of each extension with minimal implementation detail, as they are mostly straightforward and mundane and do not change the runtime complexity. Implementations are readily available in our source code.

4.3.1 Graph Pruning. As seen in Fig. 2, frame transition graphs contain many redundant edges and nodes, especially when there are few sources or shared GOP boundaries. We identify which frames are used in transitions in optimal plans and only use those nodes in our graph. This allows smaller and faster-to-search graphs. This pruning never prevents finding the optimal solution since transition graphs are created for each needed segment. The following frames are needed:

(1) The frame before the start of the range (first_t − 1)
(2) The last frame of the range (last_t)
(3) The frame prior to the start of a GOP ($\{g.\text{first_t} - 1 | g \in G\}$)
(4) The start of a GOP ($\{g.\text{first_t} | g \in G\}$)
(5) The end of a GOP ($\{g.\text{last_t} | g \in G\}$)

The first two are essential as they are the start and end points of the shortest path. Informally, the third is needed to start a new GOP from the start, the fourth to switch to another GOP after a keyframe-only decode, and the fifth to switch to another GOP after reading through the end of a GOP.

Consider the previous example source structure in Fig. 2. Frame 2 is redundant. If A1 < B2 then any path will decode GOP A1 twice, and if B2 < A1 then B2 will be decoded twice. A shorter path must always exist that does not decode the same GOP twice. Thus, the F2 and all edges from it can be pruned. The pruned graph is shown in red in Fig. 3.

We implement this pruning in our system as a set constructed at the start of the procedure that is then checked on each value of i and j. Constructing such a set is $O(|G|)$, or the equivalent $O(|S| \cdot R)$ as shown in Lemma 4.1, so this does not change the algorithm's runtime complexity.

4.3.2 Keyframe-Only Decoding. In addition to full GOPs, we add support for decoding keyframes independently, enabling up to twice as many opportunities to switch from one source to another. Hence, we extend Algorithm 1 to include keyframe-only decodes in our

implementation. To do this, we store a second zone map for the keyframe decode latency in addition to the full-GOP decode latency.

4.3.3 Stream-copy Support. The MssOpt algorithm, as presented in Algorithm 1, assumes all video must be decoded. However, some use cases and specifications allow stream-copying GOPs directly from sources to the output. To accommodate for this, we implement an additional case: if an edge goes from the frame node before the GOP's start to the last frame of the GOP, and the GOP is stream-copy eligible, then the cost is updated, and the edge is tagged as a stream-copy. We also include the equivalent operation for stream-copying a keyframe.

One observation from our preliminary analysis was that the MssOpt would jump between GOP stream copies from different video sources due to the high overlap in keyframe locations between sources and noise in our latency prediction. When GOPs are decoded, this results in a speedup; however, stream-copies are so fast that the overhead of opening additional video files overshadowed this gain. We introduce *stream-copy compaction* to address this, where we assume that maintaining locality within stream-copies exceeds the gain of marginally more efficient operations and implement a heuristic greedy algorithm to re-introduce this locality to plans. First, we find continuous spans of stream-copy operations. Then, each span is compacted: we find the longest range that starts and ends from a single source and replace the whole range with a single stream-copy. This process is repeated until we can no longer compact the span. We ensure that each newly inserted GOP meets the quality constraints during this compaction.

5 Incorporating MssOpt into an Editing System

In addition to simple decoding tasks, we wish to evaluate how Multi-Source Selection affects video editing end-to-end. This is to validate that Multi-Source Selection does not conflict with existing optimization techniques in a larger application. We integrated MssOpt into V2V [21], an optimizing declarative video editor designed to join data with videos to create query results in a VDBMS. This constitutes our baseline for comparison against our fork using Multi-Source Selection. This is a fitting system to evaluate for several reasons: Creating video results to queries is similar to our primary use case of creating compilations in response to search. It also provides existing editing optimizations via a database-style optimizer, as described in Sec. 5.0.1, which complement the components we contribute in this research. Additionally, V2V already performs GOP-level access optimizations internally, which we use as the foundation for our MssOpt integration.

5.0.1 Baseline Optimizations Overview. As a first step, an unoptimized plan is created by a planner. After this, it applies optimizations such as smart cuts [8, 13], temporal sharding (with preference towards GOP boundaries), dependency DAG parallelism, and merging operations. The architecture of this baseline optimizer is shown on the left side of Fig. 4. Each of these optimizations is lossless, sans quality loss from encoding. While most are generic data-processing optimizations, smart cuts are of note. Smart cuts are an extension of stream-copies. Since stream copies can only occur on GOP boundaries, a smart cut only stream copies GOPs strictly within the clipped segment. Then, to keep the operation frame exact, a smart cut takes the preceding and following omitted frames, transcodes them, and organizes them at the front or end of the copied segment, respectively.

5.1 MssOpt-Based Optimizer Design

We extend the baseline heuristic optimizer to function as a cost-based optimizer. Internally, we replace the source clip decoding with Quality-Aware Multi-Source selection via MssOpt and remove the "Smart Cut" step. This non-invasive upgrade is all that is needed for Quality-Aware Multi-Source selection in most editors.

5.1.1 Stream-Copy Expansion. V2V's declarative video editor makes liberal use of stream-copying and smart cuts on simple clips. To evaluate properly against it, we added stream-copying support into MssOpt, detailed in Sec. 4.3.3. With this implementation, MssOpt automatically implements smart cuts as an emergent behavior of our generalized solution.

As part of finding stream-copy eligible GOPs, MssOpt requires knowledge of the output codec. Rather than selecting the output codec upfront, we add a "Codec Selection" step, which separately attempts to optimize the entire edit for each codec used in the sources and selects the best codec to use on the output. It takes the sum cost of the shortest path from MssOpt from each decoded section. We then add an *encoder cost* to compensate for different output encoders' speeds, which we calculate by multiplying the number of output frames by an encode-time-per-frame constant for the codec and output resolution.

5.1.2 Edits Merging Multiple Input Videos. Some edits merge multiple video streams into a composition (such as evaluation edits *E*3 and *E*7). Due to a design limitation of the execution engine we are using (based on reducing plans to FFmpeg commands) in handling multi-video source splitting without using an intermediate encoding, we constrain MssOpt to select a *single source* for each input, with the lowest cost that meets the quality constraints. In Fig. 4, this is shown as "Multi-Input MssOpt Plan," which is only for edits composing multiple video inputs. Using another execution engine, such as one based on FFmpeg's libav libraries, would remove this limitation.

6 Evaluation

6.1 Datasets

To procure a set of source videos, as would exist in a VOD streaming platform, we use YouTube by downloading the internal source encodings using yt-dlp [23]. We primarily use a collection of 100 documentaries from PBS's *NOVA* & *FRONTLINE* series, publicly available on YouTube, each around 55 minutes, picked as a higher-end representative of long-form content on streaming platforms. Videos had 29.8 sources on average, with source videos averaging 162MB in size, ranging from 192 × 144 to 1920 × 1080 resolution. We use the highest quality download source as our reference copy to evaluate quality optimizations.

6.2 Experimental Setup

We used a 2×12-core (48 HT vCPUs) Intel Xeon Gold 6126 @ 2.6 GHz system with 384GB DDR4 RAM running Debian 12. We set a native resolution of 1280x720 and re-scale all sources to this

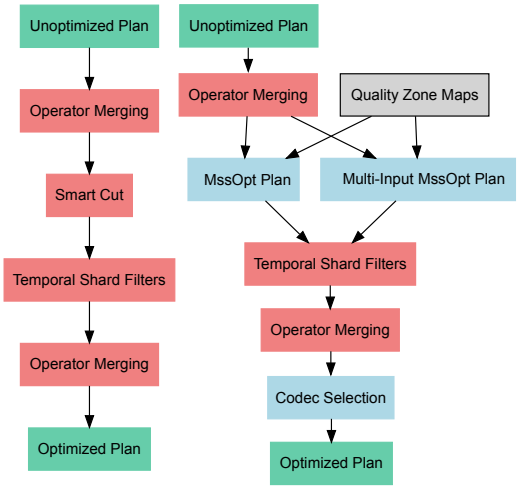


Figure 4: Optimizer Architecture - Baseline (left) & Multi-Source (right). Heuristic steps are in red and cost-based in blue.

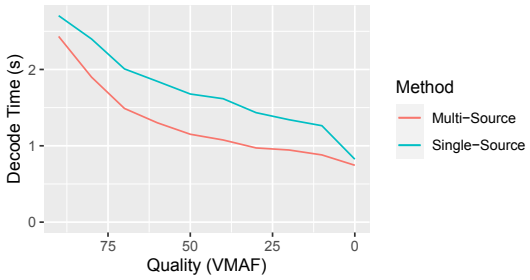


Figure 5: Decode Time vs. Quality by Method

resolution after decoding. We also use this resolution when calculating source video VMAF. Since some downloaded videos used millisecond-precision instead of exact-precision timestamps, we corrected them using a bit-stream filter. We implemented MssOpt in the Rust language with graph operations from the petgraph library. We use FFmpeg 6.0 as our underlying video processing engine. For our editing evaluation, we use the libx264 encoder with the "ultrafast" preset or libvpx-vp9 with a deadline of "realtime" and a speed of 8; the optimizer selects whichever codec allows for the lowest cost plan.

6.3 Results - Segment Decoding

We evaluate how `MssOpt` enables lower-latency decoding. The naïve approach is to pick a single source (e.g., "*lowest bitrate H.264 source at 720p*"), but this is not a good baseline to evaluate against as it leaves visual quality uncontrolled. Since finding justifiable criteria is difficult, we evaluate against what an *optimal* criteria would select. We do this by finding the lowest latency single-source clip that meets the same quality constraint as our multi-source plan. We make this selection using our pre-measured costs rather than actual time to avoid introducing one-sided bias from measurement noise. Comparing multi-source to optimal single-source decode

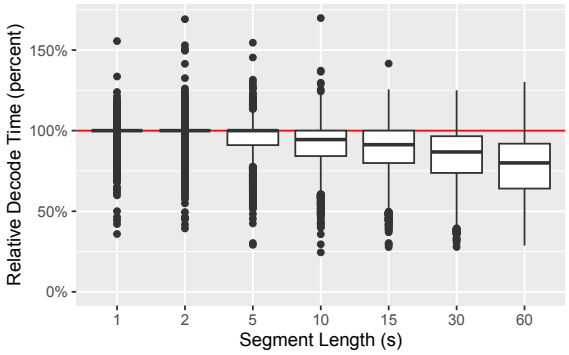


Figure 6: Relative Decode Time vs. Segment Length

Segment	Decode Time	MS≠SS	Opt. Time
1s	98.8%	16.7%	1.00 ms
2s	98.0%	29.7%	1.19 ms
5s	94.9%	60.7%	1.38 ms
10s	90.9%	82.5%	1.59 ms
15s	87.8%	89.2%	1.73 ms
30s	83.1%	95.6%	2.03 ms
60s	76.8%	97.5%	2.97 ms

Table 1: Single-to-Multi-Source Statistics by Segment Length

methods allows for a fair evaluation without relying on arbitrary source selection criteria.

We evaluated multiple quality/segment-length pairs, each having 250 samples at various points across the video dataset, and measured Multi-Source and Single-Source decode times. Fig. 5 shows that across the visual quality curve, there is a consistent drop in latency when accessing frames considering multiple sources. Table 1 shows the decode time Multi-Source/Single-Source ratio, the percentage of times when the MS and SS plans are not identical, and the time `MssOpt` takes to run across various segment lengths. In all cases, `MssOpt` runs in negligible time compared to decoding even a single frame. Our worst-case selection task, a 60s segment with a quality constraint of 0, took ~ 7.2ms on average. Short segments are highly likely to have the optimal plan be single-source, resulting in a smaller speedup than longer segments. Longer segments, more than a few times the average GOP length, consistently find lower-latency Multi-Source plans and a resulting speedup to match, up to 23.2%. These distributions are shown in Fig. 6.

6.4 Results - Video Editing

We devise an editing benchmark by building on four common editing operations: (1) A clip, (2) A splice of 4 clips, (3) 4 clips composed into quadrants, and (4) A clip with a content filter (a Gaussian blur). The first two represent our compilation use case, the fourth for additional annotation, and the third for side-by-side compilations. Each is run with 5-second and 1-minute input segments. In our tests, we term these specs $E1 - E4$ and $E5 - E8$, respectively. $E2$ and $E6$ use splices and thus result in 20-second and 4-minute videos. These specs cover each of the plan types. $E1$ and $E2$ cover the stream-copy and simple re-encode execution cases. $E3$ and $E4$ cover the filter

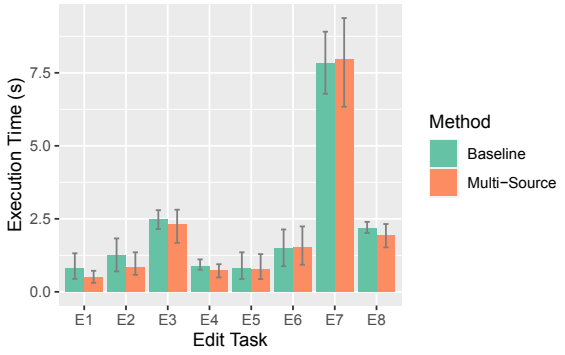


Figure 7: Declarative Video Editor - Execution Time Across Segment Access Methods

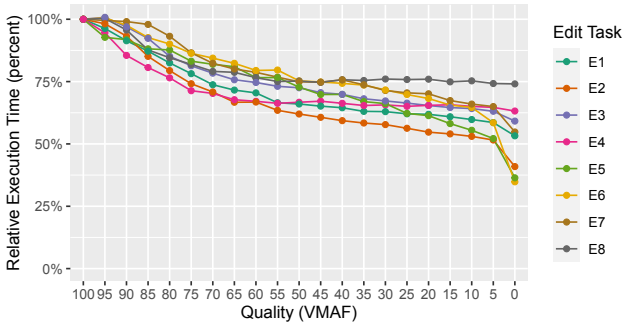


Figure 8: Declarative Video Editor - Relative Optimized Execution Time v. Visual Quality.

cases using multi- and single- input videos. The 5-second variants show latency performance, which is important in on-demand interactive use cases, and the 1-minute variants show throughput performance.

Results comparing these edits between the baseline and multi-source optimizers are shown in Fig. 7. We set the quality constraint 85 to ensure sufficiently different access plans. The error bars denote the 5% and 95% percentiles. The results show similar to slightly faster execution across all specifications. Of note are E1 and E2, which experience an average ~ 30% drop in execution time. This is more than expected considering the small drop in visual quality and decode-only evaluation expecting only ~ 5% improvement.

Fig. 8 shows the impact of visual quality on execution time. All edits saw a similar curve, including those based on stream-copying. At lower visual quality sources, edits using blur (E4 & E8) saw the least speedup, and those using stream copies (E1, E2, E5, & E6) had the most, suggesting the encoder and filters became the bottleneck.

To measure optimizer performance, we re-ran our evaluation, skipping plan execution. We set the minimum quality constraint to zero to measure the worst-case optimizer runtime. Each optimizer was slowest on E6. Both average and E6 time are reported in Table 2. While slower than the Baseline optimizer, the MssOpt-based optimizer executed under 3ms on all edits.

Optimizer	Mean Time	E6 Time
Baseline	215μs	572μs
MultiSource	1.07ms	2.18ms

Table 2: Optimizer Performance Evaluation

7 Conclusion

This paper proposes Quality-Aware Multi-Source Selection to Accelerate Video Segment Access. Decoding video clips is a fundamental primitive operation in many multimedia systems. Video on Demand streaming services maintain many sources of each video to enable bitrate streaming, and this can be exploited to accelerate access to small segments. By opportunistically using *all* video sources, needed segments can be accessed faster than any individual single source. This is similar to how modern video editors use optimized/proxy media to speed up seeking, and, since VOD services already maintain multiple sources our approach does not require additional storage. Similarly, our approach is amenable to different quality and latency prediction methods. We propose the `MssOpt` algorithm to optimally solve for the lowest latency plan to decode a specific clip given a minimum visual quality by reducing to a shortest path problem. We prove `MssOpt`'s runtime is proportional to the length of the needed segment, demonstrating that it is appropriate for use regardless of the length of needed segments. This can significantly reduce the latency of interactive applications, such as a video search returning a clip or video compilation result.

Our evaluation demonstrates that using `MssOpt` reduces decode time by up to 23% compared to single-source access methods while maintaining identical visual quality levels. This validates the effectiveness of Quality-Aware Multi-Source Selection in accelerating video segment access.

Acknowledgments

This material is based upon work supported by the National Science Foundation under Grant No. 1910356 and the NSF OAC 2118240 Imageomics Institute award.

References

[1] 2023. *editly*. https://github.com/mifi/editly
[2] Leif Andersen, Stephen Chang, and Matthias Felleisen. 2017. Super 8 languages for making movies (functional pearl). *PACMPL* 1, ICFP (Aug. 2017).
[3] Stefano Bocconi. 2004. Semantic-aware automatic video editing. In *Proceedings of the 12th Annual ACM International Conference on Multimedia* (New York, NY, USA) *(MULTIMEDIA '04)*. Association for Computing Machinery, New York, NY, USA, 971–972. https://doi.org/10.1145/1027527.1027753
[4] Cisco. 2021. VNI Complete Forecast Highlights. *Cisco Annual Reports* (2021).
[5] M Daum, B Haynes, D He, A Mazumdar, and M Balazinska. 2021. TASM: A Tile-Based Storage Manager for Video Analytics. *ICDE* (2021).
[6] Sai Deng, Jingning Han, and Yaowu Xu. 2020. VMAF Based Rate-Distortion Optimization for Video Coding. In *2020 IEEE 22nd International Workshop on Multimedia Signal Processing (MMSP)*. IEEE, Tampere, Finland, 1–6. https://doi.org/10.1109/MMSP48831.2020.9287114
[7] Haihan Duan, Junhua Liao, Lehao Lin, Abdulmotaleb El Saddik, and Wei Cai. 2024. Meetor: A Human-Centered Automatic Video Editing System for Meeting Recordings. *ACM Trans. Multimedia Comput. Commun. Appl.* (feb 2024). https://doi.org/10.1145/3648681 Just Accepted.
[8] Mikael Finstad and et al. 2023. *LosslessCut*. https://github.com/mifi/lossless-cut
[9] Daniel Chávez Heras. 2023. Creanalytics: Automating the supercut as a form of critical technical practice. *Convergence* (2023).
[10] Xian-Sheng Hua and Hong-Jiang Zhang. 2009. Automatic Home Video Editing. In *Multimedia Content Analysis: Theory and Applications*, Ajay Divakaran (Ed.). Springer US, Boston, MA, 1–35. https://doi.org/10.1007/978-0-387-76569-3_13
[11] Gaurav Kakkar, Jiashen Cao, Pramod Chunduri, Joy Arulraj, et al. 2023. EVA: An End-to-End Exploratory Video Analytics System. In *DEEM*.

[12] Daniel Kang, Francisco Romero, Peter Bailis, Christos Kozyrakis, and Matei Zaharia. 2022. VIVA: an end-to-end system for interactive video analytics. CIDR.
[13] Mario Kicherer. 2015. *avcut - frame-accurate video cutting with only small quality loss.* http://kicherer.org/joomla/index.php/de/blog/42-avcut-frame-accurate-video-cutting-with-only-small-quality-loss
[14] Yao Lu, Aakanksha Chowdhery, and Srikanth Kandula. 2016. Optasia: A Relational Platform for Efficient Large-Scale Video Analytics. In *Proceedings of the Seventh ACM Symposium on Cloud Computing* (Santa Clara, CA, USA) *(SoCC '16)*. Association for Computing Machinery, New York, NY, USA, 57–70. https://doi.org/10.1145/2987550.2987564
[15] Zhengyi Luo, Yan Huang, Xiangwen Wang, Rong Xie, and Li Song. 2019. VMAF Oriented Perceptual Optimization for Video Coding. In *2019 IEEE International Symposium on Circuits and Systems (ISCAS)*. IEEE, Sapporo, Japan, 1–5. https://doi.org/10.1109/ISCAS.2019.8702513
[16] Megha Manohara, Anush Moorthy, Jan De Cock, Ioannis Katsavounidis, , and Anne Aaron. 2018. *Optimized shot-based encodes: Now Streaming!* https://netflixtechblog.com/optimized-shot-based-encodes-now-streaming-4b9464204830
[17] Alex Poms, Will Crichton, Pat Hanrahan, and Kayvon Fatahalian. 2018. Scanner: Efficient Video Analysis at Scale. *ACM Trans. Graph.* 37, 4, Article 138 (July 2018), 13 pages. https://doi.org/10.1145/3197517.3201394
[18] Parthasarathy Ranganathan, Daniel Stodolsky, Jeff Calow, Jeremy Dorfman, Marisabel Guevara, Clinton Wills Smullen IV, Aki Kuusela, Raghu Balasubramanian, Sandeep Bhatia, Prakash Chauhan, Anna Cheung, In Suk Chong, Niranjani Dasharathi, Jia Feng, Brian Fosco, Samuel Foss, Ben Gelb, Sara J. Gwin, Yoshiaki Hase, Da-ke He, C. Richard Ho, Roy W. Huffman Jr., Elisha Indupalli, Indira Jayaram, Poonacha Kongetira, Cho Mon Kyaw, Aaron Laursen, Yuan Li, Fong Lou, Kyle A. Lucke, JP Maaninen, Ramon Macias, Maire Mahony, David Alexander Munday, Srikanth Muroor, Narayana Penukonda, Eric Perkins-Argueta, Devin Persaud, Alex Ramirez, Ville-Mikko Rautio, Yolanda Ripley, Amir Salek, Sathish Sekar, Sergey N. Sokolov, Rob Springer, Don Stark, Mercedes Tan, Mark S. Wachsler, Andrew C. Walton, David A. Wickeraad, Alvin Wijaya, and Hon Kwan Wu. 2021. Warehouse-scale video acceleration: co-design and deployment in the wild. In *Proceedings of the 26th ACM International Conference on Architectural Support for Programming Languages and Operating Systems* (Virtual, USA) *(ASPLOS '21)*. Association for Computing Machinery, New York, NY, USA, 600–615. https://doi.org/10.1145/3445814.3446723
[19] I Sandvine. 2016. Global internet phenomena report. *North America and Latin America* (2016).
[20] M Tohline. 2021. A supercut of supercuts: aesthetics, histories, databases. OpenScreens.
[21] Dominik Winecki and Arnab Nandi. 2024. V2V: Efficiently Synthesizing Video Results for Video Queries. In *2024 IEEE 40th International Conference on Data Engineering (ICDE)*.
[22] Xintian Wu, Pengfei Qu, Shaofei Wang, Lin Xie, and Jie Dong. 2021. Extend the FFmpeg Framework to Analyze Media Content. *CoRR* abs/2103.03539 (2021). arXiv:2103.03539 https://arxiv.org/abs/2103.03539
[23] yt-dlp Project. 2023. *yt-dlp.* https://github.com/yt-dlp/yt-dlp
[24] Mohamed Ziauddin, Andrew Witkowski, You Jung Kim, Dmitry Potapov, Janaki Lahorani, and Murali Krishna. 2017. Dimensions based data clustering and zone maps. *Proceedings of the VLDB Endowment* 10, 12 (Aug. 2017), 1622–1633. https://doi.org/10.14778/3137765.3137769

Secure the Stream, Not the Hosts: Attribute-Based Encryption for DRM Enabled Video Streaming

Mohammad Waquas Usmani
mohammadwaqu@umass.edu
University of Massachusetts Amherst
Massachusetts, USA

Susmit Shannigrahi
sshannigrahi@tntech.edu
Tennessee Technological University
Tennessee, USA

Michael Zink
zink@ecs.umass.edu
University of Massachusetts Amherst
Massachusetts, USA

Abstract

This paper introduces an alternate approach for encrypting video streams using attribute-based encryption (ABE), focusing on securing the data rather than the connection between streaming endpoints. This fundamental shift in the security model means we can now encrypt video segments once at the source and cache them anywhere reducing the computational load on intermediate caches by removing the need for encryption and decryption on a per-client basis. Additionally, restricting or revoking access can be done by revoking users' private keys rather than re-encrypting the entire video stream. Finally, our approach eliminates the need for decryption and encryption at the intermediate caches (currently required for TLS terminations) since an encrypted version of the content can be stored anywhere, reducing the load on the cache.

In this work, we introduce a framework for **A**ttribute-**B**ased **E**ncryption for DRM Enabled **V**ideo **S**treaming (ABEVS) that utilizes ABE for video streaming. We have integrated ABE into a DASH-based streaming client, performed Attribute-based video content encryption, and distributed them through intermediate caches. We also evaluate the performance of ABEVS by deploying a video distribution system in CloudLab. Evaluation results show that our approach significantly reduces the computational load on caches while minimally affecting the cache hit rate and video quality (VMAF) compared to standard HTTPS-based streaming.

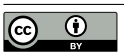
CCS Concepts

• **Information systems** → **Multimedia streaming**; • **Security and privacy** → **Public key encryption**; *Security protocols*; • **Networks** → *Network experimentation*; *Network performance analysis.*

Keywords

Video Streaming, DASH, Attribute-Based Encryption, Caching, ABR, Quality of Experience, CDN

ACM Reference Format:
Mohammad Waquas Usmani, Susmit Shannigrahi, and Michael Zink. 2025. Secure the Stream, Not the Hosts: Attribute-Based Encryption for DRM Enabled Video Streaming. In *ACM Multimedia Systems Conference 2025 (MMSys '25), March 31-April 4, 2025, Stellenbosch, South Africa.* ACM, New York, NY, USA, 11 pages. https://doi.org/10.1145/3712676.3714450



This work is licensed under a Creative Commons Attribution 4.0 International License.
MMSys '25, Stellenbosch, South Africa
© 2025 Copyright held by the owner/author(s).
ACM ISBN 979-8-4007-1467-2/25/03
https://doi.org/10.1145/3712676.3714450

1 Introduction

Digital Rights Management (DRM), controls access to copyrighted material and has contributed significantly to the success of video streaming on the Internet. An essential component of DRM is encryption, which prevents unauthorized access and distribution of copyrighted content. End-to-end encryption ensures only authorized users can decrypt and view digital content, while unauthorized viewers receive a stream they cannot decrypt.

The two aspects of current DRM approaches are securing data at transit and at rest. Most popular streaming applications today (Netflix, Amazon Prime, Disney+, etc.) achieve transport security by using Hypertext Transfer Protocol Secure (HTTPS). The HTTPS standard [13] defines a communication protocol that is encrypted using Transport Layer Security (TLS) [35].

The most common technology used for video streaming in today's Internet is adaptive bitrate (ABR) streaming (e.g., Dynamic Adaptive Streaming over HTTP (DASH) [37] or HTTP Live Streaming (HLS) [17]). In ABR, the video is chunked into segments of fixed duration, and each segment is further encoded into multiple qualities or bitrates. A list of available video segments, encoded in different qualities, is sent to the end user in a video-specific metadata file. Using the information, the player uses ABR algorithms such as BOLA [38] or Pensieve [26] to download segments at a quality that maximizes the end user's quality of experience (QoE) under current bandwidth constraints. ABR and HTTPS work well together, as the only modification needed to secure the streaming process is replacing HTTP with HTTPS for downloading chunks.

Another pillar of the success of video streaming is the support by Content Distribution Networks (CDN), which enable services like Netflix and Disney+ to serve hundreds of millions of users. Based on intricate algorithms (see, e.g., [1, 25] for more details) users are directed to caches that are in close physical proximity and store copies of the content offered by the streaming service. The disadvantage of using a combined approach of HTTPS and CDN to enable access control for video streaming is the overhead that is introduced. TLS intrinsically relies on pairwise key exchange, i.e., every streaming session to a new receiver (cache or client) is individually encrypted. To allow the use of caching, TLS termination [44] is applied, which requires the use of multiple TLS sessions (between origin and cache, and between cache and each client). TLS termination leads to a significant overhead due to a potentially high load of encryption and decryption operations at each cache. This overhead is increased by TLS's approach of total encryption of the session between sender and receiver. After the encrypted content is securely delivered, key exchanges between the client and the license server is needed to decrypt the content at rest.

In this paper, we introduce Attribute-Based Encryption for DRM Enabled Video Streaming (ABEVS), an approach that shifts the security model to securing the data itself rather than securing the transport and data separately. ABEVS incorporates Attribute-based Encryption [7] for large-scale video streaming systems. ABEVS secures content both in transit and at rest, without the need for separate transport layer security. The origin encrypts the video content using appropriate attributes, allowing content to be cached or distributed from anywhere. On receiving the content, the client requests a decryption key from the license server (as they would today). The license server returns a decryption key with attributes that allow the client to decrypt the content. A time attribute can be used to control the expiration time of the keys. We show that shifting to ABEVS not only simplifies the key management but also reduces overhead of a DRM-enabled video streaming system that can scale to the demands of popular streaming applications.

ABEVS is different from HTTPS-based ABR streaming as ABE supports point-to-multipoint encryption, removing the requirement to decrypt and re-encrypt video streams at the cache. We evaluate ABEVS potential for overhead reduction in large-scale streaming systems that employ encryption. In addition, we investigate how significantly the overhead introduced by ABEVS impacts system performance, with a specific focus on the cache. We show that while maintaining similar hitrates, midgress traffic, and VMAF-based QoE on the client side, ABEVS reduces CPU load on the cache by up to 43% compared to HTTPS in a simple setup and up to 50% in a two-level caching scenario.

This paper makes the following contributions. We introduce ABEVS, a unified solution for securing both data in transit and at rest, enabling streaming systems to serve data from anywhere without the need for re-encryption. Additionally, ABE simplifies access control by streamlining the allocation and revocation of user permissions. We demonstrate the feasibility of integrating ABEVS into a DASH player, presenting a DASH-compliant approach. Finally, we conduct an in-depth evaluation, with three distinct experiment setups: a small-scale experiment, a large-scale multi-cache hierarchical experiment with strict bandwidth limitations, and the same large-scale setup without bandwidth limitations.

2 Background and Related Work

This section discusses current video distribution techniques, related work on ABE for video streaming, and caching for video delivery.

2.1 Distributed Large-scale Video Streaming

HTTP video streaming. MPEG DASH [37] and HTTP Live Streaming (HLS) [17] are the two most popular techniques for the delivery of video on demand (VoD) and live streams on the internet. In both cases, the video is divided into fixed-duration segments, each encoded at multiple bitrates. A metadata file lists the available segments and qualities for the user. Using the information, the video player uses ABR algorithms such as BOLA [38] or Pensieve [26] to download segments at a quality that maximizes end user's QoE under the current bandwidth constraints.

HTTPS (HTTP Secure) is an extension of HTTP that encrypts communication between web servers and clients using TLS [35]. TLS establishes a short-term session key through long-term private and public keys, securing data exchange. The drawback of HTTPS is its host-centric approach that allows only point-to-point encryption. This requires individual video encoding for each receiver which increases the computational load on servers and caches.

Content Distribution Networks (CDN) CDNs help deliver digital content efficiently by caching and distributing it across multiple servers, reducing delays and improving playback. In video streaming, they minimize buffering even during high traffic. For example, Netflix's Open Connect serves videos from dedicated servers worldwide [8], ensuring smooth streaming for millions. Other platforms use similar methods, sometimes relying on third-party CDNs like Akamai and Cloudflare. All CDNs deploy caches worldwide to store content closer to users, reducing delays and improving Quality of Service (QoS). When a user requests content, a nearby CDN server delivers it. If the content is outdated or missing, the server fetches the latest version from the origin server [2].

Caching with HTTP is simple since content is stored and delivered directly. With HTTPS, encryption requires CDN servers to handle TLS/SSL termination [44]. When a client requests content, a CDN server processes the HTTPS connection. If the content is cached and fresh, it is delivered immediately. If not, the CDN fetches and caches it from the origin server over a secure connection. Before sending data to the client, the CDN re-encrypts it and establishes a new HTTPS connection [44][40].

Video QoE metrics. Several metrics exist for comparing the quality of compressed or reconstructed digital images or videos to the original uncompressed versions. These metrics can also measure the similarity between images/videos encoded at varying resolutions or bitrates. Commonly used metrics include peak signal-to-noise ratio (PSNR) [16], structural similarity index measure (SSIM) [42], and video multi-method assessment fusion (VMAF) [23]. In this study, we employ VMAF to evaluate our experiments. Nevertheless, our system is flexible and can be evaluated using other metrics like PSNR or SSIM if desired.

Encryption of content rather than the entire connection may raise concerns about exposing users' privacy such as leaks of viewing patterns. However, our work specifically addresses the streaming portion involving the download of video segments. All other control information can remain fully encrypted with HTTPS, maintaining efficiency with negligible overhead.

Research highlights that fully encrypted connections, like HTTPS, do not completely safeguard user privacy either. For instance, [11] demonstrated the detection of QoE-related issues from encrypted traffic with 92% accuracy. Machine learning models in [15, 27] predicted QoE metrics such as rebuffering and video resolution from YouTube HTTPS or QUIC traffic with 90% accuracy. Recent work [15] trained models on encrypted traffic from platforms like Netflix, YouTube, and Twitch, predicting startup delays with an error margin of just one second and resolutions with over 93% accuracy.

2.2 Attribute-based Encryption

Attribute-Based Encryption (ABE) is a public-key encryption method that manages keys and controls data access based on user attributes rather than identities. Decryption depends on the attributes linked to both the ciphertext and the user's private key. *This approach provides fine-grained access control and flexibility, as attributes and user's*

private keys can change without the need for content re-encryption. ABE works as follows.

Access Control Policy: Below is an example how a streaming service can enforce access control using Attribute-Based Encryption (ABE). **(a) Example Policy:** Only users with a *Premium subscription* and located in the *USA* or *Europe* can watch a movie. The service encrypts the video segments using ABE with this policy embedded in the ciphertext. **(b) Users and Attributes:** Each user has a set of attributes reflected in their decryption keys. **John**: {Subscription: Premium, Region: USA}, **Maria**: {Subscription: Basic, Region: Europe}. **(c) Encryption and Decryption Process:** The service provider encrypts the video segments using ABE, embedding the policy: *"Subscription must be Premium AND Region must be USA OR Europe."* The encrypted movie is stored and made available, but only users meeting the above policy (in this example, John) can decrypt it.

Revocation If the streaming service wants to revoke Maria's access (e.g., if her subscription expires), the service does not need to re-encrypt the video for all other users. Instead, it can either revoke Maria's decryption key or update the policy to exclude her attributes (e.g., change the region criteria or subscription level). It can also define an expiration time, after which each user needs to retrieve new keys. Revocation can be performed by embedding a time attribute in the key (e.g., valid until) as shown in [32–34]. If a trusted authority wants to revoke a key after a certain time, it can simply include a time-bound auxiliary component (e.g., valid until time t_1) in the decryption key. When time t_1 expires, the trusted authority updates this component. Users need both a permanent key and the current time-bound component to decrypt the ciphertext.

With ABE, access policies are embedded in data, enabling flexible updates. New regions or revoked access require only key updates, not content re-encryption. Users update keys when attributes change, reducing server interactions and eliminating TLS encryption, making the system more efficient and scalable.

2.3 ABE for video streaming:

Prior works have utilized ABE in the context of DASH video streaming [30] and media clouds [22]. However, these approaches employ ABE indirectly. They utilize Advanced Encryption Standard (AES) a symmetric encryption scheme to encrypt video content and ABE to encrypt and manage the AES keys. Evaluations in such approaches typically focus on encryption times for securing the AES key while varying the number of attributes in the ABE scheme.

However, this method has several drawbacks, primarily the reuse of the same AES key for encryption and decryption. It requires managing two keys: the AES key for content encryption and the ABE key for AES key protection. If the AES key is compromised, the ABE scheme becomes ineffective. Additionally, revoking user access requires re-encrypting all content with a new AES key and redistributing it, which is inefficient.

In contrast, our work applies ABE directly to the video segments, eliminating the need for a secondary AES key and simplifying key management. This approach avoids exposing the master key, as each user is issued a private key based on their access attributes. Content providers can use time-based ABE keys, requiring users to retrieve new keys periodically. This allows for easy revocation of access by excluding specific attributes from a user's key without needing to re-encrypt content or affect other users. For example, 4K access can be revoked for all basic plan users while retaining it for premium users. This approach ensures uninterrupted access for authorized users while effectively revoking permissions for others. This method enhances security, simplifies key management, and provides flexible, attribute-based access control.

2.4 Caching for video delivery

For video delivery, existing caching solutions fall broadly into two categories. In the first category, the algorithms exploit the sequential access patterns of video segments to make better caching decisions. For example, CoLeap [36] describes a pre-fetching-based cache algorithm that uses a QoE-oriented deep neural network model to decide the content to pre-fetch into the cache. Whereas [20] proposes performing smart prefetching by forecasting throughput values in the cache using previous client throughput data, thus optimizing resource utilization by pre-fetching video segments based on predicted demand, while improving client QoE. Another approach, presented in [24], introduces an integrated prefetching and caching proxy (ipac), a novel 0.5-competitive online pre-fetching algorithm specifically designed for HTTP-based adaptive streaming services like Netflix and YouTube, without requiring modifications to content servers or clients that improves byte-hit ratios and video rates. In the second category, the cache algorithms make decisions about specific qualities of each video segment to cache to maximize the cache hitrate for a single as well as a network of caches [5, 31]. These approaches are orthogonal to ABEVS, which does not focus on admission and eviction strategies for video caching. Since ABEVS is agnostic to these strategies, any of them can be integrated in the streaming system we propose.

3 System Architecture

In this section, we present the overall system architecture of ABEVS and introduce design choices for substantial components of this video distribution system. Figure 1 presents such an architecture that simplifies the scale of current video distribution systems while depicting the major components of such systems.

As shown in Fig. 1, our video distribution systems consist of two major organizations, Content Provider and CDN, and three major components, origin servers, CDN servers, and heterogeneous clients (laptops, tablets, smart TVs, phones, etc.). A Content Provider's role is to deliver digital video content, such as movies, TV shows, live streams, or user-generated videos, to users via CDNs. This includes but is not limited to providing a variety of encodings of each content (i.e., encoding ladder) and media descriptions (e.g., MPD) to support ABR streaming, in addition to managing digital rights.

In a video distribution system, all content that is offered in a Content Provider's library is kept on an origin server or a set of origin servers. (For reasons of simplicity and without loss of generality for our approach, we only show one in Fig. 1). CDN edge servers contact the origin server to retrieve content that is requested by clients in case that content is not already cached at the edge server. Based on CDN and Content Provider policies [25], client requests for content are directed to a specific edge server. Once a client request arrives at the edge server it can be immediately served if

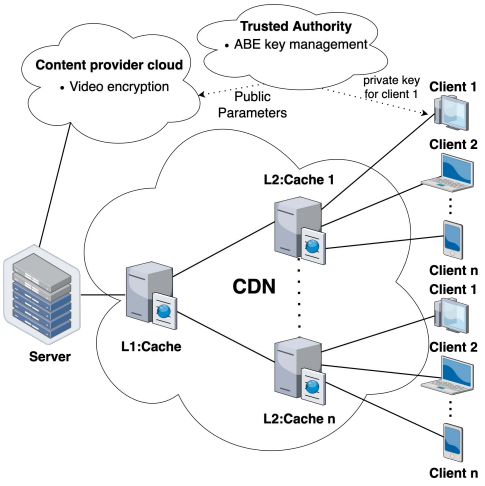


Figure 1: System Architecture

the content is already cached. If this condition is unmet, the edge server forwards the request to the origin server. Upon receipt of the content, the edge server decides if the content should be cached or not and forwards it to the client.

3.1 Integration into HTTP-based streaming

Popular video streaming applications rely on large-scale distribution infrastructures (i.e., CDNs) that support ABR streaming. We aim to create an ABE approach for such systems that integrates well with ABR. To achieve this goal, the following modifications to the standard DASH approach[1] are required.

ABE requires the generation of a Master Key and Public Key. The Master Key remains securely with the Trusted Authority (TA), which serves as the entity responsible for creating private keys for each user. Each private key is embedded with user-specific attributes, defining their access permissions. At the origin, the content provider uses the Public Key in conjunction with an ABE policy to encrypt video segments.

Second, our approach requires a mechanism that enables key exchange between the TA and Content Provider, and the TA and clients. This is different from current DASH-based approaches that make use of HTTPS, where the key exchange process is integrated into TLS. In ABEVS, origin servers and clients communicate with the TA that manages a set of keys used for content encryption and decryption using ABE. Note that caches do not engage in the key exchange process, as the content stored on them stays encrypted (no encryption or decryption is performed on the cache).

Finally, the ABR-capable player that runs on the client needs to be able to decrypt the video segments. In Sect. 3.2, we will show that it is straightforward to integrate the ABE decryption process into an existing DASH player. The encryption and decryption processes are discussed in the following section.

All other components required for DASH-based ABR streaming can be used as is, allowing for straightforward integration of ABEVS into existing streaming systems. A detailed description of the changes we made are described in more detail in Sect. 3.2.

For **Encryption**, we implemented a per-segment approach to ensure complete privacy for video content. The video is first encoded for DASH streaming, segmented into .m4s files, and accompanied by a Media Presentation Description (MPD) file. Each segment is then encrypted using the CPABE toolkit, an ABE framework (cf. Sect. 3.2). This encryption process uses a public key and an ABE policy, which defines the attributes required for content access, ensuring that only authorized users can decrypt and view the video. For instance, attributes may represent user roles (e.g., "premium" or "basic") or temporal restrictions (e.g., "access valid until 2024").

ABE introduces a size overhead with every encryption. After encrypting all segments, the MPD file is updated to account for the adjusted bitrates. This ensures compatibility with ABR, allowing clients to make optimal playback decisions.

The **Decryption** on the client side is performed per segment upon download, using the CPABE toolkit. Decryption requires the public key and the client's private key. If the client is authorized, their private key contains the attributes necessary to decrypt the segment successfully. This ensures a secure and seamless streaming experience while maintaining strict access control. Evaluation of encryption and decryption under ABE is discussed in Sect. 4.2

3.2 Prototype Implementation

The evaluation of ABEVS is based on a prototype we implemented making use of the following components:

CPABE Toolkit: To implement ABE, we utilized the CPABE toolkit [6], which is based on [7]. The process begins with the setup command, which generates the master key and public key. The public key is then used to encrypt files, specifying the ABE control policy attributes that determine who can decrypt the data. Master and public key are employed to generate user-specific private keys, which are associated with a set of attributes. A user can decrypt the file using their private key and the public key, provided their attributes satisfy the decryption policy defined during encryption.

AStream: We use the AStream DASH client, a Python-based emulated DASH player for performance evaluation. AStream supports various ABR algorithms, but for simplicity, we used the basic ABR algorithm. The implementation of the AStream client was based on [21], which is a nominally modified version of the original [19] implementation. In our work for HTTP-ABE, we made straightforward modifications to the AStream client by integrating a call to the CPABE toolkit, which performs attribute-based decryption of video segments, as described earlier in Sect.3.1. During streaming, when AStream successfully downloads a video segment, the CPABE toolkit is executed to decrypt the segment before it is written to the buffer as usual. The time spent on decryption is recorded and added to the download time to impact the overall download rate. The basic ABR algorithm then uses this combined time to decide the quality/bitrate of the next segment to download. Additionally, we modified the AStream client's download command by removing the constraint that drops a segment if its download time exceeds 10 seconds. This adjustment was made to mitigate the high rebuffering observed during our initial experiments, which was primarily caused by the strict bandwidth limitations imposed in the setup.

[1]We decided to integrate ABEVS in DASH since it is an open standard with many tools that are available to the research community.

4 Evaluation

In this section, we evaluate the performance of ABEVS concerning a set of metrics. As explained in detail in Sect. 4.1, these metrics focus on the performance of encryption and decryption, the performance of the origin server and caches, and the impact on the viewer's QoE. Note, that we can't create an infrastructure that matches the scale of today's video distribution systems. Nevertheless, we believe that results from our evaluation can be projected onto such systems to estimate the potential impact of our approach.

4.1 Setup & Metrics

4.1.1 Setup. **Videos:** For the evaluation, we use a version of the Big Buck Bunny (BBB) video in 1080p resolution in Y4M format from xiph.org [45]. The video was encoded using the H.264/AVC codec [18] in the MP4 container format and segmented in DASH with m4s segments, generating five different streams with varying quality bitrates and resolutions as denoted in Tab. 1 .

Table 1: Streams with varying bitrates and resolutions.

Stream	Bitrate (Kbps)	Resolution
1	500	426x240
2	1000	640x360
3	2000	654x480
4	3000	1280x720
5	5000	1920x1080

Each stream was encoded at a frame rate of 24 FPS with a GOP consisting of a keyframe (I-frame) every 48 frames, equating to one keyframe every 2 seconds. The video has a total duration of 10 minutes and was segmented into 4-second chunks, resulting in 150 segments per quality/bitrate stream. The MPD file for the BBB video includes 5 streaming qualities, each comprising 150 segments.

To further expand the dataset for evaluation, we created 4 additional copies of the BBB video's MPD file. For each quality segment, we generated four symbolic links, one for each additional copy of the MPD, effectively simulating five separate videos.

ABE: For evaluation, we applied ABE to every video segment within each quality bitrate of the video. For simplicity, the ABE policy includes a single attribute. [33] shows that decryption time and overhead increase linearly with number of attributes but overall impact on performance is still small. After encryption, we recalculated the new bitrates for each video quality and generated a new MPD for the encrypted videos, which reflects the updated bitrates for each encrypted stream. The encrypted bitrates we achieved were 503 Kbps at 426x240, 1035 Kbps at 640x360, 2011 Kbps at 654x480, 3005 Kbps at 1280x720, and 5001 Kbps at 1920x1080.

CloudLab: We conducted our experiments on the CloudLab testbed [12] with 2 different setups. **(a) Small-Scale Experiment:** The first experiment utilizes a simple topology consisting of seven nodes. One node hosts the origin server with the video content, another serves as a cache, and the remaining five nodes host streaming clients. Each client node run six individual streaming clients, resulting in a total of 30 streaming clients. To implement bandwidth limitations, we use Linux's tc (traffic control) utility [10]. Since the highest bitrate stream is 5 Mbps and each node hosts 6 clients, the maximum bandwidth requirement per client node is 30 Mbps, amounting to a total of 150 Mbps across all five nodes. A bandwidth limit of 30 Mbps was set between the cache and each client node to prevent contention on client-side links. However, to introduce significant stress on the server-cache link, we impose a stricter bandwidth limit of 50 Mbps, which is one-third of the total bandwidth provided to the clients. **(b) Large-Scale Hierarchical Experiment:** The second experiment scales up the setup with a caching hierarchy across eight nodes. One node hosts the origin server, one serves as the level-1 (L1) cache connecting to the server, and two level-2 (L2) caches connect to the L1 cache. Each L2 cache connects to two physical nodes, with each node hosting 20 streaming clients, totaling 80 clients. Given the larger scale, we analyzed CPU usage on CloudLab nodes to determine the maximum number of clients each could handle efficiently. We found that each AStream DASH client utilizes 100% of a single CPU core as it continuously logs streaming data. Since each physical node has 40 cores, we capped usage at 50% (20 cores) to ensure efficiency, allowing up to 20 streaming clients per node. Bandwidth configurations were as follows: 100 Mbps for the link between an L2 cache and each client node resulting in 200Mbps across all clients connected to an L2 cache (consistent with the logic used in the small-scale experiment). For the link between the L1 cache and each L2 cache a 70 Mbps limit is enforced (approx. one-third of the total bandwidth provided to all clients connected to an L2 cache). 140 Mbps was allocated for the link between the origin server and L1 cache which is the total sum of bandwidth allowed from the L1 cache to the L2 caches.

Apache HTTP Server: On the origin server, we run the Apache HTTP server [3] with TLS/SSL enabled to support HTTPS sessions.

Apache Traffic Server: To set up the caches between origin server and streaming clients, we use the Apache Traffic Server (ATS) [4]. We configure the origin server's location using the remapping plugin, which proxies all incoming requests to the origin server. In case of our large hierarchical setup, all incoming requests to the L2 cache are proxied to the parent L1 cache. We modify the default settings by disabling the RAM cache (setting it to 0) and varying the disk cache size as follows: 0 MB (disabled cache), 10 MB, 100 MB, 250 MB, 500 MB, 1000 MB, 1500 MB, and 2000 MB. For our large-scale hierarchical experiment, we also configured parent caching, where the L1 cache acts as the parent cache for both L2 caches. However, these settings do not support a 0 MB disk cache size (disabled cache); therefore, this configuration is excluded from the large-scale experiment. We configure ATS to enable SSL termination, ensuring encrypted communication between the origin server and the caches, as well as between the caches and every client.

AStream DASH Clients: On the client side, we use a modified version of AStream that incorporates ABE decryption for the HTTP-ABE experiments, while the regular AStream is used for HTTPS experiments. In the small-scale experiment, we create six instances of AStream clients on each of the five nodes, resulting in a total of 30 clients. To schedule the streaming sessions for these clients, we employ a Poisson distribution [28] with $\lambda = 30$, ensuring that each subsequent client begins streaming approximately 30 seconds after the previous one. For the large-scale experiment, similar AStream client instances are created on each node to support a total of 80 clients. To account for the increased number of clients and optimize the experiment's runtime, we adjust the Poisson distribution to use

$\lambda = 15$, reducing the interval between successive client streaming sessions. We use the Zipf distribution [29] to determine which copy of the BBB video each of the clients streams, with a skewness of 1.5 (some videos selected more than others). This simulates a real-world scenario where some videos are more popular than others, resulting in a higher frequency of streaming requests for those videos, and increasing the chances of being cached.

4.1.2 Evaluation Metrics. **CPU load:** We record CPU load to compare the performance of HTTP-ABE and HTTPS on both the caches and origin server. For this, we utilize Linux's pidstat[9] to monitor CPU usage in real time with a refresh rate of one second. For each experiment, we capture CPU usage throughout the entire session.

Hit Rate: We also compare the performance of HTTP-ABE and HTTPS in terms of the hit rates achieved for different cache sizes. To calculate the hit rate, we utilized ATS's monitoring logs, which record statistics on the number of cache hits and misses.

Midgress Traffic: Midgress traffic refers to the traffic that occurs when the cache experiences a miss and must retrieve content from the origin server. We compare our HTTP-ABE approach against HTTPS for midgress traffic while varying the cache size and completing the streaming sessions of all the clients. To measure midgress between the cache and the origin server, we use tcpdump [39], specifying the IP address of the origin server. This captures all the packets sent to and received from the origin server. Additionally, we used capinfos from Wireshark [43] to summarize the tcpdump file and provide the total data size of the packets transferred. For the large-scale experiment with a caching hierarchy, we also record the midgress traffic between each L2 cache and the L1 cache.

QoE-1 VMAF: From a streaming client's perspective, we evaluate whether HTTP-ABE or HTTPS provides a better QoE to the client by employing VMAF, a video quality metric developed by Netflix. VMAF combines human perception and machine learning to evaluate how closely a distorted video matches a reference video in terms of user experience, providing a score between 0 and 100, where higher scores indicate better quality and more enjoyable viewing [23]. For our experiment, we calculate average VMAF for each client by determining the VMAF score of each streamed segment's quality against the same segment from the highest quality stream available. After calculating the VMAF score for all segments, we compute the average score for each client. Subsequently, we average the VMAF scores of all the clients to assess overall video quality performance across different cache size configurations.

QOE-2 Rebuffering: Another metric we use to compare the QoE between HTTP-ABE and HTTPS is rebuffering. Rebuffering refers to the amount of time a video stream is interrupted due to the buffer being empty, preventing the player from continuing playback. We extract the rebuffering data for each client from the AStream DASH client's log, which records rebuffering times for each video segment. These times were summed to give the total rebuffering duration for each client's streaming session. We then average the total rebuffering times across all the clients for various cache sizes to evaluate overall performance.

4.2 Impact of Encryption under ABE

In this section, we compare the computational overhead of encryption and decryption using ABE and AES. For ABE, we utilize the CPABE toolkit with a single attribute to simplify the evaluation. AES encryption is performed in 128-bit Cipher Block Chaining (CBC) mode using the OpenSSL. As detailed in Sect. 4.1.1, the evaluation employs the BBB video encoded at five different quality bitrates, with all video segments for each quality level being encrypted.

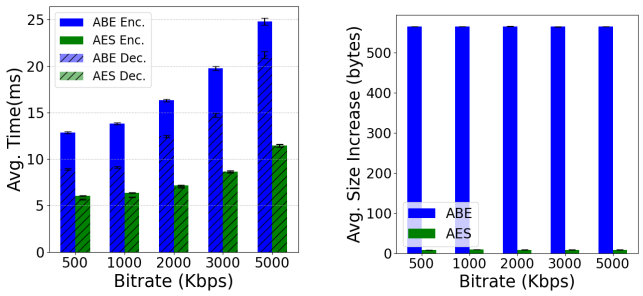


Figure 2: ABE vs AES comparison: Average Enc-Decryption times (left) and Average size increase (right)

Encryption & Decryption Total Runtime: To quantify the overhead of ABE compared to AES, we measured the Average encryption and decryption time for all the segment for each quality bitrate of the BBB video (Tab. 1). As shown in Fig. 2, encryption and decryption times rise with increasing quality bitrates. AES outperforms ABE, which aligns with our expectations due to the additional computational complexity associated with ABE policies.

The features of ABE outweigh its drawbacks, such as performing slower than AES. One key advantage is that ABE eliminates the need to reuse the same symmetric key for encryption and decryption across all users, enhancing security. Additionally, ABE allows for fine-grained access control, enabling certain users to access specific content while restricting others. Permissions can be easily revoked by removing specific attributes without impacting other users. This flexibility makes ABE a powerful tool for secure and scalable content distribution.

Average Size Change Per segment: we also analyzed the average size change of video segment across different quality bitrates, as shown in Fig. 2. Interestingly, the size change for ABE remains consistent across all bitrates. This uniformity stems from using a single attribute for encryption, as ABE operates based on access policies rather than segment size, rendering the size overhead introduced by ABE uniform. Additionally, the increase in encryption size overhead is proportional to the number of attributes used in the encryption policy, as demonstrated in [33]. In contrast, AES exhibits only a negligible size increase compared to ABE, reflecting its lower overhead in terms of encrypted data size.

While ABE introduces computational and size overhead compared to AES, our experimental results for DASH video streaming, both small and large-scale (Sect. 4.3), demonstrate that these overheads have no adverse effects on key performance metrics such as VMAF, midgress traffic, and cache hitrate. The performance remains on par with HTTP-TLS encryption while also providing numerous advantages, such as enhanced access control and simplified key management, as discussed earlier.

4.3 HTTP-ABE vs HTTPS

Here the evaluation focuses on assessing the performance of the overall distribution system by analyzing cache and origin server CPU load, cache hit rate, midgress traffic, and client QoE metrics.

We conducted three distinct experiments. A small-scale experiment and a large-scale hierarchical multi-cache experiment, both detailed in Sect. 4.1.1. The third experiment uses the same large-scale hierarchical setup but removes bandwidth limits to eliminate traffic congestion. This adjustment allows us to focus solely on comparing the performance of the two approaches—HTTP-ABE and HTTPS under unconstrained network conditions.

To collect this information for the small-scale experiment, we conducted a total of fifteen experiments: one for 0MB and two for each cache size (10MB, 100MB, 250MB, 500MB, 1000MB, 1500MB, and 2000MB). For the large-scale experiment, 0MB was excluded as it was not supported by the setup, resulting in a total of 14 experiments. In this case, the cache sizes varied simultaneously across all three caches (L1 and the two L2 caches). In each experiment setup, the first run was used as a warm-up to populate the cache whenever the cache size was changed. No metrics were collected during this initial run. The second run was executed immediately afterward, and the resulting metrics from this run are reported in the following sections. This approach was chosen to eliminate any start-up effects that could impact the collected metrics.

4.3.1 ***Small Scale Experiment***. :

CPU Load on Cache and Origin Server: Following the data collection on both cache and origin server, we plotted the average CPU usage over the whole session for each cache, accompanied by a 95% Confidence Interval, across various cache sizes.

Cache: Figure 3 shows the average CPU usage of the Cache for both HTTP-ABE (blue) and HTTPS (green). The cache's CPU load is consistently lower with HTTP-ABE compared to HTTPS. This difference becomes more pronounced as cache size increases, particularly from 1000 to 2000MB. At 2000MB, HTTP-ABE consumes, on average, only 57% of that of HTTPS (or 43% less than HTTPS).

As the cache size increases from 10 to 250MB, we do not observe a significant reduction in CPU load. This behavior can be attributed to the frequent caching and eviction processes that occur with smaller cache sizes. This also explains why the 0MB cache consumes a little less CPU than 10MB, as there is no caching or eviction process.

With increasing cache sizes, the difference in CPU load between HTTP-ABE and HTTPS widens. This is because HTTPS requires TLS termination, meaning all files retrieved from the origin server are encrypted with TLS. The cache must decrypt these files, cache them, and then re-encrypt them with TLS before forwarding to the client. With larger cache sizes, the cache under HTTPS benefits from a reduced CPU load by not having to request video segments from the origin server as frequently, nor decrypt and cache them. However, re-encryption is still needed for each segment sent to clients, even if the segment is already cached. In contrast, HTTP-ABE avoids this process entirely, as it can send cached segments directly to any requesting client without re-encryption, further emphasizing the efficiency of our approach with HTTP-ABE.

For a better depiction of the cache's CPU load behavior over the complete session, we plot CPU load over time as shown in Fig. 4 and Fig. 5 for 0MB and 2000MB cache sizes. The start of every new streaming client is denoted by a red dotted line on the time axis. We can see the rise in CPU load when most clients are streaming at the same time which is more pronounced. For 0MB cache, also there is not much difference between HTTP-ABE and HTTPS. In the 2000MB case, we see that the CPU load with HTTPS is the highest whereas the increase in the HTTP-ABE case is not very pronounced with a wide gap between both approaches.

Origin Server: Figure 6 presents the CPU load on the origin server with a 95% Confidence Interval. It is very clear from that our HTTP-ABE approach significantly reduces the server's CPU load compared to HTTPS. This is primarily because, with HTTPS, the server must encrypt each requested video segment with TLS every time it is requested. In contrast, our ABE approach allows us to store all video segments pre-encrypted. The same encrypted video segment can be used to satisfy multiple requests without the need for re-encryption. As for the receiving client, if authorized, will possess the necessary attributes to decrypt the segment.
From the graph, we can observe a similar trend to Fig.3, with a noticeable decrease in CPU load for cache sizes of 500MB and beyond. By the time the cache reaches 2000MB, the CPU load on the server becomes negligible, as the cache is large enough to store all the video segments, eliminating the need for repeated video segment retrieval from the origin server.

Comparison of Hit Rate: Figure 6 compares the hit rates across different cache sizes for both HTTP-ABE and HTTPS. We excluded the 10MB cache size for simplicity, as it would overlap with the 0MB case. The graph shows that hitrates increase with increasing cache sizes such as that on reaching 2000MB it becomes 1 (100%) which indicates that the cache size is large enough to store all the video segments. Hit rates for HTTP-ABE and HTTPS are almost similar throughout different cache sizes, although the pre-encrypted video segments in HTTP-ABE are little larger than the unencrypted segments in HTTPS as discussed earlier in Sect. 4.2. However this increase in segment size is minuscule and does not impact the cache's performance concerning hit rate.

Comparison of Midgress Traffic: Figure 6 illustrates that midgress traffic decreases as cache size increases, reaching 0 at a cache size of 2000MB, where the cache is large enough to store all segments. We also observe that midgress is similar for both HTTP-ABE and HTTPS across different cache sizes, following the same decreasing trend. Although one might expect midgress to be higher for HTTP-ABE due to the slightly larger segment sizes from encryption, this increase is too small to have a significant impact. This observation is similar to what we see in Fig. 6 for hitrates.

Comparison of QoE-1 VMAF: We compared the client's quality of experience between HTTP-ABE and HTTPS determining and comparing the VMAF segment by segment and presenting the average VMAF. Fig. 7 shows the average VMAF for 30 clients, along with a 95% confidence interval, across different cache sizes. The average VMAF is similar for both HTTP-ABE and HTTPS at each cache size, with an increase observed from the 500MB cache size onwards. This suggests that the size overhead of ABE-encrypted segments and the decryption time do not significantly affect the client's perceived video quality in our HTTP-ABE approach.

Comparison of QoE-2 Rebuffering: Figure 7 compares average rebuffering times for 30 clients across varying cache sizes, with a 95% confidence interval. Rebuffering is slightly higher for

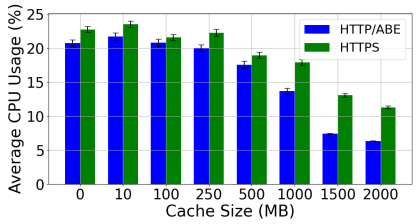


Figure 3: Cache's CPU usage for different cache sizes.

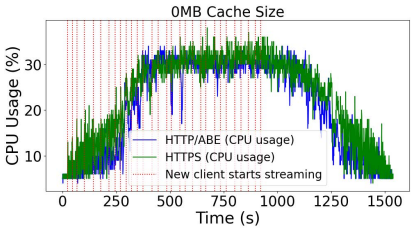


Figure 4: Cache's CPU usage for 0MB cache size

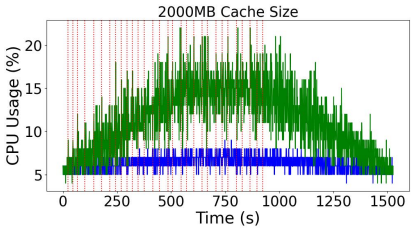
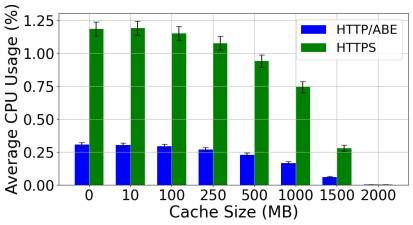
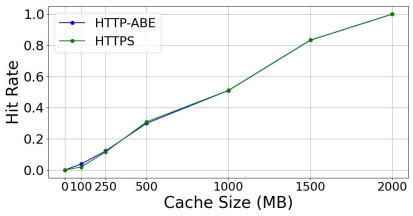


Figure 5: Cache's CPU usage for 2000MB cache size. Keys are similar to Fig. 4





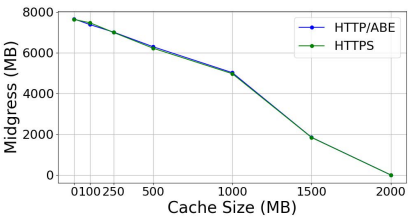


Figure 6: Origin server's CPU usage (left), Hitrates (middle) and Midgress (right) for different cache sizes.

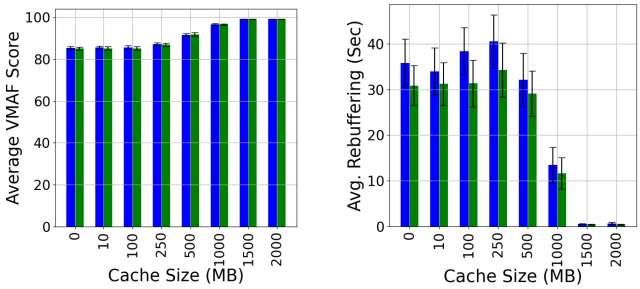


Figure 7: Average VMAF (left) and Average rebuffering (right) for different cache sizes, keys are similar to Fig. 6.

HTTP-ABE than HTTPS at smaller cache sizes, but the difference is statistically insignificant within the confidence intervals. For cache sizes of 500 MB and above, rebuffering times converge, indicating comparable performance for both.

4.3.2 ***Large Scale Hierarchal Experiment.*** :

CPU Load on level L1 and L2 Caches: Figure 8 shows the average CPU usage of L1 and L2 caches for HTTP-ABE and HTTPS. For L1, the CPU usage of HTTP-ABE is noticeably lower than HTTPS for cache sizes 10MB to 500MB, with HTTP-ABE using 25% less CPU at 0MB and 30% less at 500MB. Beyond 1000MB, CPU usage decreases for both, with minimal differences at 2000MB due to most segments being cached at both levels.

In contrast, L2 caches show slightly higher CPU usage for HTTP-ABE at smaller cache sizes (10 to 250MB), averaging 1–10% more than HTTPS. However, for larger cache sizes, HTTP-ABE consistently uses less CPU, with an average of 40% less than HTTPS at 2000MB. The higher CPU usage at smaller cache sizes is attributed to strict bandwidth limitations between L1 and L2 caches, leading to traffic congestion. This issue is further exacerbated by the larger segment sizes in HTTP-ABE and frequent caching and eviction at smaller cache sizes. We conducted an additional experiment without bandwidth limitations, as detailed in Sect. 4.3.3.

Comparison of Hit Rate: The hit rates for HTTP-ABE and HTTPS are similar, as shown in Figure 9. For both L2 caches, hit rates increase with cache size, reaching 100% at 2000MB when all segments are fully cached, following a trend similar to the small-scale experiment (Fig. 6).

However, the L1 cache behaves differently. Its hit rate initially rises with cache size, peaking at 500MB-1000MB. Beyond this point, it begins to decline, eventually reaching 0 at 2000MB when all requests are served by the L2 caches, both achieving a 100% hit rate. Since no requests are forwarded to the L1 cache at this stage, a hit rate of 0 is expected at 2000MB. A similar drop at 1500MB occurs because most segments are already cached at L2, resulting in very few misses. Specifically, L2-cache-2 achieves a 100% hit rate, while L2-cache-1 remains slightly lower at just over 80%. This minimizes the number of requests forwarded to the L1 cache. Since these requests are very rare they are extremely diverse and consequently, there is less opportunity for finding that exact content (segment & quality) on the L1 cache.

Comparison of Midgress Traffic: The midgress traffic results for HTTP-ABE are similar to HTTPS. As shown in Figure 10, the trend mirrors that of small-scale experiment (Sect.4.3.1, Fig.6). Midgress traffic for both the L1 cache (between origin and L1) and L2 caches (between L1 and L2) decreases as cache sizes increase, eventually reaching zero at 2000MB, where all segments are cached.

CPU load on Origin Server: Figure 11 illustrates the CPU load on the origin server, showing that our HTTP-ABE approach consistently requires significantly less CPU usage compared to

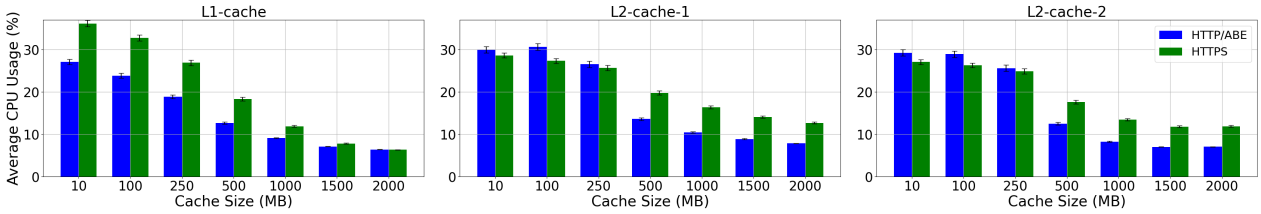


Figure 8: CPU usage of L1 and L2 caches for different cache sizes.

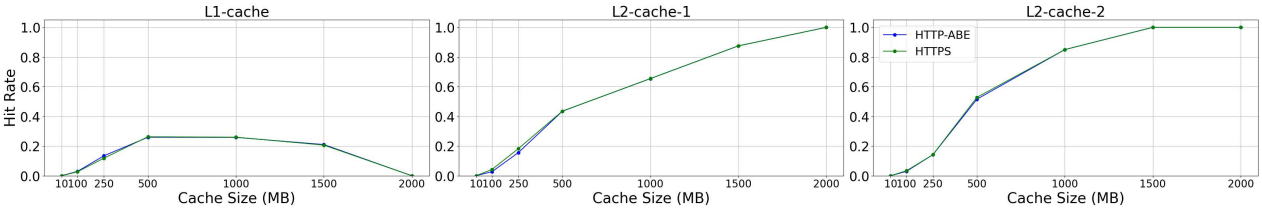


Figure 9: Hitrate of L1 and L2 caches for different cache sizes.

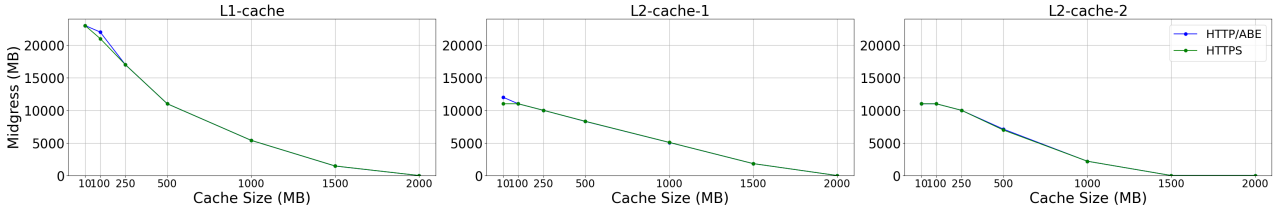


Figure 10: Midgress traffic of L1 and L2 caches for different cache sizes

HTTPS. This result aligns with the observations from the small-scale experiment (Sect.4.3.1, Fig.6).

Comparison of QoE-1 VMAF: Figure 11 shows the average VMAF scores achieved for 80 clients in this experimental setup. The results are consistent with the trends observed in the small-scale experiment, with the VMAF scores for our HTTP-ABE approach matching those of HTTPS.

Comparison of QoE-2 Rebuffering: As shown in Figure 11, rebuffering is higher for smaller cache sizes (10MB–250MB) and more pronounced for HTTP-ABE than HTTPS. This aligns with the CPU usage comparison for L2 caches in Figure 8, where HTTP-ABE exhibits slightly higher CPU usage. The strict bandwidth limitation between L1 and L2 caches intensifies congestion, exacerbated by frequent caching and evictions at smaller cache sizes. This leads to increased rebuffering and CPU usage, especially for HTTP-ABE, as its pre-encrypted segments are larger than those encrypted with HTTP-TLS, which uses AES. As shown in Sect. 4.2, ABE introduces significantly higher size overhead compared to AES.

Interestingly, for larger cache sizes (500MB-2000MB), where more segments are already cached, HTTP-ABE shows significantly lower CPU usage than HTTPS. This reduction in CPU usage corresponds with diminished rebuffering, and the differences between HTTP-ABE and HTTPS become negligible, further supporting our hypothesis about the impact of traffic congestion and cache behavior on rebuffering. To validate this hypothesis, we conducted the same large-scale hierarchical experiment without bandwidth limitations, which is detailed in the next section.

4.3.3 ***No Bandwidth Limitation Experiment:*** We conducted another version of the large-scale hierarchical experiment without any bandwidth limitations. By eliminating traffic congestion, this setup allows us to observe the clear impact of each approach—HTTP-ABE and HTTPS—on CPU usage of L1 and L2 caches and other performance metrics under ideal network conditions.

Comparison of CPU Load on L1 and L2 caches Without Traffic Congestion: Figure 12 illustrates the CPU load for L1 and L2 caches in the absence of bandwidth limitations, eliminating traffic congestion. For the L1 cache, the trend remains similar to Fig. 8 (with bandwidth limitations), but the difference between HTTP-ABE and HTTPS is more pronounced. HTTP-ABE shows up to 50% lower CPU usage than HTTPS for cache sizes of 10MB–250MB. As cache size increases, CPU usage declines for both approaches, and the difference diminishes since more segments are cached.

For the L2 caches, a significant reduction in CPU usage is observed for HTTP-ABE compared to HTTPS, with HTTP-ABE consistently using 40–45% less CPU. This contrasts sharply with the results under bandwidth-limited conditions, where HTTP-ABE

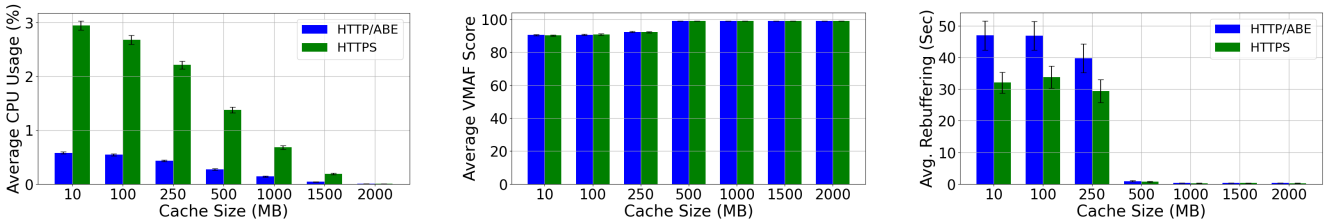


Figure 11: Origin server's CPU usage (left), Average VMAF (middle), and Average rebuffering (right) for large-scale experiment.

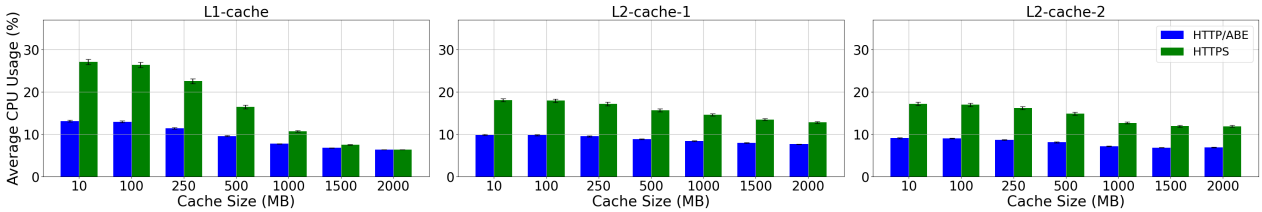


Figure 12: CPU usage of L1 and L2 caches for different cache sizes without traffic congestion.

showed slightly higher CPU usage for smaller cache sizes due to traffic congestion. These findings validate our earlier hypothesis that the elevated CPU usage of HTTP-ABE for smaller cache sizes was caused by bandwidth-induced traffic congestion coupled with slightly larger segments in case of HTTP-ABE.

Server CPU Usage, Hitrates, and Midgress: We observed no difference in these metrics compared to the results in Sec. 4.3.2.

VMAF and Rebuffering: Both approaches achieved full-quality streaming without bandwidth limitations, as expected. Rebuffering was limited to startup buffering, which is unavoidable.

Since these results do not provide any significant insights, their graphs have been excluded from this paper.

5 Discussion

In our streaming experiments, the highest quality bitrate used was 1920×1080 (FHD) at 5000 kbps. While FHD is common, modern services increasingly support higher resolutions such as 2K and 4K. Our Future work will include these higher-quality formats to align with advanced streaming practices.

Our experiments show that HTTP-ABE significantly lowers cache CPU usage compared to HTTPS while maintaining similar midgress traffic, cache hit rates, and QoE metrics. For higher-quality videos (e.g., 2K, 4K), we would adjust the server-cache bandwidth limit proportionally to the highest video bitrate, as done for FHD (one-third of total bandwidth).

In this scenario, the CPU usage gap between HTTP-ABE and HTTPS would likely widen with higher bitrates. HTTPS requires TLS termination and re-encryption at the cache, which becomes more CPU-intensive for larger video segments. HTTP-ABE, by offloading encryption, reduces cache computational load, a benefit that scales with higher resolutions.

Midgress traffic, cache hit rates, and QoE metrics are expected to remain largely unchanged even at higher resolutions, similar to as observed in bandwidth-limited (Sec. 4.3.1 & 4.3.2) and non-limited experiments (Sec. 4.3.3). Thus, while improving quality will increase bitrate and bandwidth usage, the performance difference between HTTP-ABE and HTTPS is unlikely to change significantly.

6 Conclusions

This paper proposes Attribute-Based Encryption (ABE) for HTTP-based video streaming as an alternative to HTTPS-TLS. ABE secures data at the segment level with access control policies, allowing only authorized clients to decrypt content. This approach removes reliance on transport-layer security, eliminates per-client sessions, reduces encryption/decryption overhead on caches, and enables flexible content retrieval.

We evaluated our HTTP-ABE approach using two testbed setups: (1) a small-scale deployment with an origin server, a cache, and 30 clients, and (2) a large-scale hierarchical setup with an origin server, an L1 cache, two L2 caches, and 80 clients. Results demonstrate that compared to HTTPS, HTTP-ABE reduced cache CPU load by up to 43% in the small-scale test and 50% in the large-scale setup, with the extent of reduction depending on bandwidth conditions.

While VMAF was similar for both, HTTP-ABE experienced slightly higher rebuffering under strict bandwidth and smaller caches, likely due to ABE segments being slightly larger than AES-encrypted ones in HTTPS-TLS.

Future work will explore integrating ABE with partial encryption techniques [14, 41], such as those used for privacy-preserving video (e.g., face encryption in public footage). This approach could enable controlled access, such as time-bound decryption keys for law enforcement in surveillance scenarios.

7 Acknowledgments

This material is based upon work supported by the National Science Foundation under Grant Numbers 1901137, 2106463, 2019163, and 2126148.

References

[1] Vijay K. Adhikari, Yang Guo, Fang Hao, Volker Hilt, Zhi-Li Zhang, Matteo Varvello, and Moritz Steiner. 2015. Measurement Study of Netflix, Hulu, and a Tale of Three CDNs. *IEEE/ACM Transactions on Networking* 23, 6 (2015), 1984–1997. doi:10.1109/TNET.2014.2354262

[2] Akamai. 2024. *What is a CDN (Content Delivery Network)?* Retrieved 2024 from https://www.akamai.com/glossary/what-is-a-cdn

[3] Apache. 2024. *Apache HTTP Server Project.* Retrieved 2024 from https://httpd.apache.org/

[4] Apache. 2024. *Apache Traffic Server.* Retrieved 2024 from https://trafficserver.apache.org/

[5] Andrea Araldo, Fabio Martignon, and Dario Rossi. 2016. Representation selection problem: Optimizing video delivery through caching. In *2016 IFIP Networking Conference (IFIP Networking) and Workshops*. IEEE, 323–331.

[6] John Bethencourt. 2011. *Using the cpabe Toolkit.* Retrieved 2024 from https://acsc.cs.utexas.edu/cpabe/tutorial.html

[7] John Bethencourt, Amit Sahai, and Brent Waters. 2007. Ciphertext-Policy Attribute-Based Encryption. In *2007 IEEE Symposium on Security and Privacy (SP '07)*. 321–334. doi:10.1109/SP.2007.11

[8] Timm Böttger, Felix Cuadrado, Gareth Tyson, Ignacio Castro, and Steve Uhlig. 2018. Open Connect Everywhere: A Glimpse at the Internet Ecosystem through the Lens of the Netflix CDN. *SIGCOMM Comput. Commun. Rev.* 48, 1 (apr 2018), 28–34. doi:10.1145/3211852.3211857

[9] die.net. 2024. *pidstat - Report statistics for Linux tasks.* https://linux.die.net/man/1/pidstat

[10] die.net. 2024. *tc - show / manipulate traffic control settings.* https://linux.die.net/man/8/tc

[11] Giorgos Dimopoulos, Ilias Leontiadis, Pere Barlet-Ros, and Konstantina Papagiannaki. 2016. Measuring Video QoE from Encrypted Traffic. In *Proceedings of the 2016 Internet Measurement Conference* (Santa Monica, California, USA) *(IMC '16)*. Association for Computing Machinery, New York, NY, USA, 513–526. doi:10.1145/2987443.2987459

[12] Dmitry Duplyakin, Robert Ricci, Aleksander Maricq, Gary Wong, Jonathon Duerig, Eric Eide, Leigh Stoller, Mike Hibler, David Johnson, Kirk Webb, Aditya Akella, Kuangching Wang, Glenn Ricart, Larry Landweber, Chip Elliott, Michael Zink, Emmanuel Cecchet, Snigdhaswin Kar, and Prabodh Mishra. 2019. The Design and Operation of CloudLab. In *2019 USENIX Annual Technical Conference (USENIX ATC 19)*. USENIX Association, Renton, WA, 1–14. https://www.usenix.org/conference/atc19/presentation/duplyakin

[13] Roy T. Fielding, Mark Nottingham, and Julian Reschke. 2022. HTTP Semantics. RFC 9110. doi:10.17487/RFC9110

[14] Christoph Gerhardt, Patrick Aichroth, and Sebastian Mann. 2017. Selective face encryption in H.264 encoded videos. In *2017 IEEE Visual Communications and Image Processing (VCIP)*. 1–4. doi:10.1109/VCIP.2017.8305040

[15] Craig Gutterman, Katherine Guo, Sarthak Arora, Xiaoyang Wang, Les Wu, Ethan Katz-Bassett, and Gil Zussman. 2019. Requet: real-time QoE detection for encrypted YouTube traffic. In *Proceedings of the 10th ACM Multimedia Systems Conference* (Amherst, Massachusetts) *(MMSys '19)*. Association for Computing Machinery, New York, NY, USA, 48–59. doi:10.1145/3304109.3306226

[16] Quan Huynh-Thu and Mohammed Ghanbari. 2008. The scope of validity of PSNR in image/video quality assessment. *Electronics letters* 44, 13 (2008), 800–801.

[17] Apple Inc. 2024. Apple HTTP Live Streaming. "https://developer.apple.com/streaming/".

[18] ITU-T. 2021. *H.264 : Advanced video coding for generic audiovisual services.* https://handle.itu.int/11.1002/1000/14659

[19] Parikshit Juluri, Venkatesh Tamarapalli, and Deep Medhi. 2015. SARA: Segment aware rate adaptation algorithm for dynamic adaptive streaming over HTTP. In *2015 IEEE International Conference on Communication Workshop (ICCW)*. 1765–1770. doi:10.1109/ICCW.2015.7247436

[20] Sheyda Kiani Mehr, Jogalekar Prasad, Rao Praveen, Li Zhu, Beard Cory, and Choi Baek-Yong. 2021. *Video Streaming Quality of Experience (QoE): In-Network Cache Prefetching and Moving QoE Models.* Ph. D. Dissertation. Advisor(s) Deep, Medhi,. AAI28544902.

[21] Umakant Kulkarni, Yufeng Chen, Patrick Melampy, and Sonia Fahmy. 2023. Toward QoE-based Routing Path Selection. In *2023 IEEE 24th International Conference on High Performance Switching and Routing (HPSR)*. 114–119. doi:10.1109/HPSR57248.2023.10147938

[22] Hao Li, Lianbing Deng, Cheng Yang, and Jianbo Liu. 2020. An enhanced media ciphertext-policy attribute-based encryption algorithm on media cloud. *International Journal of Distributed Sensor Networks* 16, 2 (2020), 1550147720908196. doi:10.1177/1550147720908196 arXiv:https://doi.org/10.1177/1550147720908196

[23] Zhi Li, Anne Aaron, Ioannis Katsavounidis, Anush Moorthy, and Megha Manohara. 2016. *Toward A Practical Perceptual Video Quality Metric.* Retrieved 2024 from https://netflixtechblog.com/toward-a-practical-perceptual-video-quality-metric-653f208b9652

[24] Ke Liang, Jia Hao, Roger Zimmermann, and David K. Y. Yau. 2015. Integrated prefetching and caching for adaptive video streaming over HTTP: an online approach. In *Proceedings of the 6th ACM Multimedia Systems Conference* (Portland, Oregon) *(MMSys '15)*. Association for Computing Machinery, New York, NY, USA, 142–152. doi:10.1145/2713168.2713181

[25] Bruce M Maggs and Ramesh K Sitaraman. 2015. Algorithmic nuggets in content delivery. *ACM SIGCOMM Computer Communication Review* 45, 3 (2015), 52–66.

[26] Hongzi Mao, Ravi Netravali, and Mohammad Alizadeh. 2017. Neural adaptive video streaming with pensieve. In *Proceedings of the conference of the ACM special interest group on data communication*. 197–210.

[27] M. Hammad Mazhar and Zubair Shafiq. 2018. Real-time Video Quality of Experience Monitoring for HTTPS and QUIC. In *IEEE INFOCOM 2018 - IEEE Conference on Computer Communications*. 1331–1339. doi:10.1109/INFOCOM.2018.8486321

[28] NumPy. 2024. *numpy.random.poisson.* Retrieved 2024 from https://numpy.org/doc/stable/reference/random/generated/numpy.random.poisson.html

[29] NumPy. 2024. *numpy.random.zipf.* Retrieved 2024 from https://numpy.org/doc/stable/reference/random/generated/numpy.random.zipf.html

[30] Claudiu Olteanu, Mihai Bucicoiu, and Marius Popa. 2016. SStream: An Infrastructure for Streaming Multimedia Content Efficiently and Securely in a Heterogeneous Environment. In *Exploring Services Science*, Theodor Borangiu, Monica Dragoicea, and Henriqueta Nóvoa (Eds.). Springer International Publishing, Cham, 343–354.

[31] Konstantinos Poularakis, George Iosifidis, Antonios Argyriou, Iordanis Koutsopoulos, and Leandros Tassiulas. 2016. Caching and operator cooperation policies for layered video content delivery. In *IEEE INFOCOM 2016-The 35th Annual IEEE International Conference on Computer Communications*. IEEE, 1–9.

[32] David Reddick, F. Alex Feltus, and Susmit Shannigrahi. 2022. Case Study of Attribute Based Access Control for Genomics Data Using Named Data Networking. In *IEEE CCNC*.

[33] David Reddick, Justin Presley, Frank Alex Feltus, and Susmit Shannigrahi. 2022. WiP: AABAC-Automated Attribute Based Access Control for Genomics Data. In *Proceedings of the 27th ACM on Symposium on Access Control Models and Technologies*. 217–222.

[34] David Eugene Reddick. 2021. *Design of an NDN-Based Data Distribution Infrastructure for Genomics Data Secured with a Novel Ledger Access System Founded on Attribute-Based Encryption.* Master's thesis. Tennessee Technological University.

[35] Eric Rescorla. 2018. The Transport Layer Security (TLS) Protocol Version 1.3. RFC 8446. doi:10.17487/RFC8446

[36] Wanxin Shi, Chao Wang, Yong Jiang, Qing Li, Gengbiao Shen, and Gabriel-Miro Muntean. 2020. CoLEAP: Cooperative learning-based edge scheme with caching and prefetching for DASH video delivery. *IEEE Transactions on Multimedia* 23 (2020), 3631–3645.

[37] Iraj Sodagar. 2011. The mpeg-dash standard for multimedia streaming over the internet. *IEEE multimedia* 18, 4 (2011), 62–67.

[38] Kevin Spiteri, Rahul Urgaonkar, and Ramesh K Sitaraman. 2020. BOLA: Near-optimal bitrate adaptation for online videos. *IEEE/ACM Transactions on Networking* 28, 4 (2020), 1698–1711.

[39] tcpdump. 2024. *tcpdump, a powerful command-line packet analyzer.* Retrieved 2024 from https://www.tcpdump.org/

[40] USAVPS. 2023. *CDN Basic – How CDNs handle encrypted content.* Retrieved 2024 from https://usavps.com/blog/17594/

[41] Junjue Wang, Brandon Amos, Anupam Das, Padmanabhan Pillai, Norman Sadeh, and Mahadev Satyanarayanan. 2017. A Scalable and Privacy-Aware IoT Service for Live Video Analytics. In *Proceedings of the 8th ACM on Multimedia Systems Conference* (Taipei, Taiwan) *(MMSys'17)*. Association for Computing Machinery, New York, NY, USA, 38–49. doi:10.1145/3083187.3083192

[42] Zhou Wang, Alan C Bovik, Hamid R Sheikh, and Eero P Simoncelli. 2004. Image quality assessment: from error visibility to structural similarity. *IEEE transactions on image processing* 13, 4 (2004), 600–612.

[43] Wireshark. 2024. *capinfos(1) Manual Page.* Retrieved 2024 from https://www.wireshark.org/docs/man-pages/capinfos.html

[44] Rui Xin, Shihan Lin, and Xiaowei Yang. 2023. Quantifying User Password Exposure to Third-Party CDNs. In *Passive and Active Measurement: 24th International Conference, PAM 2023, Virtual Event, March 21–23, 2023, Proceedings*. Springer-Verlag, Berlin, Heidelberg, 652–668. doi:10.1007/978-3-031-28486-1_27

[45] Xiph. 2024. *the xiph open source community.* Retrieved 2024 from https://xiph.org/



COMPACT: Content-aware Multipath Live Video Streaming for Online Classes using Video Tiles

Shubham Chaudhary, Navneet Mishra, Keshav Gambhir, Tanmay Rajore
Arani Bhattacharya, Mukulika Maity
Indraprastha Institute of Information Technology Delhi, India
{shubhamch, navneet23049, keshav19249, tanmay19118, arani, mukulika}@iiitd.ac.in

Abstract

The growing popularity of live online classes, even in remote areas, stresses the need for a good and seamless quality of experience to enhance learning. However, these bandwidth-hungry applications challenge the current cellular networks to maintain consistent bandwidth and latency. In this work, we, therefore, propose using the collaboration of multiple devices with their individual cellular networks to support such live video streaming. We design a content-aware system COMPACT that splits video into foreground and background using video tiles (independently encoded spatial blocks) and streams them over different paths. COMPACT depends on its scheduler, which exhaustively searches for the best quality based on the network estimates. We extensively evaluate our system using network traces while walking and traveling on the bus or car. Compared to the single path, COMPACT manages to reduce the median stall and E2E lag by 70.6% and 28.57%, and the tail stall and lag by 83.9% and ≈ 80% on a bus trace. Furthermore, we performed a live experiment to test COMPACT on the actual cellular network.

CCS Concepts

• **Information systems** → **Multimedia streaming**.

Keywords

Multipath, Live Video Streaming, Video Tiles, Live Online Class

ACM Reference Format:
Shubham Chaudhary, Navneet Mishra, Keshav Gambhir, Tanmay Rajore, Arani Bhattacharya, Mukulika Maity. 2025. COMPACT: Content-aware Multipath Live Video Streaming for Online Classes using Video Tiles. In *ACM Multimedia Systems Conference 2025 (MMSys'25), March 31-April 4, 2025, Stellenbosch, South Africa.* ACM, New York, NY, USA, 13 pages. https://doi.org/10.1145/3712676.3714451

1 Introduction

In recent years, the popularity of online learning has increased [16, 27, 65, 95]. Students prefer using handheld devices for online learning [1, 15] primarily due to the availability of easier internet access on smartphones. Mobile learning enables students of even remote areas to listen to lectures from renowned universities and/or colleges. Live video streaming is the technology serving online learning [22, 26, 27, 34, 69, 90], supported by platforms such as YouTube and Twitch. Such services typically provide video content with a latency of the order of 5-10 seconds [8]. However, this growth in live video streaming for online learning comes with the challenge of providing students with a good quality of experience (QoE) to enhance their learning. Since most live-streamed online classes are interactive, where the students can ask questions either verbally or through chat or Q&A, all the modalities (text, audio, and video) should be in synchrony to maintain the live experience [26, 34]. The current wireless infrastructure falls short of catering to these bandwidth-hungry video applications. Although the newer generations of wireless technologies like 5G, 6G, and WiFi-7 promise to offer high bandwidth [18, 93, 97, 98], live video streaming still suffers from high latency and poor QoE [40, 41, 55, 57].

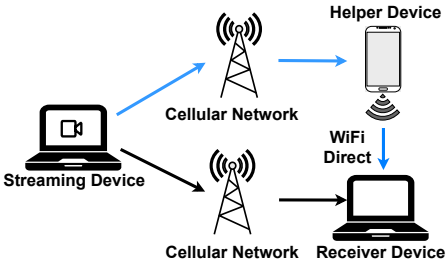


Figure 1: A schematic view of COMPACT indicating in blue the lines for additional connectivity we intend to add.

Several works [32, 33, 82, 83, 86, 87, 89, 91, 99] show the potential of using multiple connections to address the shortcomings of video streaming. Multipath streaming permits leveraging multiple available paths in conjunction. In addition, if one path goes down or has poor bandwidth (BW), the other can compensate, given it has relatively superior network characteristics [61]. Most students have access to multiple devices in their homes [3, 5, 21, 70, 77]. Live video streaming can benefit by aggregating all these devices' bandwidths. We show such a system in Fig. 1 where the user has two devices[1], a phone and a laptop connected to the Internet through two different Internet Service Providers (ISPs). Conventional techniques use only one device (called primary) at a time to retrieve video packets (shown with black arrows). Having another supporting device (called a helper) in parallel facilitates multipath transmission (shown with blue arrows). The user's devices communicate within themselves using WiFi Direct/WiFi Peer-to-Peer connection. The use case for such a multi-device setup has been utilized and

[1]As a proof of concept, this paper shows multipath with two devices only, which can be extended beyond. We leave it for future consideration.

Permission to make digital or hard copies of all or part of this work for personal or classroom use is granted without fee provided that copies are not made or distributed for profit or commercial advantage and that copies bear this notice and the full citation on the first page. Copyrights for components of this work owned by others than the author(s) must be honored. Abstracting with credit is permitted. To copy otherwise, or republish, to post on servers or to redistribute to lists, requires prior specific permission and/or a fee. Request permissions from permissions@acm.org.
MMSys'25, Stellenbosch, South Africa
© 2025 Copyright held by the owner/author(s). Publication rights licensed to ACM.
ACM ISBN 979-8-4007-1467-2/25/03
https://doi.org/10.1145/3712676.3714451

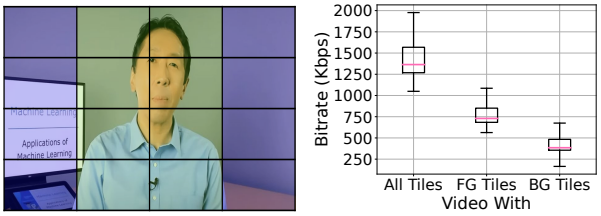
incentivized by MPBond [100], MicroCast [49], and OASIS [46], but they did not use it for live streaming.

Current multipath live streaming techniques utilize the extra bandwidth by distributing packets over the paths without considering the video content. Converge [32] used multipath WebRTC to meliorate video conferencing. Furthermore, they showed that if the current multipath protocols like MPTCP [66, 67, 85], MPQUIC [29–31], and MPRTP [72] is used as is for live video conferencing, they perform worse than single-path due to head-of-line (HoL) blocking caused by out-of-order packet arrival. TwinStar [83] sends frames in a round-robin fashion on the available paths, allocating bitrate jointly while encoding. AggDeliv [33] optimizes congestion control of the multiple paths to adapt them for mobile live video delivery. However, none of these techniques considers the relevance of spatial content. Therefore, these systems encode the whole content at a lower bitrate (BR) to accommodate it within the available BW.

The Case for Content-Aware Streaming: We first argue that the content of video matters in terms of how packets should be scheduled over a multipath network. To illustrate this argument, we consider the case of a classroom where an instructor is teaching. The body, face, and gestures of the instructor are what the students focus on the most, indicated by the fact that it has the highest saliency [50, 78, 84]. Thus, these portions of the frames/video should have the highest quality. Therefore, streaming should be content-aware. Note that just cropping the face or blurring the background is not prudent as it impairs the QoE. This is because online classes not only have scenes with the instructor talking but also slides and whiteboards. Besides, occasionally, we even see instructors' videos hovering over the background (BG) with slides. These use cases make sending and rendering the BG with the foreground (FG) indispensable. However, since the frequency of changes in the BG is lower and delays in rendering the changes in the background are not as significant as those in the instructor's body, face, and gestures, rendering an outdated BG suffices. Several works like [17, 20, 28, 48, 88] utilized content awareness in sports, lectures, gaming, and surgical videos. However, they focused only on playback speed control based on the content type and not on scheduling over multipath networks.

Our Solution: To resolve the problem of unreliable cellular networks, we propose Compact that uses content-aware multipath live video streaming for live online classes where we stream spatially segregated video content for transmission over different paths. For example, considering the case of the classroom, the portion of the video containing the teacher's face/body, the foreground (FG), is streamed at high quality through the path with the highest bandwidth and least latency (referred to as the primary path). The relatively static background (BG) content can be sent through the path with lower throughput, referred to as the secondary path. Compact reasons rendering the current FG with the previous BG in case the BG gets delayed, as the changes in BG are less frequent. Since modern smartphones have two interfaces suitable for video streaming – cellular and WiFi, it is sufficient to consider only a binary choice (FG and BG) to obtain an efficient solution.

To segregate a video into two different categories, FG and BG, we use High-Efficiency Video Codec (HEVC/H.265) [76] as it supports creating *spatially independent rectangular blocks in a video called tiles* (shown in Fig. 2a). Tiles can be removed, swapped, and/or



(a) (b)

Figure 2: (a) Video with 4×4 tiles. The foreground is highlighted in green and the background in blue. The video is publicly available under Creative Commons License [54]. (b) The bitrate of the full video is ≈1.7× and ≈4× the bitrate of foreground and background content, respectively.

added independently in real-time without affecting the rest of the video, provided the initial encoding uses tiles. Note that segregating only the speaker is computationally intensive, requiring image segmentation. Due to the changing BG, albeit with less frequency, BG subtraction gives poor accuracy. Using tiles gives us coarser yet faster FG detection using a face-detection model. We create two subsets of tiles; one has only the FG, typically containing the face and body of the speaker, while the other contains only the BG.

Content-aware Scheduler: Choosing the right path for the set of tiles is crucial because a wrong decision can lead to HoL blocking due to out-of-order delivery of tiles. For instance, if the BG tiles reach earlier, we would still have to wait for the FG tiles, causing unnecessary stalls. We found in our experiments that using conventional schedulers like MinRTT and Musher [68], used in MPQUIC and MPTCP, respectively, do not improve but lead to worse QoE (§2.3). Because the number of tiles to stream should depend on the asymmetric path characteristics. In cases where the FG set has fewer tiles than the BG set, a few tiles from the BG can be shifted to the FG if the path over which FG is being streamed has sufficient BW. Since conventional schedulers lack such knowledge, they fail to leverage the full potential of multiple available paths. Furthermore, the quality of the two tilesets can be varied independently based on the BW if streaming at a certain quality level is not possible.

Compact utilizes its own scheduler to fulfill the above requirements. The scheduler keeps track of the round-trip-time (RTT) and available bandwidth (BW) estimates of each path (we use two paths in this paper). Depending on the RTT and the BW estimates, it either shifts tiles from BG to FG, varies quality level, or does both. It finds an appropriate schedule by maximizing the utility function.

We implement Compact at the user level in Java using Stream Control Transmission Protocol (SCTP) [59, 75] under the hood. We use SCTP due to its in-order packet delivery with configurable retransmissions at the stream level to fix transport-level HoL blocking. Several prior works [86, 87, 89, 91] used it for video streaming applications. We evaluate Compact on the real network and in different scenarios using traces replayed using Mahimahi [60]. We find that Compact performs superior to the baselines, improving the median stall and E2E lag by 70.6% and 28.57%, and the tail stall and lag by 83.9% and ≈ 80% on a bus trace.

Contributions: Our main contributions are as follows:

(1) We propose to leverage the different levels of importance of the spatial regions in streaming videos for online classes. This helps

split the videos into foreground and background categories, which can be streamed through different network paths. To the best of our knowledge, we are the first to utilize such spatial segregation for multipath live video streaming.

(2) We implement our own content and path-aware scheduler to address the shortcomings of the past multipath schedulers. COMPACT is open-sourced on Github[2] to spur further research.

(3) We evaluate COMPACT on diverse network conditions, including a live test on the actual cellular network. It reduces the median stall by 3.4× and the 99*th* percentile tail stall by 6.2× compared to the single path. It further improves the median E2E lag by 28.57% and tail lag by ≈ 80%.

2 Background & Motivation

In this section, we first briefly discuss video tiles and their use cases in live video streaming. We then discuss the challenges of using tiles for multipath live streaming. Lastly, we study the use of conventional packet-level multipath schedulers at the tile level.

2.1 Video Tiles for Streaming

Modern video compression standards like HEVC [76], AV1 [39], and VVC [24] support encoding videos by splitting them spatially into a grid of rectangular blocks referred to as *tiles* (Fig. 2a). These tiles are treated as independent entities, i.e., they can be manipulated without affecting the rest of the video. Due to the ease and flexibility in manipulation, tiles are widely used in 360° and VR videos [42, 64, 74, 94]. Tiles also have the advantage that removing them reduces video filesizes [25]. We show this in Fig. 4, where shifting a certain number of tiles from BG to FG increases FG filesize while decreasing BG filesize. This allows adjusting the number of tiles in each path to handle the differences in network bandwidths. Since tiles allow independent decoding, the FG can be rendered without waiting for the BG to arrive, reducing the overall latency. Tiles can also be encoded at different quantization levels, thus allowing another parameter to adjust according to the available network BW.

The separation of spatial content into FG and BG tiles reduces the overall video streaming bandwidth requirement. We show this in Fig. 2b where for a total of 400 video segments (a bunch of frames when encoded is called a video segment), the FG tiles bitrate (BR) is ≈ 1.7×, and BG tiles bitrate (BR) is ≈ 4× lesser than the video with all tiles. This reduction in video bitrate demonstrates the utility of coupling tile-based streaming with multipath networking. This implies that if streaming a high-quality video over a single path is not viable, resorting to FG and BG tiles for multipath streaming makes it feasible without losing out on the quality of experience.

2.2 Tile-level versus Packet-level Streaming

Relying on tile-level streaming for live video delivery helps eliminate the packet-level HoL-blocking. We intuitively show this using Fig. 3. Consider the case of conventional multipath streaming at packet-level, here the encoded video segments are bifurcated into smaller packets which are distributed across the multiple paths based on the network characteristics. At the other end (receiver-side), all packets should be received to decode and render the streamed segment. If any packet gets delayed, lost, or arrives out

[2]https://github.com/shubhamchdhary/COMPACT

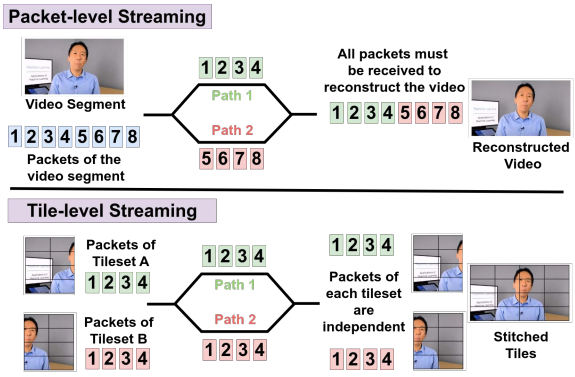


Figure 3: Difference between tile and packet-level multipath streaming. Streaming at the packet level distributes packets across the paths and mandates reception of all the video packets for video reconstruction. While in tile-level streaming, all packets of a tile-set are streamed on a single path, making it independently receivable and decodable.

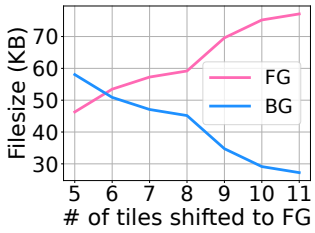


Figure 4: Shifting tiles from background (BG) to foreground (FG) reduces BG video size proportionally.

of order, the whole video reconstruction is hampered, leading to video stall/freeze. Moreover, if the paths are highly heterogeneous, the packets of the worst path will likely get delayed. Thus, besides reducing the video quality and scheduling fewer packets on the worst path, there is no way to reduce video stalls.

Piggybacking on tile-level streaming helps resolve the packet-level HoL-blocking. This is true because, unlike distributing packets across multiple paths, in tile-level streaming, all the packets of a tile are streamed through one single path. This eliminates the possibility of HoL-blocking due to inter-path packet delay. We show this in Fig. 3 where the video is divided into two tilesets, A and B. *Tileset A* is streamed over *Path* 1 while *Tileset B* is streamed over *Path* 2. All the packets of a tileset are sent over a single path. Since tiles are independently decodable, even if the paths are highly heterogeneous, the tileset reaching first can be rendered without waiting for the other. Thus, it reduces the video stalls and allows FG and BG to be encoded at different quality levels.

2.3 Limitations of Conventional Schedulers

Fig. 5 demonstrates the stall duration and the application level end-to-end (E2E) lag for different packet level and tile level schedulers (refer to §4 for details of the metrics and the schedulers' implementation). We run the experiment on a 5-minute lecture video. We replayed Pensieve's [58] open-sourced bus trace files (bandwidth patterns shown in Fig. 11b) on both paths. We note that all the packet level schedulers PMin (MinRTT), PBF (Balanced Subflow

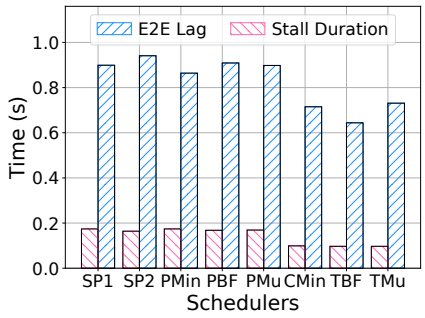


Figure 5: Median end-to-end (E2E) application-level lag and video stall duration of different packet-level (PMin, PBF, and PMu) and tile-level (CMin, TBF, and TMu) multipath schedulers compared to individual single paths (SP1 and SP2).

Completion), and PMu (Musher) incur the highest median lag and introduce median stall similar to single path streaming. This is evident because the path heterogeneity causes HoL blocking due to out-of-order packet arrivals. Thus, conventional packet-level schedulers fail to fit for live video streaming.

If these schedulers are modified to work at the tile level, then all of them (CMin, TBF, TMu) perform better than packet-level schedulers (PMin, PBF, PMu) and single paths (SP1 and SP2) in terms of both E2E lag and stall. However, this improvement is marginal because of the tile-level HoL blocking. Although the tiles sent over the better path reach earlier, it has to wait for the rest to arrive, causing unnecessary lag. This problem persists even if the scheduler is content-aware, as illustrated by CMin (Content-aware tile-level MinRTT) being worse than TBF (tile-level balanced subflow). This example shows that being content-aware is not enough. This is due to the dependence of the FG on the BG. TBF achieves a superior tradeoff among all conventional schedulers. It manages to do so because it tries to equalize the completion times (time to send all the tiles of a set) of tiles across paths. TMu (tile-level Musher) behaves similar to CMin even though it distributes tiles in proportion to the available bandwidths. Therefore, content-aware schedulers need to distribute the tiles well to minimize the completion time.

2.4 Challenges of Using Tiles

While using tiles for multipath live streaming has advantages, three major challenges are associated with tile-level live streaming.

C1) Content-Obliviousness of Tiles: Tiles themselves do not have any knowledge of their content. Using an arbitrary splitting technique to get FG and BG sets leads to suboptimal performance (discussed in §5). Since there is no foreground and background, the receiver device is, therefore, obliged to wait for all the tiles to arrive before stitching. This causes unwanted HoL blocking leading to high E2E lag and stall. Thus, it is imperative to divide tiles depending on the content so that if only the FG tiles reach, it is renderable with the recently cached BG tiles.

C2) Requirement of Fair Allocation of Tiles: After creating two sets of tiles, the BG can have more tiles than the FG. Since we stream BG over the low bandwidth path, it may not fit within the bandwidth limit. In such cases, instead of encoding the BG tiles with a lower bitrate, some of its tiles can be allocated to FG because shifting reduces filesize (Fig. 4). Such relocation of tiles gives the opportunity to stream BG tiles at relatively higher video quality as they are shifted to the FG tileset. It is, therefore, crucial for the system designer to consider appropriate tile allocation.

C3) Dependencies among FG and BG Tiles: Upon receiving all the tiles, the receiver stitches them as a single video and renders them on the screen. However, delayed reception of BG tiles can cause unwanted video stalls and HoL blocking. Although playing only the FG tiles is possible, it leads to worse QoE. It is important to break the strong dependency between FG and BG tiles.

3 Design of Compact

Compact's design is based on the following key observations: (1) at a particular time and location, all ISPs generally do not suffer equally from poor throughput [35], (2) a significant number of people often carry multiple smart devices that can be collaborated to leverage their aggregated bandwidth [77], (3) background in live online classes is mostly static, and changes are less frequent than the foreground, therefore, FG can be rendered with an outdated BG, (4) content-aware segregation of tiles allows streaming at a higher bitrate (better quality), and (5) BG can be encoded at a lower bitrate than FG because people pay more attention to FG [20, 48].

3.1 System Architecture

We show Compact's architecture and workflow in Fig. 6. At the streamer side, the camera sends raw frames to the system where *FG Detector* identifies the tiles containing any human face using a DNN (§4). Thereafter, the *Scheduler* uses the marked tiles to create FG and BG sets with tile indices. Based on the network statistics (RTT and BW), the scheduler decides what quality/bitrate to keep for the two categories (FG & BG), along with the number of tile shifts if needed (§3.2). Besides, it chooses the path with the minimum completion time = $\left(\frac{RTT}{2} + \frac{Filesize}{BW}\right)$, called the primary path, to stream the FG tiles. The other path is called the secondary. *BG tiles need not be streamed through the helper device only. It can be sent through the direct path too, depending on the network characteristics.*

Finally, *Scheduler* notifies the *Tile Segregator*, which passes the raw frames received from the camera to the FG and BG encoder to generate two tiled video segments, one for FG and another for BG. These encoded segments are stored in their corresponding sending queues (*FG Segment Queue* and *BG Segment Queue*). Although, called *FG Tile Encoder*, the encoded segments can also have a few relocated tiles of BG. Subsequently, the *Streamer* module streams the video segments from the queues over the network through primary and secondary paths. In addition, it performs another important task of maintaining the network's RTT and BW estimates of each path using separate probe channels discussed in the next subsection. The *Helper Device* just relays the received tiles to the *Primary Device*.

As on the streamer side, the received tiles are first stored in queues on the primary device (where the user watches the video). The decoders for FG and BG fetch the segments sequentially from these queues and pass on the decoded frames to the *Tile Stitcher*. On receiving the FG and BG frames, the stitcher clubs these two subframes together to play as a single video (shown in Fig. 7). At times when the BG segments get delayed, the stitcher uses the last

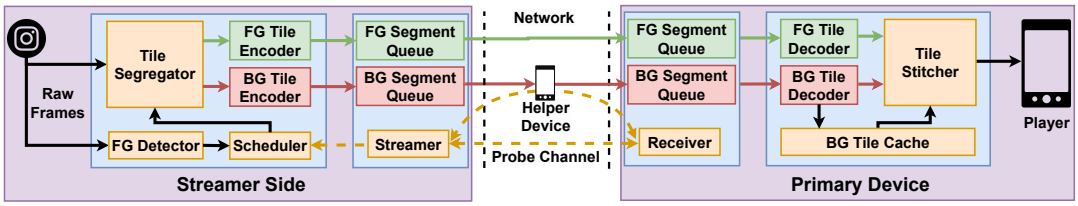


Figure 6: Architecture and workflow of COMPACT.

rendered BG stored in *BG Tile Cache* for rendering. The *Receiver* module communicates with the *Streamer* to echo back probe packets required for network RTT and BW estimation.

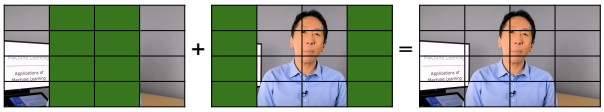
To get the right schedule, having a good enough estimate of network characteristics like RTT and BW is crucial. We create a total of four probe channels, two for RTT and two for BW. To estimate the RTT of each path, we send timestamps from the streamer machine to the primary, which is echoed back. The difference between the receiving time and the sent timestamp is the RTT. We attach timestamps in the custom headers of the sent video segment tiles to probe the network BW. The primary device dissects the header and replies with the demarcated timestamp. We use this timestamp to calculate the total sending time to infer network BW. The instantaneous RTT and BW are noisy in nature. Similar to prior works [45, 96], we use the harmonic mean (HM) of the past five values to smoothen the estimates.

Once the tiles are streamed from the sender based on an appropriate schedule, they might arrive at different moments of time at the primary device. If the BG tiles are delayed too much (more than inter-frame delay), it can lead to stalls because we cannot render the FG tiles alone. Therefore, we need some way to handle the out-of-order arrival of the two sets of tiles. In online classes, the background (slides, whiteboard, or texts) is mostly static, and the changes are less frequent than those of the speaker (foreground). Therefore, the same BG can be rendered for multiple foregrounds. This opens the possibility of breaking the tight coupling between FG and BG tiles. In cases where the BG tiles are delayed, the previous BG can be stitched along with the current FG tiles to render.

3.2 Tile Scheduler

We now discuss our content-aware scheduler. Once we get the two sets of tiles for foreground (FG) and background (BG), respectively, timely delivery of both of them to the player side is crucial. Any delay caused increases the stall duration between two consecutive segments. For a video encoded at $25fps$, the inter-frame delay δ is $40ms$. Any extra time required beyond $40ms$ is, therefore, considered a stall/freeze. To have the best quality of experience, the quality (bitrate) of the played video should be highest with the least stall and quality switches within and across the segments. Borrowing from prior works on tiled streaming [62], we define a utility function to capture users' QoE as a weighted sum of user-perceived video quality, stall, inter and intra-segment quality switches.

Let there be total N tiles with m tiles in FG and n tiles in BG set. The variables Q_{FG} and Q_{BG} denote the quality defined in terms of the video quantization parameter (QP) of tiles in FG and BG, respectively. Let the size of each tile in FG and BG be $R_{FG}(Q_{FG})$ and $R_{BG}(Q_{BG})$ for a certain quality level Q_{FG} and Q_{BG} of foreground and background.



(a) (b) (c)

Figure 7: Tile stitching: (a) background tiles and (b) foreground tiles stitched into a single video in subfigure (c).

Filesize of Tiles: We note that our scheduler maps all tiles of FG and a limited number of tiles of BG to the primary path. Assuming a total of o BG tiles use the primary (i.e., the higher BW) path, the data F_P and F_S sent through primary and secondary paths are:

$$F_P = R_{FG}(Q_{FG}) \times m + R_{BG}(Q_{BG}) \times o; F_S = R_{BG}(Q_{BG}) \times (n-o) \quad (1)$$

User-perceived Video Quality: The video quality Q_t of a segment t, the user observes, depends on the quality of the FG and BG tiles rendered. Mathematically, if β_1 and β_2 denote the weights given to the quality of FG and BG, respectively, the video quality Q_t is:

$$\begin{aligned} Q_t &= \beta_1 Q_{FG} + \beta_2 Q_{BG} \\ &= \beta_1 (m+o) q_t(l_{FG}) + \beta_2 (n-o)(\mathbb{1}(K) q_t(l_{BG}) - p(t)) \end{aligned} \quad (2)$$

Here, the function $q_t(.)$ returns a normalized quality value for a given quantization level l_{FG} and l_{BG} for the FG and BG, respectively. In case of delayed delivery of BG tiles beyond a threshold, the player can choose to render received FG tiles by stitching them with the previous segment's BG tiles, caching the current ones. This depends on the completion time captured using the indicator function $\mathbb{1}(K)$ which is 1 if $\left(\left(\frac{RTT_S}{2} + \frac{F_S}{BW_S}\right) - \left(\frac{RTT_P}{2} + \frac{F_P}{BW_P}\right)\right) < \delta$ else 0. The path with the least completion time is referred to as the primary path P where FG tiles are scheduled. Here, BW_P and BW_S are the estimated bandwidths, and RTT_S and RTT_P are estimated RTTs of primary and secondary paths, respectively. $p(t) = max(Q_t)$ penalizes if the BG tiles of past segments are rendered. We set β_1 and β_2 to 0.6 and 0.4, respectively to prioritize FG tiles.

Quality Switches: The quality levels of the tiles on the player side can be different across and within a segment. There should be a graceful degradation of quality for better QoE. We use I_1 and I_2 to penalize abrupt changes to control inter-segment and intra-segment quality switches, respectively. We define these as:

$$I_1 = |Q_t - Q_{t-1}| = |q_t(l_{FG+BG}) - q_{t-1}(l_{FG+BG})| \quad (3)$$

$$I_2 = |Q_{FG} - Q_{BG}| = |q_t(l_{FG}) - q_t(l_{BG})| \quad (4)$$

Stall Duration: The stall duration depends on the completion time of a video segment. A stall occurs when the tiles of the current segment reach after the play time of the previous segment P_{t-1}. Therefore, assuming primary path (P) is the fast path, for a segment

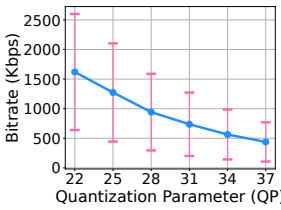


Figure 8: Video bitrate reduces on increasing quantization parameter (QP).

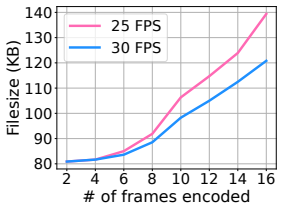


Figure 9: Increasing the # of frames in the encoded video segment increases filesize.

t the stall L_t can defined as:

$$K_{max} = max\left(\frac{RTT_P}{2} + \frac{F_P}{BW_P}, \mathbb{1}(K)\left(\frac{RTT_S}{2} + \frac{F_S}{BW_S}\right)\right) \quad (5)$$

$$L_t = max((K_{max} - P_{t-1} - \delta), 0) \quad (6)$$

Here, K_{max} is the maximum of the sending times on the primary and secondary paths, indicator function $\mathbb{1}(K)$ becomes zero when all BG tiles are shifted to FG, otherwise, it is one. The objective of the scheduler is to split the total tiles into primary and secondary paths in such a way as to maximize the utility function below.

$$\textbf{Maximize } Utility = \alpha_1 Q_t - \alpha_2 I_1 - \alpha_3 I_2 - \alpha_4 L_t - \alpha_5 S_t \quad (7)$$

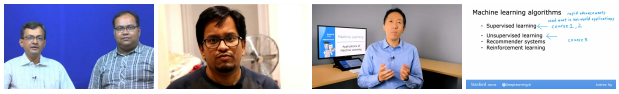
Here, S_t is the absolute difference in the number of tiles shifted from the background to the primary path between the last and the current segment. It refrains from abrupt tile relocations between two adjacent segments. We borrow the weights assignment strategy of [19, 58] keeping $\alpha_1 = max(Q_t)$, α_2 to α_4 equal to 1, and make $\alpha_5 = 0.1$. We use an exhaustive search over all possible tile shifts, FG, and BG QP changes to get the right schedule for every segment, which takes <= $1ms$.

Moreover, we need to estimate the filesize of tiles for each QP value while scheduling because the segment is encoded after getting the right schedule. To estimate the filesize, we rely on Fig. 8, which depicts that for a QP reduction of 3, the filesize/bitrate increases by 1.2×. We use the filesize of the previously streamed segment and a multiplying factor of 1.2 to approximate the filesize for every QP during the exhaustive search.

4 Implementation

We implement the host/streamer and the joinee/viewer side of the Compact in Java. We use a Linux machine as the host device and another two Linux PCs as helper and primary devices. We now discuss each of the components in detail.

(1) **Video Encoding:** Since Compact uses tiled videos for streaming, we use an open-source HEVC encoder, Kvazaar [80], to encode the raw frames into a grid of 4×4 tiles. Other configurations are also possible. However, we empirically found this configuration works well for us as it balances encoding overhead and stitching complexities similar to prior works utilizing tiles [25, 36]. Our encoding pipeline begins by capturing raw frames from the screen (simulating camera) using FFmpeg [2]. We then encode these frames into a tiled HEVC video. We use GPAC [52] for tile manipulation and packing segments into an mp4 file. This process repeats for each video segment. We keep the framerate fixed at 25 (used by most live video streaming works [9, 32]). Each segment contains four frames balancing encoding time and compression efficiency. We note from Fig. 9 that segments with 4 frames have approximately the same mean filesize as 2 frames, but it escalates faster from 6.



(a) Video 1 **(b) Video 2** **(c) Video 3** **(d) Video 4**

Figure 10: Screenshots of the four videos used for evaluations.

(2) **Foreground Detection:** We argued in §3 that Compact is content aware. We use a pre-trained ResNet model available at [7] for face detection to detect the FG content. Once we have the coordinates of the bounding boxes of the detected faces, we identify the tiles containing these boxes and consider them the FG tiles and the rest the BG tiles. In the case of presentation slides, the model detects no human face, we keep a horizontal split with eight tiles in both FG and BG.

(3) **Transport Layer:** We use Stream Control Transmission Protocol (SCTP) to stream the video segments because it provides in-order delivery, configurable retransmission, flow, and congestion control. We use UDP under the hood for all the control channels. Apart from SCTP, QUIC [51] is another transport protocol that precludes transport layer HoL-blocking. QUIC is built on top of UDP borrowing ideas from SCTP. Since SCTP is a relatively older protocol, multiple live-streaming techniques [44, 81, 86, 87, 89, 91, 92] have used it. QUIC, till date, is mostly explored in video-on-demand setups. We intend to explore the feasibility of its multipath variant [29–31] in the future.

(4) **Helper & Primary Device:** The helper's job is just to relay the received tiles to the primary device. To keep the overhead minimal, we use the Linux utility *socat (SOcket CAT)* [6], which acts as a socket-based bidirectional stream relay running on the helper. The primary device receives all the tiles, stitches, and plays them. Once the tiles of both paths are synchronized, we use OpenCV to stitch/join and render the FG and BG tiles.

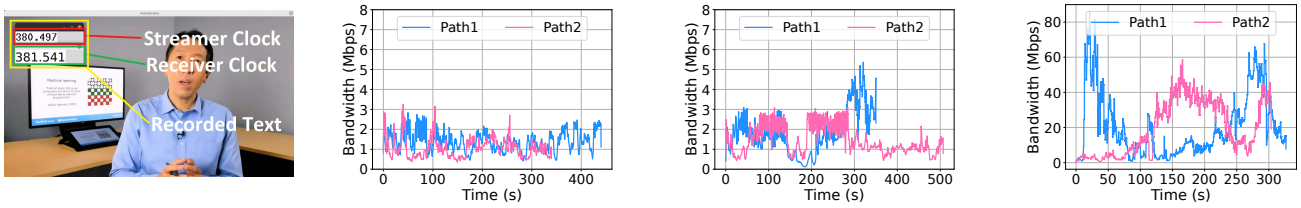
The different tasks performed on the streamer and the receiver devices are asynchronous. We use multi-threading to parallelize the tasks. For instance, the video segment encoder can independently capture and encode videos, while the sender thread can parallelly stream the encoded videos. The encoder should not wait for the sender thread to complete sending. We use evicting queues[3] of size two to exchange information between threads.

Baselines: We compare Compact with the different conventional schedulers at both the packet level and tile level. All the tile-level schedulers, except CMin, are unaware of the video content.

1) Single Path: For the single path, we always stream on one of the paths without any scheduler. We denote as SP1 when using only Path 1 and SP2 when streamed on Path 2 only.

2) MinRTT: This is the default scheduler in MPQUIC and MPTCP. At the packet level (PMin), it sends all the segments' chunks (MTU size packets) through the path with minimum RTT. We even make it content-aware at the tile level (CMin) to show that without tile shifting and out-of-sync FG and BG playback, it behaves similar to oblivious tile-level schedulers. CMin sends FG tiles to the minimum RTT path and BG tiles to the other one.

[3]FIFO queues with auto-dequeuing on exceeding queue size are referred to as evicting queues.



(a) E2E Lag Computation (b) Bus trace (c) Car trace (d) Walk Trace

Figure 11: (a) Screenshot showing both the streamed digital clock and the local one. We record only the portion inside the yellow box for text detection to calculate lag. (b), (c), and (d) depicts the bandwidth patterns of the replayed trace for the experiments.

3) Musher: This is another MPTCP scheduler that allocates packets proportional to paths' bandwidth (PMu) [68]. At the tile level (TMu), we assign tiles based on the available bandwidth of the paths.

4) BFlow: We implement balanced subflow completion (DEMS [37]) at both packet (PBF) and tile level (TBF). The core objective of BFlow is to minimize the difference between the completion times of the sent data (packet for PBF and tiles for TBF) on both paths.

Dataset: We selected four lecture videos from YouTube [11–13, 43], each lasting 4-5 minutes. Fig. 10 shows screenshots of the videos used for the experiments. The videos have a mix of scenes, with Video 1 having two speakers delivering a lecture. Video 2 and Video 3 have one speaker in the FG with different BGs. Video 4 is similar to Video 3, with the only difference being that it has slides showing up frequently. We used these videos as they represent the typical online class setup and provide us with sufficient diversity to check the robustness of our technique. For each session, we replay these downloaded videos with a live digital clock to calculate E2E lag.

Performance Metrics: For extensive evaluations, we employ E2E lag and video stall to quantify the quality of experience. We do not report the quality of baselines individually in the evaluation because we kept it identical to Compact. This ensures a fair comparison of factors, such as end-to-end latency and stalls. Thus, we first ran Compact, logged its FG and BG quality for each segment, and then reused it in all other baselines while streaming.

Our application-level E2E lag is defined as the difference between the time a frame appears on the streamer screen/camera and when it reflects on the viewer's screen. It depicts the actual lag that the user perceives. To measure E2E lag, we follow the strategy shown in [79], where we put a live digital clock on the video we stream. We also run the same clock on the receiver end and put it on the playing video. Afterward, we use a screen recorder to capture only the portion of the screen displaying the clocks (yellow box in Fig. 11a). Thereafter, we extract frames from the recorded video and run PyTessaract [53] OCR to recover the clocks' text. The difference of the detected texts is the E2E lag. While recording the clock texts, we reduced the capture rate to $10fps$ to reduce the number of OCR inferences, which is time-intensive. Note that we discard a few frames where the OCR fails to detect texts ($\approx$ 8% on average). Our stall represents the inter-segment delay, which is the interval between the render time of the last frame and the first frame of two consecutive video segments.

5 Evaluation

We now discuss Compact's performance under various network conditions and the overheads of its different components.

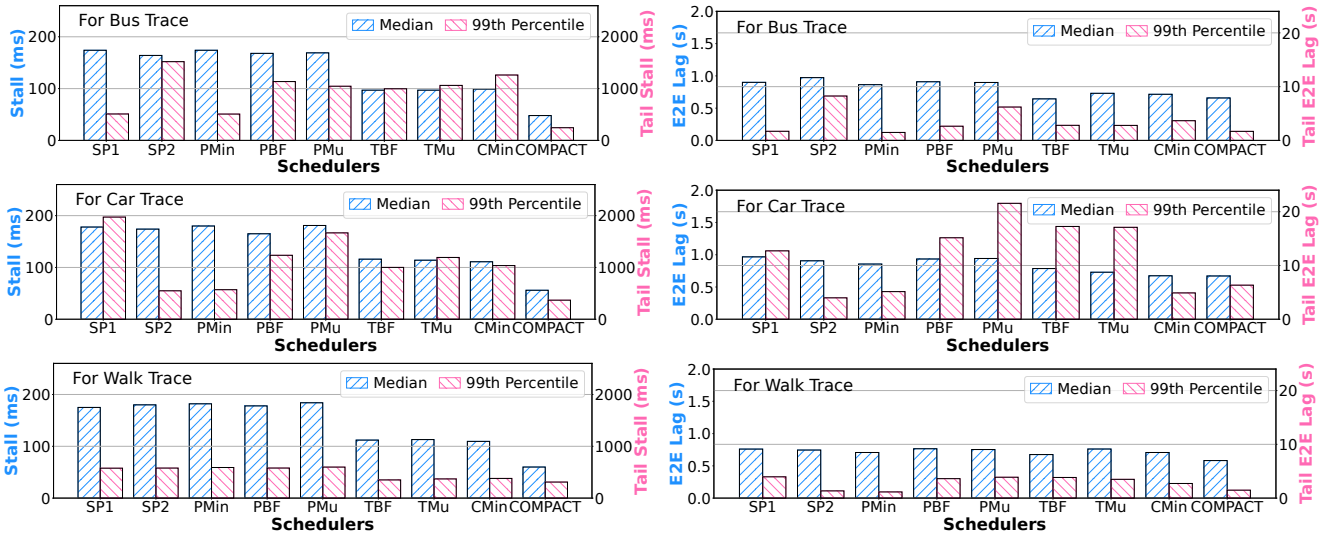
5.1 Trace-driven Results

To extensively evaluate Compact under various network conditions, we use trace files [14] open soured by Pensieve [58] collected while traveling in a bus and a car. Besides, we even collected two walking traces ourselves. We chose these traces to evaluate Compact under challenging network cases with high bandwidth fluctuations and RTTs. Since many countries in the world still witness poor bandwidth (< 3Mbps) [10, 73], we scale down the bus and car trace accordingly to the same range to evaluate for such scenarios. We report the bandwidth patterns in Fig. 11. In each case, we replay two traces of the same category on the two available paths using Mahimahi. If a trace ends early, it gets replayed automatically. We show the video stall and E2E lag in Fig. 12.

Stall: When streaming over a single path, we note that the median stall reaches 180ms on all traces. The median performance of packet-level schedulers (PMin, PBF, and PMu) is similar to or slightly worse than single paths. This is due to the HoL blocking caused by the out-of-order delivery of packets because of path heterogeneity. Therefore, packet-level schedulers fail to leverage the benefits of multiple paths. The network characteristics of bus and car traces are distinct, therefore, the tail values (99th percentile) of single paths differ by $\approx$ 3× while on walk trace, it is almost the same for packet-level schedulers and single paths. Except for the PMin, the rest of the packet-level baselines experience larger tail stalls. Since PMin picks the path with minimum RTT to stream all packets through that path only, it manages to avoid out-of-order packets.

At the tile level, conventional schedulers reduce the median stall by at least 35% compared to the best single path (single path with the least stall) and observe a lower stall than packet-level baselines. Even the basic content-aware tile-level MinRTT (CMin) scheduler shows a trend similar to oblivious tile-level schedulers on all traces, validating that mere content awareness is insufficient. Tile-level schedulers are superior to packet levels because tiles intended for a path are sent only through that path, thus completely removing inter-path HoL blocking because only smaller size BG is scheduled on poorer paths, unlike packet levels that do not consider this. However, the tail stall still dominates due to the dependency between FG and BG tiles.

Compared to all baselines Compact induces the least median and tail stall with 48ms and 245ms, respectively, on the bus trace, 59ms and 379ms on the car, and 59ms and 307ms on the walking trace. On the bus trace, it experiences 3.4× less median stall than the single path best, 3.5× lesser than the best packet-level scheduler (PBF), and 2× than the best tile-level scheduler (TBF). Similarly, the reduction is 2.9× (single path best), 2.4× (PBF), and 1.9× (CMin) on car trace and $\approx$ 3× (single path best), 3× (PMin), and 1.9× (CMin)



(a) Stall (b) E2E Lag

Figure 12: (a) Stall and (b) E2E lag with traces while traveling in a bus, traveling in a car, and walking

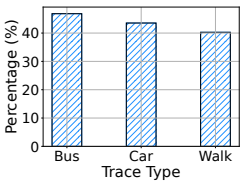


Figure 13: Out-of-sync Playback: shows the proportion of time FG and BG tiles played out-of-sync.

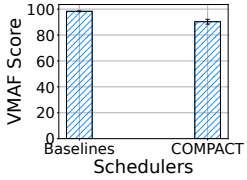


Figure 14: Perceived mean video quality (VMAF score) for all four videos on the bus trace.

for the walking trace. In terms of tail stalls, on the bus and car traces, Compact manages to reduce it by at least 1.5× compared to the single path best and packet-level best, while 2.6× in contrast to tile-level best. On walking trace, its tail is 44*ms* smaller than TBF (best) and 84*ms* than PMin. The asynchronous playback of FG and BG tiles of Compact helps alleviate the video stalls dramatically.

E2E Lag: In terms of E2E lag, on all traces Compact experiences either the least lag or is comparable to tile-level schedulers. However, in terms of tail lag, it is always better than TBF and TMu on all the traces. The lag (median E2E lag) of single paths and packet-level baselines is almost similar in all the scenarios. Tile-level schedulers are marginally better than packet levels in terms of lag. Except for MinRTT (PMin and CMin), the rest of the schedulers witness large tail lags on the car trace. Since the bandwidth is superior in our collected walking trace, all the schedulers manage to keep that median lag below 1*s* and tail lag below 4*s*.

Compact's median lag is 28% lower than single path best, 23% than the lowest of packet-level schedulers, and similar the lowest of tile-level. Compared to TMu and TBF (both perform almost the same), the 99*th* percentile lag is 1.7×, 1.1×, and 2.5× less on bus, car, and walking trace, respectively. We note that all the schedulers are consistently better on the walking trace due to the higher bandwidth (Fig. 11d). The evenness vanishes when the paths' characteristics differ under the bus and car trace. Due to the same streaming video quality, all schedulers encounter a similar median E2E lag. All the above observations vouch for using tiles and multipath-based live-streamed classes.

Out-of-order FG and BG stitching: In the cases where BG gets delayed, Compact stitches the current FG with the cached most recent BG. This helps significantly reduce video stalls and lag. We quantify in Fig. 13 the number of times such out-of-order stitching occurs while running each of the traces for all four videos. We note that these out-of-sync playbacks happened 46.8%, 43.6%, and 40.3% on the bus, car, and walk trace, respectively. These events are fewer on the walking trace due to the more consistent bandwidth compared to the bus and car traces.

Perceived Video Quality: For all the packet and tile-level schedulers, the player renders the streamed video without any changes. Therefore, there is no scope for quality degradation compared to the encoded video streamed from the streamer. However, Compact stitches the previous BG in case it gets delayed. Now, we show that such asynchronous stitching does not hurt the user experience because the BG remains mostly static. VMAF is known to capture human-perceived video quality more accurately than PSNR and SSIM. Therefore, we report the VMAF score. We show the mean video quality of all four videos perceived by the viewer in terms of VMAF score [23] for the bus trace in Fig. 14. We utilize the open source [4] implementation of VMAF for calculations. We note that compared to the baselines, the VMAF score of Compact drops marginally by 8.15% even though it renders a previous BG 46.8% (Fig. 13) of the time on the bus trace. This implies that although out-of-order FG and BG stitching theoretically degrades the video quality, it is marginal in practice. Note that the quality for all the baselines remains the same because we use the same streaming quality for all of them.

5.2 Adaptability to Path Fluctuations

To validate Compact's adaptability to drastic and abrupt network fluctuations, we create two traces for each path. We fixed *Path*1's

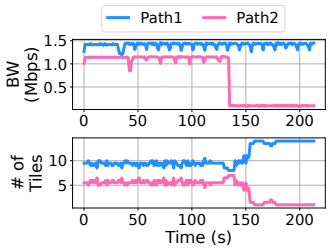


Figure 15: Number of tiles scheduled on each path when network changes drastically.



(a) Our Setup

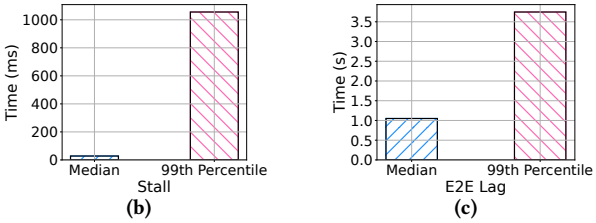


Figure 16: (a) shows our live experiment setup. Compact observed (b) stall and (c) E2E lag while streaming over the actual cellular network during the live experiment.

bandwidth to 1.6$Mbps$ and $Path2$'s to 1.2$Mbps$ with a standard deviation of 10$Kbps$. We make the $Path2$'s bandwidth suddenly dip to 100$Kbps$. We replay these traces and run Compact for 4 min on a video with only slides. To confirm the desired trace patterns, we run Iperf to get the bandwidths of both paths in Fig. 15.

To capture Compact's response to the abrupt change in the BW, we show the number of tiles scheduled on each path in Fig. 15. We observe that before 135s, our scheduler distributes the tiles proportionally to the paths' estimated bandwidths. However, soon after, when $Path2$ went down, Compact detected the change and adapted accordingly by gradually shifting most of the tiles to $Path1$ only, demonstrating robustness to sudden network variations.

5.3 Live Experiment

For the live experiment, we run Compact on the actual cellular network. As shown in Fig. 1, we keep the same setup and use three Linux desktops: one acting as the streamer to live stream videos (with 15th Gen Intel i7 having 16GB RAM), another as the helper (with AMD Ryzen 5 with 16GB RAM), and the third one as the client playing the video (equipped with 12 Gen Intel i7 with 32GB RAM). All machines access the internet through the cellular network obtained using three smartphones via USB tethering. Furthermore, we connect the primary and helper desktops together using a separate WiFi connection to relay the tiles. In actual deployment, the application on the primary can be made to discover and connect to available helper device. Like trace-driven experiments, we streamed one of the four videos running a live digital clock to get E2E lag and stall (reported in Fig. 16). We utilized the publicly accessible IPv6 addresses from the helper and the client to connect to the streamer. We note from Fig. 16 that the experienced median stall is 28ms with a tail of 1055ms. The median lag is close to 1s with a tail latency of 3.75s. The mean QP for the FG and BG tiles during the streaming was 33. Here, the lag values are worse than trace-driven results because of the poor cellular network with a mean BW and RTT in order of 833.4$Kbps$ and 425ms, respectively.

Table 1: Overheads of different components of Compact

Encoding	FG Detection	Scheduling	Stitching
277 ms	56 ms	1 ms	1 ms

5.4 Microbenchmarks

Overheads: We log in Table 1 the mean overheads (in ms) of each individual component of Compact. We observe that scheduling and stitching take almost negligible time of 1ms. Tiled video encoding incurs greatest latency due to the software-based encoder Kvazaar. We rule out using a faster preset as it degrades the final video quality. Moreover, screen capture using FFmpeg also contributes to the encoding overhead.

Our encoding happens in two steps: raw frame capture and HEVC encoding, and FG and BG generation, where the former takes most of the time (> 85%). On average, the model used for foreground detection takes 56ms to recognize human faces in the streamed videos. We parallelize detection with the frame capture thread to amortize this overhead.

Video-wise Results: We show Compact's median lag and stall on individual videos in Fig. 17 for the bus trace to show the effect of different video types. $Video1$ consists of two speakers with no slides, and the other three videos have only one speaker. Specifically, in $Video2$ and $Video3$, the instructor uses slides to convey whatever he is teaching. $Video4$ has animation playing in the video in between.

We note that Compact provides consistently lower lag and stall compared to the baseline schedulers on all the videos. All the packet-level schedulers incur stalls similar to single-path streaming, thus failing to benefit from multiple paths. Due to the absence of inter-path out-of-order packet delivery, tile-level schedulers are better than packet-level schedulers. Among the tile-level schedulers, CMin gives the least stall. The out-of-order playback helps Compact to have the least stall. Since only $Video2$, $Video3$, and $Video4$ have frequently changing screens from slides to the speaker, they are encoded at a relatively higher bitrate than $Video1$. Therefore, the E2E lag and stall are highest for them for SP2 (worse path).

6 Related Works

Content-aware video streaming: Streaming a video with consistent quality over a cellular network is difficult due to its random variations in bandwidth, loss, and latency. HotDASH [71] relies on the notion of temporal content awareness for adaptive video streaming. They decide the video's encoding rate based on the network characteristics. This can lead to poor streaming quality due

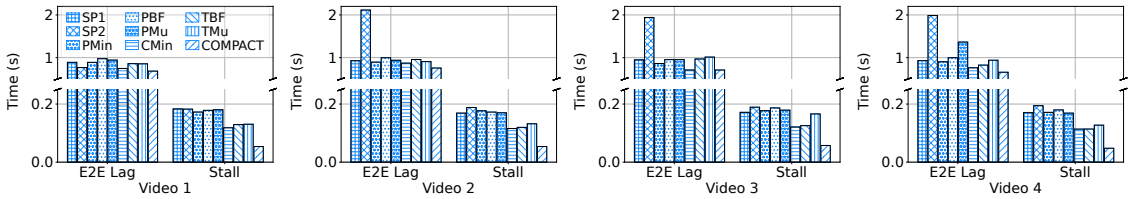


Figure 17: End-to-end (E2E) application-level lag and video stall duration of the different schedulers on individual videos used in the large-scale evaluation for the bus trace. Compact consistently performs better than the baseline schedulers

to the limited bandwidth of the single path. Works like [63, 94] prioritize content based on user viewport for quality adjustment in 360° videos using tiles. ZGaming [88] predicts frames based on content type to improve cloud gaming. However, none of these works employ multipath.

Multipath Video Streaming: Several works [56, 82, 86, 87, 89, 91] utilized multipath to bolster the video streaming performance. XLINK [99] improved the quality of short video streaming using MPQUIC. MPDASH [38] employs MPTCP [66, 67, 85] to use multiple interfaces while streaming video adaptively. MPRTP [72] creates multiple subflows to deliver media packets over multiple paths. However, live streaming is more challenging than video-on-demand due to its tight deadline requirement. Converge [32] and TwinStar [83] used multipath WebRTC for live video conferencing and live cloud gaming, respectively. AggDeliv [33] optimize congestion control in MPQUIC for mobile live video delivery. Chorus [56] integrates adaptive bitrate streaming with multipath networks to optimize the quality of experience. However, none of these works utilize content-awareness to address the packet-level HoL blocking.

Multipath Schedulers: There have been several multipath schedulers like Musher [68], DEMS [37], ECF [47], and DAMS [101] proposed, which work at the packet level. Although DEMS and DAMS consider the deadline requirements, they are incognizant of the content of the packets. Thus, they fail to mitigate inter-path HoL blocking at the application level. Employing these schedulers in live video streaming worsens the situation (§2.3). Therefore, the scheduler tailored for live streaming should be content-aware to distribute packets well across multiple paths.

7 Discussion and Limitations

Having multiple peers: Compact currently supports peer-to-peer video streaming with only one participant. Extending to a multi-participant setup will require getting the network statistics of each attendee for scheduling. This can incur overheads on the streamer side. We intend to look for possible mitigation strategies to amortize the overheads.

Catering to diverse applications: Currently, we show results for live online classes. However, the setup can serve different applications. The major challenge is correct foreground detection, which depends on the video application. A possible direction is to use a saliency-based FG segragator. Moreover, Compact captures a few frames and encodes them as segments (similar to HTTP low latency live streaming) that are streamed over different paths. Although the encoding is parallelized using pipelining, the encoding overhead of the first segment dominates, leading to large latency. One possible mitigation strategy is to stream on a frame basis, as done by WebRTC. Here, frames are captured, encoded, and streamed continuously in real time. Such an architecture can make Compact suitable for video conferencing-like applications.

Accurate bandwidth estimation: Our scheduling decision relies on bandwidth estimation. An underestimation causes poor quality, and overestimation can lead to unwanted video stalls. Compact currently uses a relatively simple technique of bandwidth estimation, leaving the use a of more sophisticated and accurate predictor for future work. An accurate predictor can provide better link utilization, albeit at the cost of more computation.

Scope of using feedback from primary device: One way to address inaccurate bandwidth estimation is to incorporate feedback from the primary device. On receiving a complete video segment, the receiver can send the completion time and other QoE metrics as part of the feedback, which the streamer can utilize to offset its bandwidth and scheduling inaccuracies. Such feedback has been shown to work well by works like Converge [32] and XLINK [99]. We intend to explore appropriate feedback metrics and mechanisms.

Energy aware scheduling over the other device: Utilizing multiple devices for collaboration comes with the cost of additional energy consumption. However, as validated by [100], the energy vs. performance improvement tradeoff amortizes the additional overheads. Compact can address such overheads by incorporating an appropriate utility function that puts a constraint on the amount of data and energy drawn from the helper device. We intend to explore designing such a power-aware scheduler in the future.

8 Conclusions

In this paper, we propose a system Compact for live video streaming intended for online classes over cellular networks that utilize multiple devices at the user end to distribute video tiles. We observe that the conventional packet-level scheduler incurs inter-path HoL blocking. Therefore, Compact utilizes a video content-aware scheduler that splits a video into foreground and background using tiles and sends these tiles over different network paths. We evaluate Compact under various network conditions using simulated, controlled, and real setups. On a bus trace Compact manages to limit the median stall by 3.4× and the tail stall by 6.2× compared to the single path. It also reduces the median E2E lag by 28.57% and tail lag by ≈ 80%. Compact, therefore, is likely to increase access to education in connectivity-challenged regions.

9 Acknowlegement

Shubham Chaudhary's research in this paper is funded by iHub Anubhuti-IIITD Foundation, New Delhi, India, through the CHANAKYA Doctoral Fellowship. This work is supported by the Science and Engineering Research Board, Department of Science and Technology, Government of India, under Grant CRG/2022/005096.

References

[1] 2020. 50 Online Education Statistics: 2024 Data on Higher Learning & Corporate Training.

[2] 2024. FFmpeg. https://www.ffmpeg.org/.

[3] 2024. Multiple device ownership means more smartphone usage. https://cdne.kantar.com/north-america/inspiration/technology/multiple-device-ownership-means-more-smartphone-usage. Accessed on April 8, 2024.

[4] 2024. Netflix/vmaf. original-date: 2016-02-08T18:41:38Z.

[5] 2024. Teens and Internet, Device Access Fact Sheet. https://www.pewresearch.org/internet/fact-sheet/teens-and-internet-device-access-fact-sheet/. Accessed on April 8, 2024.

[6] 2024-02-14. socat(1): Multipurpose relay - Linux man page. https://linux.die.net/man/1/socat.

[7] 2024-04-01. javacv/samples/DeepLearningFaceDetection.java at master · bytedeco/javacv. https://github.com/bytedeco/javacv/blob/master/samples.

[8] 2024-04-04. Live streaming latency - YouTube Help. https://support.google.com/youtube/answer/7444635. Accessed on April 4, 2024.

[9] 2024-04-08. Understanding and Choosing The Right Frame Rates for You. https://riverside.fm/blog/frame-rate-guide, https://riverside.fm/blog/frame-rate-guide. Accessed on April 8, 2024.

[10] 2024-04-09. Worldwide Broadband Speed League 2023. https://www.cable.co.uk/broadband/speed/worldwide-speed-league/.

[11] 2024-04-12. #2 Machine Learning Specialization [Course 1, Week 1, Lesson 1] - YouTube. https://www.youtube.com/watch?v=wiNXzydta4c.

[12] 2024-04-12. #3 Machine Learning Specialization [Course 1, Week 1, Lesson 2] - YouTube. https://www.youtube.com/watch?v=XtlwSmJfUs4.

[13] 2024-04-12. RL 3: Upper confidence bound (UCB) to solve multi-armed bandit problem. https://www.youtube.com/watch?v=RPbtzWgzD9M.

[14] 2024-11-14. 4G/LTE Bandwidth Logs. https://users.ugent.be/ jvdrhoof/dataset-4g/.

[15] 2024-11-14. The Evolving Landscape of Students' Mobile Learning Practices in Higher Education.

[16] 2024-11-14. Online Learning Statistics: The Ultimate List in 2024 | Devlin Peck.

[17] Irshad Abibouraguimane, Kakeru Hagihara, Keita Higuchi, Yuta Itoh, Yoichi Sato, Tetsu Hayashida, and Maki Sugimoto. 2019. CoSummary: adaptive fast-forwarding for surgical videos by detecting collaborative scenes using hand regions and gaze positions. In *Proceedings of the 24th International Conference on Intelligent User Interfaces* (2024-02-22). ACM, Marina del Ray California, 580–590. https://doi.org/10.1145/3301275.3302284

[18] Shivang Aggarwal, Moinak Ghoshal, Piyali Banerjee, Dimitrios Koutsonikolas, and Joerg Widmer. 2021. 802.11ad in Smartphones: Energy Efficiency, Spatial Reuse, and Impact on Applications. In *IEEE INFOCOM 2021 - IEEE Conference on Computer Communications* (2024-03-29). IEEE, Vancouver, BC, Canada, 1–10. https://doi.org/10.1109/INFOCOM42981.2021.9488763

[19] Zahaib Akhtar, Yun Seong Nam, Ramesh Govindan, Sanjay Rao, Jessica Chen, Ethan Katz-Bassett, Bruno Ribeiro, Jibin Zhan, and Hui Zhang. 2018. Oboe: auto-tuning video ABR algorithms to network conditions. In *Proceedings of the 2018 Conference of the ACM Special Interest Group on Data Communication* (2023-12-28). ACM, Budapest Hungary, 44–58. https://doi.org/10.1145/3230543.3230558

[20] Omer F. Aladag, Deniz Ugur, Mehmet N. Akcay, and Ali C. Begen. 2021. Content-Aware Playback Speed Control for Low-Latency Live Streaming of Sports. In *Proceedings of the 12th ACM Multimedia Systems Conference* (2024-02-22). ACM, Istanbul Turkey, 344–349. https://doi.org/10.1145/3458305.3478437

[21] Kisten Schmitt Alexus Bazen. 2023. Cell phone statistics 2024. https://www.consumeraffairs.com/cell_phones/cell-phone-statistics.html. Published on 28 September 2023; Accessed on April 8, 2024.

[22] Research Article, Arbaz Khan, Mubashir Saeed, Muhammad Anwar, and Lareb Kanwal. 2023. Journal of Mass Communication & Journalism Unleashing the Potential: A Study of the Effectiveness and Impact of YouTube Educational Content on Student Learning Outcomes. *Journal of Mass Communication and Journalism* 13:05, 2023 (September 2023). https://doi.org/10.37421/2165-7912.2023.13.538

[23] Netflix Technology Blog. 2017. Toward A Practical Perceptual Video Quality Metric.

[24] Benjamin Bross, Ye-Kui Wang, Yan Ye, Shan Liu, Jianle Chen, Gary J. Sullivan, and Jens-Rainer Ohm. 2021. Overview of the Versatile Video Coding (VVC) Standard and its Applications. *IEEE Transactions on Circuits and Systems for Video Technology* 31, 10 (October 2021), 3736–3764. https://doi.org/10.1109/TCSVT.2021.3101953

[25] Shubham Chaudhary, Aryan Taneja, Anjali Singh, Purbasha Roy, Sohum Sikdar, Mukulika Maity, and Arani Bhattacharya. 2024. TileClipper: Lightweight Selection of Regions of Interest from Videos for Traffic Surveillance. In *2024 USENIX Annual Technical Conference (USENIX ATC 24)*. 967–984.

[26] Di (Laura) Chen, Dustin Freeman, and Ravin Balakrishnan. 2019. Integrating Multimedia Tools to Enrich Interactions in Live Streaming for Language Learning. In *Proceedings of the 2019 CHI Conference on Human Factors in Computing Systems* (2024-11-15) *(CHI '19)*. Association for Computing Machinery, New York, NY, USA, 1–14. https://doi.org/10.1145/3290605.3300668

[27] Xinyue Chen, Si Chen, Xu Wang, and Yun Huang. 2021. "I was afraid, but now I enjoy being a streamer!": Understanding the Challenges and Prospects of Using Live Streaming for Online Education. *Proc. ACM Hum.-Comput. Interact.* 4, CSCW3 (January 2021), 237:1–237:32. https://doi.org/10.1145/3432936

[28] Kai-Yin Cheng, Sheng-Jie Luo, Bing-Yu Chen, and Hao-Hua Chu. 2009. SmartPlayer: user-centric video fast-forwarding. In *Proceedings of the SIGCHI Conference on Human Factors in Computing Systems* (2024-02-22). ACM, Boston MA USA, 789–798. https://doi.org/10.1145/1518701.1518823

[29] Quentin De Coninck. 2022. The packet number space debate in multipath QUIC. *ACM SIGCOMM Computer Communication Review* 52, 3 (July 2022), 2–9. https://doi.org/10.1145/3561954.3561956

[30] Quentin De Coninck and Olivier Bonaventure. 2017. Multipath QUIC: Design and Evaluation. In *Proceedings of the 13th International Conference on emerging Networking EXperiments and Technologies* (2024-03-29). ACM, Incheon Republic of Korea, 160–166. https://doi.org/10.1145/3143361.3143370

[31] Quentin De Coninck and Olivier Bonaventure. 2017. Multipath QUIC: Design and Evaluation. In *Proceedings of the 13th International Conference on emerging Networking EXperiments and Technologies* (2022-11-16). ACM, Incheon Republic of Korea, 160–166. https://doi.org/10.1145/3143361.3143370

[32] Sandesh Dhawaskar Sathyanarayana, Kyunghan Lee, Dirk Grunwald, and Sangtae Ha. 2023. Converge: QoE-driven Multipath Video Conferencing over WebRTC. In *Proceedings of the ACM SIGCOMM 2023 Conference* (2023-09-08). ACM, New York NY USA, 637–653. https://doi.org/10.1145/3603269.3604822

[33] Jinlong E, Lin He, Zongyi Zhao, Yachen Wang, Gonglong Chen, and Wei Chen. 2024. AggDeliv: Aggregating Multiple Wireless Links for Efficient Mobile Live Video Delivery. In *IEEE INFOCOM 2024 - IEEE Conference on Computer Communications* (2024-09-25). 1173–1180. https://doi.org/10.1109/INFOCOM52122.2024.10621184

[34] Travis Faas, Lynn Dombrowski, Alyson Young, and Andrew D. Miller. 2018. Watch Me Code: Programming Mentorship Communities on Twitch.tv. *Proceedings of the ACM on Human-Computer Interaction* 2, CSCW (November 2018), 1–18. https://doi.org/10.1145/3274319

[35] Keshav Gambhir, Tanmay Rajore, Shubham Chaudhary, Taral Jain, Avishi Gupta, Mukulika Maity, and Arani Bhattacharya. 2023. NATIVE: Network Aggregation based Tiled Live Video Streaming. In *2023 15th International Conference on COMmunication Systems & NETworkS (COMSNETS)*. 219–221. https://doi.org/10.1109/COMSNETS56262.2023.10041371

[36] Hongpeng Guo, Shuochao Yao, Zhe Yang, Qian Zhou, and Klara Nahrstedt. 2021. CrossRoI: cross-camera region of interest optimization for efficient real time video analytics at scale. In *Proceedings of the 12th ACM Multimedia Systems Conference*. 186–199. https://doi.org/10.1145/3458305.3463381

[37] Yihua Ethan Guo, Ashkan Nikravesh, Z. Morley Mao, Feng Qian, and Subhabrata Sen. 2017. Accelerating Multipath Transport Through Balanced Subflow Completion. In *Proceedings of the 23rd Annual International Conference on Mobile Computing and Networking* (2024-04-03). ACM, Snowbird Utah USA, 141–153. https://doi.org/10.1145/3117811.3117829

[38] Bo Han, Feng Qian, Lusheng Ji, and Vijay Gopalakrishnan. 2016. MP-DASH: Adaptive Video Streaming Over Preference-Aware Multipath. In *Proceedings of the 12th International on Conference on emerging Networking EXperiments and Technologies* (2024-04-10). ACM, Irvine California USA, 129–143. https://doi.org/10.1145/2999572.2999606

[39] Jingning Han, Bohan Li, Debargha Mukherjee, Ching-Han Chiang, Adrian Grange, Cheng Chen, Hui Su, Sarah Parker, Sai Deng, Urvang Joshi, Yue Chen, Yunqing Wang, Paul Wilkins, Yaowu Xu, and James Bankoski. 2021. A Technical Overview of AV1. *Proc. IEEE* 109, 9 (September 2021), 1435–1462. https://doi.org/10.1109/JPROC.2021.3058584

[40] Ahmad Hassan, Arvind Narayanan, Anlan Zhang, Wei Ye, Ruiyang Zhu, Shuowei Jin, Jason Carpenter, Z. Morley Mao, Feng Qian, and Zhi-Li Zhang. 2022. Vivisecting mobility management in 5G cellular networks. In *Proceedings of the ACM SIGCOMM 2022 Conference* (2024-03-29). ACM, Amsterdam Netherlands, 86–100. https://doi.org/10.1145/3544216.3544217

[41] Jia He, Mostafa Ammar, and Ellen Zegura. 2023. A Measurement-Derived Functional Model for the Interaction Between Congestion Control and QoE in Video Conferencing. In *Passive and Active Measurement*, Anna Brunstrom, Marcel Flores, and Marco Fiore (Eds.). Springer Nature Switzerland, Cham, 129–159. https://doi.org/10.1007/978-3-031-28486-1_7

[42] Jeroen Van Der Hooft, Maria Torres Vega, Stefano Petrangeli, Tim Wauters, and Filip De Turck. 2019. Tile-based Adaptive Streaming for Virtual Reality Video. *ACM Transactions on Multimedia Computing, Communications, and Applications* 15, 4 (November 2019), 1–24. https://doi.org/10.1145/3362101

[43] IIT Kharagpur July 2018. 2018. Prof Soumya Kanti Ghosh & Prof Sandip Chakraborty. https://www.youtube.com/watch?v=O–rkQNKqls.

[44] J.R. Iyengar, P.D. Amer, and R. Stewart. 2006. Concurrent Multipath Transfer Using SCTP Multihoming Over Independent End-to-End Paths. *IEEE/ACM Transactions on Networking* 14, 5 (October 2006), 951–964. https://doi.org/10.1109/TNET.2006.882843

[45] Junchen Jiang, Vyas Sekar, and Hui Zhang. 2012. Improving fairness, efficiency, and stability in HTTP-based adaptive video streaming with FESTIVE. In *Proceedings of the 8th international conference on Emerging networking experiments and technologies* (2023-12-28). ACM, Nice France, 97–108. https://doi.org/10.1145/2413176.2413189

[46] Shuowei Jin, Ruiyang Zhu, Ahmad Hassan, Xiao Zhu, Xumiao Zhang, Z. Morley Mao, Feng Qian, and Zhi-Li Zhang. 2024. OASIS: Collaborative Neural-Enhanced Mobile Video Streaming. In *Proceedings of the 15th ACM Multimedia Systems Conference* (Bari, Italy) *(MMSys '24)*. Association for Computing Machinery, New York, NY, USA, 45–55. https://doi.org/10.1145/3625468.3647610

[47] Baptiste Jonglez, Martin Heusse, and Bruno Gaujal. 2020. SRPT-ECF: challenging Round-Robin for stream-aware multipath scheduling. In *2020 IFIP Networking Conference (Networking)*. 719–724.

[48] Chen-Tai Kao, Yen-Ting Liu, and Alexander Hsu. 2014. Speeda: adaptive speed-up for lecture videos. In *Proceedings of the adjunct publication of the 27th annual ACM symposium on User interface software and technology - UIST'14 Adjunct* (2024-02-22). ACM Press, Honolulu, Hawaii, USA, 97–98. https://doi.org/10.1145/2658779.2658794

[49] Lorenzo Keller, Anh Le, Blerim Cici, Hulya Seferoglu, Christina Fragouli, and Athina Markopoulou. 2012. MicroCast: cooperative video streaming on smartphones. In *Proceedings of the 10th international conference on Mobile systems, applications, and services* (2024-04-04). ACM, Low Wood Bay Lake District UK, 57–70. https://doi.org/10.1145/2307636.2307643

[50] René F Kizilcec, Kathryn Papadopoulos, and Lalida Sritanyaratana. 2014. Showing face in video instruction: effects on information retention, visual attention, and affect. In *Proceedings of the SIGCHI conference on human factors in computing systems*. 2095–2102. https://doi.org/10.1145/2556288.2557207

[51] Adam Langley, Alistair Riddoch, Alyssa Wilk, Antonio Vicente, Charles Krasic, Dan Zhang, Fan Yang, Fedor Kouranov, Ian Swett, Janardhan Iyengar, Jeff Bailey, Jeremy Dorfman, Jim Roskind, Joanna Kulik, Patrik Westin, Raman Tenneti, Robbie Shade, Ryan Hamilton, Victor Vasiliev, Wan-Teh Chang, and Zhongyi Shi. 2017. The QUIC Transport Protocol: Design and Internet-Scale Deployment. In *Proceedings of the Conference of the ACM Special Interest Group on Data Communication* (Los Angeles, CA, USA) *(SIGCOMM '17)*. Association for Computing Machinery, New York, NY, USA, 183–196. https://doi.org/10.1145/3098822.3098842

[52] Jean Le Feuvre, Cyril Concolato, and Jean-Claude Moissinac. 2007. GPAC: open source multimedia framework. In *Proceedings of the 15th ACM international conference on Multimedia*. 1009–1012. https://doi.org/10.1145/1291233.1291452

[53] Matthias A. Lee. 2022. Python Tesseract. https://github.com/madmaze/pytesseract. original-date: 2010-10-27T23:02:49Z.

[54] Creative Commons License. Dec 27, 2021. What is Machine Learning Andrew Ng. https://www.youtube.com/watch?v=yeII6KWenWU.

[55] Hyoyoung Lim, Jinsung Lee, Jongyun Lee, Sandesh Dhawaskar Sathyanarayana, Junseon Kim, Anh Nguyen, Kwang Taik Kim, Youngbin Im, Mung Chiang, Dirk Grunwald, Kyunghan Lee, and Sangtae Ha. 2024. An Empirical Study of 5G: Effect of Edge on Transport Protocol and Application Performance. *IEEE Transactions on Mobile Computing* 23, 4 (April 2024), 3172–3186. https://doi.org/10.1109/TMC.2023.3274708

[56] Gerui Lv, Qinghua Wu, Yanmei Liu, Zhenyu Li, Qingyue Tan, Furong Yang, Wentao Chen, Yunfei Ma, Hongyu Guo, Ying Chen, and Gaogang Xie. 2024. Chorus: Coordinating Mobile Multipath Scheduling and Adaptive Video Streaming. In *Proceedings of the 30th Annual International Conference on Mobile Computing and Networking* (Washington D.C., DC, USA) *(ACM MobiCom '24)*. Association for Computing Machinery, New York, NY, USA, 246–262. https://doi.org/10.1145/3636534.3649359

[57] Kyle MacMillan, Tarun Mangla, James Saxon, and Nick Feamster. 2021. Measuring the performance and network utilization of popular video conferencing applications. In *Proceedings of the 21st ACM Internet Measurement Conference* (2024-03-29). ACM, Virtual Event, 229–244. https://doi.org/10.1145/3487552.3487842

[58] Hongzi Mao, Ravi Netravali, and Mohammad Alizadeh. 2017. Neural Adaptive Video Streaming with Pensieve. In *Proceedings of the Conference of the ACM Special Interest Group on Data Communication* (2023-12-27). ACM, Los Angeles CA USA, 197–210. https://doi.org/10.1145/3098822.3098843

[59] Preethi Natarajan, Professor Paul D. Amer, Jonathan Leighton, and Fred Baker. 2009. *Using SCTP as a Transport Layer Protocol for HTTP*. Internet Draft draft-natarajan-http-over-sctp-02. Internet Engineering Task Force. Num Pages: 13.

[60] Ravi Netravali, Anirudh Sivaraman, Somak Das, Ameesh Goyal, Keith Winstein, James Mickens, and Hari Balakrishnan. 2015. Mahimahi: Accurate Record-and-Replay for HTTP. In *2015 USENIX Annual Technical Conference (USENIX ATC 15)*. Santa Clara, CA, 417–429. https://www.usenix.org/conference/atc15/technical-session/presentation/netravali

[61] Yunzhe Ni, Feng Qian, Taide Liu, Yihua Cheng, Zhiyao Ma, Jing Wang, Zhongfeng Wang, Gang Huang, Xuanzhe Liu, and Chenren Xu. 2023. {POLYCORN}: Data-driven Cross-layer Multipath Networking for High-speed Railway through Composable Schedulerlets. In *20th USENIX Symposium on Networked Systems Design and Implementation (NSDI 23)*. 1325–1340.

[62] Sohee Park, Arani Bhattacharya, Zhibo Yang, Samir R. Das, and Dimitris Samaras. 2021. Mosaic: Advancing User Quality of Experience in 360-Degree Video Streaming With Machine Learning. *IEEE Transactions on Network and Service Management* 18, 1 (2021), 1000–1015. https://doi.org/10.1109/TNSM.2021.3053183

[63] Sohee Park, Arani Bhattacharya, Zhibo Yang, Samir R. Das, and Dimitris Samaras. 2021. *Mosaic* : Advancing User Quality of Experience in 360-Degree Video Streaming With Machine Learning. *IEEE Transactions on Network and Service Management* 18, 1 (March 2021), 1000–1015. https://doi.org/10.1109/TNSM.2021.3053183

[64] Sohee Park, Minh Hoai, Arani Bhattacharya, and Samir R. Das. 2021. Adaptive Streaming of 360-Degree Videos With Reinforcement Learning. In *Proceedings of the IEEE/CVF Winter Conference on Applications of Computer Vision (WACV)*. 1839–1848. https://doi.org/10.1109/WACV48630.2021.00188

[65] Genevieve Carlton Ph.D. 2024. 2024 Online Learning Statistics. Section: Online Colleges.

[66] Costin Raiciu, Mark Handley, and Damon Wischik. 2011. *Coupled congestion control for multipath transport protocols*. Technical Report.

[67] Costin Raiciu, Christoph Paasch, Sebastien Barre, Alan Ford, Michio Honda, Fabien Duchene, Olivier Bonaventure, and Mark Handley. [n. d.]. How Hard Can It Be? Designing and Implementing a Deployable Multipath TCP. ([n. d.]).

[68] Swetank Kumar Saha, Shivang Aggarwal, Rohan Pathak, Dimitrios Koutsonikolas, and Joerg Widmer. 2019. MuSher: An Agile Multipath-TCP Scheduler for Dual-Band 802.11ad/ac Wireless LANs. In *The 25th Annual International Conference on Mobile Computing and Networking* (2023-12-27). ACM, Los Cabos Mexico, 1–16. https://doi.org/10.1145/3300061.3345435

[69] Pranit Sarda. 2020. Why YouTube still rules the edtech roost. https://www.forbesindia.com/article/edtech-special/why-youtube-still-rules-the-edtech-roost/57761/1. Published on Feb 18, 2020; Accessed on Apr 12, 2024.

[70] Katherine Schaeffer. 2019. Most U.S. teens who use cellphones do it to pass time, connect with others, learn new things. https://www.pewresearch.org/short-reads/2019/08/23/most-u-s-teens-who-use-cellphones-do-it-to-pass-time-connect-with-others-learn-new-things/. Accessed on April 8, 2024.

[71] Satadal Sengupta, Niloy Ganguly, Sandip Chakraborty, and Pradipta De. 2018. HotDASH: Hotspot Aware Adaptive Video Streaming Using Deep Reinforcement Learning. In *2018 IEEE 26th International Conference on Network Protocols (ICNP)* (2024-04-12). IEEE, Cambridge, 165–175. https://doi.org/10.1109/ICNP.2018.00026

[72] Varun Singh, Saba Ahsan, and Jörg Ott. 2013. MPRTP: multipath considerations for real-time media. In *Proceedings of the 4th ACM Multimedia Systems Conference* (2024-03-29). ACM, Oslo Norway, 190–201. https://doi.org/10.1145/2483977.2484002

[73] Vibhaalakshmi Sivaraman, Pantea Karimi, Vedantha Venkatapathy, Mehrdad Khani, Sadjad Fouladi, Mohammad Alizadeh, Frédo Durand, and Vivienne Sze. 2023. Gemino: Practical and Robust Neural Compression for Video Conferencing. arXiv:2209.10507 [cs].

[74] Jangwoo Son, Dongmin Jang, and Eun-Seok Ryu. 2018. Implementing Motion-Constrained Tile and Viewport Extraction for VR Streaming. In *Proceedings of the 28th ACM SIGMM Workshop on Network and Operating Systems Support for Digital Audio and Video* (2024-03-07). ACM, Amsterdam Netherlands, 61–66. https://doi.org/10.1145/3210445.3210455

[75] Randall R. Stewart. 2007. *Stream Control Transmission Protocol*. Request for Comments RFC 4960. Internet Engineering Task Force. https://doi.org/10.17487/RFC4960 Num Pages: 152.

[76] Gary J Sullivan, Jens-Rainer Ohm, Woo-Jin Han, and Thomas Wiegand. 2012. Overview of the high efficiency video coding (HEVC) standard. *IEEE Transactions on circuits and systems for video technology* 22, 12 (2012), 1649–1668. https://doi.org/10.1109/TCSVT.2012.2221191

[77] Petroc Taylor. Jan 19, 2023. Average number devices and connections per person worldwide in 2018 and 2023. https://www.statista.com/statistics/1190270/number-of-devices-and-connections-per-person-worldwide/. Accessed on Nov 22, 2024.

[78] Margot Van Wermeskerken, Susanna Ravensbergen, and Tamara Van Gog. 2018. Effects of instructor presence in video modeling examples on attention and learning. *Computers in Human Behavior* 89 (2018), 430–438. https://doi.org/10.1016/j.chb.2017.11.038

[79] Matteo Varvello, Hyunseok Chang, and Yasir Zaki. 2022. Performance characterization of videoconferencing in the wild. In *Proceedings of the 22nd ACM Internet Measurement Conference* (2023-05-26). ACM, Nice France, 261–273. https://doi.org/10.1145/3517745.3561442

[80] Marko Viitanen, Ari Koivula, Ari Lemmetti, Arttu Ylä-Outinen, Jarno Vanne, and Timo D Hämäläinen. 2016. Kvazaar: open-source HEVC/H. 265 encoder. In *Proceedings of the 24th ACM international conference on Multimedia*. 1179–1182. https://doi.org/10.1145/2964284.2973796

[81] T. Daniel Wallace and Abdallah Shami. 2014. Concurrent Multipath Transfer Using SCTP: Modelling and Congestion Window Management. *IEEE Transactions on Mobile Computing* 13, 11 (November 2014), 2510–2523. https://doi.org/10.1109/TMC.2014.2307330

[82] Chengke Wang, Hao Wang, Feng Qian, Kai Zheng, Chenglu Wang, Fangzhu Mao, Xingmin Guo, and Chenren Xu. 2023. Experience: A Three-Year Retrospective of Large-scale Multipath Transport Deployment for Mobile Applications. In *Proceedings of the 29th Annual International Conference on Mobile Computing and Networking* (2023-07-15). ACM, Madrid Spain, 1–15. https://doi.org/10.1145/3570361.3592506

[83] Haiping Wang, Zhenhua Yu, Ruixiao Zhang, Siping Tao, Hebin Yu, and Shu Shi. 2023. TwinStar: A Practical Multi-path Transmission Framework for Ultra-Low Latency Video Delivery. In *Proceedings of the 31st ACM International Conference on Multimedia* (2024-03-18). ACM, Ottawa ON Canada, 9234–9242. https://doi.org/10.1145/3581783.3613443

[84] Jiahui Wang, Pavlo Antonenko, and Kara Dawson. 2020. Does visual attention to the instructor in online video affect learning and learner perceptions? An eye-tracking analysis. *Computers & Education* 146 (2020), 103779. https://doi.org/10.1016/j.compedu.2019.103779

[85] Damon Wischik, Costin Raiciu, Adam Greenhalgh, and Mark Handley. 2011. Design, Implementation and Evaluation of Congestion Control for Multipath TCP. (March 2011). https://www.usenix.org/conference/nsdi11/design-implementation-and-evaluation-congestion-control-multipath-tcp

[86] Jiyan Wu, Bo Cheng, and Ming Wang. 2018. Improving Multipath Video Transmission With Raptor Codes in Heterogeneous Wireless Networks. *IEEE Transactions on Multimedia* 20, 2 (February 2018), 457–472. https://doi.org/10.1109/TMM.2017.2741425

[87] Jiyan Wu, Bo Cheng, Chau Yuen, Yanlei Shang, and Junliang Chen. 2015. Distortion-Aware Concurrent Multipath Transfer for Mobile Video Streaming in Heterogeneous Wireless Networks. *IEEE Transactions on Mobile Computing* 14, 4 (April 2015), 688–701. https://doi.org/10.1109/TMC.2014.2334592

[88] Jiangkai Wu, Yu Guan, Qi Mao, Yong Cui, Zongming Guo, and Xinggong Zhang. 2023. ZGaming: Zero-Latency 3D Cloud Gaming by Image Prediction. In *Proceedings of the ACM SIGCOMM 2023 Conference* (New York, NY, USA) *(ACM SIGCOMM '23)*. Association for Computing Machinery, New York, NY, USA, 710–723. https://doi.org/10.1145/3603269.3604819

[89] Jiyan Wu, Chau Yuen, Ming Wang, and Junliang Chen. 2016. Content-Aware Concurrent Multipath Transfer for High-Definition Video Streaming over Heterogeneous Wireless Networks. *IEEE Transactions on Parallel and Distributed Systems* 27, 3 (March 2016), 710–723. https://doi.org/10.1109/TPDS.2015.2416736

[90] Man Wu and Qin Gao. 2020. Using live video streaming in online tutoring: Exploring factors affecting social interaction. *International Journal of Human-Computer Interaction* 36, 10 (2020), 964–977. https://doi.org/10.1080/10447318.2019.1706288

[91] Changqiao Xu, Zhuofeng Li, Jinglin Li, Hongke Zhang, and Gabriel-Miro Muntean. 2015. Cross-Layer Fairness-Driven Concurrent Multipath Video Delivery Over Heterogeneous Wireless Networks. *IEEE Transactions on Circuits and Systems for Video Technology* 25, 7 (July 2015), 1175–1189. https://doi.org/10.1109/TCSVT.2014.2376138

[92] Changqiao Xu, Tianjiao Liu, Jianfeng Guan, Hongke Zhang, and Gabriel-Miro Muntean. 2013. CMT-QA: Quality-Aware Adaptive Concurrent Multipath Data Transfer in Heterogeneous Wireless Networks. *IEEE Transactions on Mobile Computing* 12, 11 (November 2013), 2193–2205. https://doi.org/10.1109/TMC.2012.189

[93] Dongzhu Xu, Anfu Zhou, Xinyu Zhang, Guixian Wang, Xi Liu, Congkai An, Yiming Shi, Liang Liu, and Huadong Ma. 2020. Understanding Operational 5G: A First Measurement Study on Its Coverage, Performance and Energy Consumption. In *Proceedings of the Annual conference of the ACM Special Interest Group on Data Communication on the applications, technologies, architectures, and protocols for computer communication* (2024-03-29). ACM, Virtual Event USA, 479–494. https://doi.org/10.1145/3387514.3405882

[94] Praveen Kumar Yadav and Wei Tsang Ooi. 2020. Tile Rate Allocation for 360-Degree Tiled Adaptive Video Streaming. In *28th ACM International Conference on Multimedia* (Seattle, WA, USA). 3724–3733. https://doi.org/10.1145/3394171.3413550

[95] Alex Yelenevych. [n. d.]. The Future Of EdTech. https://www.forbes.com/sites/forbesbusinesscouncil/2022/12/26/the-future-of-edtech/?sh=4ed280296c2f. Published on Dec 26, 2022; Accessed on Apr 12, 2024.

[96] Xiaoqi Yin, Abhishek Jindal, Vyas Sekar, and Bruno Sinopoli. 2015. A Control-Theoretic Approach for Dynamic Adaptive Video Streaming over HTTP. In *Proceedings of the 2015 ACM Conference on Special Interest Group on Data Communication* (2023-12-28). ACM, London United Kingdom, 325–338. https://doi.org/10.1145/2785956.2787486

[97] Xinjie Yuan, Mingzhou Wu, Zhi Wang, Yifei Zhu, Ming Ma, Junjian Guo, Zhi-Li Zhang, and Wenwu Zhu. 2022. Understanding 5G performance for real-world services: a content provider's perspective. In *Proceedings of the ACM SIGCOMM 2022 Conference* (2024-03-29). ACM, Amsterdam Netherlands, 101–113. https://doi.org/10.1145/3544216.3544219

[98] Yunze Zeng, Parth H. Pathak, and Prasant Mohapatra. 2014. A first look at 802.11ac in action: Energy efficiency and interference characterization. In *2014 IFIP Networking Conference* (2024-03-29). IEEE, Trondheim, Norway, 1–9. https://doi.org/10.1109/IFIPNetworking.2014.6857103

[99] Zhilong Zheng, Yunfei Ma, Yanmei Liu, Furong Yang, Zhenyu Li, Yuanbo Zhang, Jiuhai Zhang, Wei Shi, Wentao Chen, Ding Li, Qing An, Hai Hong, Hongqiang Harry Liu, and Ming Zhang. 2021. XLINK: QoE-driven multi-path QUIC transport in large-scale video services. In *Proceedings of the 2021 ACM SIGCOMM 2021 Conference* (2022-09-30). ACM, Virtual Event USA, 418–432. https://doi.org/10.1145/3452296.3472893

[100] Xiao Zhu, Jiachen Sun, Xumiao Zhang, Y Ethan Guo, Feng Qian, and Z Morley Mao. 2020. MPBond: efficient network-level collaboration among personal mobile devices. In *Proceedings of the 18th International Conference on Mobile Systems, Applications, and Services*. 364–376. https://doi.org/10.1145/3386901.3396600

[101] Xutong Zuo, Yong Cui, Xin Wang, and Jiayu Yang. 2022. Deadline-aware Multipath Transmission for Streaming Blocks. In *IEEE INFOCOM 2022 - IEEE Conference on Computer Communications*. 2178–2187. https://doi.org/10.1109/INFOCOM48880.2022.9796942

eCHFD : extended Ceasefire Hierarchical Firearm Dataset

Loubna Lechelek
Sylvie Chambon
Alain Crouzil
Saddam Abdulwahab
firstname.lastname@irit.fr
Institut de Recherche en Informatique de Toulouse (IRIT)
Toulouse, France

Grégory Jalabert
Christian Brocard
Charles-Edouard Coquillard
Laurence Abadie
Bruno Sera
firstname.lastname@interieur.gouv.fr
Service National de Police Scientifique (SNPS) -
Laboratoire de Police Scientifique de Toulouse (LPS 31)
Toulouse, France

Thierry Hartmann
thierry.hartmann@interieur.gouv.fr
Direction de la Coopération Internationale de Sécurité
Nanterre, France

Marjorie Le Bras
marjorie.le-bras@interieur.gouv.fr
Direction Nationale de la Police Judiciaire
Nanterre, France

Abstract

In recent years, the demand for large, diverse, and well-annotated firearm datasets has grown, particularly for advancing firearm recognition systems. However, existing datasets often lack sufficient variety in viewpoints, configurations, and model versions. To address these challenges, this paper introduces an extented version of the Ceasefire Hierarchical Firearm Dataset (eCHFD), a large image dataset of firearms consisting of over 93,000 images in 505 classes. It was constructed from more than 240 videos filmed at the Toulouse Forensics Laboratory (France) and further enriched with images from the existing CHFD dataset and additional downloaded images. Automated post-processing techniques were employed to efficiently build this large-scale dataset. In addition, a hierarchical annotation scheme validated by forensics experts was used to classify firearms at multiple levels, enabling its utility for fine-grained classification tasks. Experimental results was conducted using models such as HERBS and FGVC-PIM to demonstrate the effectiveness of the dataset.

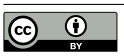
CCS Concepts

• **Computing methodologies** → **Computer vision**.

Keywords

Extented firearm dataset, Hierarchical dataset, Fine-grained classification, Firearm recognition.

ACM Reference Format:
Loubna Lechelek, Sylvie Chambon, Alain Crouzil, Saddam Abdulwahab, Grégory Jalabert, Christian Brocard, Charles-Edouard Coquillard, Laurence Abadie, Bruno Sera, Thierry Hartmann, and Marjorie Le Bras. 2025. eCHFD : extended Ceasefire Hierarchical Firearm Dataset. In *ACM Multimedia Systems Conference 2025 (MMSys '25), March 31-April 4, 2025, Stellenbosch, South Africa.* ACM, New York, NY, USA, 7 pages. https://doi.org/10.1145/3712676.3718333



This work is licensed under a Creative Commons Attribution 4.0 International License.
MMSys '25, Stellenbosch, South Africa
© 2025 Copyright held by the owner/author(s).
ACM ISBN 979-8-4007-1467-2/2025/03
https://doi.org/10.1145/3712676.3718333

1 Introduction

Deep learning has transformed computer vision, enabling substantial advances in tasks such as object recognition and image classification. Unlike earlier methods that relied heavily on handcrafted features [20], deep learning techniques [4] are designed to automatically learn features and extract meaningful representations. This pivotal shift in Artificial Intelligence (AI) has gained significant attention in sensitive areas such as security, forensic science and law enforcement. An example of this interest is the Horizon 2020 European project, CEASEFIRE [16][1], within which our research was conducted. CEASEFIRE aims to leverage AI to develop an advanced technical system that supports Law Enforcement Agencies (LEAs) in combating firearms trafficking. One of the objectives is to develop a mobile application with an embedded AI module as shown in Figure 1, trained to recognize and classify firearm models from captured images. The goal is to enable the rapid and accurate identification of critical firearm attributes, including type, brand, and specific model to assist LEA in their work.

The pioneering AI architectures for image classification, including ResNet [8], VGG [22], and EfficientNet [24], are based on Convolutional Neural Networks (CNNs) [10]. These methods can be considered as multi-scale analysis, as features are extracted through successive convolutional layers. More recently, Vision Transformers (ViTs) have introduced a paradigm shift by replacing convolutional operations with self-attention mechanisms. Models such as the original ViT [5] and Swin Transformer [15] capture relationships across different regions of an image, enabling more global feature representations. Both CNNs and ViTs have demonstrated exceptional performance in standard image classification tasks.

More recently, fine-grained visual recognition has emerged as a specialized domain focused on distinguishing highly similar object that belong to separate classes. These methods integrate semantic and low-level features. For instance, the Cross-layer Mutual Attention Learning Network, CMAL-Net [14], combines shallow

[1] https://ceasefire-project.eu/

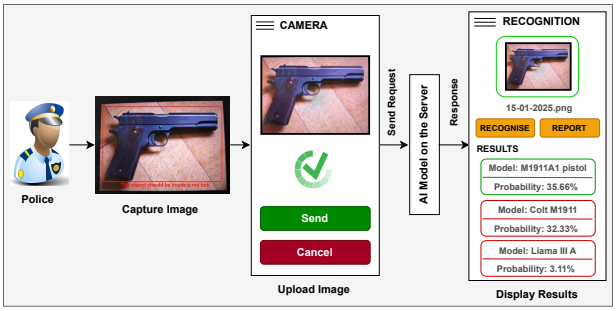


Figure 1: Overview of the firearm recognition tool developed within the Ceasefire European Project. A police officer uses a mobile application to capture an image of a seized firearm. The captured image is sent as a request to an AI system hosted on a server, which processes the image to identify the firearm model. The result is then sent back to the mobile application, allowing the police officer to view the recognition outcome directly on the device.

and deep CNN layers, using attention regions as a form of data augmentation in order for the model to focus on discriminative features. Hybrid approaches like High-temperature Refinement and Background Suppression HERBS [2] further enhance fine-grained recognition by integrating Swin Transformers for global feature extraction with Feature Pyramid Networks (FPNs) [13] to build a multi-scale feature hierarchy. This synergy effectively bridges high-level semantics and low-level details, producing rich and robust data representations.

For firearm classification, most methods focus predominantly on coarse-grained categorization [9, 18], by classifying firearms into broad categories such as `pistol` or `revolver`. Consequently, the available databases are annotated according to these limited categories. Furthermore, this restricted classification does not meet the recognition requirements for LEA. To our knowledge, Firearms14K [7] was the first publicly available dataset specifically designed for fine-grained firearm classification. It includes 14,674 images of 167 firearm models, collected from online sources. The dataset offers a variety of instances within the same category, presenting firearms under different lighting conditions, in various colors, and across diverse environments (both indoor and outdoor). However, each class includes only a single viewpoint per weapon instance and provides just one configuration per firearm, limiting the range of representations for each example. This is one of the reasons we have previously introduced the Ceasefire Hierarchical Firearm Database (CHFD) [17] - an RGB image database of firearms consisting of over 40,000 images across 260 classes. A significant portion of the dataset was derived from video recordings captured from diverse perspectives at the forensic science laboratory in Toulouse, France. Frames were extracted from these videos and processed to preserve only the most relevant ones. To enhance the database diversity, these images were merged with the existing Firearms 14k dataset and complemented with additional images manually downloaded from the internet. However, only one configuration[2] per firearm model was considered. In addition, the creation of CHFD required time-consuming manual post-processing, which proved to be inefficient and impractical for expanding the database. Consequently, we need to automate this post-processing.

Building upon the success of the CHFD, we introduce eCHFD, a significantly expanded version of the Ceasefire Hierarchical Firearm Dataset. While CHFD offered a diverse collection of 260 firearm models, eCHFD further enhances this diversity by adding 245 new firearm models, selected from the firearm collection of the police forensic laboratory in Toulouse. This expansion results in a total of 93,380 images across 505 distinct models. eCHFD not only expands the number of models but also introduces multiple configurations for each physical model. These configurations include pistols with open and closed slides, revolvers with swing-out cylinders, break-action rifles, firearms with foldable cocking levers, Firearms with or without magazines and firearms equipped with accessories like scopes or carrying straps. Moreover, eCHFD provides a broader range of versions for some firearm models. For example it includes multiple versions of semi-automatic Beretta pistols such as the Beretta 70, Beretta 81, Beretta 82BB, Beretta 951 brigadier, Beretta 93-R etc. Such variations, which were absent in the original CHFD and the Firearms14k dataset, offer a more comprehensive and detailed representation of firearms. To address the challenges of constructing a large-scale dataset, we have incorporated automated post-processing techniques, significantly reducing manual intervention and improving processing efficiency.

The remaining of this of this paper is structured as follows. Section 2 provides an in-depth review of existing weapon datasets, identifying their limitations and gaps. Section 3 explains the reason behind the development of the expanded Ceasefire Hierarchical Firearm Dataset (eCHFD), details the video acquisition protocol and outlines the hierarchical annotation process. Section 4 describes how the proposed dataset was built using automated post-processing. Section 5 elaborates on the automated data collection methods employed to further enrich the dataset. To validate the effectiveness of our dataset, Section 6 presents the experimental results. Finally, Section 7 concludes the paper and outlines potential directions for future research.

2 Existing firearm datasets

To effectively train a deep learning model for fine-grained firearm classification, a large and diverse dataset is needed. This dataset should encompass a wide range of firearm models, with sufficient high-quality images per model to enable accurate recognition. More precisely, we identify that an ideal database should include at least 20 to 30 images per class; feature at least one physical model per class and ensure that the firearm is the primary subject in each photograph. Existing weapon datasets present significant challenges for this task due to several limitations in both image diversity and quantity. For instance, the Internet Movie Firearms Database (IMFDB) [3] offers a vast collection of over 400,000 images. However, this

[2] specific design and assembly of a firearm, including its components, features, and accessories
[3] https://imfdb.org

Ref	Objective	Nb classes	Nb images	Nb levels	Multiple viewpoints	Multiple configurations	Multiple versions
[21]	Detection and classification	2	51K	1	×	×	×
[1]	Detection	2	8327	1	×	×	×
[25]	Detection	2	7767	1	×	×	×
[18]	Coarse classification	5	8843	1	×	×	×
[9]	Coarse classification	7	5214	1	×	×	×
[7]	Fine-grained classification	167	14K	1	×	×	✓
[17]	Fine-grained classification	260	40,789	4	✓	×	✓
Ours	Fine-grained classification	505	93,380	4	✓	✓	✓

Table 1: Summary of datasets used in existing works. This table provides an overview of the datasets employed in previous studies, indicating whether the three key requirements—variety of viewpoints, versions, and configurations—are met for each weapon dataset.

database typically provides fewer than five images per model, often captured from a single viewpoint. Similarly, the ATF website [4] provides information on over 100 firearm models but includes only one image per model.

Some datasets are specifically designed to address the challenge of firearm detection in surveillance video footage. For example, Qi al. [21] introduced a large-scale CCTV (Closed-Circuit Television) dataset containing two subsets of 51,000 firearm images. The first subset consists of annotated firearm images designed for gun detection, while the second subset includes cropped firearm images derived from the first subset for firearm classification. The database was collected primarily from the IMFDB website, and then completed with images extracted from public websites related to these papers [12, 19]. Bhatti et al. [1] introduced a weapon dataset specifically designed for detecting weapons in real time CCTV videos. The images were collected from a variety of platforms, including their own camera captures, manually curated images from the internet, extracted frames from YouTube CCTV videos and the IMFDB website. Yellapragada et al. [25] introduced a handgun image database designed to detect handguns in CCTV footage. The database was curated by selecting images from three existing CCTV imagery datasets [6, 11, 23]. These three datasets provide images captured from CCTV viewpoints that do not align with our intended use case. Specifically, the weapons are typically very small, held by persons, with low visual salience, and are often occluded or easily confused with other small objects. Additionally, the weapons do not appear to be the primary focus of the images and are annotated with drawn rectangles just to indicate their locations.

The authors of [18] have introduced a weapon dataset designed to detect and classify five distinct weapon categories: assault rifles, handguns, knives, shotguns, and sniper rifles. The images for this dataset were collected through webscraping techniques. Similarly, Volkan et al. [9] presented a database tailored for weapon detection and classification, consisting of images downloaded from the internet. These images were meticulously labeled and organized into seven distinct classes: assault rifles, bazookas, grenades, hunting rifles, knives, pistols, and revolvers. Both datasets focus on general weapon categories, which limits the ability of classification models to capture the subtle visual distinctions between specific firearm models within these general categories. The paper by Hao et al. [7] presents the first publication to address the specific challenge of fine-grained firearm recognition. Their work introduced the Firearm 14k dataset, which includes 14,674 images covering 167 distinct firearm models. This dataset marks a significant advancement by enabling models to differentiate between closely related firearm models within the same category. To create the Firearm 14k dataset, the authors meticulously curated a list of 167 commonly used keywords from websites and forums. For each keyword, they automatically downloaded between 300 to 500 publicly available images, resulting in numerous instances for each model. In most images, the firearm is the primary focus, fully visible, and occupies a substantial portion of the frame. These datasets are summarized in Table 1. For each weapon dataset, we indicate whether the three requirements, regarding the variety of viewpoints, versions and configurations are satisfied.

3 Ceasefire Hierarchical Firearm Dataset (eCHFD)

In the scope of the Ceasefire project, we presented the Ceasefire Hierarchical Firearm Dataset (CHFD). The dataset was constructed using over 23,000 images acquired at the Toulouse Forensics Laboratory (France). These images were extracted from video captured on-site, offering multiple viewpoints for each firearm model. The dataset was further enriched with the Firearm 14k dataset, complemented by an additional 2,526 images sourced from the web. Each model was meticulously annotated by an expert across four hierarchical levels: family, group, type, and model. The key advantages of the (CHFD) is that it provides multiple viewpoints for each acquired model, which enhances the AI model ability to recognize the weapon from any angle. However, only one configuration per model was considered. Additionally, the creation of this dataset required manual post-processing. To address these limitations, we present (eCHFD), a comprehensive expansion of the Ceasefire Hierarchical Firearm Dataset, designed to overcome the constraints of previous datasets. The dataset leverages the strengths of Firearms 14k and CHFD while offering a significant expansion by incorporating a diverse array of new firearm models. The originality of this dataset lies in the fact that each model is presented with at least one configuration and captured from multiple viewpoints. Moreover,

[4] https://www.atf.gov



(a) Variations of Beretta semi-automatic pistols. Different versions of Beretta pistols are shown, featuring models with closed slides in the top row and open slides in the bottom row.



(b) Illustration of firearm configurations. The top row showcases each firearm model in its primary configuration, while the bottom row presents an alternate configuration of the same models, highlighting design or functional variations.

Figure 2: Example images from the extended Ceasefire Hierarchical Weapon Dataset (eCHFD).

several firearm models are available in different versions as shown in Figures 2a and 2b. The resulting dataset is publicly available at the following url: https://zenodo.org/records/14908294. The subsections below provide a detailed explanation of how we developed eCHFD and how automated post-processing was introduced to encompass 505 models and 93,380 images.

3.1 Firearm acquisitions at police premises

As part of the Ceasefire project, we had the opportunity to visit the Toulouse forensics laboratory, (Service National de Police Scientifique, SNPS), which houses a vast collection of over 2,000 firearms of diverse types. Most of these firearms are available in multiple versions and can be acquired with varying configurations. From this collection, we successfully captured 245 firearm models. The acquisition protocol for these firearms followed the same methodology used during the creation of the CHFD dataset [17]. This protocol involves the following steps:

- The firearm is placed on a flat surface, such as a table.
- The firearm is filmed along a path that captures it from multiple perspectives.
- Each firearm is filmed from both sides.
- Each video lasts approximately 2 minutes, with 1 minute dedicated to each side of the firearm.

The camera used for acquisitions was the compact camera, Olympus Om System Tough TG-7. For firearms with multiple versions or configurations, the acquisition protocol is repeated for each variant. We extracted images with a resolution of 1920×1080 pixels from each video every 0.4 seconds using FFmpeg (an open-source software used for handling multimedia files, including video and audio processing, conversion, and extraction.) [5], resulting in approximately 300 frames per video. These frames are then processed using an automatic post-processing as exposed in section 4

3.2 Hierarchical annotation scheme

To ensure consistency with the CHFD database [17], we adopted the same annotation protocol to label the 245 firearm models. This protocol, developed by experts at the Toulouse Forensic Science Laboratory, classifies firearms using a four-level hierarchy: family, group, type, and model. Figure 4 illustrates the first three hierarchical levels, while the full hierarchy is detailed as follows:

- Family: This binary attribute consists of two subcategories: handgun and long gun.
- Group: Seven distinct groups are defined within the firearm hierarchy, two under handgun: pistol and revolver, and five under long gun: submachine gun, rifle, assault rifle, shotgun, and light machine gun. This level often reflects characteristics such as ammunition type or magazine design.
- Type: The firearm operation and barrel shape are specified, comprising 34 distinct types. For example, 'psa' denotes a semi-automatic pistol.
- Model: The most granular level, distinguishing between individual firearm models such as the Marlin 1894 and Browning BL-22. The dataset includes 505 models.

[5] https://www.ffmpeg.org/

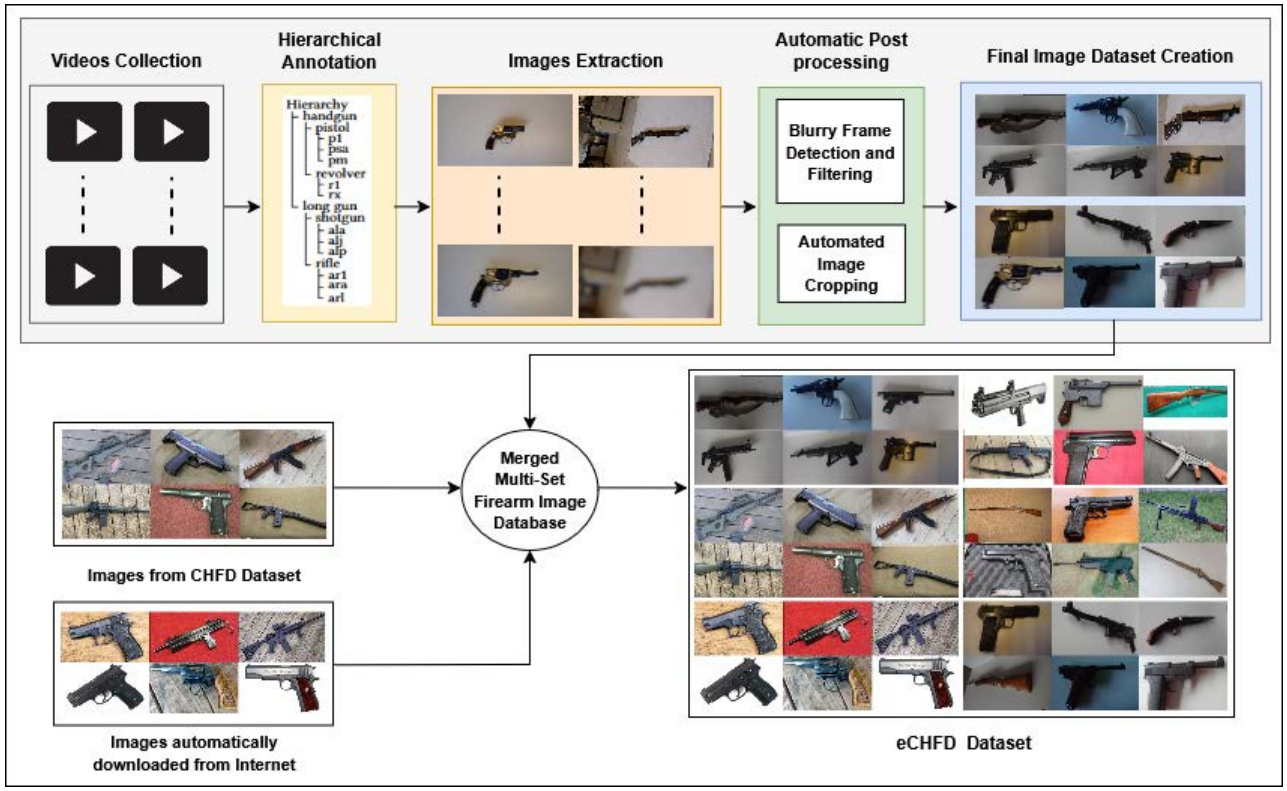


Figure 3: Workflow to creat the eCHFD Dataset. The upper part of the figure outlines the initial pipeline, starting with the collection of videos featuring firearms. Each video is hierarchically annotated and categorized, followed by frame extraction. Blurry images are automatically filtered out, and a trained model isolates firearms by removing unrelated objects, resulting in a clean and robust dataset. The lower part of the figure illustrates the integration process, where the newly created dataset is merged with the existing CHFD dataset and supplemented with additional automatically downloaded images.

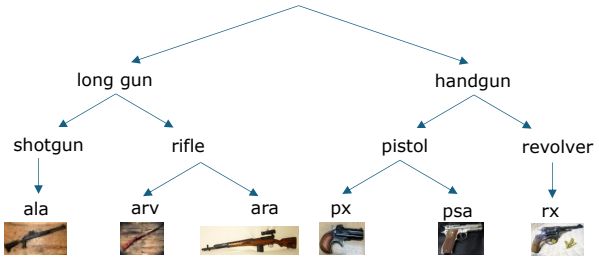


Figure 4: Firearm hierarchy defined for the eCHFD. The figure presents some subcategories from the first three levels of the hierarchy. The model level is represented by a single example, shown alongside an image

.

4 Automatic post-processing of extracted frames

After extracting the frames, we found a significant number of irrelevant images in the dataset. Frames where the firearm was not fully visible were manually removed. To handle additional issues, such as blurriness or the presence of non-firearm objects, we implemented an automatic post-processing pipeline, as illustrated in Figure 3, comprising the following steps:

4.1 Blurry frame detection and filtering

To filter out blurry frames, we employ the variance of Laplacian method to measure the amount of blur. The Laplacian operator is a second order derivative filter used to detect edges in an image by highlighting regions of rapid intensity change. We apply a convolution process to each frame with the Laplacian operator and we compute the variance of the resulting image. A high variance corresponds to a sharp image with well-defined edges, while a low variance indicates minimal edge information. If the calculated variance falls below a predefined threshold, the image is marked as blurry. Through experimentation, a threshold value of 12 has been found to be effective for our dataset.

4.2 Automated image cropping

In order to detect and locate the firearm in each frame, we employed the YOLOv8 object detector from Ultralytics [6]. The YOLOv8 model, pre-trained to detect firearms, was used to generate bounding boxes around the firearms. We developed a custom script to process the

[6]https://docs.ultralytics.com/fr

detected bounding boxes, crop the identified objects from the original images, and save each cropped object as a separate image. To ensure that the cropped firearms were not too close to the edges of the bounding box, we applied a scale factor to adjust the bounding box dimensions.

5 eCHFD enrichment using webscrapping

We have used a pre-existing module : simple_image_download to scrape images from the internet. The script functionality is as follows:

- We provide a list of search keywords and specify the desired number of images to download for each keyword.
- The keywords are transformed into search URLs that link to Google Image Search pages. The HTML content of these URLs is then downloaded.
- The downloaded HTML content is scanned to locate image URLs, which are filtered to retain only those with supported file extensions (jpg and png).
- A dedicated directory is created for each keyword to organize the downloaded images. The validated image files are subsequently downloaded and saved in their respective directories.

After scraping the images, a firearm expert from law enforcement identified and removed those that were not relevant to the specified category, resulting in a final set of 2,133 downloaded images.

6 Experimental results

We have used two different models for fine-grained image classification, trained and evaluated on our dataset (eCHFD). The models employed for the experimental analysis are HERBS [2] and FGVC-PIM [3]. For both models, the dataset was divided into three groups: 85 % for training, 10% for validation, and 5% for testing. The training and validation datasets were used during the training process. Top-1 to Top-3 accuracies were obtained by testing the models on the test dataset. The training was performed for exactly 30 epochs for each model in order to compare them on equal grounds. Data augmentation was applied to enhance the robustness and generalization capabilities of the models. Techniques such as blurring, rotation, flipping, and color jittering were employed. Training was performed on GPU NVIDIA with 64 GB of memory.

Looking at the training and validation losses of Figures 5a and 5b, it can be seen that the two models exhibit consistent decreases in loss over the epochs. Both training and validation losses converge, with a small gap between them, indicating effective learning with the absence of overfitting and good generalization to the validation dataset. Regarding the training and validation accuracy curves, it can be seen that both models achieve high and stable accuracy over the epochs. The training accuracy increases during the initial epochs and achieves 100%, indicating effective learning of the training dataset. Similarly, the validation converges near the training accuracy with negligible gap between the two curves. That means that both models perform well on our dataset.

We also evaluated the performance of both HERBS and FGVC models on the validation set using multiple metrics, including top-1 accuracy, F1-score, precision, and recall. As shown in the Table 2 both models achieved good results which demonstrate their robustness and effectiveness on our dataset. As for the test set, We evaluated the models performance using top-1, top-3, and top-5 accuracy. As shown in Table 3, the results highlight the models capability to make accurate predictions on our dataset.

Method	Top-1 Accuracy	F1-score	Precision	Recall
HERBS	98.77%	98.1%	98.2%	98.1%
FGVC-PIM	97.93%	90.3%	90.3%	90.2%

Table 2: Performance of HERBS and FGVC-PIM on the validation set. This table presents the performance metrics of HERBS and FGVC-PIM methods, including Top-1 accuracy, F1-score, precision, and recall

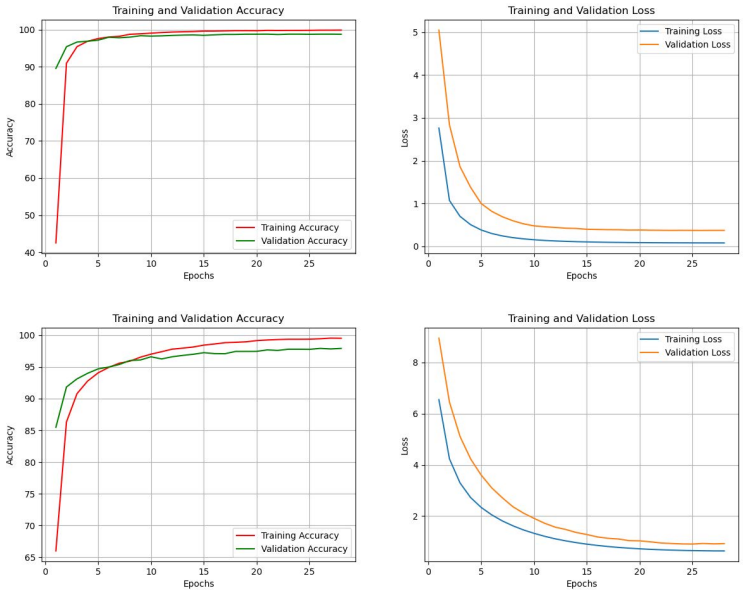
Method	Top-1	Top-3	Top-5
HERBS	98.33%	99.80%	99.86%
FGVC-PIM	97.2%	98.6%	98.8%

Table 3: Top-1, Top-3, and Top-5 accuracies of HERBS and FGVC-PIM on the test dataset.

Our dataset is highly imbalanced; the number of images in a class varies from 21 to over 1376. To study the influence of class imbalance on the performance of the model, we evaluated the F1-score, precision, recall, and the false positive rate (FPR) for each class. The results for the minority class (the class with the fewest images) obtained with HERBS demonstrated high performance, with an F1-score of 99%, precision of 100%, recall of 0.99, and an FPR of 0.0. These results indicate that the model performs well even for classes with limited representation. As a result, we opted not to weight the loss function based on the number of samples per class, as the models effectively handles minority classes without the need for additional weighting. Furthermore, in our context, there is no specific firearm model for which we aim to assign a great importance.

7 Conclusion

In this paper, we introduced the extented Ceasefire Hierarchical Firearm Dataset (eCHFD), a significant enhancement of the original Ceasefire Hierarchical Firearm Dataset (CHFD). This expansion was achieved by capturing 245 additional firearm models at the Toulouse Forensics Laboratory, which houses a vast collection of firearms. The eCHFD offers a substantial improvement by providing images of firearms from multiple viewpoints, representing various configurations, and including multiple versions for most firearms. Experiments were conducted on eCHFD and demonstrated promising results in terms of fine-grained firearm classification. Looking ahead, further acquisitions following the same protocol could expand the eCHFD to include the full range of firearm models from the police forensic laboratory collection in Toulouse (France).



(a) HERBS

(b) FGVC-PIM

Figure 5: Accuracy and loss curves during training

Acknowledgments

The research leading to these results received funding from the European Union's Horizon Europe research and innovation programme. This publication reflects only the authors' views. The European Union is not liable for any use of the information contained herein. We also thank the SNPS for their assistance in facilitating the acquisitions and for granting permission to publish the images.

References

[1] M. T. Bhatti, M. G. Khan, M. Aslam, and M. J. Fiaz. 2021. Weapon Detection in Real-Time CCTV Videos Using Deep Learning. *IEEE Access* 9 (2021), 34366–34382.

[2] P.-Y. Chou, Y.-Y. Kao, and C.-H. Lin. 2023. Fine-grained Visual Classification with High-temperature Refinement and Background Suppression. *arXiv* 2303.06442 (2023).

[3] P.-Y. Chou, C.-H. Lin, and W.-C. Kao. 2022. A Novel Plug-in Module for Fine-Grained Visual Classification. *ArXiv* abs/2202.03822 (2022).

[4] L. Deng and D. Yu. 2014. Deep Learning: Methods and Applications. *Foundations and Trends in Signal Processing* 7, 3–4 (2014), 197–387.

[5] A. Dosovitskiy, L. Beyer, A. Kolesnikov, D. Weissenborn, X. Zhai, T. Unterthiner, M. Dehghani, M. Minderer, G. Heigold, S. Gelly, J. Uszkoreit, and N. Houlsby. 2021. An Image is Worth 16x16 Words: Transformers for Image Recognition at Scale. In *International Conference on Learning Representations*.

[6] J. L. S. González, Z. Carlos, J. A. Álvarez-García, L. M. S. Morillo, and F. S. Caparrini. 2020. Real-time gun detection in CCTV: An open problem. *Neural Networks* 132 (2020), 297–308.

[7] J. Hao, J. Dong, W. Wang, and T. Tan. 2018. DeepFirearm: Learning Discriminative Feature Representation for Fine-grained Firearm Retrieval. In *International Conference on Pattern Recognition*. 3335–3340.

[8] K. He, X. Zhang, S. Ren, and J. Sun. 2016. Deep Residual Learning for Image Recognition. In *IEEE Conference on Computer Vision and Pattern Recognition*. 770–778.

[9] V. Kaya, S. Tuncer, and A. Baran. 2021. Detection and Classification of Different Weapon Types Using Deep Learning. *Applied Sciences* 11 (2021).

[10] A. Krizhevsky, I. Sutskever, and G. E. Hinton. 2017. ImageNet classification with deep convolutional neural networks. *Commun. ACM* 60 (2017), 84–90.

[11] J.Y. Lim, M. I. Al Jobayer, V. M. Baskaran, J. M.Y. Lim, J. See, and K.S. Wong. 2021. Deep multi-level feature pyramids: Application for non-canonical firearm detection in video surveillance. *Engineering Applications of Artificial Intelligence* 97 (2021), 104094.

[12] J. Lim, M. I. Al Jobayer, V. M. Baskaran, J. M Lim, K. Wong, and J. See. 2019. Gun Detection in Surveillance Videos using Deep Neural Networks. In *Asia-Pacific Signal and Information Processing Association Annual Summit and Conference*. 1998–2002.

[13] T.-Y. Lin, P. Dollár, R. Girshick, K. He, B. Hariharan, and S. Belongie. 2017. Feature Pyramid Networks for Object Detection. In *IEEE Conference on Computer Vision and Pattern Recognition*. 936–944.

[14] D. Liu, L. Zhao, Y.u Wang, and J. Kato. 2023. Learn from each other to Classify better: Cross-layer mutual attention learning for fine-grained visual classification. *Pattern Recognition* 140 (2023), 109550.

[15] Z. Liu, Y. Lin, Y. Cao, H. Hu, Y. Wei, Z. Zhang, S. Lin, and B. Guo. 2021. Swin Transformer: Hierarchical Vision Transformer using Shifted Windows. In *International Conference on Computer Vision*. 9992–10002.

[16] I. Mademlis, J. Cani, M. Mancuso, C. Paternoster, E. Adamakis, G. Margetis, S. Chambon, A. Crouzil, L. Lechelek, G. Dede, S. Evangelatos, G. Lalas, F. Mignet, P. Linardatos, K. Kentrotis, H. Gierszal, P. Tyczka, S. Karagiorgou, G. Pantelis, G. Stavropoulos, K.s Votis, and G. Th. Papadopoulos. 2024. CEASEFIRE: An AI-powered system for combatting illicit firearms trafficking. In *Workshop on Big Data Applications for Fight against Crime and Terrorism, IEEE Big Data, BD*.

[17] T. Malon, S. Chambon, A. Crouzil, L. Lechelek, G. Jalabert, C. Brocard, N. Bernardeau, L. Abadie, B. Sera, T. Hartmann, and M. Le Bras. 2024. Ceasefire Hierarchical Weapon Dataset. In *ACM Multimedia Systems Conference*. 243–249.

[18] S. B. Mane. 2024. Weapon Detection and Classification Using Deep Learning. *Journal of Engineering and Technology for Industrial Applications* 10 (2024), 19–26.

[19] R. Olmos, S. Tabik, and F. Herrera. 2018. Automatic handgun detection alarm in videos using deep learning. *Neurocomputing* 275 (2018), 66–72.

[20] N. O'Mahony, S. Campbell, A. Carvalho, S. Harapanahalli, G. V. Hernandez, V. Gustavo, L. Krpalkova, D. Riordan, and J. Walsh. 2019. Deep Learning vs. Traditional Computer Vision. In *Computer Vision Conference, CVC*. 128–144.

[21] D. Qi, W. Tan, Z. Liu, Q. Yao, and J. Liu. 2021. A Dataset and System for Real-Time Gun Detection in Surveillance Video Using Deep Learning. In *IEEE International Conference on Systems, Man, and Cybernetics*. 667–672.

[22] K. Simonyan and A. Zisserman. 2015. Very Deep Convolutional Networks for Large-Scale Image Recognition. In *International Conference on Learning Representations*.

[23] W. Sultani, C. Chen, and M. Shah. 2018. Real-World Anomaly Detection in Surveillance Videos. In *IEEE Conference on Computer Vision and Pattern Recognition*. 6479–6488.

[24] M. Tan and Q. V. Le. 2019. EfficientNet: Rethinking Model Scaling for Convolutional Neural Networks. In *International Conference on Machine Learning*, Vol. 97. 6105–6114.

[25] S. Yellapragada, Z. Li, K. B. Doshi, P. M. Mhasakar, H. Fan, J. Wei, E. Blasch, B. Zhang, and H. Ling. 2023. CCTV-Gun: Benchmarking Handgun Detection in CCTV Images. *arXiv preprint arXiv:2303.10703* (2023).

WIDE-VR: An open-source prototype for web-based VR through adaptive streaming of 6DoF content and viewport prediction

May Lim
maylim@comp.nus.edu.sg
National University of Singapore
Singapore

Abdelhak Bentaleb
abdelhak.bentaleb@concordia.ca
Concordia University
Canada

Roger Zimmermann
rogerz@comp.nus.edu.sg
National University of Singapore
Singapore

Abstract

This paper tackles the challenge of designing and implementing a complete, operational, and extensible web-based virtual reality (VR) system for stored six degrees-of-freedom (6DoF) content. We present a unique, open-source volumetric video streaming system, termed WIDE-VR, that leverages cutting-edge web technologies, including a WebGL-based rendering engine, Draco's real-time in-browser decoder, and HTTP/3 over QUIC transport. We also explore various adaptive streaming and viewport prediction strategies to achieve improvements in bandwidth efficiency, reduced rebuffering, and minimized quality fluctuations. Our experimental results demonstrate the system's capabilities and potential in delivering immersive, high-quality VR experiences directly via web platforms, fostering broader accessibility and scalability for VR applications.

CCS Concepts

• **Information systems** → **Multimedia streaming**.

Keywords

Virtual reality, Adaptive streaming, DASH, Mesh compression, Web

ACM Reference Format:
May Lim, Abdelhak Bentaleb, and Roger Zimmermann. 2025. WIDE-VR: An open-source prototype for web-based VR through adaptive streaming of 6DoF content and viewport prediction. In *ACM Multimedia Systems Conference 2025 (MMSys '25), March 31-April 4, 2025, Stellenbosch, South Africa.* ACM, New York, NY, USA, 7 pages. https://doi.org/10.1145/3712676.3718334

1 Introduction

The advancement of web-based virtual reality (VR) has revolutionized user interaction with digital content, enabling immersive experiences that were previously accessible only through specialized hardware. At the core of this evolution lies six degrees-of-freedom (6DoF) content, which allows users to navigate and interact within virtual environments with full spatial freedom. However, the data-intensive nature of 6DoF content [11] presents significant challenges for real-time delivery over the Internet [1], necessitating efficient streaming solutions that ensure high-quality experiences while optimizing bandwidth, latency, and reliability. Addressing these challenges is critical to meeting the growing demand for accessible and scalable web-based immersive applications [18].

Adaptive streaming, also commonly known as adaptive bitrate streaming (ABR), is essential in streaming volumetric (6DoF) videos as it allows for adjustments to video quality based on real-time network conditions and/or user interactions [21]. DASH is a widely adopted adaptive streaming standard. In contrast to traditional 2D videos, by breaking down the video not only into segments but also into objects of varying quality, ABR helps ensure that users can access content with minimal interruptions. This is important for volumetric videos with 6DoF content, where large data sizes and computational complexity in their data handling can easily lead to performance issues. Alongside ABR, viewport prediction strategies improve streaming efficiency by anticipating which regions of the 6DoF scene a user is likely to view next [10]. By preloading relevant content based on the user's viewport movement in the scene, these strategies help reduce unnecessary data transmission and optimize resource use.

This paper introduces a unique volumetric video streaming system designed to optimize web-based delivery of 6DoF content and is also compatible with DASH. By integrating adaptive streaming techniques for bitrate selection and viewport prediction strategies for content preloading into a purpose-built architecture, the system demonstrates improvements in key quality-of-experience (QoE) metrics, including visual quality, rebuffering, and quality stability. Through our experiments, we validate the system's capability to address practical challenges in volumetric video delivery, offering insights into achieving seamless and immersive experiences in web-based environments. Our contributions advance the field of web-based streaming for 6DoF content as follows:

(1) We designed and implemented a unique and operational web-based volumetric video streaming system called WIDE-VR—Web-based Immersive Delivery Engine for VR applications that leverages the latest web technologies, including Babylon.js (an in-browser 3D graphics engine that supports WebGL, WebXR, and a Draco-based mesh decoder) and HTTP/3 (which is built on QUIC [13]). This implementation not only enhances the user experience and enables experimentation under real-world conditions, but also sets a new standard for future streaming applications.
(2) We adapted and implemented a suite of well-established adaptive bitrate (ABR) and viewport prediction algorithms within our DASH-based system to facilitate the efficient streaming of 6DoF content and provide a valuable resource for researchers and industry practitioners to further work on. The system is also easily extensible to support other ABR and viewport prediction algorithms. The components in (1) and (2) make up a

Permission to make digital or hard copies of all or part of this work for personal or classroom use is granted without fee provided that copies are not made or distributed for profit or commercial advantage and that copies bear this notice and the full citation on the first page. Copyrights for third-party components of this work must be honored. For all other uses, contact the owner/author(s).
MMSys '25, Mar 31-Apr 4, 2025, Stellenbosch, South Africa
© 2025 Copyright held by the owner/author(s).
ACM ISBN 979-8-4007-1467-2/2025/03
https://doi.org/10.1145/3712676.3718334

complete system that streams stored 6DoF content efficiently on a modern laptop. The source code of the system is made available at [16].

(3) Finally, we conduct experimental investigations to demonstrate and assess the performance of these algorithms within our system. These evaluations provide insights and recommendations that can inform future research on WIDE-VR and more generally in 6DoF streaming and practical applications, ultimately contributing to the evolution of streaming technologies.

The rest of the paper is organized as follows: Section 2 compares the system against related work. Section 3 details the design of the implemented system with a focus on the player, followed by Section 4 which describes the algorithms used for ABR and viewport prediction. Section 5 discusses the experimental investigations. Finally, Section 6 concludes the paper.

2 Related Work

This section provides an overview of other systems designed for immersive content delivery using web-based technologies. Table 1 summarizes the differences between WIDE-VR and these systems.

Gunkel et al. [8] present a web-based framework for social VR applications (named VRComm) that supports real-time photorealistic user representations. It combines video conferencing technologies like WebRTC with social VR capabilities like multi-user connection through a pipeline that captures, processes, transmits, and renders 3D user representations.

Farrugia et al. [6] present a DASH-based framework for adaptive streaming of 3D content in web-based VR scenarios. Their work integrates multiple utility metrics and adaptive streaming strategies to improve the delivery of 3D meshes based on user viewport and bandwidth conditions. The source code is available at [12]. While their work serves as the key inspiration to our system, one major limitation is that it only considers static 3D objects. Our work extends support for dynamic 3D objects, which requires additional considerations such as buffering and continuous playback, as well as additional HTTP/3 compatibility.

Cabrero et al. [4] present a prototype of a web-based platform that transmits and renders volumetric videos by leveraging web technologies like WebSockets for ease of compatibility across devices. Their work includes an empirical study evaluating the performance of the platform on eight different device models, revealing heterogeneity in device performance and identifying key bottlenecks like in-browser rendering limitations.

De et al. [5] present a streaming architecture designed for one-to-many applications like virtual classrooms and concerts. The work includes a capture-to-render pipeline built on WebRTC, however, the pipeline does not support in-browser rendering but instead renders the content in the Unity game engine. The source code is available at [9].

3 System Implementation

3.1 System Components

The diagram in Figure 1 provides a visual representation of the key components within the WIDE-VR system. Each component's purpose in the web-based player of the system is further elaborated below. We also discuss their interactions in the next section.

	[8] (2021)	[6] (2023)	[4] (2024)	[5] (2024)	WIDE-VR
Web-based rendering	✔	✔	✔	✗	✔
Dynamic 3D objects	✔	✗	✔	✔	✔
DASH-compatible	✗	✔	✗	✗	✔
Adaptive streaming	✔	✔	✗	✔	✔
Viewport prediction	✗	✔	✗	✗	✔
Code made available	✗	✔	✗	✔	✔

Table 1: Summary of differences between our system (WIDE-VR) and existing web-based immersive content delivery systems.

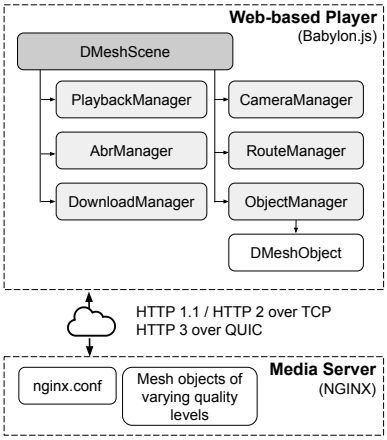


Figure 1: Key components of the WIDE-VR system focusing on streaming, decoding and playback.

- The **`DMeshScene`** (DMesh is an abbreviation of dynamic mesh) is the central component that coordinates various managers and objects within the VR scene. It has a constructor that takes instances of `ObjectManager`, `AbrManager`, `PlaybackManager`, `DownloadManager`, `CameraManager`, and `RouteManager` as parameters and relies on these managers to handle different aspects of the scene.
- The **`PlaybackManager`** class manages playback of the scene. It provides methods to begin the playback and controls the media playback within the scene, such as passing the relevant mesh object segments to the rendering engine and removing previously rendered segments when they are no longer needed.
- The **`AbrManager`** class handles the ABR algorithms. It includes methods to initialize the manager and execute a selected algorithm that determines the quality level to stream each mesh object segment. Depending on the ABR logic, the algorithm uses parameters such as network conditions and/or buffer level to perform bitrate decisions.
- The **`DownloadManager`** class handles downloading operations. It manages the downloading of media or other resources as required by the scene, such as the mesh object segments.
- The **`ObjectManager`** class manages `DMeshObject` instances. It provides methods to add, remove, and retrieve objects. This class ensures that all objects within the scene are properly managed and can be accessed easily.
- The **`DMeshObject`** class represents objects within the scene. It has attributes representing the metadata and media data of each mesh object, such as name, position, vertices, and textures. It also

contains a buffer to organize the media data by frames. This class is fundamental for representing and manipulating objects within the scene.
- The **`CameraManager`** class is responsible for camera operations, which also determines the viewport of the scene. It provides methods to initialize the camera and retrieve its position and rotation. This class ensures that the camera is properly set up and can be controlled within the scene. The class also executes the viewport prediction algorithms, which return the predicted camera position and rotation given a future timestamp.
- The **`RouteManager`** class manages routes within the scene which translate to the automated movement of the camera (and the viewport) along predetermined paths primarily for automated testing purposes. It provides methods to initialize a desired route at the beginning of playback.

3.2 System Interactions

The key system interactions can be broadly defined in the four steps described below. An accompanying sequence diagram, which illustrates these interactions in greater detail, is included in [16].

- **Initialization**: `ObjectManager` and `DMeshObject` are initialized for the scene. `ObjectManager` contains a list of references to all `DMeshObject` instances.
- **Segment Retrieval**: `DMeshScene` instructs `DownloadManager` to begin segment retrieval. `DownloadManager` queries `AbrManager` for quality level, determined based on the chosen ABR algorithm and the predicted visibility of each object. `DownloadManager` then retrieves segments for all objects and adds the segment data to the buffer of each `DMeshObject`.
- **Playback**: `DMeshScene` initiates playback on `PlaybackManager` which sets up a loop using `onBeforeRenderObservable` that notifies its `Observers` before the Babylon.js engine starts the rendering of each frame. On each loop iteration, `PlaybackManager` checks if the buffer is sufficient for playback. If so, it retrieves object data from the buffer of each `DMeshObject` and creates a mesh for the object in the scene, thereby rendering it.
- **Camera Movement**: `DMeshScene` instructs `RouteManager` to move the camera along the chosen route. The camera (and viewport) movement is continuously updated within the loop that is also constructed using `onBeforeRenderObservable`.

4 Algorithm Design

There are two key classes of algorithms implemented in WIDE-VR and they are responsible for rate adaptation and viewport prediction respectively. When a playback session is initiated, the first segments for all objects in the scene are downloaded in parallel as separate HTTP requests, and once the downloads for these object segments are completed, the second segments are downloaded, and so on. At each segment download boundary, the quality levels to be downloaded for each object-segment have to be decided. First, a viewport prediction algorithm is triggered to determine the viewport (coordinates and angles) and hence the visibility of each object at the point that the respective segment is to be played. For objects that are not visible, they are downloaded at the lowest possible quality, and for objects that are deemed visible, their quality levels are determined based on a second algorithm known as the object-level adaptation algorithm. Below, we describe in detail the alternative strategies implemented for both classes of algorithms.

4.1 Object-level Adaptation Algorithms

Adaptation algorithms (ABRs) for HTTP adaptive streaming [2] can be subcategorized into rate-based, buffer-based, and hybrid. Rate-based adaptation focuses on real-time monitoring of the available network bandwidth to make decisions about the quality of the content to be delivered. Buffer-based adaptation prioritizes maintaining an adequate buffer level (a preconfigured value) rather than directly responding to bandwidth fluctuations. These algorithms focus on the amount of content already downloaded and stored in the buffer to make quality decisions. In this work, we adapted and implemented strategies that fall under both categories to study the impact of rate-based versus buffer-based adaptation in our case of volumetric videos. Nonetheless, the system can be further extended with the implementation of other ABR algorithms.

4.1.1 Rate-based: Uniform and Greedy. For rate-based adaptation algorithms, we adopt the Uniform and Greedy algorithms in [21] to support mesh-based data instead of point clouds. The Uniform algorithm takes an even distribution approach to resource allocation. It assumes that all visible objects in the scene are equally important, regardless of their spatial relevance. In this work, we adopt the distance between the object and the user's viewport as the utility metric to signal spatial relevance. In this method, the estimated available bandwidth is divided uniformly across the objects. Specifically, starting with the object of the highest utility (i.e., of closest proximity to the viewport), the lowest available quality level is assigned to all objects before the next quality level is considered while keeping within the available bandwidth constraint. The Uniform algorithm is simple and predictable, making it uncomplicated to implement and manage. However, one limitation lies in its inability to prioritize critical regions in the scene, such as objects that are closer to the user's viewport, potentially leading to inefficient use of bandwidth in certain scenarios. In contrast, the Greedy algorithm allocates bandwidth greedily to objects in the scene based on their utility. Specifically, starting with the object of the highest utility (i.e., of closest proximity to the viewport), the highest possible quality level is assigned to each object before the next object is considered while keeping within the available bandwidth constraint. By focusing resources on critical regions in the scene, the Greedy algorithm could improve the perceptual quality. However, it tends to result in uneven quality across all objects, which may be undesirable in certain applications. Note that, for both algorithms, each object is initialized with the lowest quality level to ensure that no object is omitted from the scene.

4.1.2 Buffer-based: BOLA. BOLA (Buffer Occupancy based Lyapunov Algorithm), as proposed in [19] for 2D videos and adapted as BOLA360 in [24] for 360° videos, is a modern adaptation algorithm used in HTTP adaptive streaming to balance video quality and playback stability while minimizing rebuffering events. Unlike traditional rate-based or simplistic buffer-based approaches, BOLA incorporates a mathematical framework, known as the Lyapunov optimization framework, that dynamically optimizes quality selection based on both buffer occupancy and playback utility. The key

idea behind BOLA is that the higher the buffer level, the more liberty the algorithm would have to download higher bitrates. In this work, we adapt BOLA and BOLA360 to cater to volumetric videos. Specifically, the algorithm selects the quality level m for each object j in the segment k by solving the single-slot maximization problem:

$$\arg\max_{a(k)} \rho(k, a(k)) = \sum_{j=1}^{J} \sum_{m=1}^{M} \frac{a_{j,m}(k)(V v_m + V\gamma\delta - Q(t_k)/\delta)}{S_m}$$

$$\text{subject to: } \sum_{m=1}^{M} a_{j,m}(k) \leq 1 \quad \forall k, j, \quad a_{j,m}(k) \in 0, 1,$$

where ρ is the value to be maximized, t_k is the download time for segment k, $a(k)$ is the decision vector indicating the quality levels selected for all objects, $a_{j,m}(k)$ is the binary decision variable where 1 indicates that quality level m is selected for object j (and 0 otherwise), δ is the segment duration, $Q(t_k)$ is the buffer level at time t_k, v_m is the utility value of the segment with quality level m given as $log(2S_m/S_1)$ [24], S_m is the size of the segment with quality level m, V is an input parameter that tunes the trade-off between algorithm performance and buffer utilization (higher values suit on-demand streaming while lower values suit real-time streaming), γ is an input parameter that tunes the importance given to high bitrates (lower values) relative to playback smoothness (higher values).

4.2 Viewport Prediction Algorithms

Next, we discuss two alternative strategies implemented for the viewport prediction algorithm. The inputs to the regression models discussed below are the camera position and rotation samples recorded at each frame playback and tagged with a timestamp. The models are then used to predict the camera position and rotation given a future timestamp. Viewport prediction is crucial in the streaming of immersive formats due to the presence of a buffer, which implies that the segments that are to be retrieved next will not be played back until buffer = x seconds later, and the viewport then may not be the same viewport that the user is viewing right now. Effective viewport prediction algorithms would allow us to accurately predict the viewport that the user is likely to be viewing at x seconds later so that the adaptation algorithm can select a more appropriate quality level for each object (e.g., based on the object visibility and distance to the predicted viewport), thereby improving bandwidth efficiency and user experience.

4.2.1 Linear Predictor. Linear predictor [6] derives a simple linear regression model on the last two samples collected. The model is then used to predict the viewport into the future. The model is derived using the least squares method, in which we solve for the intercept and slope for each of the positional and rotational axes.

4.2.2 Weighted Linear Regression. Weighted linear regression, as proposed in [17] for 360° videos and adapted for volumetric videos in this work, is an extension of linear regression where each data point is assigned a weight, which influences its contribution to the final model. This approach is particularly useful when we wish to consider more samples yet allow recent samples to have a greater influence on the regression line which fits the viewport prediction problem [17]. We use the weighted least squares method in our implementation.

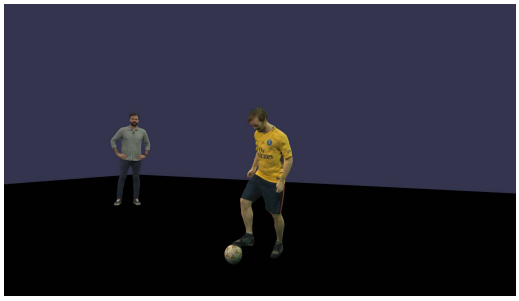


Figure 2: The 6DoF scene in the experiment setup. The camera pans across the scene while the volumetric video plays. The user can also initiate movement in the scene.

5 Performance Evaluation

5.1 Experimental Setup

For this study, we use pre-encoded textured meshes from the vsenseVVDB2 dataset [23]. The geometry data was encoded using the Draco [7] mesh compression format, while the texture data was compressed using JPEG. We chose the Draco mesh format due to the computational efficiency of its web-based decoder that allows the volumetric video to stream at 24 fps in the highest quality, on a modern laptop, which is regarded as one of the standard frame rates for film [3]. The setup of the volumetric video scene in our experiments is illustrated in Figure 2, where two mesh objects, Matis and Rafa, are placed a distance apart in the scene, with Matis being closer to the camera representing the user's viewport. Each video contains 600 frames, where each frame translates to one segment, and is streamed at 24 fps, thereby giving a video duration of 25 seconds. A total of four different quality levels are generated for each mesh object. The average bitrates for quality level 0–3 of each object are – Matis: { 42.5 Mbps, 60.1 Mbps, 91.2 Mbps, 120.9 Mbps }; Rafa: { 42.0 Mbps, 57.7 Mbps, 85.0 Mbps, 111.4 Mbps }. Other configuration parameters include: buffer size = 2s, $V = 200$ and $\gamma = 0.9$ for BOLA, $n = 24$ for weighted linear regression and, unless otherwise stated, the default HTTP version is HTTP/2. Parameters are determined empirically.

To simulate real-world network conditions, we incorporate two 5G network traces from [14], 5G Lumous (Driving) and 5G Lumous (Walking), which exhibit different characteristics. 5G Lumous (Driving) is characterized by high variability and steep fluctuations while 5G Lumous (Walking) is more consistent with lower average and maximum bandwidth values. Their bitrate statistics are – 5G Lumous (Driving): { average: 136.4 Mbps, min: 0.5 Mbps, max: 408.0 Mbps }; 5G Lumous (Walking): { average: 109.3 Mbps, min: 86.0 Mbps, max: 124.0 Mbps }. The traces are incorporated using the bandwidth throttling feature provided by Chrome DevTools. The player estimates the available bandwidth using a moving average of the segment download speeds given by (*download_size/download_time*). The evaluation assessed QoE through several key metrics including average selected bitrate, average perceptual quality (using SSIM [22] as the experiments are conducted on a 2D display), rebuffering duration and count, and playback smoothness (given by frequency of quality switches as a percentage of the playback session). These metrics can be extended as needed.

Network Profile	ABR	Avg. Selected Bitrate (Mbps)	Avg. Perceptual Quality (SSIM)	Avg. Rebuffering Duration (s)	Avg. Rebuffering Count (#)	Avg. Switches (% of session)
Lumous 5G (Driving)	**Uniform**	61.16 ±1.51	0.972 ±0.002	4.73 ±0.70	2.00 ±0.00	14.46 ±1.41
	Greedy	59.94 ±1.40	0.970 ±0.002	3.91 ±0.44	2.00 ±0.00	18.76 ±1.10
	BOLA	59.60 ±0.42	0.986 ±0.000	6.18 ±0.17	2.20 ±0.45	0.45 ±0.07
Lumous 5G (Walking)	**Uniform**	57.22 ±1.28	0.983 ±0.002	0.00 ±0.00	0.00 ±0.00	7.80 ±1.15
	Greedy	55.72 ±1.01	0.978 ±0.002	0.00 ±0.00	0.00 ±0.00	15.73 ±2.31
	BOLA	57.04 ±0.08	0.986 ±0.000	0.00 ±0.00	0.00 ±0.00	0.43 ±0.15

Table 2: Performance metrics derived in investigating the impact of network conditions on adaptation algorithms.

The software environment consists of Nginx (v1.25.4) as a web media server, configured to support HTTP/1.1, HTTP/2, and HTTP/3 protocols while storing the media files of our encoded mesh objects. The player is implemented using Typescript (v5.2.2) and Babylon.js (v6.34.3) with its graphics library for rendering and built-in Draco mesh decoder. The experiments were automated using Puppeteer (v22.8.0) on Chrome browser (v131.0.6778.86), and each experiment was run five times with their average results reported. The system that ran the experiments is a MacBook Pro that includes an Apple M2 Max processor with 12 cores and 32 GB of RAM.

5.2 Results and Discussion

We conducted this investigation across three dimensions. First, we evaluated the performance of ABR algorithms under varying network conditions, examining whether their rate-based or buffer-based design influences their effectiveness. Second, we studied the behavior of viewport prediction algorithms with user viewport movements. Third, we investigated the impact of different HTTP variants, including HTTP1/TCP, HTTP2/TCP, and the newly introduced HTTP3/QUIC, on overall system performance. We highlight key insights, recommendations, and opportunities for future work.

5.2.1 Impact of Network Condition on Adaptation Algorithms. Table 2 shows the results of how each of the three ABR algorithms performed under both network profiles. We observe that one distinguishing trade-off between rate-based and buffer-based adaptation is between the extent of rebuffering and the extent of switches during the playback sessions. Specifically, in the Lumous 5G (Driving) profile, Uniform and Greedy (rate-based) achieved lower average rebuffering duration of 24% – 37% reduction as compared to BOLA (buffer-based), but at a cost of higher switch frequencies (expressed as a percentage of the session), which are given by 14.46% (Uniform), 18.76% (Greedy), and 0.45% (BOLA). This behavior is also reflected in the Lumous 5G (Walking) profile, where the average switches are reduced from 7.80% (Uniform) and 15.73% (Greedy) to 0.43% (BOLA), even when no rebuffering occurs. The adverse impact of the switches is further demonstrated in Figure 3 where the perceptual quality ranges for the rate-based and buffer-based adaptation algorithms differ significantly. Note that the reference frames in the SSIM computation are derived by selecting the highest quality level available for the objects in the frame. The reason for this behavior is that, for rate-based algorithms, as they are modeled after changes in the bandwidth, network fluctuations would cause the algorithms to react immediately. The benefit of this is reduced rebuffering as the ABR reacts quickly to sudden network dips and requests for lower quality levels which would avert or at least reduce the extent of rebuffering. Conversely, for buffer-based algorithms, network fluctuations would have an indirect and smoothened impact on the algorithm's behavior but may also result in a slower response to network dips as downshifting happens in response to the buffer instead of the network.

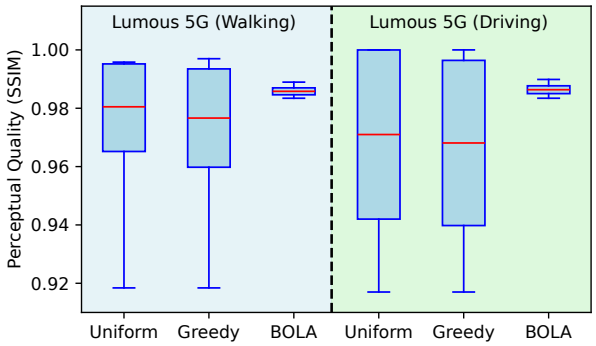
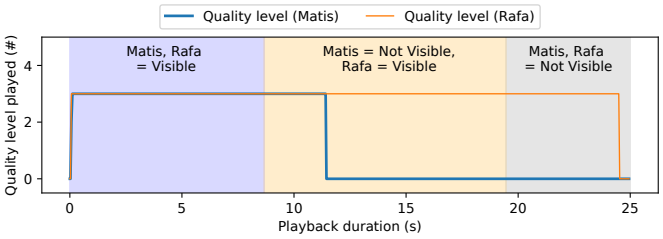
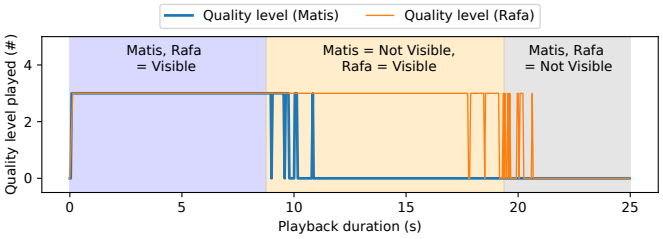


Figure 3: Range of SSIM values attained in sample runs for each ABR and network profile.

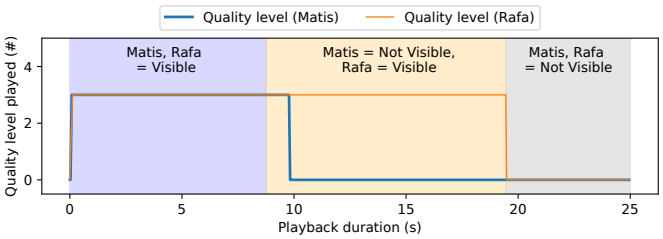
5.2.2 Impact of Viewport Changes on Viewport Prediction Algorithms. In this experiment, we investigate how viewport changes influence the outcomes of applying different viewport prediction algorithms. As introduced in Section 5.1, this set of experiments applies a leftward camera movement to the scene while the volumetric video is playing. The playback outcomes for the control setup (no viewport prediction) and the two viewport prediction algorithms (linear predictor and weighted linear regression) are visualized in Figure 4. In this setup, the scene begins with both objects visible (the purple region in the figure), and as the camera pans left, Matis disappears from the scene at about the 8.5s-mark (entering the orange region). Finally, as the camera pans further left, Rafa also disappears from the scene at about the 19s-mark (entering the grey region). Recall that when an object is deemed not visible in the predicted viewport, its lowest quality level will be requested. With no viewport prediction (Figure 4a), we observe that although Matis disappeared from the scene at the 8.5s-mark, the quality level for Matis only drops about 3s later at the 11.5s-mark. This is because the video segment at the 11.5s-mark was downloaded at the 8.5s-mark (i.e., the buffer contained 3s worth of video data). Hence, there is a delay in reflecting the viewport change on the selected



(a) No viewport prediction.



(b) Linear predictor model for viewport prediction.



(c) Weighted linear regression model for viewport prediction.

Figure 4: Sample runs with and without viewport prediction.

quality levels. With viewport prediction, however, we are able to bring that delay down. As seen in Figure 4b and Figure 4c, the drop in Matis' quality level comes earlier than the 11.5s-mark observed in Figure 4a. Of the two algorithms, linear predictor seems to be more responsive here as the noise may at times cause the linear predictor to overestimate the camera movement. This is also evident when Rafa disappears, where the drop in Rafa's quality level comes earlier at the 18s-mark than when it actually happens at the 19.5s-mark. Similarly, the noise may also at times cause the linear predictor to underestimate the camera movement which explains the occurrences of increasing the quality level after it has dropped (i.e., leading to the up-and-down fluctuations in the quality levels). Hence, the responsiveness may not be a good thing here. Overall, weighted linear regression should be a recommended approach over linear predictor. To further quantify the benefits of applying weighted linear regression against having no viewport prediction, results from our experiments show that by predicting the disappearance of Matis and Rafa objects earlier in the timeline, there was a bandwidth saving of 9.55% (with Uniform ABR) while achieving the same perceptual quality. In this regard, more comprehensive investigations can be done in the future to study the accuracy of viewport prediction algorithms using real-world viewport movement traces collected from users on the WIDE-VR system.

5.2.3 Varying the HTTP Version. To investigate the potential impact of different HTTP versions on the system, we repeated the experimental setup described in Section 5.2.1 (which used HTTP/2) on HTTP/1.1 and HTTP/3 versions. The HTTP version is configured in the Nginx server. The results consistently revealed the same trends regarding the influence of network conditions on the three ABR algorithms. Across the three HTTP versions, Uniform and Greedy (rate-based) experience noticeably more switches than BOLA (buffer-based), while being able to reduce rebuffering duration in Lumous 5G (Driving) (the network profile with steep fluctuations) due to their responsiveness to the dips. This suggests that while the choice of HTTP version may affect specific protocol-level behaviors, the overarching adaptation trends remain robust in these bandwidth conditions. Studies [15, 20] have also suggested that using HTTP/3 simplistically at the application layer may not lead to performance improvements as it requires careful tuning of parameters like congestion control, packet prioritization, and taking advantage of new features such as with respect to multiplexing strategies, no head-of-line blocking, etc., to fully leverage its potential benefits over HTTP/2 or HTTP/1.1. In this regard, more comprehensive investigations can be done in the future to study how such enhancements can be leveraged in the WIDE-VR system.

6 Conclusions

This paper presents the design and implementation of a unique, operational, and extensible web-based 6DoF streaming system that comprehensively addresses multiple essential aspects of volumetric video delivery. By integrating DASH-based adaptive streaming and viewport prediction techniques with modern web technologies, WIDE-VR sets a new benchmark for delivering immersive 6DoF experiences seamlessly through the browser, ensuring enhanced cross-platform compatibility. This system bridges critical gaps in the deployment of bandwidth- and computation-intensive VR applications. Through our range of evaluations, we demonstrate the effectiveness of these techniques in enhancing key performance metrics, including bandwidth efficiency, rebuffering duration, and quality stability. The insights gained underline the practicality and scalability of deploying high-quality VR experiences via the web, overcoming traditional limitations of 6DoF streaming.

The WIDE-VR system also opens avenues for future research, including areas such as refining adaptive streaming and viewport prediction algorithms using machine learning to further improve user experiences. Additionally, exploring advanced features of emerging HTTP standards, such as QUIC in HTTP/3, offers opportunities to tackle latency and scalability challenges. Conducting user studies with the implemented system will also be instrumental in evaluating subjective experience metrics and collecting viewport movement data, providing deeper insights into optimizing web-based VR for diverse use cases and audiences.

Acknowledgments

This research is supported by Singapore Ministry of Education Academic Research Fund Tier 2 under MOE's official grant number T2EP20221-0023. The authors thank Nazanin Azarian for her assistance with the implementation of this work.

References

[1] Abdelhak Bentaleb, May Lim, Sarra Hammoudi, Saad Harous, and Roger Zimmermann. 2024. Solutions, Challenges and Opportunities in Volumetric Video Streaming: An Architectural Perspective. *ACM Transactions on Multimedia Computing, Communications, and Applications* (Nov. 2024). https://doi.org/10.1145/3705321

[2] Abdelhak Bentaleb, Bayan Taani, Ali C Begen, Christian Timmerer, and Roger Zimmermann. 2018. A survey on bitrate adaptation schemes for streaming media over HTTP. *IEEE Communications Surveys & Tutorials* 21, 1 (2018), 562–585.

[3] John Andrew Berton and Kai-Lin Chuang. 2016. Effects of very high frame rate display in narrative CGI animation. In *2016 20th International Conference Information Visualisation (IV)*. IEEE, IEEE, 395–398.

[4] Sergio Cabrero Barros, Ander Elosegi, Iñigo Tamayo, Ana Dominguez, and Mikel Joseba Zorrilla. 2024. Volumetric Video on the Web: a platform prototype and empirical study. In *Proceedings of the 29th International ACM Conference on 3D Web Technology*. ACM, 1–10.

[5] Matthias De Fré, Jeroen van der Hooft, Tim Wauters, and Filip De Turck. 2024. Scalable MDC-Based Volumetric Video Delivery for Real-Time One-to-Many WebRTC Conferencing. In *Proceedings of the 15th ACM Multimedia Systems Conference*. ACM, 121–131.

[6] Jean-Philippe Farrugia, Luc Billaud, and Guillaume Lavoué. 2023. Adaptive streaming of 3D content for web-based virtual reality: an open-source prototype including several metrics and strategies. In *Proceedings of the 14th Conference on ACM Multimedia Systems*. ACM, 430–436.

[7] Google. 2025. Google's Draco 3D Data Compression. [Online] Available: https://google.github.io/draco/. Accessed on Nov. 22, 2024.

[8] Simon NB Gunkel, Rick Hindriks, Karim M El Assal, Hans M Stokking, Sylvie Dijkstra-Soudarissanane, Frank ter Haar, and Omar Niamut. 2021. VRComm: an end-to-end web system for real-time photorealistic social VR communication. In *Proceedings of the 12th ACM multimedia systems conference*. ACM, 65–79.

[9] IDLab DISCOVER. 2025. Many-to-many WebRTC-based point cloud video conferencing. [Online] Available: https://github.com/idlab-discover/pc-webrtc-m2m. Accessed on Jan. 21, 2025.

[10] Yili Jin, Junhua Liu, Kaiyuan Hu, and Fangxin Wang. 2024. A networking perspective of volumetric video service: Architecture, opportunities and case study. *IEEE Network* (2024).

[11] Preetish Kakkar and Hariharan Ragothaman. 2024. The evolution of volumetric video: A survey of smart transcoding and compression approaches. *arXiv preprint arXiv:2411.02095* (2024).

[12] LIRIS XR Team. 2023. Adaptive streaming of 3D content for web-based virtual reality. [Online] Available: https://github.com/liris-xr/AdaptiveStreaming. Accessed on Jan. 21, 2025.

[13] Mike Bishop. 2022. HTTP/3. [Online] Available: https://datatracker.ietf.org/doc/rfc9114/. Accessed on Nov. 22, 2024.

[14] Arvind Narayanan, Xumiao Zhang, Ruiyang Zhu, Ahmad Hassan, Shuowei Jin, Xiao Zhu, Xiaoxuan Zhang, Denis Rybkin, Zhengxuan Yang, Zhuoqing Morley Mao, et al. 2021. A variegated look at 5G in the wild: performance, power, and QoE implications. In *Proceedings of the 2021 ACM SIGCOMM 2021 Conference*. ACM, 610–625.

[15] Ryuichi Niitsu and Saneyasu Yamaguchi. 2023. Performance Evaluation and Improvement of HTTP/3 Considering TLS. In *2023 Eleventh International Symposium on Computing and Networking Workshops (CANDARW)*. IEEE, IEEE, 368–370.

[16] NUStreaming. 2025. WIDE-VR. [Online] Available: https://github.com/NUStreaming/WIDE-VR. Accessed on Jan. 21, 2025.

[17] Feng Qian, Lusheng Ji, Bo Han, and Vijay Gopalakrishnan. 2016. Optimizing 360 video delivery over cellular networks. In *Proceedings of the 5th Workshop on All Things Cellular: Operations, Applications and Challenges*. 1–6.

[18] Xiuquan Qiao, Pei Ren, Schahram Dustdar, Ling Liu, Huadong Ma, and Junliang Chen. 2019. Web AR: A promising future for mobile augmented reality—State of the art, challenges, and insights. *Proc. IEEE* 107, 4 (2019), 651–666.

[19] Kevin Spiteri, Rahul Urgaonkar, and Ramesh K Sitaraman. 2020. BOLA: Near-optimal bitrate adaptation for online videos. *IEEE/ACM transactions on networking* 28, 4 (2020), 1698–1711.

[20] Sreeni Tellakula. 2020. Comparing HTTP/3 vs. HTTP/2 Performance. [Online] Available: https://blog.cloudflare.com/http-3-vs-http-2. Accessed on Nov. 22, 2024.

[21] Jeroen Van Der Hooft, Tim Wauters, Filip De Turck, Christian Timmerer, and Hermann Hellwagner. 2019. Towards 6dof http adaptive streaming through point cloud compression. In *Proceedings of the 27th ACM International Conference on Multimedia*. ACM, 2405–2413.

[22] Zhou Wang, Alan C Bovik, Hamid R Sheikh, and Eero P Simoncelli. 2004. Image quality assessment: from error visibility to structural similarity. *IEEE transactions on image processing* 13, 4 (2004), 600–612.

[23] Emin Zerman, Cagri Ozcinar, Pan Gao, and Aljosa Smolic. 2020. Textured mesh vs coloured point cloud: A subjective study for volumetric video compression. In *Twelfth International Conference on Quality of Multimedia Experience (QoMEX)*. IEEE.

[24] Ali Zeynali, Mohammad H Hajiesmaili, and Ramesh K Sitaraman. 2024. BOLA360: Near-optimal View and Bitrate Adaptation for 360-degree Video Streaming. In *Proceedings of the 15th ACM Multimedia Systems Conference*. ACM, 12–22.

HockeyAI: A Multi-Class Ice Hockey Dataset for Object Detection

Mehdi Houshmand Sarkhoosh
OsloMet & Forzasys
Oslo, Norway

Sushant Gautam
SimulaMet & OsloMet
Oslo, Norway

Cise Midoglu
Forzasys
Oslo, Norway

Saeed Shafiee Sabet
Forzasys
Oslo, Norway

Tomas Kupka
Forzasys
Oslo, Norway

Pål Halvorsen
SimulaMet, OsloMet & Forzasys
Oslo, Norway

Abstract

The fast paced nature of ice hockey presents unique challenges for object detection, particularly in tracking the puck—a small, fast moving object that is critical to gameplay analysis. This paper introduces HockeyAI, a novel open source dataset specifically designed for multi-class object detection in ice hockey. The dataset includes 2,101 high resolution frames extracted from professional games in the Swedish Hockey League (SHL), annotated in the You Look Only Once (YOLO) format. Annotations span 7 classes, covering dynamic objects such as the players and the puck, as well as static rink elements such as goalposts and face-off circles. The dataset is derived from diverse SHL games across multiple seasons and teams, ensuring a rich variety of scenarios reflective of real world gameplay. A fine tuned YOLOv8 medium model is also provided, demonstrating high performance across all classes. Key comparisons highlight the dataset's advancements over existing resources, addressing their limitations such as incomplete class coverage, low resolution, and inconsistent annotations. The dataset, model, and an interactive demo are publicly available on Hugging Face under an open source license, fostering further collaboration in sports related computer vision applications (https://huggingface.co/SimulaMet-HOST/HockeyAI).

CCS Concepts

• **Information systems**; • **Computing methodologies** → **Machine translation**; **Computer vision**; Scene understanding; **Video summarization**; *Activity recognition and understanding*; **Visual inspection**; • **Human-centered computing** → *Visualization*;

Keywords

AI, computer vision, ice hockey, object detection, gameplay analysis, smart cropping, summarization, tracking

ACM Reference Format:
Mehdi Houshmand Sarkhoosh, Sushant Gautam, Cise Midoglu, Saeed Shafiee Sabet, Tomas Kupka, and Pål Halvorsen. 2025. HockeyAI: A Multi-Class Ice Hockey Dataset for Object Detection. In *ACM Multimedia Systems Conference 2025 (MMSys '25), March 31-April 4, 2025, Stellenbosch, South Africa.* ACM, New York, NY, USA, 7 pages. https://doi.org/10.1145/3712676.3718335

Permission to make digital or hard copies of all or part of this work for personal or classroom use is granted without fee provided that copies are not made or distributed for profit or commercial advantage and that copies bear this notice and the full citation on the first page. Copyrights for components of this work owned by others than the author(s) must be honored. Abstracting with credit is permitted. To copy otherwise, or republish, to post on servers or to redistribute to lists, requires prior specific permission and/or a fee. Request permissions from permissions@acm.org.
MMSys '25, March 31-April 4, 2025, Stellenbosch, South Africa
© 2025 Copyright held by the owner/author(s). Publication rights licensed to ACM.
ACM ISBN 979-8-4007-1467-2/2025/03
https://doi.org/10.1145/3712676.3718335

1 Introduction

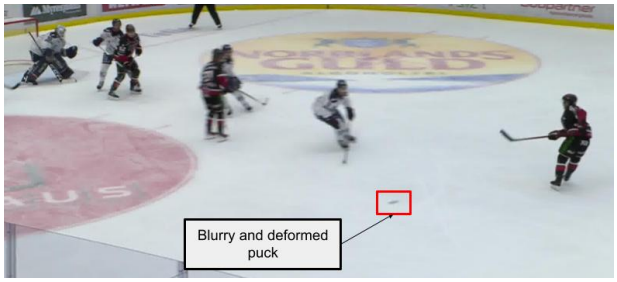
Ice hockey is a high speed sport characterized by rapid player movements, frequent occlusions, varying jersey designs, diverse backgrounds and ambient atmospheres, and specific action centered areas of the rink, such as face-off circles. Combined with the small size and high velocity of the puck, these factors make ice hockey one of the most challenging domains for computer vision applications [13, 23]. Accurate object detection and tracking in ice hockey are crucial for applications such as player analysis, game strategy evaluation, and fan engagement [19, 28, 36]. Furthermore, generating highlights and summarizations of different events in ice hockey—whether as videos or textual descriptions—requires a robust understanding of the game's objects and context [16, 24, 25]. However, the fast paced nature of the game, coupled with motion blur and rapid camera movements, poses significant challenges for traditional computer vision techniques [27].

Despite growing interest in sports analytics, publicly available datasets tailored to ice hockey remain limited in both scale and scope. Existing datasets often lack comprehensive class annotations, consistent puck tracking in challenging scenarios such as motion blurred frames, or high resolution imagery. Additionally, they are often collected from cameras other than the broadcast cameras, which may limit their applicability in diverse settings. These gaps hinder the development of robust models for ice hockey analysis, as well as applications such as event detection [20], game summarization [7], and highlight generation [34]. To address these limitations, we present HockeyAI, a novel open source dataset specifically designed for multi-class object detection in ice hockey.

The HockeyAI dataset consists of 2,101 high quality frames, each with a resolution of 1920×1080, extracted from SHL [12] games. The frames are annotated using Label Studio [32] in YOLO [22] format across 7 diverse classes, which include not only moving objects within the rink but also key static elements of the rink itself. The dataset spans the 2022, 2023, and 2024 seasons and various teams, emphasizing wide angle shots to effectively capture the entire rink. A key advancement of the dataset is its facilitation of puck tracking through sequential frame analysis, which compensates for blur (see Figure 1) in individual frames by leveraging up to 10 consecutive frames. Overall, the dataset demonstrates strong object detection accuracy for all labeled categories.

Alongside the HockeyAI dataset, we offer a fine tuned YOLOv8 model specifically trained on this dataset to address the distinctive challenges present in ice hockey. Our model is publicly available on Hugging Face [8].

The contributions of this work are as follows:



(a) Blurry and deformed puck



(b) Normal and vivid puck

Figure 1: Comparison of puck visualizations in ice hockey: (a) deformed due to motion blur (fast camera movement) vs. (b) clear under normal conditions.

(1) HockeyAI, an open source dataset with high quality annotations for 7 key object classes in ice hockey.
(2) A comparison of HockeyAI with existing datasets, highlighting its advantages in class coverage, image resolution, and consistent puck annotations.
(3) A fine tuned YOLOv8 medium model for object detection, achieving high performance on the HockeyAI dataset.
(4) A publicly available Hugging Face Space that enables real-time inference on HockeyAI and serves as an interactive demo.

By curating a novel ice hockey dataset, we aim to foster further innovation in sports related computer vision research and practical applications. The HockeyAI dataset and the associated YOLOv8 model are hosted on Hugging Face under an open source license, ensuring accessibility and encouraging their use across diverse research and industrial domains.

2 Background and Related Work

Ice hockey is played on a standardized rink with distinct zones, each serving a critical role in the game. The rink measures approximately 60 meters in length and 30 meters in width, divided into three primary zones: the defensive zone, the neutral zone, and the offensive zone. Key structural elements include the center ice circle, which is used for face-offs to restart play; the goal crease, protecting the goaltender's space; face-off circles and dots, which facilitate controlled play restarts; and the blue lines, which delineate zone boundaries and determine offside violations (see Figure 2). These features are strategically significant, frequently influencing gameplay dynamics, making their precise detection essential for

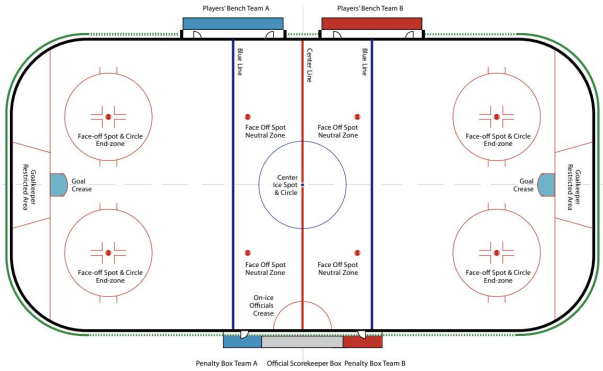


Figure 2: Illustration of a standard hockey rink with key areas (e.g., face-off circles, goalposts, center ice circle) [10].

advanced performance analysis. Additionally, the reflective ice surface and dynamic lighting conditions introduce complexities in object detection tasks, further challenging automated analysis.

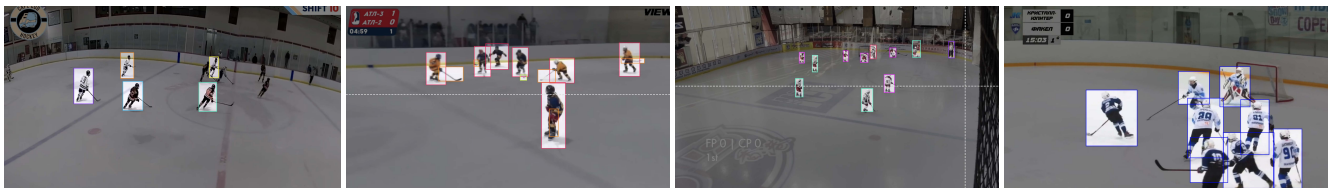
2.1 Object Detection in Ice Hockey

The field of object detection and tracking in sports has seen rapid advancements, largely driven by deep learning innovations and the increasing demand for automated analytics in sports broadcasting, training, and fan engagement. Ice hockey, however, presents unique challenges for computer vision due to its high speed gameplay, frequent occlusions, and the small size of the puck, which often moves at extreme velocities. While significant progress has been made in sports such as soccer and basketball—where publicly available datasets and domain specific models have accelerated research [6, 25, 26]—ice hockey remains comparatively underrepresented. The limited availability of high quality datasets tailored to its specific challenges highlights the need for targeted research efforts to enhance automated analytics in this domain.

2.2 Ice Hockey Datasets

Existing ice hockey datasets [21, 30, 31, 39] often lack comprehensive annotations for key objects and areas of interest. For example, the puck, arguably the most critical object for tracking in ice hockey, is frequently omitted or inconsistently annotated due to its small size and rapid movement (see Figure 3). Moreover, these datasets rarely include annotations for rink specific elements such as face-off circles or goalposts, despite their importance in gameplay analysis. Many datasets are also constrained to low resolution imagery, limiting their utility for developing robust models capable of detecting fine grained details. Additionally, the lack of diversity in camera angles and contexts, as most datasets do not rely on broadcast footage, further hinders the generalizability of models trained on existing datasets (see Table 1).

Several efforts have been made to address object detection challenges in sports through the development of specialized models and datasets. For example, datasets such as SoccerNet [26] and SpaceJam [6] have provided annotated images and videos for soccer and basketball [9], respectively, enabling advancements in player tracking, event detection, and strategy analysis. However, ice hockey



(a) Inconsistent player annotations (b) Missing puck annotations (c) Low resolution imagery (d) Poor coverage of rink elements

Figure 3: Examples from 4 hockey datasets, illustrating common challenges.

Table 1: Comparison of HockeyAI with existing hockey datasets, in terms of scale (number of frames), scope (number of classes), quality (image resolution), availability of annotations for puck and rink elements, annotation consistency (uniform labeling and precise bounding boxes), and dataset source.

Dataset Names	# Frames	# Classes	Resolution	Puck	Rink Elements	Consistency	Source
McGill Hockey Player Tracking Dataset (MHPTD) [41]	82,305	3	1280×720	✗	✗	✓	NHL Broadcast
Player tracking dataset [37]	91,807	2	1280×720	✗	✗	✓	NHL Broadcast
Ice Hockey Computer Vision Project [4]	3,155	13	640×640	◑	◑	✗	Unknown
Hockey Computer Vision Project [18]	1,583	14	640×640	◑	◑	✗	Unknown
Hockey8 Computer Vision Project [33]	1,448	10	1280×1280	◑	◑	✗	Unknown
Hockey FullIce Computer Vision Project [11]	1,361	3	640×640	◑	✗	✗	Unknown
Hockey Computer Vision Project [5]	1,959	1	416×416	✓	✗	✗	Unknown
HockeyAI	2,100	7	1920×1080	✓	✓	✓	SHL Broadcast

introduces additional complexities, such as the presence of blurred frames caused by rapid player and puck movements, reflective surfaces, and frequent camera panning. These factors necessitate datasets specifically designed for ice hockey, incorporating annotations that reflect the nuances of the sport and facilitate the development of reliable detection and tracking models.

The recent emergence of YOLO based models has further advanced the field of object detection. Models such as YOLOv8 provide state of the art performance across a variety of domains, offering a strong foundation for tackling the unique challenges of ice hockey. However, the success of these models depends heavily on the quality and relevance of the datasets used for training and fine tuning. By addressing the shortcomings of existing datasets, HockeyAI aims to provide a robust resource for advancing computer vision research in ice hockey, focusing on comprehensive annotations, high resolution imagery, and diverse scenarios reflective of real world gameplay.

3 The HockeyAI Dataset

The HockeyAI dataset was curated by extracting and annotating high resolution frames from broadcast videos of SHL games. This collection includes 2,101 frames, each with a resolution of 1920×1080 pixels, selected to represent diverse scenarios in ice hockey. Key features of HockeyAI include:

- **Diverse gameplay scenarios:** The selected frames cover key moments such as face-offs, goals, and referee interventions, providing a comprehensive view of game dynamics.
- **Wide angle shots:** Most frames capture the entire rink, ensuring the visibility of multiple object classes and spatial interactions.
- **Sequential frames:** Up to 10 consecutive frames are included, enabling robust tracking of fast moving objects like the puck.
- **Class diversity:** Annotations cover both dynamic objects (e.g., players, puck) and static rink elements (e.g., goalposts, center ice).
- **Diverse jersey colors:** The dataset includes frames featuring a wide range of jersey colors, capturing variability across teams and under different lighting and environmental conditions.

3.1 Sources and Collection

Partnering with SHL and Sportality [29], we utilized Forzasys APIs [1] to access M3U8 URLs for SHL matches from the 2022–23, 2023–24, and 2024–25 seasons. Permission from Sportality was obtained to create the dataset, which is made open-source provided that full videos are not shared and only non-sequential frames that do not reconstruct the original video are publicly available. The original videos were encoded in H.264 format with a resolution of 1920×1080 pixels and a frame rate of 50 fps. Frames were stored as JPEG files with a quality setting of 90 to preserve high fidelity.

Our data collection methodology involved selecting random games across the specified seasons to ensure representation of diverse conditions, jersey colors, and gameplay scenarios. For each selected game, we extracted 10 minute segments and sampled frames at regular intervals. Specifically, one frame was extracted at the midpoint of each second, followed by additional frames at consistent intervals (e.g., every 500 frames) to ensure adequate coverage of the segment. This approach minimized redundancy while capturing a variety of transitions, wide shots, and background frames. All

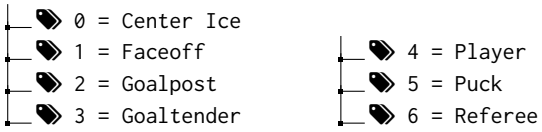


Figure 4: Class index mapping for HockeyAI.

extracted frames underwent manual inspection to ensure that the puck was visible in each selected frame.

To address the challenges of detecting the puck in fast paced scenarios involving motion blur, we employed an additional targeted methodology. Randomly selected games from the dataset were processed to identify specific one minute segments where motion blur or deformation of the puck was evident due to rapid camera movements. From these segments, 10 consecutive frames were extracted to provide robust training data for scenarios where traditional detection methods might fail. The restriction to 10 consecutive frames was implemented to prevent the reconstruction of original video footage, adhering to licensing agreements. These blur focused sequences were selected across different games and seasons to ensure diversity.

By combining these two methodologies, we curated a dataset encompassing a wide array of frames, each representing distinct scenarios, conditions, and gameplay dynamics observed in professional ice hockey games.

3.2 Annotations

HockeyAI includes annotations for 7 key object classes in YOLO format. The annotated classes and their corresponding identifiers are presented in Figure 4. The annotations were generated using Label Studio [32] and underwent multiple review cycles to guarantee precision and uniformity. Each bounding box received a manual inspection, with special focus on the puck. Altogether, our dataset includes 24,277 unique annotations across all classes, with an average of 11.55 annotations per image, ranging from a minimum of 2 to a maximum of 23. This diversity, illustrated in Figure 5 as a scatter plot of bounding box dimensions, highlights the variety of scenarios represented in HockeyAI, covering both wide and close-up shots. The class distribution of annotations is illustrated in Figure 6.

3.3 Public Dataset

The HockeyAI dataset is hosted on Hugging Face, and serves as a collection of annotated frames where each frame in the dataset is paired with a single annotation file in YOLO format, specifying the bounding boxes for all detected objects. The dataset is organized in a hierarchical manner, beginning with the SHL league, and is then divided into two main directories: "frames" and "annotations". Within these directories, files are organized by their respective unique IDs, ensuring easy navigation and efficient use (see Figure 7).

Coordinate system: The coordinates `(x,y)` used in the annotation files are normalized relative to the dimensions of the frame. This normalization follows the format:

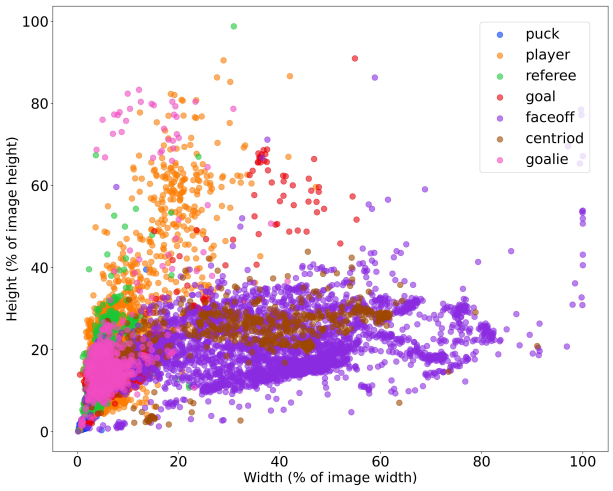


Figure 5: Bounding box dimensions in HockeyAI, with colors indicating different object classes.

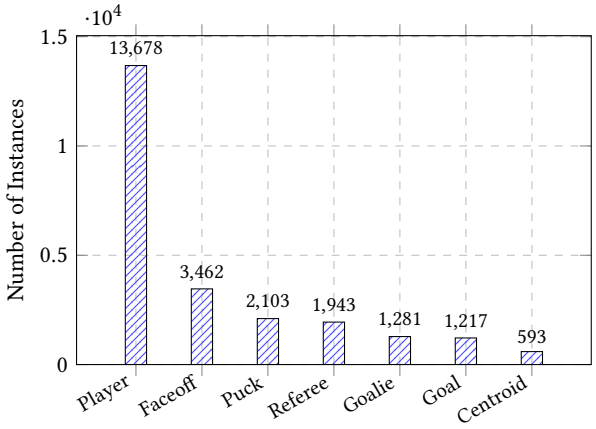


Figure 6: Class distribution in HockeyAI. `Player` has the highest number of annotations, `Centroid` has the lowest.

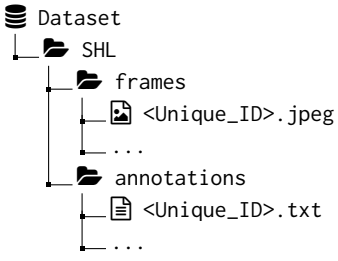


Figure 7: HockeyAI directory structure.

$$x = \frac{\text{absolute_x}}{\text{image_width}} \qquad y = \frac{\text{absolute_y}}{\text{image_height}}$$

where `absolute_x` and `absolute_y` are the pixel coordinates of the object in the image, and `image_width` and `image_height` represent the dimensions of the image.

Sample line in the annotation file: In YOLO's object detection format, each line in the annotation file corresponds to one detected object in the associated frame. The format for each line is as follows:

```
<class_id> <x> <y> <w> <h>
```

where:

- **`class_id`** is an integer representing the object's class (e.g., player, referee, puck).
- **`x,y`** are the normalized coordinates of the bounding box center.
- **`w,h`** are the normalized dimensions (width and height) of the bounding box.

3.4 Novelty

The HockeyAI dataset contributes to ice hockey computer vision research by addressing the following challenges in the field:

- **Comprehensive annotations:** Labels for dynamic objects (e.g., players, puck) and static rink elements (e.g., goalposts, center ice) support detailed gameplay analysis.
- **Sequential frames:** Inclusion of consecutive frames enhances tracking capabilities, particularly for fast moving objects under challenging conditions such as motion blur.
- **High resolution imagery:** Frames with a resolution of 1920×1080 enable precise object detection and analysis.
- **Diverse scenarios:** Coverage of multiple seasons, teams, and gameplay events ensures versatility and contextual richness.

3.5 Model Performance

We present an optimized YOLOv8 medium model that has been fine-tuned on the HockeyAI dataset, serving as a baseline. This model is specifically configured for identifying the seven dataset classes: player, goaltender, referee, puck, goalpost, center ice, and face-off. To assess its effectiveness, we performed evaluations on a separate test set of 106 frames, reporting metrics including mean average precision (mAP), precision, and recall.

Figures 8 and 9 present the confusion matrix and precision-recall curve, respectively, illustrating the model's accuracy in detecting small objects such as the puck, and resolving challenging scenarios involving motion blur and occlusions. Figure 10 further illustrates the effectiveness of the model by presenting a sample model output, highlighting the detection of players, puck, and rink elements in a high speed gameplay scenario.

4 Dataset Applications

The HockeyAI dataset is a versatile resource for a wide range of applications in ice hockey analytics and broader domains. Its high-resolution imagery, comprehensive annotations, and diverse scenarios empower researchers and practitioners to address challenges in sports-related computer vision tasks. Below, we group the key use cases into five major categories.

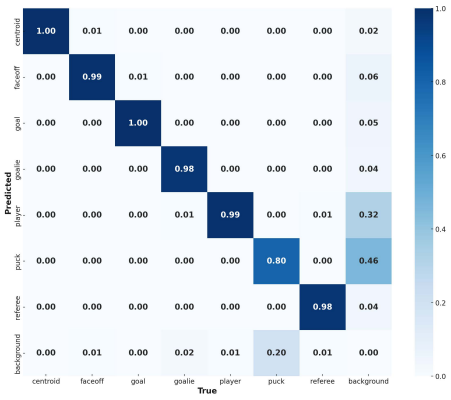


Figure 8: Confusion matrix for the baseline model.

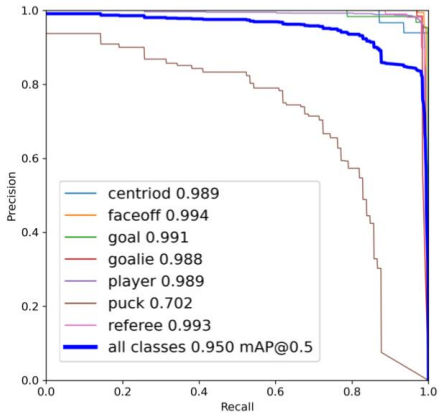


Figure 9: Precision-recall curve for the baseline model.

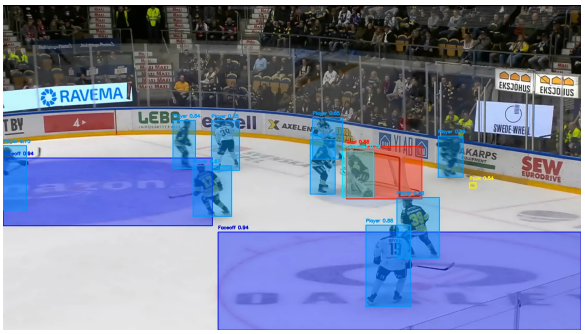


Figure 10: Sample output from the baseline model.

4.1 Object Detection and Tracking

HockeyAI serves as a benchmark dataset for object detection in sports, particularly in challenging conditions like motion blur and rapid object movement. Its structure and methodology inspire similar datasets for other sports, fostering advancements in general-purpose object detection models [3, 38].

HockeyAI enables advanced tracking of players and the puck, addressing challenges such as motion blur, occlusions, and rapid

movements. Accurate tracking is critical for tactical analysis, player performance evaluation, and identifying key game events [35].

4.2 Gameplay Analysis

The dataset also supports automated gameplay analysis by modeling spatial interactions between players, the puck, and rink elements. This enables applications such as evaluating team formations, identifying offensive and defensive patterns, and generating heatmaps for player activity [14]. These capabilities are particularly useful for coaches, analysts, and broadcasters aiming to optimize strategies and enhance game coverage.

4.3 Event Detection and Video Summarization

The dataset facilitates the detection of gameplay events like goals, penalties, face-offs, and referee interventions. Such event detection can be used to automate the generation of highlight reels, summarize key moments, and assist broadcasters with event-driven storytelling [7, 15].

4.4 Content Adaptation and Personalization

HockeyAI supports innovative applications for adapting and personalizing hockey content.

Smart cropping: Algorithms can use the dataset to develop smart cropping techniques for reframing videos to fit different aspect ratios, ensuring key elements like players and the puck remain in focus under different viewing scenarios [23]. This is particularly beneficial for enhancing viewing experiences on devices with varying screen orientations.

Personalization: The dataset enables the creation of customized highlight reels and game summaries tailored to individual players, teams, or fan preferences [27]. By detecting and tracking specific elements, algorithms can generate content that meets the unique needs of users [24, 25].

4.5 Broader Applications

The dataset's insights and techniques extend beyond ice hockey to other high-speed environments, such as automotive applications (e.g., tracking objects on highways), surveillance systems, and fast-paced sports like basketball and tennis [17, 40]. Additionally, HockeyAI supports the development of smart stadium and broadcast solutions, such as automated camera systems that track the puck and players in real-time, ensuring key moments are always captured and framed correctly [2]. These innovations have the potential to enhance both live broadcasts and fan engagement tools, such as augmented reality (AR) overlays and in-depth analysis platforms.

Through its various potential use cases, HockeyAI establishes itself as a valuable resource for broader computer vision research and industrial applications.

5 Discussion

The HockeyAI dataset fills a gap in ice hockey computer vision research by addressing the unique challenges posed by the sport, such as the puck's rapid movement, motion blur, and frequent occlusions. Its high-resolution imagery, diverse scenarios, and sequential frame support provide a robust foundation for developing models capable of handling these complexities. Unlike existing datasets, HockeyAI incorporates both dynamic objects and static rink elements, enabling holistic gameplay analysis.

One of the key strengths of the dataset lies in its ability to support applications beyond simple object detection. For example, the inclusion of sequential frames enhances the accuracy of tracking models, while detailed annotations enable advanced event detection and strategic gameplay insights. Moreover, the dataset's adaptability to tasks such as smart cropping and personalized content generation highlights its versatility in real-world applications.

However, certain limitations should be considered. One notable constraint is that all frames in the dataset contain the puck, which limits the model's ability to learn from negative examples where the puck is absent or occluded. This could impact the generalization of trained models, particularly in scenarios where the puck is hidden behind players or temporarily out of the camera's field of view. Future dataset expansions could address this by incorporating frames without the puck or augmenting the dataset with synthetic occlusions to improve robustness in real-world gameplay conditions. Additionally, the reliance on broadcast footage and the dataset's size relative to other large-scale datasets point to opportunities for further expansion. Looking forward, the methodologies employed in creating HockeyAI can serve as a blueprint for similar datasets in other sports, particularly those requiring fast object detection and tracking. Integrating advanced features such as temporal annotations or 3D data could further enhance its utility. Collaboration with professional leagues and technology providers, as demonstrated in this project, is crucial for maintaining the relevance and applicability of sports datasets.

6 Conclusion

This paper introduced HockeyAI, a novel open source dataset for multi-class object detection in ice hockey. Comprising 2,101 high quality frames annotated across 7 classes, the dataset addresses the challenges of fast paced gameplay, motion blur, and small object detection. By providing rich annotations, high resolution imagery, and sequential frame support, HockeyAI enables a range of applications, from player and puck tracking to smart video cropping and automated gameplay analysis.

The dataset's public availability, along with an accompanying fine tuned YOLOv8 model, fosters collaboration and innovation in sports analytics and computer vision research. While designed for ice hockey, the insights and methodologies behind HockeyAI offer broader relevance for other fast paced sports and high speed environments.

Future work will focus on expanding the dataset's size, incorporating temporal and spatial annotations, and exploring its application in real-time systems. HockeyAI aims to remain at the forefront of sports analytics research, driving advancements in both academic and industrial domains.

Acknowledgments

This research was funded by the Research Council of Norway, projects #354154 (Smart Sports Studio) and #346671 (AI-Storyteller). The authors would like to thank the Swedish Hockey League (SHL) for making frames from the videos available for the research.

References

[1] Forzasys AS. 2024. Forzasys DIGITALIZING SPORTS. https://forzasys.com/.

[2] Boone Ashworth. 2024. The Paris Olympics Will Show Us the Future of Sports on TV. https://www.wired.com/story/paris-olympics-broadcast-tech-ai/

[3] Yutao Cui, Chenkai Zeng, Xiaoyu Zhao, Yichun Yang, Gangshan Wu, and Limin Wang. 2023. SportsMOT: A Large Multi-Object Tracking Dataset in Multiple Sports Scenes. https://doi.org/10.48550/arXiv.2304.05170 arXiv:2304.05170

[4] Ravirajsinh Dabhi. 2024. Ice hockey Dataset. https://universe.roboflow.com/ravirajsinh-dabhi-6mq2l/ice-hockey-drjvv. visited on 2025-01-09.

[5] Eagle Eye. 2022. Hockey Dataset. https://universe.roboflow.com/eagle-eye/hockey-jgl1w visited on 2025-01-09.

[6] Simone Francia. 2024. SpaceJam: a Dataset for Basketball Action Recognition. https://github.com/simonefrancia/SpaceJam.

[7] Sushant Gautam, Cise Midoglu, Saeed Shafiee Sabet, Dinesh Baniya Kshatri, and Pål Halvorsen. 2022. Soccer Game Summarization using Audio Commentary, Metadata, and Captions. In *Proceedings of the 1st Workshop on User-Centric Narrative Summarization of Long Videos* (Lisboa, Portugal) *(NarSUM '22)*. Association for Computing Machinery, New York, NY, USA, 13–22. https://doi.org/10.1145/3552463.3557019

[8] Mehdi Houshmand Sarkhoosh. 2025. HockeyAI: A Multi-Class Ice Hockey Dataset with YOLOv8 Medium Model for Object Detection. https://huggingface.co/SimulaMet-HOST/HockeyAI. Accessed: 2025-01-08.

[9] Q. Hu, A. Scott, C. Yeung, et al. 2024. Basketball-SORT: an association method for complex multi-object occlusion problems in basketball multi-object tracking. , 86281–86297 pages. https://doi.org/10.1007/s11042-024-20360-2

[10] International Ice Hockey Federation. 2024. IIHF Official Rulebook 2024/25 – Version 1.0, May 2024. https://blob.iihf.com/iihf-media/iihfmvc/media/contentimages/4_sport/officiating/rule_book/24_25/2024_iihf_rulebook_24052024_v1.pdf. Accessed: January 8, 2025.

[11] KellerWells. 2024. Hockey_FullIce Dataset. https://universe.roboflow.com/kellerwells-allyr/hockey_fullice. visited on 2025-01-09.

[12] Professional Swedish Hockey Leauge. 2025. https://www.shl.se/.

[13] Mohammad Majidi, Mehdi Houshmand Sarkhoosh, Cise Midoglu, Saeed S. Sabet, Tomas Kupka, Dag Johansen, and Pål Halvorsen. 2024. SmartCrop-H: AI-Based Cropping of Ice Hockey Videos. In *Proceedings of the 15th ACM Multimedia Systems Conference* (Bari, Italy) *(MMSys '24)*. Association for Computing Machinery, New York, NY, USA, 471–477. https://doi.org/10.1145/3625468.3652195

[14] Nazanin Mehrasa, Yatao Zhong, Frederick Tung, Luke Bornn, and Greg Mori. 2017. Learning Person Trajectory Representations for Team Activity Analysis. https://doi.org/10.48550/arXiv.1706.00893 arXiv:1706.00893

[15] Cise Midoglu, Steven A. Hicks, Vajira Thambawita, Tomas Kupka, and Pål Halvorsen. 2022. MMSys'22 Grand Challenge on AI-based Video Production for Soccer. https://doi.org/10.48550/arXiv.2202.01031 arXiv:2202.01031

[16] Cise Midoglu, Saeed Shafiee Sabet, Mehdi Houshmand Sarkhoosh, Mohammad Majidi, Sushant Gautam, Håkon Maric Solberg, Tomas Kupka, and Pål Halvorsen. 2024. AI-Based Sports Highlight Generation for Social Media. In *Proceedings of the 3rd Mile-High Video Conference* (Denver, CO, USA) *(MHV '24)*. Association for Computing Machinery, New York, NY, USA, 7–13. https://doi.org/10.1145/3638036.3640799

[17] Germán Mora-Martín, Alex Turpin, Alice Ruget, Abderrahim Halimi, Robert Henderson, Jonathan Leach, and Istvan Gyongy. 2021. High-speed object detection with a single-photon time-of-flight image sensor. https://doi.org/10.1364/OE.435619 arXiv:2107.13407

[18] N/A. 2024. Hockey Dataset. https://universe.roboflow.com/assignments-dojac/hockey-fwm0b-4ikzl-kpjyo. visited on 2025-01-09.

[19] Helmut Neher. 2018. Hockey Pose Estimation and Action Recognition using Convolutional Neural Networks to Ice Hockey.

[20] Olav Andre Nergård Rongved, Markus Stige, Steven Alexander Hicks, Vajira Lasantha Thambawita, Cise Midoglu, Evi Zouganeli, Dag Johansen, Michael Alexander Riegler, and Pål Halvorsen. 2021. Automated Event Detection and Classification in Soccer: The Potential of Using Multiple Modalities. *Machine Learning and Knowledge Extraction* 3, 4 (2021), 1030–1054. https://doi.org/10.3390/make3040051

[21] Playerstracker. 2024. Players Tracking System Dataset. https://universe.roboflow.com/playerstracker/players_tracking_system. Visited on 2025-01-08.

[22] Joseph Redmon, Santosh Kumar Divvala, Ross B. Girshick, and Ali Farhadi. 2015. You Only Look Once: Unified, Real-Time Object Detection. arXiv:1506.02640 http://arxiv.org/abs/1506.02640

[23] Mehdi Houshmand Sarkhoosh, Sayed Mohammad Majidi Dorcheh, Cise Midoglu, Saeed Shafiee Sabet, Tomas Kupka, Dag Johansen, Michael A. Riegler, and Pål Halvorsen. 2024. AI-Based Cropping of Ice Hockey Videos for Different Social Media Representations. *IEEE Access* 12 (2024), 118227–118249. https://doi.org/10.1109/ACCESS.2024.3449152

[24] Mehdi Houshmand Sarkhoosh, Sushant Gautam, Cise Midoglu, Saeed Shafiee Sabet, and Pål Halvorsen. 2024. Multimodal AI-Based Summarization and Storytelling for Soccer on Social Media. In *Proceedings of the 15th ACM Multimedia Systems Conference* (Bari, Italy) *(MMSys '24)*. Association for Computing Machinery, New York, NY, USA, 485–491. https://doi.org/10.1145/3625468.3652197

[25] Mehdi Houshmand Sarkhoosh, Sushant Gautam, Cise Midoglu, Saeed Shafiee Sabet, Thomas Torjusen, and Pål Halvorsen. 2024. The SoccerSum Dataset for Automated Detection, Segmentation, and Tracking of Objects on the Soccer Pitch. In *Proceedings of the 15th ACM Multimedia Systems Conference* (Bari, Italy) *(MMSys '24)*. Association for Computing Machinery, New York, NY, USA, 353–359. https://doi.org/10.1145/3625468.3652180

[26] Mohieddine Amine Silvio Giancola. 2018. SoccerNet: A Scalable Dataset for Action Spotting in Soccer Videos. , 1792–179210 pages. https://doi.org/10.1109/cvprw.2018.00223

[27] Håkon Maric Solberg, Mehdi Houshmand Sarkhoosh, Sushant Gautam, Saeed Shafiee Sabet, Pål Halvorsen, and Cise Midoglu. 2024. PLayerTV: Advanced Player Tracking and Identification for Automatic Soccer Highlight Clips. arXiv:2407.16076 [cs.CV] https://arxiv.org/abs/2407.16076

[28] Konstantin Sozykin, Stanislav Protasov, Adil Khan, Rasheed Hussain, and Jooyoung Lee. 2018. Multi-label Class-imbalanced Action Recognition in Hockey Videos via 3D Convolutional Neural Networks. , 146-151 pages. https://doi.org/10.1109/SNPD.2018.8441034

[29] Sportality. 2024. Sportality - Web publishing for sports. https://www.sportality.se/.

[30] SportContract. 2023. Hockey Dataset. https://universe.roboflow.com/sportcontract/hockey-fwm0b. Visited on 2025-01-08.

[31] Sportobal. 2023. Hockey Players Dataset. https://universe.roboflow.com/sportobal-xe8fu/hockey-players-i23se. Visited on 2025-01-08.

[32] Label Studio. 2025. https://labelstud.io/.

[33] Roboflow Universe. 2024. Hockey8 Dataset. https://universe.roboflow.com/yolotest-olm2o/hockey8. visited on 2025-01-16.

[34] Joakim O. Valand, Haris Kadragic, Steven A. Hicks, Vajira Thambawita, Cise Midoglu, Tomas Kupka, Dag Johansen, Michael A. Riegler, and Pål Halvorsen. 2021. Automated Clipping of Soccer Events using Machine Learning. , 210-214 pages. https://doi.org/10.1109/ISM52913.2021.00042

[35] Kanav Vats, Mehrnaz Fani, David A Clausi, and John Zelek. 2021. Puck localization and multi-task event recognition in broadcast hockey videos. , 4567–4575 pages.

[36] Kanav Vats, Pascale Walters, Mehrnaz Fani, David A. Clausi, and John Zelek. 2021. Player Tracking and Identification in Ice Hockey. https://doi.org/10.48550/arXiv.2110.03090 arXiv:2110.03090 [cs.CV]

[37] Kanav Vats, Pascale Walters, Mehrnaz Fani, David A. Clausi, and John S. Zelek. 2023. Player tracking and identification in ice hockey. https://www.sciencedirect.com/science/article/pii/S0957417422022680. , 119250 pages. https://doi.org/10.1016/j.eswa.2022.119250

[38] Jie Wang, Yuzhou Peng, Xiaodong Yang, Ting Wang, and Yanming Zhang. 2022. SportsTrack: An Innovative Method for Tracking Athletes in Sports Scenes. https://doi.org/10.48550/arXiv.2211.07173 arXiv:2211.07173

[39] Xan989. 2023. Autocamera_hokk2 Dataset. https://universe.roboflow.com/xan989/autocamera_hokk2. Visited on 2025-01-08.

[40] Tomoya Yamaguchi, Kohei Arai, Tomoaki Niiyama, Atsushi Uchida, and Satoshi Sunada. 2023. Ultrafast single-channel machine vision based on neuro-inspired photonic computing. https://doi.org/10.1038/s42005-023-01368-w arXiv:2302.07875

[41] Kua Chen Yingnan Zhao, Zihui Li. 2020. A Method for Tracking Hockey Players by Exploiting Multiple Detections and Omni-Scale Appearance Features. https://github.com/grant81/hockeyTrackingDataset.



uvgVPCCenc: Practical Open-Source Encoder for Fast V-PCC Compression

Louis Fréneau
louis.freneau@tuni.fi
Tampere University
Tampere, Finland

Guillaume Gautier
guillaume.gautier@tuni.fi
Tampere University
Tampere, Finland

Alexandre Mercat
alexandre.mercat@tuni.fi
Tampere University
Tampere, Finland

Jarno Vanne
jarno.vanne@tuni.fi
Tampere University
Tampere, Finland

Abstract

Video-based Point Cloud Compression (V-PCC) standard offers state-of-the-art tools and efficiency for volumetric video compression. The V-PCC reference software, TMC2, is able to demonstrate the compression efficiency of V-PCC, but with computational complexity that is impractical for real-life applications. This paper introduces a novel open-source V-PCC encoder, named uvgVPCCenc, that brings V-PCC compression speed to a practical level. To reduce computational bottlenecks, uvgVPCCenc is implemented from the ground up in C++ with advanced multi-threading and streamlined algorithms. It also supports a flexible parameterization and different presets that facilitate its adaptation to various content and target bitrates. Our comparative evaluations show that uvgVPCCenc achieves substantial speedups over TMC2 while maintaining competitive rate-distortion-complexity tradeoff. To the best of our knowledge, uvgVPCCenc is the first open-source encoder explicitly designed for practical V-PCC compression, paving the way from theoretical research to practical volumetric video applications.

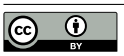
CCS Concepts

• **Computing methodologies** → **Image compression**; **Point-based models**; • **Information systems** → **Open source software**.

Keywords

Practical encoding, Point cloud, Visual Volumetric Video-based Coding (V3C), Video-based Point Cloud Compression (V-PCC), MPEG Immersive Video (MIV), Extended reality (XR)

ACM Reference Format:
Louis Fréneau, Guillaume Gautier, Alexandre Mercat, and Jarno Vanne. 2025. uvgVPCCenc: Practical Open-Source Encoder for Fast V-PCC Compression. In *ACM Multimedia Systems Conference 2025 (MMSys '25), March 31-April 4, 2025, Stellenbosch, South Africa.* ACM, New York, NY, USA, 7 pages. https://doi.org/10.1145/3712676.3718336



This work is licensed under a Creative Commons Attribution 4.0 International License.
MMSys '25, Stellenbosch, South Africa
© 2025 Copyright held by the owner/author(s).
ACM ISBN 979-8-4007-1467-2/2025/03
https://doi.org/10.1145/3712676.3718336

1 Introduction

Recent advances in visual volumetric media technologies have enabled the creation of detailed 3D digital representations of real-world objects and scenes. These developments enable immersive *extended reality* (*XR*) experiences with *six degrees of freedom* (*6DoF*), which can be applied, e.g., in communication, remote supervision, gaming, broadcasting, and motion pictures. However, the storage and transmission of volumetric data require efficient compression, which is a key enabler for these XR applications. To address this, *Moving Picture Experts Group* (*MPEG*) introduced the *Visual Volumetric Video-based Coding* (*V3C*) standard suite, specified as ISO/IEC 23090-5 [5]. A major component of V3C is *Video-based Point Cloud Compression* (*V-PCC*) [5], which compresses dynamic point clouds using conventional 2D video coding schemes, such as the *High Efficiency Video Coding* (*HEVC*) standard [8].

Test Model Category 2 (*TMC2*) [24] is currently the only publicly available V-PCC implementation. As a V-PCC reference encoder, it implements all V-PCC coding tools, but lacks sufficient coding speed for practical usage. The computational complexity of TMC2, particularly its patch generation process and reliance on a non-optimized 2D encoder, makes it impractical for real-world applications.

A recent study [15] attempted to achieve real-time V-PCC encoding by using a closed-source, modified version of TMC2 on a distributed platform with 13 CPUs. Although this setup achieved real-time encoding, it did not offer a practical solution. Similarly, the KDDI company promoted another closed-source, real-time solution [10]. In addition, several works have investigated the high computational complexity of specific stages in the V-PCC encoding pipeline, such as 2D projection [20, 26, 29–31], refined segmentation [9, 11, 13, 18, 21, 27], patch segmentation [14, 28], map generation [21], and 2D encoding [17, 25, 34–37]. However, all these studies were based on the unoptimized TMC2 encoder. Thus, the reported speedups do not necessarily generalize to real-world scenarios where a fully optimized pipeline is expected.

To tackle the need for efficient V-PCC implementations, this paper presents uvgVPCCenc. To the best of our knowledge, this is the first open-source V-PCC encoder designed for real-time performance and practical use. uvgVPCCenc is licensed under BSD 3-Clause and is available online at:

https://github.com/ultravideo/uvgVPCCenc.

uvgVPCCenc is designed from the ground up with numerous optimizations, including efficient multithreading, simplified patch

generation, and the default use of our widely adopted Kvazaar HEVC encoder [16]. While an earlier work has demonstrated preliminary real-time performance for encoding voxel 9 sequences under All Intra (AI) condition [26], this paper provides a comprehensive overview of the software and presents results from an extensive set of experiments.

This paper is organized as follows: Section 2 provides a brief overview of the V-PCC encoding pipeline. Section 3 introduces uvgVPCCenc and highlights its key contributions. Section 4 analyzes its computational complexity and quality. Section 5 discusses practical usage, and Section 6 concludes the paper.

2 V-PCC Overview

Fig. 1 illustrates an overview of the V-PCC encoding pipeline as implemented in its reference software TMC2[1].

2.1 Patch Generation

Patch generation transforms each input point cloud frame into a collection of 3D patches, establishing a correspondence between regions of the point cloud and their 2D projections. The key steps of this process are depicted in Fig. 2.

2.1.1 Normal Computation & Orientation. For each point in a point cloud frame, a normal vector is computed through a matrix-based calculation applied to its neighborhood, with neighboring points identified using a kd-tree structure. Although the direction of each normal can be determined locally, its orientation, i.e., whether it points inward or outward, cannot. To ensure the normals point outward from the shape being captured, a propagation algorithm selectively flips individual normals, resulting in a globally coherent orientation.

2.1.2 Initial Segmentation. During this step, each point in the point cloud is assigned a *Projection Plane Index* (*PPI*). The PPI is determined by selecting the bounding box plane whose normal minimizes the dot product with the point's normal.

2.1.3 Refined Segmentation. This step smooths the PPI selection, ensuring that neighboring points are projected onto the same plane. To produce a PPI selection optimized for the subsequent patch segmentation process, the smoothing is intentionally aggressive, creating large regions of coherent PPI selection. This approach minimizes the formation of small patches, thereby enhancing encoding efficiency. However, it also increases occlusion, leading to a higher number of missing points in the decoded point cloud, as shown in our previous V-PCC tool space exploration [2].

The *refined segmentation* process is the most computationally demanding stage of the V-PCC patch generation pipeline. Extensive effort has been invested to improve its algorithm [12]. To reduce computational overhead, this process operates on a downscaled version of the point cloud. A preprocessing step then focuses iterative computations on a selected subset of points that are most likely to undergo changes in their PPI.

2.1.4 Patch Segmentation. This step employs a propagation algorithm to group neighboring points that share the same PPI. The connected components obtained through this process are then refined to satisfy specific size and depth constraints, resulting in the formation of patches. This propagation algorithm iterates until no additional points can be assigned to a patch.

[1]Fig. 4 can help illustrate the V-PCC pipeline, although its content was generated by the proposed encoder, uvgVPCCenc.

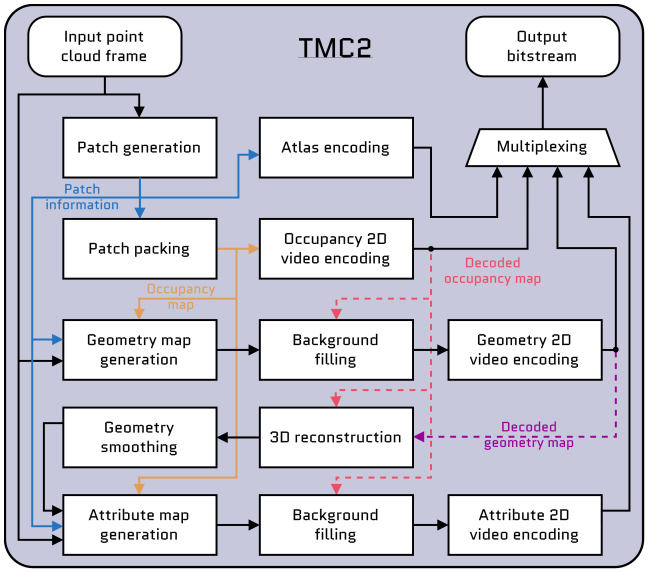


Figure 1: TMC2 encoding workflow.

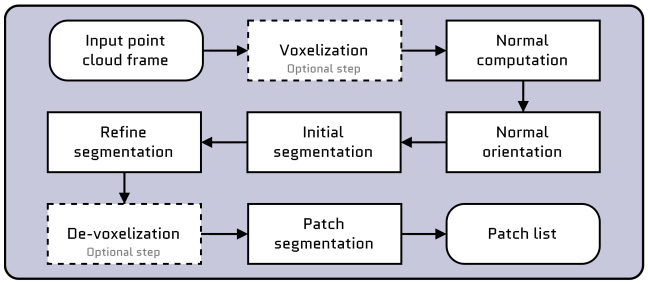


Figure 2: Main steps of the *patch generation* process.

2.1.5 Voxelization & Devoxelization. To reduce computational complexity and leverage the high spatial correlation between adjacent points, the entire PPI selection process can be performed on a downscaled version of the input point cloud (see Fig. 2). Specifically, each point is mapped to the centroid of the voxel in which it resides. All subsequent operations are then carried out on these centroid points. After the *refined segmentation* step, the PPI associated with each voxel is propagated to all original points within that voxel.

2.2 Patch Packing

In AI condition, *patch packing* is performed independently for each frame. The patches are first sorted by size, with larger patches prioritized. Each patch is then placed at the first available top-left position on a 2D canvas where it does not overlap with any previously placed patches.

Table 1: CTC *rate* scale for V-PCC

V-PCC Rates	R1	R2	R3	R4	R5
Geometry encoding QP	32	28	24	20	16
Attribute encoding QP	42	37	32	27	22
Occupancy downscaling factor	4	4	4	4	2

In *random access* (*RA*) condition, similar patches across the frames of a *group of frames* (*GOF*) are identified using an intersection-over-union metric [29]. These matched patches, which represent the same region over the point cloud sequence, are then placed at the same location in each frame canvas. This spatial alignment across frames allows the 2D video encoder to better exploit temporal correlations. Once the matched patches are packed, the remaining unmatched patches are packed for each frame following the *patch packing* method used in AI condition.

2.3 Map Generation

Map generation creates the final three videos that encode occupancy, geometry, and attributes information.

2.3.1 Occupancy. For each frame, the 2D location of the patches on the canvas obtained during patch packing is used to generate the occupancy map, i.e., a binary map indicating whether each pixel belongs to a patch. During decoding, it serves as a mask to indicate which pixels in the geometry and attribute maps are used to reconstruct the point cloud.

2.3.2 Geometry & Attribute. In V-PCC, a patch represents a 3D object with small thickness (up to four points by default), rather than a surface. As a results, multiple points may project onto the same 2D location. To tackle these occlusions, V-PCC splits each patch into two layers, storing the farthest and nearest points relative to the projection plane. Consequently, each point cloud frame generates two geometry maps and two attribute maps, all sharing the same occupancy map. This so-called *double-layer* strategy improves encoding quality with minimal impact on computational complexity or bitstream size. Indeed, the layers being similar, 2D video encoders can process them efficiently.

2.3.3 Occupancy Downscaling & 3D Reconstruction. The occupancy map is downscaled by a factor of two or four, depending on the selected *rate* (see Table 1). As a result, each occupied pixel in the downscaled occupancy map represents 4 or 16 pixels in the geometry and attribute maps, which are then reconstructed during decoding. While this approximation can introduce artifacts in the decoded point cloud, the V-PCC decoder mitigates them through post-processing steps such as geometry smoothing. To further improve coding efficiency and ensure consistency with the decoder, the TMC2 encoder decodes the occupancy and geometry maps, applies geometry smoothing, and then uses the reconstructed 3D point cloud to generate the attribute map.

2.3.4 Background Filling. To improve the efficiency of 2D video encoding, a *background filling* process is applied to fill empty pixels in both the geometry and attribute maps. This approach also minimizes quality loss caused by extra points reconstructed through occupancy map downscaling.

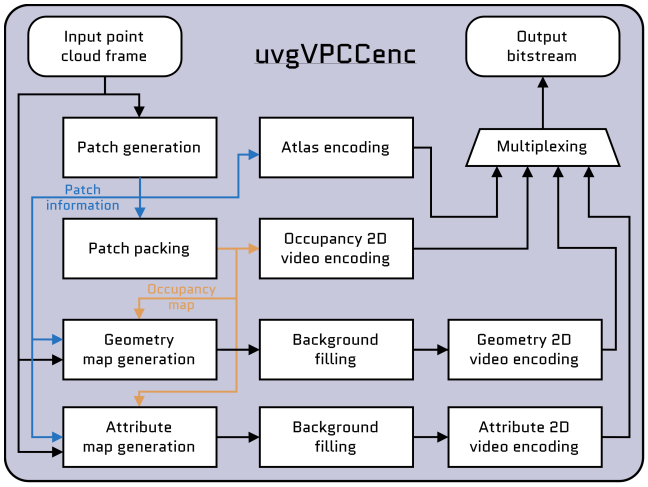


Figure 3: uvgVPCCenc encoding workflow.

2.4 2D Video Encoding & Multiplexing

The occupancy, geometry, and attribute 2D videos are encoded separately. The occupancy video is encoded using lossless compression, while the geometry and attribute videos use lossy compression. V-PCC is codec-agnostic, meaning it is independent of the 2D codec used. Under the *common test condition* (*CTC*) for V-PCC [7], TMC2 utilizes the HEVC reference encoder *HEVC Test Model* (*HM*) [23]. The final bitstream is compliant with V3C standards [5], carrying the three video bitstreams alongside entropy-encoded patch information (also called atlas information) required to reconstruct point cloud frames from their 2D projections.

3 uvgVPCCenc Software Description

Fig. 3 gives an overview of the uvgVPCCenc encoding process, while Fig. 4 illustrates its intermediate steps.

3.1 Streamlined Encoding Pipeline

The proposed initial release of uvgVPCCenc is designed to achieve practical encoding times within an accessible development environment. To this end, it implements only the essential tools for V-PCC encoding, omitting optional features like lossless compression and computationally intensive processes such as 3D reconstruction. Eliminating 3D reconstruction removes dependencies between V-PCC encoding steps, significantly simplifying the pipeline (see Fig. 1 and Fig. 3). Consequently, all *map generation* and *2D video encoding* steps are processed independently in uvgVPCCenc.

This independent treatment of videos, along with the frame-level independence of the *patch generation* processes, is leveraged through a custom thread-pool mechanism. Additionally, to handle the computational demands of *2D encoding* [2], uvgVPCCenc uses Kvazaar by default, which is our award-winning, open-source practical HEVC encoder [16].

3.2 Algorithmic Optimizations

Refined segmentation and *patch segmentation* are the most computationally intensive steps of patch generation [2], mainly because they

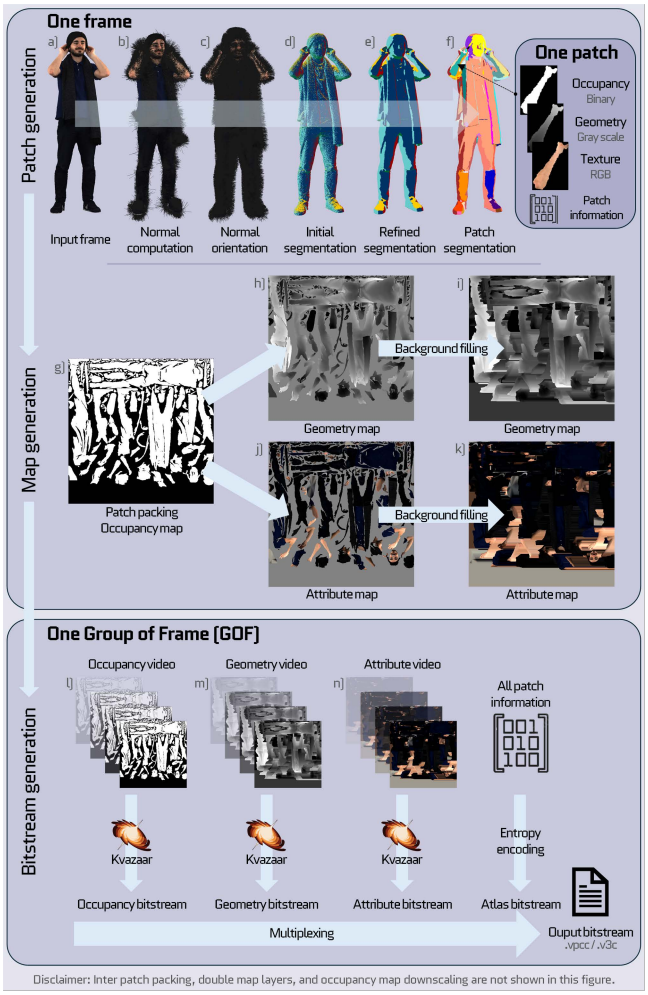


Figure 4: Intermediate steps of the uvgVPCCenc pipeline.

rely on neighboring-search algorithms. Unlike TMC2, which employs a kd-tree search, uvgVPCCenc uses a LUT-based method that reduces computational costs, as proposed in related works [13, 14].

To accelerate *patch packing*, uvgVPCCenc increases the step size between tested positions and integrates an early-termination mechanism that sequentially checks corners, edges, and finally the entire patch location. Under RA condition, uvgVPCCenc simplifies the exhaustive matching algorithm of TMC2 by classifying patches as "inter-patch" only if they consistently match across all frames within a GOF. Although this approach reduces inter-frame correlation in the 2D videos, it achieves significant computational savings.

In addition, uvgVPCCenc employs a simplified RGB-to-YUV color conversion and implements a straightforward pixel-value extension for background filling, which are less extensive than the corresponding schemes used by TMC2 [6].

3.3 Parameter Tuning and Presets

As noted in our previous tool space exploration [2], the CTC for V-PCC are suboptimal with respect to computational complexity. Significant speedups can be achieved by adjusting encoding tool parameters, with only a minimal impact on quality. Below are some key parameterization strategies for uvgVPCCenc.

3.3.1 Normal Computation. To reduce computational complexity, uvgVPCCenc generates low-quality normals. Although this introduces some noise, the aggressive PPI smoothing in *refined segmentation* largely compensates for it, resulting in minimal impact on subsequent *patch segmentation*.

3.3.2 Refined Segmentation. A significant reduction in computational complexity is achieved by decreasing the number of iterations in the *refined segmentation* process. To compensate potential quality loss, uvgVPCCenc applies more aggressive PPI smoothing at each iteration.

3.3.3 Voxelization. Although *voxelization* is not used in the CTC for V-PCC, our previous tool space exploration [2] highlights its importance for practical encoding. In particular, as the voxel resolution of the input sequence increases, neighboring points become more similar, reducing the significance of the approximation introduced by voxelization. For voxels 10, 11, or higher, which contain large numbers of points, *voxelization* substantially reduces computational complexity with minimal impact on encoding efficiency.

3.3.4 Neighboring Research. The *patch generation* processes rely heavily on neighboring-search algorithms, whose computational complexity increases with the number of neighboring points considered. The high search ranges specified by the CTC often gather redundant information. By reducing theses search ranges, uvgVPCCenc significantly lowers computational complexity with minimal impact on encoding quality [2].

3.3.5 Patch Generation. Patch generation is the most computationally demanding step of the V-PCC pipeline. However, its effect on overall encoding efficiency is relatively smaller than that of the subsequent 2D encoding steps, especially for low-resolution point clouds or under real-time constraints. Consequently, uvgVPCCenc prioritizes faster processing by lowering the quality of the generated patches. This approach produces smaller patches with less smooth boundaries compared to those from TMC2, yet it significantly accelerates the overall *patch generation* process.

To clarify the encoding pipeline parametrization, six presets have been defined for Voxel 9, 10, and 11, each available in "Fast" and "Slow" variants. These presets were manually crafted, initially derived from Kvazaar's presets, and refined through extensive tool space exploration using stochastic methods. While they are not exhaustive and do not address certain features, such as single- and double-layer options, they serve as a practical reference for parameter adjustments. Notably, such presets are exclusive to uvgVPCCenc and do not have an equivalent for TMC2.

4 Encoding Complexity and Quality Analysis

The evaluation of encoding complexity and overall quality was performed using TMC2 version 24.0 [24] and uvgVPCCenc version 1.0 [33].

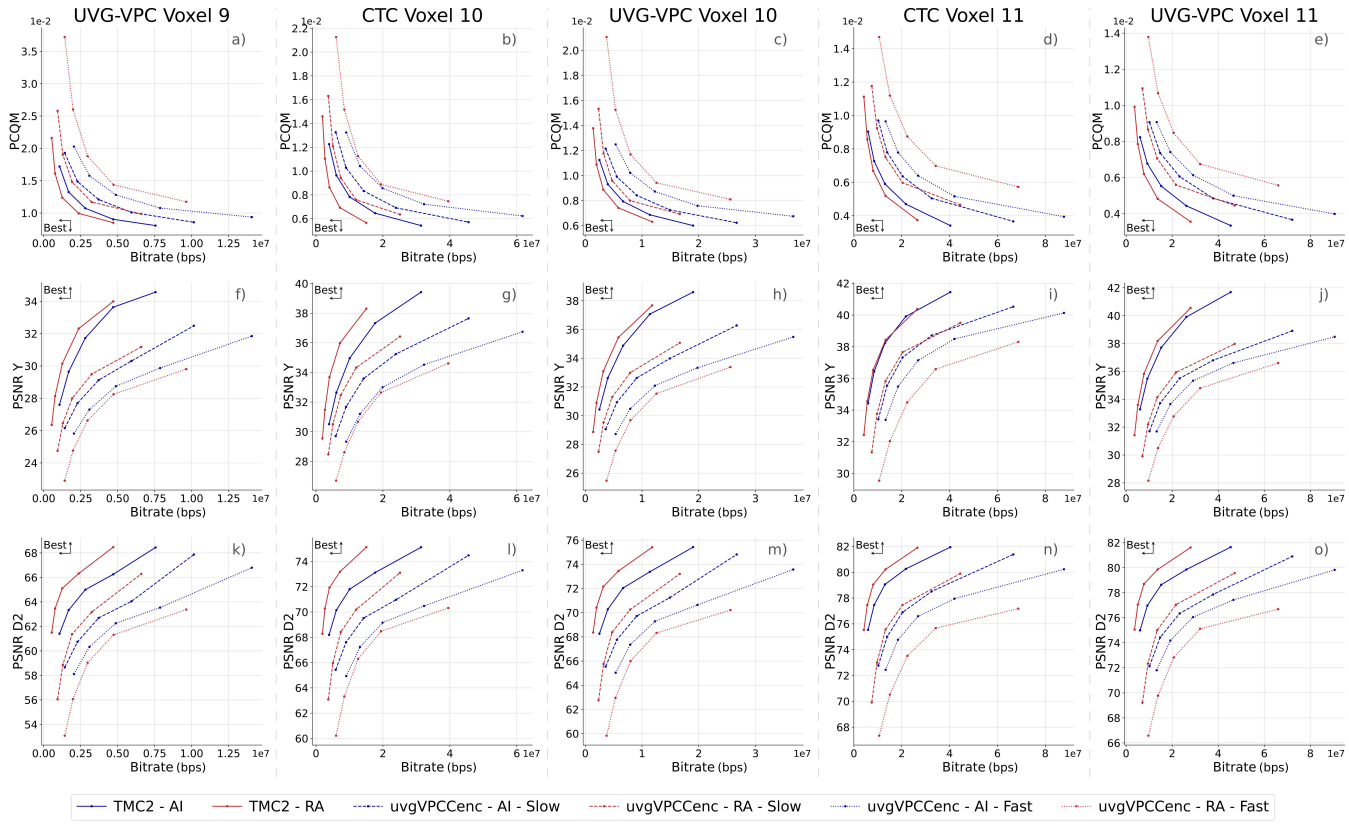


Figure 5: uvgVPCCenc and TMC2 comparison of quality distortion against bitrate across dataset.

Table 2: Encoding complexity and efficiency comparison between TMC2 and uvgVPCCenc

			AI							RA						
			TMC2	uvgVPCCenc		uvgVPCCenc vs. TMC2				TMC2	uvgVPCCenc		uvgVPCCenc vs. TMC2			
				Fast	Slow	Fast		Slow			Fast	Slow	Fast		Slow	
Dataset	**Nb. of seq.**	**Avg. #pts/f**	time /frame	fps	fps	speed up	BD-rate PCQM	speed up	BD-rate PCQM	time /frame	fps	fps	speed up	BD-rate PCQM	speed up	BD-rate PCQM
Vox9 UVG-VPC	12	186k	17.1s	26.1	4.6	447×	170%	79×	70%	49.0s	23.3	3.8	1143×	418%	188×	120%
Vox10 CTC A & B	5	883k	22.2s	6.8	1.6	152×	158%	35×	68%	62.5s	6.2	1.1	385×	414%	68×	120%
Vox10 UVG-VPC	12	709k	20.8s	5.4	1.1	112×	164%	22×	64%	61.9s	4.9	0.8	304×	385%	52×	89%
Vox 11 CTC C	2	2.8M	61.6s	2.1	0.8	128×	147%	52×	83%	242.1s	1.7	0.4	421×	329%	94×	100%
Vox11 UVG-VPC	12	2.4M	58.0s	1.0	0.6	60×	158%	37×	91%	209.2s	1.0	0.4	203×	383%	75×	126%

4.1 Experimental Setup

Experiments were conducted under AI and RA conditions specified by the CTC. All sequences from the CTC for V-PCC [7] and from the UVG-VPC dataset (voxel 9, 10, and 11) [3] were encoded at the *rates* (R1–R5) defined in the CTC and listed in Table 1. Sequences with voxel 9 and 10 each have 250 frames, while those with voxel 11 have 64 frames.

Geometry and attribute quality were evaluated using the PSNR D2 and PSNR Y metrics [7], respectively. In addition, *Point Cloud Quality Metric* (*PCQM*) was used as it better correlates with subjective quality [22]. Coding efficiency was assessed via *Bjøntegaard delta bitrate* (*BD-rate*) [1] based on PCQM, using TMC2 as anchor.

Speed measurements were carried out on an Intel i7-12700 CPU (16×4.8 GHz + 4×3.6 GHz) with 20 threads, encoding the first 64 frames of each sequence at all *rates* (R1–R5). TMC2 encoding speeds were based on a single run due to its long encoding times, which provide enough consistency, while uvgVPCCenc results were averaged over ten runs to ensure accuracy.

4.2 Experimental Results

Fig. 5 shows distortion-versus-bitrate curves for TMC2 and uvgVPCCenc across all datasets, using the PCQM, PSNR Y, and PSNR D2 metrics. Each point on these graphs corresponds to a specific V-PCC rate. In addition, Fig. 2 compares encoding speeds,

where TMC2 performance is expressed as encoding time per frame, and uvgVPCCenc performance is given in *frames per second* (*fps*) along with relative speedups. Finally, BD-rate results reported in Table 2 synthetize the coding efficiency outcomes presented in graphs a) to e) of Fig. 5.

Because the V-PCC standard leverages the 2D coding efficiency of conventional video codecs, the decrease in coding efficiency of uvgVPCCenc primarily stems from using Kvazaar [16], which has lower 2D encoding efficiency than HM [23]. However, by employing a practical video encoder along with structural and algorithmic optimizations, as well as parameter tuning (detailed in Section 3), uvgVPCCenc consistently achieves substantial encoding speedups across all tested configurations and sequences.

Notably, speedups averaged over the five *rates* (R1–R5) do not reflect variations across rates. For both TMC2 and uvgVPCCenc, lower-quality rates achieve faster encoding speeds than higher-quality ones. For instance, on voxel 9 sequences under the AI Fast configuration, uvgVPCCenc averages 25.3 fps at R5 but increases to 27.1 fps at R1.

5 Practical Use of uvgVPCCenc

Beyond delivering real-time compression performance, uvgVPCCenc is designed as an open experimental platform. Its open-source and modular architecture encourages experimentation with new coding tools and facilitates rapid prototyping of volumetric video applications.

5.1 User Instructions

uvgVPCCenc is a V-PCC encoder designed to process positive integer point cloud sequences. While it supports all point cloud resolutions, it is specifically tuned for voxel 9, 10, and 11. Developed from scratch in C++, the encoder is compatible with Linux and can be compiled using either Clang or GCC. The software is available on GitHub, with CMake build instructions, tests, and detailed usage in the `README.md` file.

The repository [33] offers several resources to facilitate the adoption of the library for V-PCC encoding applications. The library's public API documentation, generated with Doxygen, is hosted on GitHub [32]. Additionally, the `src/app` folder includes an application that demonstrates common usage of the library, with usage instructions provided in the `README.md`.

Efforts have been made to enhance the software's usability. For instance, the ability to export intermediary files, as displayed in Fig. 4, streamlines tasks for developers working with the uvgVPCCenc pipeline. Additionally, while Kvazaar is the default 2D encoder, the integration of alternative video encoders into the uvgVPCCenc framework is well-supported.

5.2 Achievable Range of Quality

The *rate* scale defined in the CTC for V-PCC (Table 1) aims to offer diverse compression scenarios for standardization. However, these settings are not absolute and may be suboptimal in certain contexts, particularly since they remain fixed regardless of input resolution. For instance, a voxel 9 sequence may exhibit more pronounced artifacts from geometry 2D video compression and occupancy map downscaling than the same sequence at voxel 10. In such cases, even at the same bitrate, allocating bits more effectively among occupancy, geometry, and attribute maps could yield better overall quality [4, 19].

In addition, when bitstream size is less of a constraint, using lossless 2D encoding for both geometry and attribute videos can deliver outstanding objective and subjective quality. Although this does not result in fully lossless V-PCC encoding, due to missing points in *patch generation*, its improved quality and faster processing speed compared with TMC2 offer a viable option for users. Moreover, employing lossless 2D encoding underscores the pivotal role of 2D coding in determining final V-PCC quality.

5.3 Non-Standardized Tools

The V-PCC standard defines the structure of a V-PCC bitstream but does not specify how it should be generated. TMC2 serves both as an implementation platform during the standardization process and as a baseline reference for researchers exploring V-PCC compression. However, its encoding pipeline, especially the *patch generation* process, is not intended to be strictly followed. This is particularly relevant because computational complexity, while not a primary concern during standardization, is crucial for practical applications. As a result, several works [29, 30, 35] have substantially modified parts of the TMC2 encoding pipeline freely diverging from the reference implementation. Consequently, it is anticipated that significant parts of the proposed encoder will also evolve or be replaced with innovative solutions in the future. Therefore, this first release of uvgVPCCenc does not presents a fully optimized V-PCC encoder, but rather serves as a modular platform for developing and testing tools within a practical encoding pipeline.

6 Conclusion

This paper introduces uvgVPCCenc, an open-source encoder designed for practical V-PCC compression. By bridging the gap between theoretical encoding models and practical implementations, uvgVPCCenc addresses the need for a flexible development framework that suits the non-normative nature of volumetric video compression.

Future work will build upon the uvgVPCCenc framework to enable real-time compression of high-resolution volumetric video and better exploit temporal correlations between point cloud frames. Focused research in these areas would be of utmost importance to foster the adoption of real-world volumetric video applications.

Acknowledgments

This work was supported in part by the XR Simulation and Presence at the Cloud Edge (XR-SPACE) project led by Nokia and funded by Business Finland, and the Research Council of Finland (decision no. 349216).

References

[1] G. Bjøntegaard. 2008. *Improvements of the BD-PSNR model.* Technical Report VCEG-AI11. VCEG, Berlin, Germany.

[2] L. Fréneau, A. Mercat, G. Gautier, J. Sainio, and J. Vanne. 2024. Tool space exploration of the V-PCC patch generation for practical point cloud encoding. *Proc. Picture Coding Symp.* (Jun. 2024).

[3] G. Gautier, A. Mercat, L. Fréneau, M. Pitkänen, and J. Vanne. 2023. UVG-VPC: voxelized point cloud dataset for visual volumetric video-based coding. *Proc. Int. Conf. Qual. Multimedia Exper.* (Jun. 2023).

[4] Y. et al. Hui. 2024. Optimized quantization parameter selection for video-based point cloud compression. *Frontiers Signal Process.* 4 (Jul. 2024).

[5] ISO/IEC 23090-5:2023. 2023. Information technology — coded representation of immersive media — part 5: visual volumetric video-based coding (V3C) and video-based point cloud compression (V-PCC).

[6] ISO/IEC JTC 1/SC 29/WG 7. 2020. *V-PCC codec description.* Technical Report N00100. MPEG. Online.

[7] ISO/IEC JTC1/SC29/WG11. 2020. *Common test conditions for V3C and V-PCC.* Technical Report N19518. MPEG. Online.

[8] ITU-T Recommendation H.265. 2013. High Efficiency Video Coding.

[9] Q. Jia, K. Kim, M. K. Lee, and E. S. Jang. 2024. Simple grid-based refinement segmentation algorithm for MPEG video-based point cloud compression. *IEEE Access* 12 (Feb. 2024), 23695–23706.

[10] KDDI. 2022. KDDI Research News Release. Online. https://www.kddi-research.jp/english/newsrelease/2022/102401.html.

[11] J. Kim and Y.-H. Kim. 2021. Fast grid-based refining segmentation method in video-based point cloud compression. *IEEE Access* (May 2021).

[12] J. Kim, Y.-H. Kim, and E.-S. Ryu. 2021. *Fast grid-based refining segmentation method in V-PCC.* Technical Report m56635. ISO/IEC JTC 1/SC 29/WG 7. Online.

[13] Y. Kim and Y.-H. Kim. 2022. Low complexity fast grid-based refining segmentation in the V-PCC encoder. In *Proc. IEEE Int. Conf. Consum. Electronics-Asia.* Yeosu, South Korea.

[14] Y. Kim and Y.-H. Kim. 2023. A low-complexity patch segmentation in the VPCC encoder. In *Proc. Int. Tech. Conf. Circuits/Syst., Comput., Commun.* (Jeju, South Korea).

[15] Y. Kim and Y.-H. Kim. 2023. Real-time video-based point cloud encoding system on a distributed platform. In *Proc. Int. Conf. Ubiquitous and Future Networks.* Paris, France, 593–595.

[16] A. Lemmetti, M. Viitanen, A. Mercat, and J. Vanne. 2020. Kvazaar 2.0: fast and efficient open-source HEVC inter encoder. In *Proc. ACM Multimedia Syst. Conf.* (Istanbul, Turkey).

[17] Y. Li, J. Huang, C. Wang, and H. Huang. 2023. Unsupervised learning-based fast CU size decision for geometry videos in V-PCC. *J. Real-Time Image Process.* 21 (Dec. 2023).

[18] T. L. Lin, C. H. Lin, Y. S. Chiou, and S. L. Chen. 2024. Improved grid refine segmentation for 3D point cloud in video-based point cloud compression (V-PCC). *Multimedia Tools Applications* 83 (Jul. 2024), 62701–62720.

[19] Q. Liu, H. Yuan, J. Hou, R. Hamzaoui, and H. Su. 2020. Model-based joint bit allocation between geometry and color for video-based 3D point cloud compression. *IEEE Trans. Multimedia* 23 (Sep. 2020).

[20] E. Lopes, J. Ascenso, C. Brites, and F. Pereira. 2019. Adaptive plane projection for video-based point cloud coding. In *Proc. IEEE Int. Conf. Multimedia Expo.* Shanghai, China.

[21] H. Luo, Y. Chi, S. Lu, Y. Ding, and C. Han. 2024. Improved patch packing and refining segmentation for the V-PCC standard. *Applied Sciences* (2024).

[22] G. Meynet, Y. Nehmé, J. Digne, and G. Lavoué. 2020. PCQM: a full-reference quality metric for colored 3D point clouds. *Proc. Int. Conf. Qual. Multimedia Exper.* (May 2020).

[23] MPEG. [n. d.]. HEVC Reference Software Version 16.20. https://vcgit.hhi.fraunhofer.de/jct-vc/HM/-/tags/HM-16.20. Online.

[24] MPEG. 2024. Test Model Category 2 Version 24. Online. https://github.com/MPEGGroup/mpeg-pcc-tmc2.

[25] S. Que and Y. Li. 2023. Lightweight fully connected network-based fast CU size decision for video-based point cloud compression. *J. Computers & Graphics* 117 (Oct. 2023), 20–30.

[26] S. Rhyu, J. Kim, G. H. Park, and K. Kim. 2024. Low-complexity patch projection method for efficient and lightweight point-cloud compression. *ETRI Journal* 46 (May 2024), 683–696.

[27] I. Seidel, D. R. Freitas, C. Dorea, D. C. Garcia, R. U. B. Ferreira, and R. Higa. 2021. Memory-friendly segmentation refinement for video-based point cloud compression. In *Proc. IEEE Int. Conf. Image Process.* Anchorage, AK, USA.

[28] N. Sheikhipour, M. Pesonen, S. Schwarz, and V. K. M. Vadakital. 2019. Improved single-layer coding of volumetric data. In *Proc. European Workshop Visual Information Process.* (Roma, Italy).

[29] F. Tohidi and M. Paul. 2023. Video-based point cloud compression using density-based variable size hexahedrons. In *Proc. IEEE Int. Conf. Multimedia Expo Workshops.* Brisbane, Australia.

[30] F. Tohidi, M. Paul, and F. Afsana. 2024. Efficient dynamic point cloud compression through adaptive hierarchical partitioning. *IEEE Access* 12 (Oct. 2024), 152614–152629.

[31] F. Tohidi, M. Paul, A. Ulhaq, and S. Chakraborty. 2024. Improved video-based point cloud compression via segmentation. *Sensors* 24 (Jul. 2024), 4285.

[32] Ultra Video Group. 2025. uvgVPCCenc Doxygen. https://ultravideo.github.io/uvgVPCCenc/. Online.

[33] Ultra Video Group. 2025. uvgVPCCenc Git. https://github.com/ultravideo/uvgVPCCenc Online.

[34] W. Wang, D. Gongchun, and D. Ding. 2024. Leveraging occupancy map to accelerate video-based point cloud compression. *J. Visual Comm. Image Representation* 104 (Sep. 2024).

[35] Y. Wang, Y. Wang, T. Cui, and Z. Fang. 2024. Fast video-based point cloud compression based on early termination and transformer model. *IEEE Trans. Emerging Topics Computational Intelligence* 8, 3 (Jun. 2024), 2336–2348.

[36] J. Xiong, H. Gao, M. Wang, H. Li, and W. Lin. 2022. Occupancy map guided fast video-based dynamic point cloud coding. *IEEE Trans. Circuits Syst. Video Technol.* 32, 2 (Feb. 2022), 813–825.

[37] H. Yuan, W. Gao, G. Li, and Z. Li. 2022. Rate-distortion-guided learning approach with cross-projection information for V-PCC fast CU decision. In *Proc. ACM Int. Conf. Multimedia* (New York, USA).

OLED-EQ: A Dataset for Assessing Video Quality and Energy Consumption in OLED TVs Across Varying Brightness Levels

Minh Nguyen
Fraunhofer FOKUS
Berlin, Germany
minh.nguyen@fokus.fraunhofer.de

Raphael Koch
Fraunhofer FOKUS
Berlin, Germany
raphael.koch@fokus.fraunhofer.de

Alexander Fischer
Fraunhofer FOKUS
Berlin, Germany
alexander.fischer@fokus.fraunhofer.de

Moustafa Ghaddar
Fraunhofer FOKUS
Berlin, Germany
moustafa.ghaddar@fokus.fraunhofer.de

Görkem Güclü
Fraunhofer FOKUS
Berlin, Germany
goerkem.gueclue@fokus.fraunhofer.de

Martin Lasak
Fraunhofer FOKUS
Berlin, Germany
martin.lasak@fokus.fraunhofer.de

Robert Seeliger
Fraunhofer FOKUS
Berlin, Germany
robert.seeliger@fokus.fraunhofer.de

Stefan Arbanowski
Fraunhofer FOKUS
Berlin, Germany
stefan.arbanowski@fokus.fraunhofer.de

Stephan Steglich
Fraunhofer FOKUS
Berlin, Germany
stephan.steglich@fokus.fraunhofer.de

Abstract

The climate crisis has highlighted the environmental impact of information and communication technologies (ICT), underscoring the need for sustainable solutions to reduce carbon emissions from ICT. As video streaming continues to dominate Internet traffic, research in this field has become increasingly important. The energy consumption of OLED TVs relies on the video brightness. Thus, an approach to optimized the energy consumption is reducing the video brightness. In this work, we provide an open dataset for energy consumption in OLED TVs while playing videos with various brightness levels. The dataset comprises the energy data of four OLED TVs with different screen sizes and manufacturers in playing 176 videos in a range of dark and bright content. As results, 704 data traces of energy consumption are collected.

We also conduct a subjective test to assess the quality of videos displayed in OLED TVs when they are reduced the brightness. Ten levels of brightness reduction are investigated four different video bitrates and video characteristics. The rating scores are attached to our dataset so that they can be used to develop QoE prediction models in the context of the tradeoff between energy consumption and video brightness. We collected opinion scores from 28 participants, resulting in 2240 responses in total. The dataset is available at **https://github.com/minhkstn/OLED-EQ**.

CCS Concepts

• **Information systems** → **Multimedia databases**; **Multimedia content creation**; • **General and reference** → **Evaluation**.



This work is licensed under a Creative Commons Attribution-NonCommercial-NoDerivatives 4.0 International License.
MMSys '25, March 31-April 4, 2025, Stellenbosch, South Africa
© 2025 Copyright held by the owner/author(s).
ACM ISBN 979-8-4007-1467-2/2025/03
https://doi.org/10.1145/3712676.3718337

Keywords

Adaptive Video Streaming, Green Streaming, Subjective test, Energy Consumption

ACM Reference Format:
Minh Nguyen, Raphael Koch, Alexander Fischer, Moustafa Ghaddar, Görkem Güclü, Martin Lasak, Robert Seeliger, Stefan Arbanowski, and Stephan Steglich. 2025. OLED-EQ: A Dataset for Assessing Video Quality and Energy Consumption in OLED TVs Across Varying Brightness Levels. In *ACM Multimedia Systems Conference 2025 (MMSys '25), March 31-April 4, 2025, Stellenbosch, South Africa.* ACM, New York, NY, USA, 7 pages. https://doi.org/10.1145/3712676.3718337

1 Introduction

In today's digital landscape, video streaming services have become a common feature, catering to a wide range of devices with different display capabilities. According to the Sandvine Phenomena report [23], video streaming—encompassing both on-demand and live content—constitutes 68% of total downstream traffic. This statistic underscores the critical need to address energy consumption associated with video streaming [5]. Recent approaches have mainly focused on minimizing the bitrate and resolution of videos during the encoding (on the server side) and decoding (on the player side) processes [3, 30, 39]. However, these elements are not the primary drivers of energy use in playback devices, especially televisions [25]. Since videos are encoded only once on the server, the opportunities for energy savings through these methods are limited. In our previous work [18], we showed that reducing video brightness can significantly influence the energy consumption of display devices while also affecting viewers' perceived quality. However, an open dataset for energy consumption of OLED TVs and subjective scores of people watching brightness-changed videos on these TVs is still missing.

HTTP Adaptive Streaming (HAS) is essential for video streaming [2, 29]. In the HAS framework, the original video is encoded into multiple quality levels defined by different bitrates and resolutions. Generally, higher bitrates and resolutions yield superior visual quality. Each quality tier is divided into equally timed segments. The

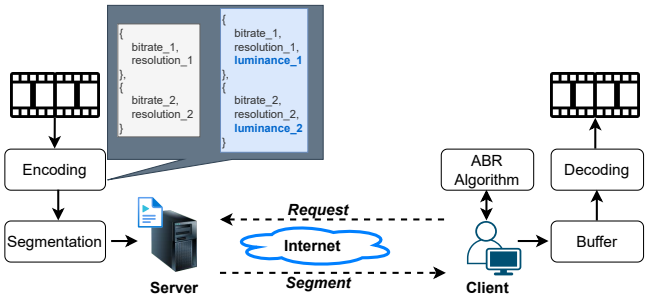


Figure 1: HAS principle system. The blue box with blue text are our proposal to be added to HAS system for the purpose of saving energy consumption at the client side.

client is tasked with downloading the appropriate quality levels for these segments to enhance the quality of experience (QoE) while adjusting to variations in throughput using an Adaptive Bitrate (ABR) algorithm. The architecture of HAS is illustrated in Figure 1.

Conventional ABR algorithms primarily concentrate on adjusting the chosen quality levels in response to network fluctuations, often overlooking energy consumption aspects [7, 19, 26]. Recent developments in energy-aware video encoding [12, 14] and ABR algorithms [10, 34] have aimed to reduce energy use in client devices by lowering the selected bitrate while still ensuring an acceptable QoE. However, the impact of video brightness has not been adequately explored, despite its significant role in energy reduction for many display types, particularly OLED (Organic Light Emitting Diode) televisions [18, 25].

According to a report in [22], the global market for OLED technology was valued at USD 56.37 billion in 2024. By 2034, the market is expected to reach approximately USD 344.58 billion, reflecting a compound annual growth rate (CAGR) of 19.85% from 2024 to 2034.

In this work, we provide a dataset named OLED-EQ of energy consumption for various OLED TVs while playing videos with different brightness levels. The tested videos are encoded at various quality levels determined by bitrates and resolutions. Each quality level is then reduced in brightness for the purpose of saving energy. We also include the subjective score retrieved from our subjective test to evaluate the impact of video brightness reduction on perceptual quality. This dataset can be used for many applications such as energy consumption prediction and QoE prediction models. Especially, our dataset is useful for determining the sweet spot of video brightness reduction and QoE in the HAS system, where the server prepares various representations in terms of brightness, besides bitrate and resolution (see the blue text in Fig. 1). To the best of our knowledge, this is the first open dataset for energy consumption and visual quality for OLED TVs.

The contributions of the paper are three-fold:

- We provide a database of the energy consumption of four different OLED TVs while playing 176 videos at various content, bitrate, and brightness.
- We also provide 176 tested videos for reproducibility.
- We conducted subjective tests to evaluate the impact of video brightness reduction on the perceptual quality of end users. The rating scores are made publicly available to help train and validate QoE prediction models as well as develop objective quality metrics.

The remaining of the paper is constructed as follows. Section 3 draws an overview of the background and related work, followed by the motivation of the paper. Section 4 describes our data acquisition methods (energy consumption and subjective scores). Section 5 presents a dataset analysis and usage scenarios. Finally, Section 6 concludes the paper.

2 Motivation

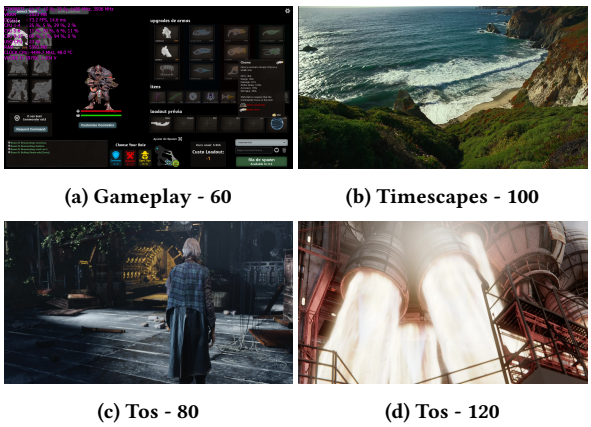
Energy consumption of a video streaming system can be split into four elements: (*i*) data center, (*ii*) network, (*iii*) end device, and (*iv*) manufacturing [20]. The power consumption varies from 50W/stream to 250W/stream [16]. The energy for manufacturing is usually estimated by 60% to 80% of the energy of the end device [11]. This end device's energy varies based on the device type such as 1W for a smartphone, 22W for a Laptop, and 100W for a TV [20]. Finally, the energy consumption of the network can be calculated based on the amount of data with 100 W/GB [27].

Thus, the energy consumption of TVs accounts for a significant part of the energy consumption of the whole system. According to [25], OLED TVs show the most remarkable increase in energy consumption when the displayed video becomes brighter, which implies that there is a positive correlation between video brightness and energy consumption of OLED TVs. In this paper, we provide an open dataset to validate the statements and support further research directions related to environmental-friendly video streaming.

3 Background and Related Work

3.1 Energy-Aware Encoding Approaches

Various strategies have been proposed to enhance energy efficiency in video encoding. Menon *et al.* [14] presented a dynamic resolution encoding method aimed at minimizing both the carbon footprint and energy usage on encoding servers in video streaming applications. Their approach involves an optimized resolution prediction system that estimates appropriate bitrate and resolution pairs for specific video segments, taking into account the spatial and temporal characteristics of the content. With an encoding time capped at 200 seconds, their method can achieve energy savings of up to 84%. In another study [12], the authors introduced an online energy-efficient Multi-Codec Bitrate ladder Estimation scheme (MCBE) for adaptive video streaming. Like their previous work, this scheme focuses solely on determining the bitrate and resolution during encoding, aiming to conserve energy in both encoding and storage by minimizing encoding time and the volume of data produced. However, the energy savings were projected based on models rather than actual energy measurements. Nguyen *et al.* [18] explored a novel approach to energy conservation by lowering video brightness. During the encoding process, in addition to adjusting bitrate and resolution, the encoding server can modify video brightness to assist client devices in reducing their energy consumption. Nonetheless, a limitation of this research is the absence of subjective testing to assess how the reduction in brightness affects viewers' perception of video quality.



(a) Gameplay - 60 (b) Timescapes - 100

(c) Tos - 80 (d) Tos - 120

Figure 2: Frames from the test sequences. The capture of each frame includes the video name and its brightness.

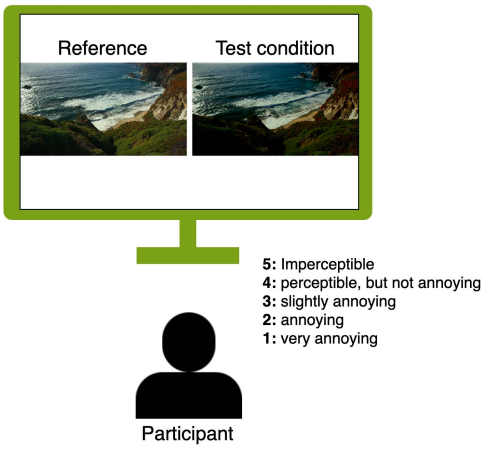


Figure 3: Subjective test setup and rating scores.

3.2 Energy-Aware ABR Algorithms

The brightness of a video frame corresponds to its luminance, meaning that screens displaying brighter videos will consume more energy as they need to increase brightness levels accordingly [38]. Yan *et al.* [37] suggested lowering the brightness of video content and developed an Adaptive Bitrate (ABR) algorithm tailored for mobile devices. However, their research overlooked the effect of brightness on commonly used objective quality metrics such as VMAF [21] and SSIM [36]. Moreover, because mobile devices typically consume much less energy than televisions, it is essential to assess video brightness in relation to TV energy consumption to realize more significant energy savings.

Varghese *et al.* [34] introduced eDASH, an energy-efficient plugin created for MPEG-DASH players on mobile devices. This plugin aims to minimize battery usage by factoring in elements like bitrate and video brightness when selecting the next segment to download. However, the study did not specify the extent to which video brightness should be reduced, nor did it address the potential degradation in quality resulting from changes in brightness.

Turkkan *et al.* [32] presented a reinforcement learning-based ABR algorithm called GreenABR, which considers energy consumption, network throughput, buffer levels, recently downloaded bitrates, and VMAF scores. An Artificial Neural Network was utilized to estimate the energy consumption during video playback based on segment features such as bitrate, resolution, framerate, file size, and VMAF. They further developed a new ABR algorithm, GreenABR+ [31], using a Deep Reinforcement Learning model that encompasses a wider range of bitrates to address the challenge of retraining the model for each new bitrate set.

Lorenzi [10] introduced an energy-aware ABR algorithm called E–WISH. This algorithm considers *(i)* throughput, *(ii)* buffer status, *(iii)* video quality, and *(iv)* energy costs to improve the quality of experience (QoE) while decreasing energy consumption on end-user devices. However, this evaluation was conducted solely on mobile phones, neglecting TV devices and the implications of video brightness.

3.3 TV display technology

Television display technologies have evolved significantly over the past few decades, offering various options that cater to different consumer needs in terms of picture quality, energy efficiency, and price. The three most prominent technologies in the market today are LCD (Liquid Crystal Display), OLED (Organic Light Emitting Diode), and QLED (Quantum Dot Light Emitting Diode).

3.3.1 LCD. LCD technology has been the foundation of the TV market for many years. LCDs rely on a backlight to illuminate liquid crystals, which modulate light to form an image. Despite being the most widely used technology, LCDs have limitations in contrast ratios and viewing angles due to the reliance on a separate backlight.

3.3.2 OLED TV. In comparison to LCD displays, OLED displays provide higher contrast, increased brightness, wider viewing angles, faster response times, and greater energy efficiency [22]. OLED displays are able to turn on or off each pixel individually due to their biological makeup. As a result, they do not need a backlight and consume less power when displaying darker images, as each pixel can be turned off completely. OLED technology has seen significant growth, capturing around 43% of the premium TV market and 20% in the overall TV market [4, 9].

3.3.3 QLED. QLED technology, pioneered by Samsung, improves upon traditional LCDs by incorporating quantum dots into the backlighting system. These microscopic semiconductor particles enhance color accuracy and brightness, making QLED displays a strong competitor to OLED, especially in terms of peak brightness and color volume. QLED TVs do not offer the same deep blacks as OLEDs due to the reliance on a backlight. This display technology occupies only 10% of the TV market [4].

4 Data Acquisition

4.1 Original Sequences

In this work, we use four sequences extracted from three different videos with different brightness levels: Gameplay [17][1], Timescapes [18],

[1]https://www.youtube.com/watch?v=gkIYZCmD-40

Table 1: Bitrates and resolutions used in this work.

Resolution (p)	540	720	1080	2160
Bitrate (kbps)	600	2400	4500	11600

Tears of steel (Tos)[2]. Here, we denote tested sequences as their original video name and their average brightness among frames calculated by the video complexity analyzer tool VCA [13], *i.e.*, *(i)* Gameplay_60, *(ii)* Tos_80, *(iii)* Timescapes_100, and *(iv)* Tos_120. Fig 2 shows one frame of each video to illustrate their brightness.

For the video streaming use case, we encode the tested sequences at multiple different bitrates and resolutions based on Apple's bitrate ladder[3] as shown in Table 1 with HEVC codec. Each bitrate is then reduced in brightness with the FFmpeg tool [1]. Based on the observation from [18], we only consider the first ten percentages of brightness reduction (*i.e.*, from 1% to 10%) as higher reduction can be recognized straightforwardly.

4.2 Objective Quality Metrics

To quantify the quality of the videos under different brightness, we calculate the following objective metrics:

- PSNR (Peak Signal-to-Noise Ratio) [6]: PSNR is a metric that measures the ratio between the maximum possible power of a video frame and the power of the noise that degrades its quality. This metric evaluates the pixel values of both the original and distorted frames directly.
- SSIM (Structural Similarity Index) [36]: SSIM assesses the similarity between two images or videos by examining the structural alterations between the original and distorted versions. Unlike PSNR, SSIM takes into account perceived changes in structural details and luminance, making it more aligned with human perceptions of image and video quality [24, 28].
- VMAF (Video Multimethod Assessment Fusion) [21]: VMAF, developed by Netflix, is designed to predict video quality as experienced by viewers. It utilizes a machine learning model that has been trained on an extensive dataset of videos, making it a reliable predictor of human subjective evaluations [28, 33, 35].

4.3 Subjective Test

4.3.1 Test Setup. The subjective testing was conducted using the double-stimulus method, where each participant was shown two videos simultaneously: a reference video on the left side of the screen and a test video on the right. This double-stimulus setup allows for direct and immediate comparison between the unimpaired reference video, which represented the highest achievable video quality, and the test video, which included various impairments. This method is integral to ensuring that participants have a clear baseline for comparison, enhancing the reliability of their assessments of video quality. Following the recommendations of ITU-R BT.500-15 [8], our experiments are conducted in a room with black walls and low illumination.

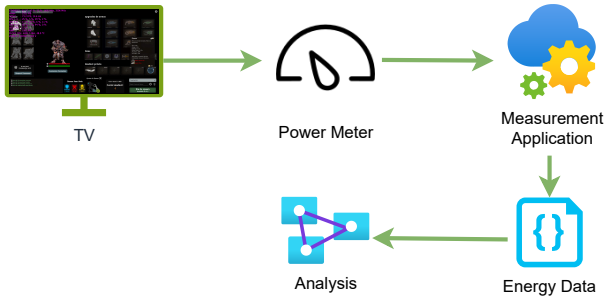


Figure 4: The architecture of our measurement setup.

4.3.2 Procedure. Before the test, participants read the introduction about the project and gave permission to collect, process, and store personal data for the purposes of this study. The personal data includes their gender and age group.

During each testing session, participants were presented with pairs of video sequences and were instructed to rate the test video while considering the quality of the reference video. Ratings were based on the Double-Stimulus Impairment Scale (DSIS), which ranges from 1 ("Very Annoying") to 5 ("Imperceptible") (see Fig. 3). This 5-point impairment scale, detailed in ITU-R BT.500-15 [8], allows participants to effectively quantify the subjective impact of video impairments.

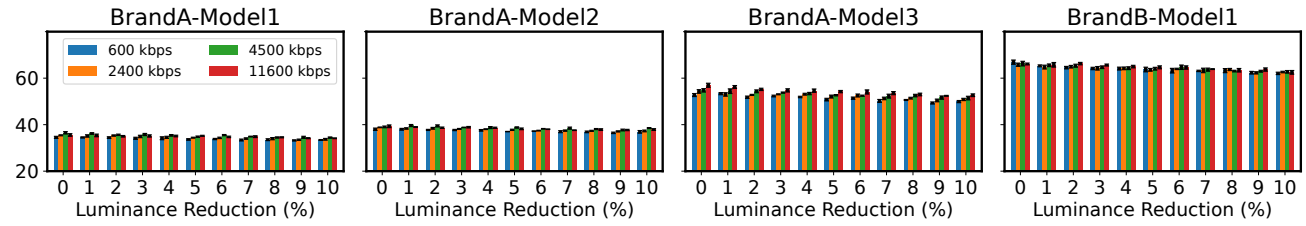
To avoid order bias, the sequence of video presentations was randomized. The video impairments varied across different clips to cover a broad range of scenarios potentially affecting perceived video quality. Each testing session was designed to last no longer than 30 minutes to avoid fatigue, which could skew the assessment results. A rehearsal session was included prior to actual testing, where participants practiced rating video clips using the provided scale; however, scores from this rehearsal were not recorded.

This methodological approach ensures that the subjective data collected are reflective of true viewer perceptions, providing valuable insights into the effectiveness of objective quality metrics and the viewer's experience under varying brightness conditions.

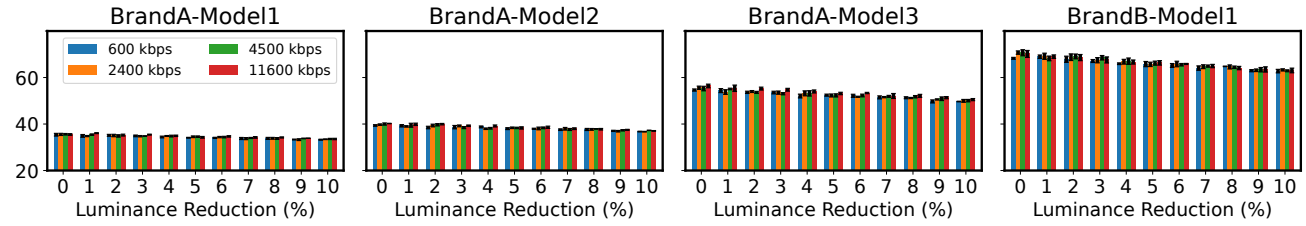
4.4 Energy Consumption Measurement

We set up a measurement configuration as illustrated in Fig. 4. This configuration includes a 4-way Power Meter (NETIO PowerPDU 4KS[4]) along with an Android TV measurement application built on ExoPlayer[5]—a widely used player found in numerous applications, including YouTube and Netflix [15]. Our measurement application plays videos on the connected devices (i.e., smart TVs), concurrently retrieves energy measurement data through the JSON API provided by the power meter, and generates a measurement data log on a per-second basis. This log captures not only energy consumption figures but also player metrics such as player state, playback position, media URLs, and more, which are recorded for further analysis.

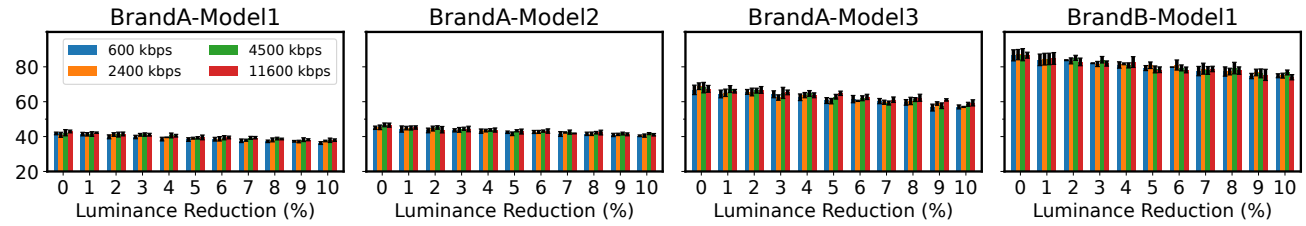
[2]https://mango.blender.org/download/
[3]https://developer.apple.com/documentation/http-live-streaming/hls-authoring-specification-for-apple-devices
[4]https://www.netio-products.com/en/device/powerpdu-4ks. Accessed 09 January 2025.
[5]https://github.com/google/ExoPlayer. Accessed 09 January 2025



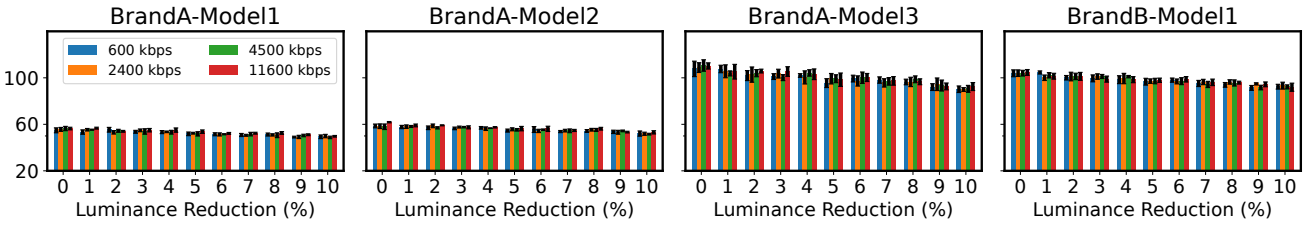
(a) Gameplay_60



(b) Tos_80



(c) Timescapes_100



(d) Tos_120

Figure 5: Average power (W) of different devices for each test sequence. The error bars show the standard deviation for five runs. All subplots in the same row share a legend on the left side. The title of each subplot indicates the TV model.

We employed four OLED TV models as detailed in Table 2, all set to 50% brightness. The experiment was conducted five times to ensure accuracy.

5 Test Results and Dataset

5.1 Energy Consumption

Fig. 5 illustrates the average power consumption of the tested OLED TVs with the test sequences across various bitrates/resolutions and levels of brightness reduction. It is evident that the influence of bitrate and video resolution is minimal. In most instances, a higher bitrate/resolution results in a slight increase in power load or energy

Table 2: Tested TV information. The brand and model are anonymized.

Branch	Model	Screen Size (inches)	Energy class	Power (W)	Release Year
A	1	42	G	51.9	2023
A	2	42	G	52.0	2024
A	3	55	A	111.0	2019
B	1	55	B	131.0	2019

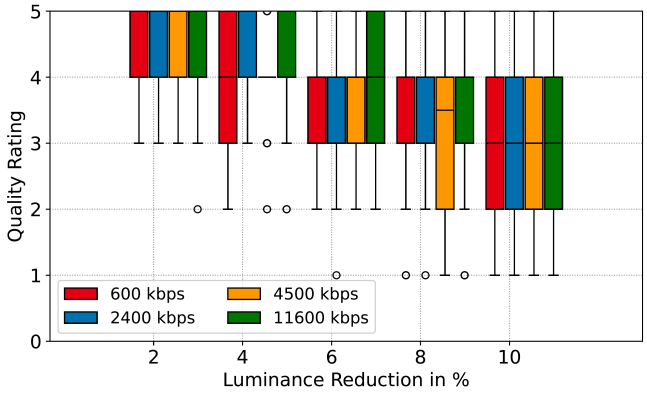


Figure 6: Quality ratings for different brightness reduction and video bitrates for all videos.

consumption. For instance, BrandA-Model1 TV playing original tos_120 consumes 54.8 W, 55.6 W, 56.4 W, and 56.4 W at bitrates of 600 kbps, 2400 kbps, 4500 kbps, and 11600 kbps, respectively. This indicates an energy increase of only about 2.9% when the TV switches from playing the video at 600 kbps@540p to 11600 kbps@2160p.

Conversely, video brightness significantly impacts the energy consumption of OLED TVs. BrandB-Model1 TV playing the original tos_120 video consumes 104.6 W at a bitrate of 11600 kbps (see Fig. 5 (d)). When the brightness of this sequence is reduced by 5% and 10%, the power load drops to 97.9 W and 91.9 W, respectively, translating to energy savings of 6.4% and 12.1%.

5.2 Subjective Test

5.2.1 Participants. A total of 28 participants attended the subjective test, including 24 males, and 4 females. 5 participants were in the age group from 18 to 24 years, 13 were between 25 and 34, 4 were between 34 and 44, 4 were between 45 and 54, and 2 were between 55 and 64.

5.2.2 Opinion Score. Fig. 6 illustrates the quality ratings associated with various levels of brightness reduction and video bitrates. It is evident that the quality ratings remain relatively stable when participants view videos with a specific brightness reduction across different bitrates. For instance, with a 2% reduction in brightness, the lowest rating scores for all bitrates are identical at 3, and 75% of the participants rated these bitrates between 4 and 5. Similarly, at a 10% brightness reduction, the pattern persists, as the box plot for the analyzed bitrates shows a lower quartile of 2, an upper quartile of 4, a minimum score of 1, and a maximum score of 5.

5.3 Objective Metrics vs. Subjective Metric

Fig. 7 visualizes the correlation of considered objective metrics with MOS. It can be seen that each objective metric (*i.e.*, PSNR, SSIM, and VMAF) has a different correlation with the MOS where the PSNR has the most positive correlation while VMAF has the worst one. It is reasonable because VMAF relies on a machine learning model trained with artifacts caused by the encoding process. Thus, VMAF cannot reflect the MOS of the viewer for darkened videos.

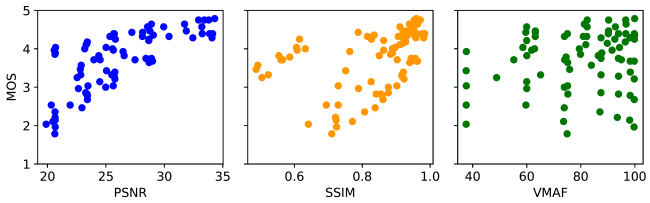


Figure 7: Scatter plot of objective metrics and MOS

5.4 Applications and Usage Scenarios

Our open dataset can benefit diverse research directions that explore the QoE of end users and energy consumption of OLED TVs in various video brightness. This includes, but is not limited to, the following applications:

- The impaired videos can be used to conduct further tests for energy consumption of other TV models and other display technologies.
- These videos can also be used for subjective tests to evaluate the perceptual quality of these videos on various TVs.
- The energy measurement together with video characteristics and TV models can be used to build energy consumption prediction models/digital twin to estimate the energy consumption of different TVs.
- The rating scores collected from participants with a wide variety of backgrounds (*i.e.*, age, gender) can be used to train and validate QoE prediction models.

6 Conclusions and Future Work

In this study, we introduced the OLED-EQ dataset, a pioneering resource aimed at assessing energy consumption and video quality in OLED TVs under varying brightness levels. Our comprehensive experiments demonstrated that reducing video brightness significantly decreases energy consumption while maintaining acceptable perceptual quality, highlighting the importance of video brightness management in energy-efficient streaming solutions. The dataset comprises energy consumption data from four different OLED TV models while playing 176 videos, alongside subjective quality ratings from participants, making it a valuable asset for future research in the fields of adaptive video streaming and green computing.

Future work will focus on several key areas. First, we aim to expand the OLED-EQ dataset by including additional OLED TV models and content types to enhance its applicability across a broader range of devices and scenarios. Second, we plan to investigate energy consumption and quality assessment in other display technologies, such as QLED and LCD, to provide a comparative understanding of energy efficiency across various platforms. Finally, we will develop advanced Quality of Experience (QoE) prediction models that leverage the dataset to aid in optimizing streaming protocols and enhancing user satisfaction.

References

[1] FFmpeg. [Online] Available: https://ffmpeg.org. Accessed 04 April 2024.

[2] Abdelhak Bentaleb, Bayan Taani, Ali C Begen, Christian Timmerer, and Roger Zimmermann. A Survey on Bitrate Adaptation Schemes for Streaming Media Over HTTP. *IEEE Communications Surveys & Tutorials*, 21(1):562–585, 2019.

[3] Taieb Chachou, Wassim Hamidouche, Sid Ahmed Fezza, and Ghalem Belalem. Energy Consumption and Carbon Emissions of Modern Software Video Encoders. *IEEE Consumer Electronics Magazine*, 2023.

[4] Barry Elad. Television Statistics By Revenue, Delivery Platform, Sales Channel Share, Time Spent, Vendor, Brands, Users And Market Share. [Online] Available: https://www.coolest-gadgets.com/television-statistics/. Accessed: 10 September 2024.

[5] Christian Herglotz, Werner Robitza, Matthias Kränzler, Andre Kaup, and Alexander Raake. Modeling of Energy Consumption and Streaming Video QoE using a Crowdsourcing Dataset. In *2022 14th International Conference on Quality of Multimedia Experience (QoMEX)*, pages 1–6, 2022.

[6] Alain Hore and Djemel Ziou. Image Quality Metrics: PSNR vs. SSIM. In *2010 20th International Conference on Pattern Recognition*, pages 2366–2369. IEEE, 2010.

[7] T.Y. Huang, R. Johari, N. McKeown, M. Trunnell, and M. Watson. A Buffer-Based Approach to Rate Adaptation: Evidence From a Large Video Streaming Service. In *ACM SIGCOMM Computer Communication Review*, volume 44, pages 187–198. ACM, 2014. https://doi.org/10.1145/2619239.2626296.

[8] ITU. Methodologies for the Subjective Assessment of the Quality of Television Images, Recommendation ITU-R BT. 500-15 (05/2023), 2023.

[9] LG. LG's OLED Leadership Holds Firm in First Half of 2023. [Online] Available: https://www.lg.com/global/newsroom/lg-story/beyond-news/lgs-oled-leadership-holds-firm-in-first-half-of-2023/: :text=OLED Accessed: 10 September 2024.

[10] Daniele Lorenzi, Minh Nguyen, Farzad Tashtarian, and Christian Timmerer. E-WISH: An Energy-aware ABR Algorithm For Green HTTP Adaptive Video Streaming. In *Proceedings of the 3rd Mile-High Video Conference*, pages 28–33, 2024.

[11] P-VP Clément Louis-Philippe, Quentin ES Jacquemotte, and Lorenz M Hilty. Sources of Variation in Life Cycle Assessments of Smartphones and Tablet Computers. *Environmental Impact Assessment Review*, 84:106416, 2020.

[12] Vignesh V Menon, Reza Farahani, Prajit T Rajendran, Samira Afzal, Klaus Schoeffmann, and Christian Timmerer. Energy-Efficient Multi-Codec Bitrate-Ladder Estimation for Adaptive Video Streaming. In *2023 IEEE International Conference on Visual Communications and Image Processing (VCIP)*, pages 1–5. IEEE, 2023.

[13] Vignesh V Menon, Christian Feldmann, Hadi Amirpour, Mohammad Ghanbari, and Christian Timmerer. VCA: Video Complexity Analyzer. In *Proceedings of the 13th ACM multimedia systems conference*, pages 259–264, 2022.

[14] Vignesh V Menon, Amritha Premkumar, Prajit T Rajendran, Adam Wieckowski, Benjamin Bross, Christian Timmerer, and Detlev Marpe. Energy-efficient Adaptive Video Streaming with Latency-Aware Dynamic Resolution Encoding. In *Proceedings of the 3rd Mile-High Video Conference*, pages 21–27, 2024.

[15] Chayabanjonglerd Minseo. Playing Video by ExoPlayer. [Online] Available: https://medium.com/fungjai/playing-video-by-exoplayer-b97903be0b33. Accessed: 09 April 2024.

[16] Joanna Moulierac, Guillaume Urvoy-Keller, Marco Dinuzzi, and Zhejiayu Ma. *What is the Carbon Footprint of One Hour of Video Streaming?* PhD thesis, Université Côte d'Azur, 2023.

[17] Minh Nguyen, Ekrem Çetinkaya, Hermann Hellwagner, and Christian Timmerer. Super-Resolution Based Bitrate Adaptation for HTTP Adaptive Streaming for Mobile Devices. In *Proceedings of the 1st Mile-High Video Conference*, pages 70–76, 2022.

[18] Minh Nguyen, Görkem Güclü, Martin Lasak, Robert Seeliger, Stefan Arbanowski, and Stephan Steglich. Impact of Video Luminance on Perceptual Quality and Energy Consumption in Video Streaming. In *2024 Sixteenth International Conference on Quality of Multimedia Experience (QoMEX)*. IEEE, 2024.

[19] Minh Nguyen, Ekrem Çetinkaya, Hermann Hellwagner, and Christian Timmerer. WISH: User-centric Bitrate Adaptation for HTTP Adaptive Streaming on Mobile Devices. In *2021 IEEE 23rd International Workshop on Multimedia Signal Processing (MMSP)*, pages 1–6, 2021.

[20] Tania Pouli, Khaoula Ounajim, and Daniel Menard. Evaluating the Energy Impact of Different Dimensions of UHD. In *2024 IBC*, 2024.

[21] Reza Rassool. VMAF Reproducibility: Validating a Perceptual Practical Video Quality Metric. In *2017 IEEE international symposium on broadband multimedia systems and broadcasting (BMSB)*, pages 1–2. IEEE, 2017.

[22] Precedence Research. OLED Market Size, Share, and Trends 2024 to 2034. [Online] Available: https://www.precedenceresearch.com/oled-market. Accessed: 10 January 2025.

[23] Sandvine. Global Internet Phenomena Report 2024. [Online] Available: https://www.sandvine.com/phenomena. Accessed: 02 April, 2024.

[24] Umme Sara, Morium Akter, and Mohammad Shorif Uddin. Image Quality Assessment through FSIM, SSIM, MSE and PSNR – A Comparative Study. *Journal of Computer and Communications*, 7(3):8–18, 2019.

[25] Robert Seeliger. Decoding Green Streaming – Part II of III Evaluation of Measurements. https://websites.fraunhofer.de/video-dev/decoding-green-streaming-part-ii-of-iii-evaluation-of-measurements/. Accessed: 10 April 2024.

[26] K. Spiteri, R. Urgaonkar, and R. K. Sitaraman. BOLA: Near-Optimal Bitrate Adaptation for Online Videos. In *IEEE INFOCOM 2016-The 35th Annual IEEE Int'l. Conf. on Computer Communications*. IEEE, 2016.

[27] Andie Stephens, Chloe Tremlett-Williams, Liam Fitzpatrick, Luca Acerini, Matt Anderson, and Noor Crabbendam. Carbon Impact of Video Streaming. [Online] Available: https://www.carbontrust.com/our-work-and-impact/guides-reports-and-tools/carbon-impact-of-video-streaming. Accessed: 10 September 2024.

[28] Michela Testolina, Evgeniy Upenik, João Ascenso, Fernando Pereira, and Touradj Ebrahimi. Performance Evaluation of Objective Image Quality Metrics on Conventional and Learning-Based Compression Artifacts. In *2021 13th International Conference on Quality of Multimedia Experience (QoMEX)*, pages 109–114. IEEE, 2021.

[29] T. C. Thang, H. T. Le, A. T. Pham, and Y. M. Ro. An Evaluation of Bitrate Adaptation Methods for HTTP Live Streaming. *IEEE Journal on Selected Areas in Communications*, 32(4):693–705, 2014. https://doi.org/10.1109/JSAC.2014.140403.

[30] Ramona Trestian, Arghir-Nicolae Moldovan, Olga Ormond, and Gabriel-Miro Muntean. Energy Consumption Analysis of Video Streaming to Android Mobile Devices. In *2012 IEEE Network Operations and Management Symposium*, pages 444–452. IEEE, 2012.

[31] Bekir Oguzhan Turkkan, Ting Dai, Adithya Raman, Tevfik Kosar, Changyou Chen, Muhammed Bulut, Jaroslav Zola, and Daby Sow. GreenABR+: Generalized Energy-Aware Adaptive Bitrate Streaming. *ACM Transactions on Multimedia Computing, Communications and Applications*, 20(9):1–24, 2024.

[32] Bekir Oguzhan Turkkan, Ting Dai, Adithya Raman, Tevfik Kosar, Changyou Chen, Muhammed Fatih Bulut, Jaroslaw Zola, and Daby Sow. GreenABR: Energy-Aware Adaptive Bitrate Streaming with Deep Reinforcement Learning. In *Proceedings of the 13th ACM Multimedia Systems Conference*, pages 150–163, 2022.

[33] Sam Van Damme, Maria Torres Vega, and Filip De Turck. A Full-and No-Reference Metrics Accuracy Analysis for Volumetric Media Streaming. In *2021 13th International Conference on Quality of Multimedia Experience (QoMEX)*, pages 225–230. IEEE, 2021.

[34] Benoy Varghese, Guillaume Jourjon, Kanchana Thilakarathne, and Aruna Seneviratne. E-DASH: Modelling an Energy-Aware DASH Player. In *2017 IEEE 18th International Symposium on A World of Wireless, Mobile and Multimedia Networks (WoWMoM)*, pages 1–9, 2017.

[35] Abhinau K Venkataramanan, Cosmin Stejerean, Ioannis Katsavounidis, and Alan C Bovik. One Transform To Compute Them All: Efficient Fusion-Based Full-Reference Video Quality Assessment. *IEEE Transactions on Image Processing*, 2023.

[36] Zhou Wang, Alan C Bovik, Hamid R Sheikh, and Eero P Simoncelli. Image Quality Assessment: from Error Visibility to Structural Similarity. *IEEE transactions on image processing*, 13(4):600–612, 2004.

[37] Zhisheng Yan and Chang Wen Chen. RnB: Rate and Brightness Adaptation for Rate-Distortion-Energy Tradeoff in HTTP Adaptive Streaming over Mobile Devices. In *Proceedings of the 22nd Annual International Conference on Mobile Computing and Networking*, pages 308–319, 2016.

[38] Kun Yin, En-Lin Hsiang, Junyu Zou, Yannanqi Li, Zhiyong Yang, Qian Yang, Po-Cheng Lai, Chih-Lung Lin, and Shin-Tson Wu. Advanced Liquid Crystal Devices for Augmented Reality and Virtual Reality Displays: Principles and Applications. *Light: Science & Applications*, 11(1):161, 2022.

[39] Chaoqun Yue, Subhabrata Sen, Bing Wang, Yanyuan Qin, and Feng Qian. Energy Considerations for ABR Video Streaming to Smartphones: Measurements, Models and Insights. In *Proceedings of the 11th ACM Multimedia Systems Conference*, pages 153–165, 2020.

HockeyRink: A Dataset for Precise Ice Hockey Rink Keypoint Mapping and Analytics

Mehdi Houshmand Sarkhoosh
OsloMet & Forzasys
Oslo, Norway

Sushant Gautam
SimulaMet & OsloMet
Oslo, Norway

Cise Midoglu
Forzasys
Oslo, Norway

Saeed Shafiee Sabet
Forzasys
Oslo, Norway

Tomas Kupka
Forzasys
Oslo, Norway

Pål Halvorsen
SimulaMet, OsloMet & Forzasys
Oslo, Norway

Abstract

Precise mapping of ice hockey rinks is critical for applications such as player tracking, game strategy analysis, and broadcast enhancements. Traditional methods often rely on manual annotations or simplistic models that fail to account for the rink's complex geometry and dynamic in-game conditions. To address these limitations, we present HockeyRink, a novel dataset comprising 56 meticulously annotated keypoints corresponding to significant landmarks on a standard hockey rink, including face-off dots, goalposts, and blue lines. Leveraging the YOLOv8-Large pose estimation architecture, we adapted the model to treat the rink as a single 'pose' object, enabling accurate keypoint predictions tailored to the nuances of hockey rink imagery. Our dataset, derived from diverse hockey game footage from the Swedish Hockey League (SHL), facilitates applications in homography estimation, 2D/3D scene mapping, and tactical overlays. By detecting and mapping rink keypoints, one can compute precise transformations between the image plane and the rink's physical dimensions, enabling the overlay of player trajectories and tactical insights onto broadcast footage. This work addresses the unique challenges of ice hockey, such as occlusions, rapid camera movements, and varying lighting conditions, and offers a robust foundation for future research and innovation in sports analytics. The HockeyRink dataset and trained model are openly available at: https://huggingface.co/SimulaMet-HOST/HockeyRink.

CCS Concepts

• **Information systems**; • **Computing methodologies** → **Machine translation**; **Computer vision**; Scene understanding; **Video summarization**; *Activity recognition and understanding*; **Visual inspection**; • **Human-centered computing** → *Visualization*;

Keywords

AI, computer vision, ice hockey, gameplay analysis, homography, player tracking, summarization, 2D mapping

Permission to make digital or hard copies of all or part of this work for personal or classroom use is granted without fee provided that copies are not made or distributed for profit or commercial advantage and that copies bear this notice and the full citation on the first page. Copyrights for components of this work owned by others than the author(s) must be honored. Abstracting with credit is permitted. To copy otherwise, or republish, to post on servers or to redistribute to lists, requires prior specific permission and/or a fee. Request permissions from permissions@acm.org.
MMSys '25, March 31-April 4, 2025, Stellenbosch, South Africa
© 2025 Copyright held by the owner/author(s). Publication rights licensed to ACM.
ACM ISBN 979-8-4007-1467-2/2025/03
https://doi.org/10.1145/3712676.3718338

ACM Reference Format:
Mehdi Houshmand Sarkhoosh, Sushant Gautam, Cise Midoglu, Saeed Shafiee Sabet, Tomas Kupka, and Pål Halvorsen. 2025. HockeyRink: A Dataset for Precise Ice Hockey Rink Keypoint Mapping and Analytics. In *ACM Multimedia Systems Conference 2025 (MMSys '25), March 31-April 4, 2025, Stellenbosch, South Africa.* ACM, New York, NY, USA, 7 pages. https://doi.org/10.1145/3712676.3718338

1 Introduction

Ice hockey analytics presents unique technical challenges due to the sport's intrinsic complexity, rapidly evolving gameplay dynamics, and the demanding requirements for precise spatial tracking. While computer vision and machine learning have revolutionized sports analytics across many domains, ice hockey remains distinctly challenging due to its fast-paced nature, frequent player occlusions, variable rink geometries, and dynamic camera movements. A fundamental prerequisite for advanced hockey analytics is the accurate mapping and registration of the rink's spatial features, yet existing approaches often rely on simplistic models or manual annotations that fail to capture the sport's intricate geometric relationships.

Adding to these challenges is the variability in rink dimensions and markings across different leagues and regions. The International Ice Hockey Federation (IIHF) [17] specifies rinks measuring 60 meters by 30 meters for international competitions and European leagues (see Figure 1), while the National Hockey League (NHL) [27] employs smaller rinks of 200 feet by 85 feet (61 meters by 26 meters) for faster-paced North American hockey. Even within leagues like the SHL [20], which traditionally follows IIHF standards, some arenas have adopted hybrid dimensions with reduced widths of 28-29 meters to balance technical and physical gameplay aspects. This variability extends to critical features such as neutral zones, blue line placement, and goal crease design, posing additional challenges for computer vision applications as models trained on one league's data may not generalize effectively to others.

To address these limitations, we introduce HockeyRink, a comprehensive dataset comprising 661 high-resolution frames annotated with 56 precisely labeled keypoints corresponding to critical rink landmarks such as face-off dots, goal posts, blue lines, and zone boundaries (see Figure 2). The dataset leverages broadcast footage from the SHL, incorporating diverse game situations, lighting conditions, and camera angles to ensure robust real-world applicability. Each frame is annotated with visibility flags to handle partial occlusions and out-of-frame keypoints, with an average of 13.36 visible keypoints per image and 8,884 unique annotations across the dataset.

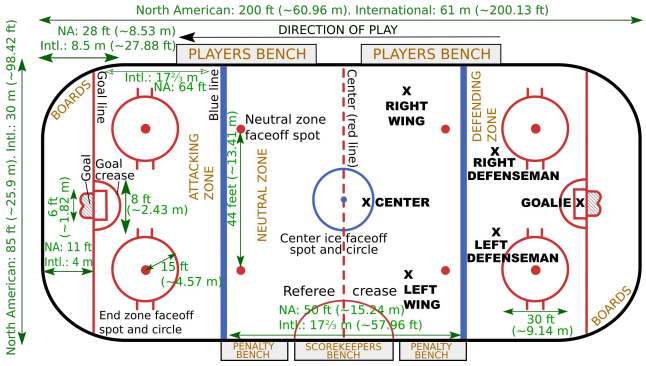


Figure 1: Standard hockey rink dimensions as defined by IIHF (60m x 30m) [17].

Our primary contributions include:

(1) A novel dataset specifically designed for ice hockey rink registration and mapping, featuring dense keypoint annotations that capture the sport's unique geometric properties
(2) A trained YOLOv8-Large pose estimation [39] architecture adapted to treat the rink as a single 'pose' object, enabling accurate keypoint predictions
(3) A publicly available Hugging Face Space that enables real-time inference on HockeyRink, allowing users to visualize predictions and interact with the model without local setup, facilitating experimentation and usability.
(4) Comprehensive evaluation metrics and benchmarks for rink keypoint detection

The HockeyRink dataset enables multiple downstream applications, from automated tracking systems and tactical overlay generation to immersive broadcast experiences. By providing a robust foundation for precise rink mapping, our work facilitates advances in player trajectory analysis, team formation assessment, and automated game statistics generation. The dataset and trained model are openly available to foster further research and innovation in ice hockey analytics.

2 Background and Related Work

The convergence of computer vision and sports analytics has driven substantial progress in player tracking, scene understanding, and performance analysis [3, 12, 38]. While sports like soccer have benefited from extensive datasets and established methodologies [33], ice hockey poses unique challenges due to its rapid gameplay, frequent occlusions, and complex rink geometry.

2.1 Open Datasets for Sports Analytics

Datasets are foundational to advancements in sports analytics. Soccer analytics, for example, has leveraged resources like TS-WorldCup [6] and SoccerNet [7], which provide extensive keypoint mappings and annotated frames for field registration and player tracking. These datasets have facilitated robust computer vision solutions, including homography estimation and camera calibration.

In contrast, the development of datasets for ice hockey analytics has been limited. Existing datasets, such as the HockeyPlayer Dataset [15], primarily focus on specific tasks like player detection, while others, like the dataset presented by Fani et al. [11], are designed for basic rink localization. However, these datasets lack comprehensive annotated keypoints necessary for precise mapping of hockey rinks. The HockeyRink dataset introduced in this study bridges this gap by providing 56 annotated keypoints corresponding to significant landmarks on the rink, such as face-off dots, goalposts, and blue lines. This dataset, derived from diverse SHL footage, is specifically designed to support homography estimation, 2D/3D scene mapping, and tactical overlays, enabling more accurate and insightful analysis.

2.2 Computer Vision for Sports Analytics

Recent trends in computer vision for sports analytics have significantly influenced ice hockey research [38, 40]. Unified pose estimation has gained traction[19, 25, 28], and in hockey the entire rink can be treated as a single 'pose' object, enabling the prediction of all keypoints simultaneously. This approach, exemplified by YOLOv8 [6, 25], improves coherence and efficiency in keypoint detection. Temporal modeling techniques, such as Temporal 1D Convolutional Networks and transformers, have also been applied effectively to analyze sequential data, enhancing player identification and event recognition [40].

The integration of object detection and tracking has further strengthened the field [4, 26]. For instance, ByteTrack [44] has been utilized to maintain reliable player trajectories even in the presence of occlusions and rapid movements [43]. Additionally, there has been a push toward real-time analytics, with models achieving immediate processing capabilities to provide tactical insights during live games [8, 45].

2.3 Challenges in Ice Hockey Analytics

Several methodologies have been employed for sports field registration, ranging from points-and-lines optimization [13, 14, 18] to instance segmentation [6, 16]. While these approaches have proven successful in sports like soccer, they face significant limitations when applied to ice hockey due to the unique characteristics of the sport. Ice hockey's rapid gameplay, with players reaching speeds of over 30 km/h and pucks traveling at more than 160 km/h [10], creates considerable motion blur, complicating object detection. Frequent player clustering during face-offs and board battles introduces additional occlusion challenges, which traditional tracking algorithms struggle to overcome [23]. Furthermore, dynamic camera movements, such as frequent zooming and panning to follow the action, make homography estimation particularly challenging.

Recent research has addressed some of these issues through innovative approaches. Prakash et al. [29] demonstrated improved player tracking by leveraging homographic projections. Similarly, Vats et al. [40] developed transformer-based approaches for player identification, which have proven effective in recognizing jersey numbers from broadcast footage. For rink localization, Fani et al. [11] introduced a ResNet18-based regression framework that achieved a mean Intersection over Union (IoU) of 94.6%. This method predicts four control points on the ice rink model but faces limitations in scenarios with zoomed-in footage where fewer control points are visible. Partial views of the rink remain a persistent challenge,

especially during dynamic game situations, highlighting the need for more robust solutions. Additionally, Lopes et al. [24] demonstrated that event detection systems could achieve an accuracy of 80% in identifying game elements, showing promise for automated analytics.

Despite these advances, current solutions often rely on frame-by-frame processing, which can lead to temporal inconsistencies and jittering in video sequences. Maintaining stable keypoint detection across consecutive frames, especially in the presence of occlusions and varying lighting conditions, remains an active area of research.

2.4 Research Gaps and Future Directions

Several challenges persist in ice hockey analytics, which future research must address:

- Lack of a comprehensive, publicly available dataset capturing rink keypoint detection and mapping under real-world conditions (e.g., occlusions, lighting variations, and different rink standards).
- Limited generalization of existing models across rink dimensions, league standards, and environmental variations [32].
- Absence of standardized evaluation metrics for rink registration accuracy [9].
- Challenges posed by ice surface reflections, varying lighting, and dynamic camera angles in detection tasks [24].
- Insufficient multi-modal data and temporal markers for long-term trajectory analysis and event prediction [31].

Future work should focus on developing diverse datasets, improving model robustness against occlusions and rapid movements, and integrating adaptive preprocessing techniques like exposure correction. Additionally, domain adaptation methods could enhance model generalization across sports and environments, bridging gaps in ice hockey analytics.

3 The HockeyRink Dataset

The HockeyRink constitutes a novel dataset comprising high-resolution frames extracted from SHL broadcast footage. The dataset encompasses 661 frames, each with spatial dimensions of 1920×1080 pixels, selected to represent diverse scenarios in ice hockey. Key features include:

- Diverse scenarios showcasing different actions in ice hockey.
- Team variability across multiple SHL venues and rink configurations.
- Dense keypoint annotations, particularly valuable in medium and zoomed-in shots where fewer rink points are visible.

3.1 Data Collection

Through collaboration with SHL and Sportality [34], we leveraged Forzasys' [1] API infrastructure to access game footage spanning three years (2022-2024). Source material was encoded using H.264 codec with spatial resolution of 1920×1080 pixels and temporal resolution of 50 fps. Frame preservation utilized JPEG compression with quality factor 90 to maintain fidelity while optimizing storage requirements.

Our sampling methodology involved selecting random games across seasons and extracting frames from 10-minute segments at 1 fps, yielding approximately 500 frames per segment. Manual inspection ensured diversity in scenarios, resulting in a final collection of 661 frames. This approach ensures compliance with SHL licensing by preventing video reconstruction from extracted frames.

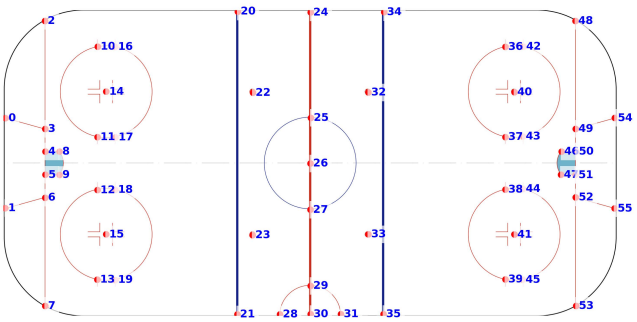


Figure 2: Annotated hockey rink showing 56 (0-55) labeled keypoints corresponding to significant landmarks. Keypoints are marked with red circles, and their indices are shown in blue.

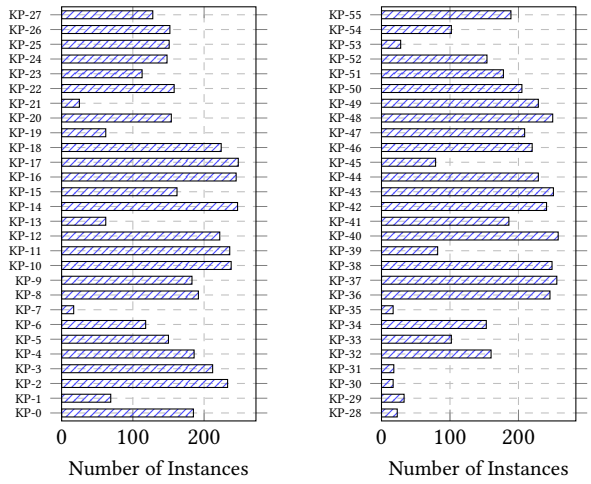


Figure 3: Keypoint distribution in the HockeyRink dataset.

3.2 Annotations

The annotation process employs Label Studio [36] with a specialized configuration for rink geometry capture. Each frame is annotated with a bounding box $\mathbf{B} \in \mathbb{R}^4$ encompassing the entire rink, followed by a set of 56 keypoints $\mathbf{K} = \{k_j\}_{j=0}^{55}$, where each keypoint contains specific (x, y) coordinates. Figure 3 depicts the keypoint distribution within the dataset. It reveals that although the majority of keypoints are adequately represented, a few such as KP-7 and KP-21, have fewer instances. This discrepancy arises from inherent limitations in broadcast footage, where some areas of the rink are shown less often because of camera positioning or close-up views.

The annotation schema follows YOLO pose estimation format:

$$\begin{aligned} \text{bbox} &= [c, x_c, y_c, w, h] \in \mathbb{N} \times [0,1]^4 \\ \text{keypoints} &= \{(x_i, y_i, v_i)\}_{i=0}^{55} \\ \text{where } v_i &\in \{0,1,2\} \end{aligned} \tag{1}$$

Here, `class_id` is 0 (denoting ice hockey rink), followed by normalized bounding box coordinates, and then the keypoint coordinates with their respective visibility flags. The visibility flag $v \in \{0,1,2\}$ indicates:

- `0`: Keypoint is not present in the frame
- 1: Keypoint is present but occluded (e.g., by players or equipment)
- 2: Keypoint is present and visible

A schematic representation of the annotation schema is shown below:

- Annotation Schema
  - Bounding Box: $[c, x_c, y_c, w, h]$
    - c: Class ID (e.g., $c = 0$ for hockey rink)
    - x_c, y_c: Normalized center coordinates
    - w, h: Normalized width and height
  - Keypoints: $\{(x_i, y_i, v_i)\}_{i=0}^{55}$
    - x_i, y_i: Normalized coordinates of keypoint i
    - v_i: Visibility flag
      - 0: Not present
      - 1: Occluded
      - 2: Visible

This approach ensures the capture of all keypoints, even when occluded, providing robust ground truth for pose estimation tasks.

3.3 Public Dataset

The HockeyRink dataset, hosted on Hugging Face, is a collection of annotated frames where each frame in the dataset is paired with a corresponding annotation file formatted according to the YOLO pose estimation model's dataset schema. The dataset is hierarchically organized, ensuring ease of access and usability for researchers and developers.

The directory structure begins with a top-level `SHL` folder containing two main subdirectories: `frames` and `annotations`. Each frame file in the `frames` directory is uniquely identified by an ID, which directly maps to its corresponding annotation file in the `annotations` directory. This one-to-one mapping facilitates seamless dataset management and enables users to easily split the dataset into training, validation, and test subsets according to their specific requirements. The directory structure schema is shown on Figure 4.

This structured design ensures compatibility with machine learning pipelines, making it straightforward to integrate the dataset for model training and evaluation.

3.4 Model Performance

To complement the HockeyRink dataset, we provide a trained YOLOv8-Large pose estimation model, openly available on Hugging Face[1]. The model treats each frame as a single object representing the ice hockey rink and its corresponding keypoints. The dataset

[1]https://huggingface.co/SimulaMet-HOST/HockeyRink

- Dataset
  - SHL
    - frames
      - <Unique_ID>.jpeg
      - ...
    - annotations
      - <Unique_ID>.txt
      - ...

Figure 4: HockeyRink directory structure, illustrating the hierarchical organization of frames and annotations. Each frame has a corresponding annotation file with the same unique identifier, facilitating seamless data pairing.

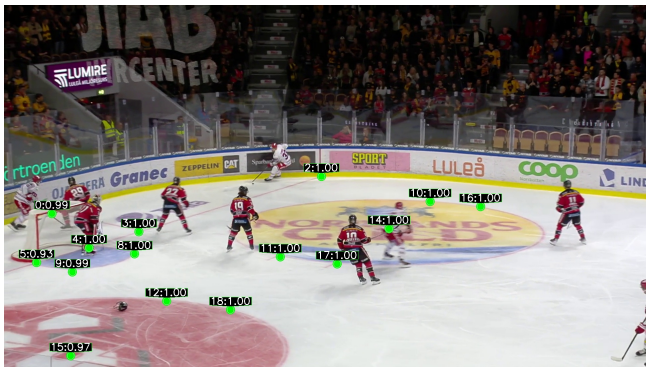


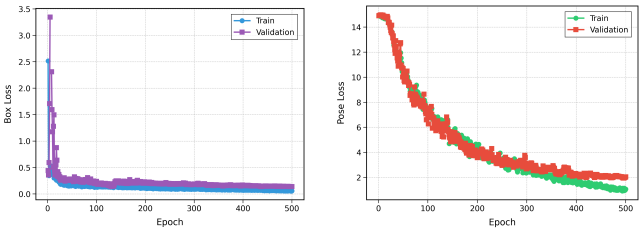
Figure 5: Visualization of model predictions for keypoint detection on an ice hockey rink with their class id and confidence score.

was split into training, validation, and test sets with an 80/10/10 ratio. Training was conducted for 500 epochs on 4 NVIDIA A100 GPUs (80GB each), completing in approximately 35 minutes. During training, GPU memory usage reached 25% per GPU, indicating efficient resource utilization. The predictions generated by the model on a sample from the test set are illustrated in Figure 5.

The model's performance was quantitatively evaluated using Precision (P), Recall (R), and Mean Average Precision (mAP) for both bounding box and keypoint detection. The results, presented in Table 1, demonstrate the model's strong ability to capture keypoints accurately. Additionally, system metrics were assessed to ensure usability across diverse hardware setups. The model achieved 13.64 FPS on a Tesla T4 GPU and 6.4 FPS on an M3 MacBook Pro, offering practical benchmarks for users deploying the model on different devices.

Table 1: Model Performance Metrics

Bounding Box Detection		Pose Keypoint Detection	
$mAP_{@0.5}$	99.50%	$mAP_{@0.5}$	97.48%
$mAP_{@0.5:0.95}$	99.48%	$mAP_{@0.5:0.95}$	76.45%
Precision	99.97%	Precision	96.21%
Recall	100.00%	Recall	96.24%



(a) Box Loss Curve **(b) Pose Loss Curve**

Figure 6: Loss curves for (a) box loss and (b) pose loss, illustrating the convergence of the model during training. Box loss represents the accuracy of bounding box predictions, while pose loss evaluates keypoint localization.

Key insights during training revealed that adjusting the weights for pose loss and keypoint objectness loss to optimal values of 15 and 25, respectively, significantly improved keypoint detection precision. Data augmentation techniques, including random flipping and rotation, did not yield noticeable performance gains, likely due to the dataset's inherent variability. The use of the visibility flag ($v_i \in \{0, 1, 2\}$) was particularly beneficial, enabling the model to account for occluded keypoints effectively. Figure 6 showcases the training loss curves, highlighting the optimization process.

Qualitatively, the model demonstrated robust performance under challenging conditions, such as occlusions caused by players or equipment, motion blur during fast-paced gameplay, and varying lighting conditions across rinks. These observations further validate the model's practical applicability to real-world scenarios. However, limitations remain; expanding the dataset to include scenarios with greater variability in lighting conditions and reflective surfaces could further enhance accuracy and generalization. Adding more diverse gameplay scenarios is a promising direction for future work.

4 Dataset Applications

The HockeyRink dataset provides a unique foundation for advancing research and applications in ice hockey analytics and beyond. Its detailed keypoint annotations of the rink's geometry enable precise spatial mapping, unlocking innovative solutions in computer vision, sports analytics, and immersive technologies. Below, we outline key applications enabled by the dataset.

4.1 Camera Calibration and Homography

The dense keypoint annotations in HockeyRink serve as ground truth for calibrating cameras used in hockey games. By aligning detected keypoints with their real-world counterparts, researchers can compute accurate homographies to transform 2D video frames into a bird's-eye view of the rink. This transformation is essential for reconstructing player and puck trajectories in a unified coordinate system [29]. Such capabilities can enhance automated tracking systems and improve multi-camera setups for comprehensive coverage of gameplay (see Figure 7).

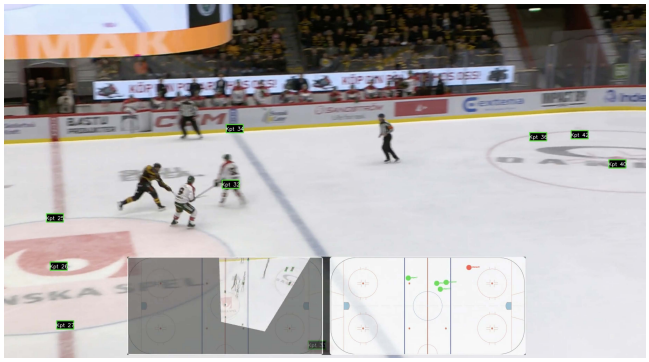


Figure 7: Homography transformation. the top-down perspective obtained through homography, mapping the keypoints to their real-world positions. This transformation enables accurate 2D spatial analysis of player and puck trajectories.

4.2 2D Mapping for Analytical Insights

By integrating object detection models with HockeyRink's keypoint annotations, all rink objects (e.g., players, pucks, referees) can be mapped onto a 2D radar view of the rink. This mapping provides a platform for advanced analytical insights, such as:

- **Player Formation Analysis:** Evaluating team formations during power plays or defensive setups [5].
- **Speed and Distance Metrics:** Calculating player and puck speeds, distances covered, and acceleration patterns [35].
- **Zone Utilization:** Analyzing puck possession and player activity in specific rink zones [30].

These insights empower coaches and analysts to refine tactics, assess player performance, and identify game trends.

4.3 Immersive AR and VR Applications

The precise mapping capabilities of HockeyRink enable immersive experiences in augmented reality (AR) and virtual reality (VR) [2]. By overlaying detected objects onto the rink's keypoint framework, it is possible to create interactive visualizations of gameplay in real-time. For instance:

- **AR Overlays:** Highlighting player positions, puck trajectories, and tactical zones directly on a live broadcast [42].
- **VR Simulations:** Recreating game scenarios for training or fan engagement, allowing users to observe plays from any perspective [22].

These technologies bridge the gap between on-ice action and digital interaction, offering new ways to engage fans and train athletes.

4.4 Foundation for Advanced Research

Beyond direct applications, the HockeyRink dataset provides a benchmark for exploring fundamental research challenges, such as:

- **Dynamic Object Tracking:** Enhancing multi-object tracking systems by incorporating spatial constraints defined by rink keypoints [21].

- **Biomechanics and Motion Analysis:** Studying player movements in relation to rink features, enabling insights into athletic performance [37].
- **Domain Generalization:** Applying lessons learned from hockey rink mapping to other sports fields with fixed geometries, such as soccer or basketball [41].

5 Discussion

The dynamic nature of ice hockey introduces several challenges for computer vision systems, even with a well-annotated dataset like HockeyRink. While our work addresses key aspects of rink mapping, several limitations and edge cases highlight areas for future exploration.

One significant challenge lies in handling extreme occlusions caused by players, equipment, or fast-moving pucks. Although visibility flags in the dataset account for partially or fully occluded keypoints, real-world scenarios often involve prolonged occlusions or scenarios where keypoints become indistinguishable. Developing models that can leverage temporal data across video sequences, rather than relying solely on single-frame inputs, could improve robustness under such conditions.

Lighting variability across different arenas also poses a critical challenge. The reflective nature of the ice surface and dynamic lighting conditions during games can introduce significant noise in keypoint detection. While the dataset includes diverse lighting scenarios, augmenting the dataset with synthetic data to simulate extreme conditions—such as glare, shadows, or low-contrast regions—could enhance model generalization. Additionally, incorporating adaptive preprocessing techniques, such as exposure correction or domain-specific normalization, could mitigate lighting inconsistencies.

Edge cases, such as frames captured during abrupt camera transitions, introduce another layer of complexity. These transitions often distort spatial relationships within the frame, making homography estimation unreliable. Future work could explore leveraging multi-camera systems or integrating inertial sensors to provide supplementary geometric constraints during transitions.

The absence of temporal annotations in HockeyRink limits its applicability for event-based analytics, such as goal detection or power-play analysis. Expanding the dataset to include temporal markers and synchronized multi-frame sequences could unlock applications in predictive modeling and long-term trajectory analysis. Furthermore, extending the dataset to incorporate multi-modal data, such as player and puck tracking, would enable comprehensive scene understanding and facilitate advanced analytics.

From a methodological perspective, the exclusion of data augmentation, while contextually justified, restricts the exploration of synthetic diversity in training. Investigating augmentation techniques, such as geometric transformations or generative models for simulated frames, could reveal insights into the balance between dataset realism and diversity. Similarly, exploring domain adaptation techniques to generalize the model for other sports or varying rink geometries could broaden its applicability.

Additionally, real-time deployment of HockeyRink requires addressing practical system-level constraints. While the YOLOv8-based model achieves competitive inference speeds (13.64 FPS on a Tesla T4 and 6.4 FPS on an M3 MacBook Pro), deploying it for live analytics necessitates further optimizations. Techniques such as model distillation, edge inference deployment, and latency-aware processing could enhance real-time applicability. Future work should explore these optimizations to facilitate seamless integration with existing sports analytics pipelines.

Finally, scalability remains an open question. While HockeyRink demonstrates feasibility within the constraints of SHL games, its performance across leagues with differing rink dimensions, camera setups, and gameplay styles remains untested. Creating benchmarks for cross-league performance or standardizing datasets for different leagues could help establish a unified framework for rink mapping in ice hockey.

In summary, while HockeyRink provides a strong foundation for precise rink mapping, addressing these challenges and edge cases will be essential for realizing its full potential in sports analytics. Future work should prioritize robustness under different conditions, integration with complementary data sources, and the exploration of new methodologies to expand its impact.

6 Conclusion

We presented HockeyRink, a novel dataset designed to advance the precision of ice hockey rink mapping. Comprising 661 high-resolution images annotated with 56 keypoints representing significant landmarks, this dataset addresses challenges unique to ice hockey, such as occlusions, rapid gameplay, and varying rink geometries. Using the **YOLOv8l-pose** architecture, we developed a custom model tailored to hockey rink imagery, enabling applications in homography estimation, 2D/3D scene reconstruction, and player tracking. Our work establishes a strong foundation for the integration of precise spatial analytics into sports broadcasting, coaching, and performance analysis. By enabling seamless integration into machine learning pipelines, **HockeyRink** supports researchers and developers in creating robust, scalable solutions for ice hockey and beyond. Looking ahead, we envision this dataset serving as a cornerstone for future innovations, including multi-modal data fusion, domain adaptation for other sports, and integration with wearable technologies. By addressing critical gaps in data availability and leveraging state-of-the-art methodologies, HockeyRink contributes to the growing field of computer vision in sports analytics and paves the way for transformative advancements in underexplored domains like ice hockey.

Acknowledgments

This research was funded by the Research Council of Norway, projects #354154 (Smart Sports Studio) and #346671 (AI-Storyteller). The authors wish to thank the Swedish Hockey League (SHL) for making frames from the videos available for the research.

References

[1] Forzasys AS. 2024. Forzasys DIGITALIZING SPORTS. https://forzasys.com/.

[2] Chris Ashraf. 2021. Verizon tests how 5G & edge compute can bring hockey fans real-time stats in arenas. https://www.verizon.com/about/news/verizon-5g-edge-compute-hockey-real-time-stats [Online; accessed 16. Jan. 2025].

[3] Sian Barris and Chris Button. 2008. A Review of Vision-Based Motion Analysis in Sport. *Sports Med.* 38, 12 (Dec. 2008), 1025–1043. https://doi.org/10.2165/00007256-200838120-00006

[4] Matija Buric, Marina Ivasic-Kos, and Miran Pobar. 2019. Player Tracking in Sports Videos. , 11–13 pages. https://doi.org/10.1109/CloudCom.2019.00058

[5] Qiao Chen and Charalambos Poullis. 2023. Tracking and Identification of Ice Hockey Players. , 3–16 pages. https://doi.org/10.1007/978-3-031-44137-0_1

[6] Yen-Jui Chu, Jheng-Wei Su, Kai-Wen Hsiao, Chi-Yu Lien, Shu-Ho Fan, and Min-Chun Hu. 2022. Sports Field Registration via Keypoints-aware Label Condition. , 19–20 pages. https://doi.org/10.1109/CVPRW56347.2022.00396

[7] Anthony Cioppa, Adrien Deliège, Floriane Magera, Silvio Giancola, Olivier Barnich, Bernard Ghanem, and Marc Van Droogenbroeck. 2021. Camera Calibration and Player Localization in SoccerNet-v2 and Investigation of their Representations for Action Spotting. https://doi.org/10.48550/arXiv.2104.09333 arXiv:2104.09333

[8] Alec Cook and Oktay Karakuş. 2024. LLM-Commentator: Novel fine-tuning strategies of large language models for automatic commentary generation using football event data. *Knowledge-Based Systems* 300 (Sept. 2024), 112219. https://doi.org/10.1016/j.knosys.2024.112219

[9] Jesse Davis, Lotte Bransen, Laurens Devos, Arne Jaspers, Wannes Meert, Pieter Robberechts, Jan Van Haaren, and Maaike Van Roy. 2024. Methodology and evaluation in sports analytics: challenges, approaches, and lessons learned. *Mach. Learn.* 113, 9 (Sept. 2024), 6977–7010. https://doi.org/10.1007/s10994-024-06585-0

[10] Xin Duan. 2011. *Automatic determination of puck possession and location in broadcast hockey video.* Ph. D. Dissertation. University of British Columbia. https://doi.org/10.14288/1.0052111

[11] Mehrnaz Fani, Pascale Berunelle Walters, David A. Clausi, John Zelek, and Alexander Wong. 2021. Localization of Ice-Rink for Broadcast Hockey Videos. https://doi.org/10.48550/arXiv.2104.10847 arXiv:2104.10847

[12] Indrajeet Ghosh, Sreenivasan Ramasamy Ramamurthy, Avijoy Chakma, and Nirmalya Roy. 2023. Sports analytics review: Artificial intelligence applications, emerging technologies, and algorithmic perspective. *WIREs Data Min. Knowl. Discovery* 13, 5 (Sept. 2023), e1496. https://doi.org/10.1002/widm.1496

[13] Marc Gutiérrez-Pérez and Antonio Agudo. 2024. Pnlcalib: Sports Field Registration Via Points and Lines Optimization. https://doi.org/10.2139/ssrn.4998149 [Online; accessed 9. Jan. 2025].

[14] Marc Gutiérrez-Pérez and Antonio Agudo. 2024. No Bells Just Whistles: Sports Field Registration by Leveraging Geometric Properties. , 3325-3334 pages.

[15] HockeyDetection. 2024. HockeyPlayer Dataset. https://universe.roboflow.com/hockeydetection/hockeyplayer. visited on 2025-01-09.

[16] Namdar Homayounfar, Sanja Fidler, and Raquel Urtasun. 2017. Sports Field Localization via Deep Structured Models. , 4012-4020 pages. https://doi.org/10.1109/CVPR.2017.427

[17] International Ice Hockey Federation. 2024. IIHF Official Rulebook 2024/25 – Version 1.0, May 2024. https://blob.iihf.com/iihf-media/iihfmvc/media/contentimages/4_sport/officiating/rule_book/24_25/2024_iihf_rulebook_24052024_v1.pdf. Accessed: January 8, 2025.

[18] Igor Jánoš and Wanda Benesova. 2024. Football Playing Field Registration Using Distance Maps. , 106-110 pages. https://doi.org/10.1109/ICFSP62546.2024.10785382

[19] Donghyun Kim, Kaihong Wang, Kate Saenko, Margrit Betke, and Stan Sclaroff. 2022. A Unified Framework for Domain Adaptive Pose Estimation. In *Computer Vision – ECCV 2022*. Springer, Cham, Switzerland, 603–620. https://doi.org/10.1007/978-3-031-19827-4_35

[20] Professional Swedish Hockey Leauge. 2025. https://www.shl.se/.

[21] Hui Li, Haoran Yang, Xiaoxue Ai, Zhong Chen, and Yanli Wu. 2024. A Dynamic 3D Multi-Object Tracking Method Based on Spatiotemporal Features. , 17 pages. https://doi.org/10.1109/TIV.2024.3508743

[22] Tica Lin, Yalong Yang, Johanna Beyer, and Hanspeter Pfister. 2020. SportsXR – Immersive Analytics in Sports. https://doi.org/10.48550/arXiv.2004.08010 arXiv:2004.08010

[23] Jingbin Liu, Gege Huang, Juha Hyyppä, Jing Li, Xiaodong Gong, and Xiaofan Jiang. 2023. A survey on location and motion tracking technologies, methodologies and applications in precision sports. *Expert Syst. Appl.* 229 (Nov. 2023), 120492. https://doi.org/10.1016/j.eswa.2023.120492

[24] Jorge Miguel Lopes, Luis Paulo Mota, Samuel Marques Mota, José Manuel Torres, Rui Silva Moreira, Christophe Soares, Ivo Pereira, Feliz Ribeiro Gouveia, and Pedro Sobral. 2024. Object and Event Detection Pipeline for Rink Hockey Games. *Future Internet* 16, 6 (May 2024), 179. https://doi.org/10.3390/fi16060179

[25] Debapriya Maji, Soyeb Nagori, Manu Mathew, and Deepak Poddar. 2022. YOLO-Pose: Enhancing YOLO for Multi Person Pose Estimation Using Object Keypoint Similarity Loss. https://doi.org/10.48550/arXiv.2204.06806 arXiv:2204.06806

[26] Banoth Thulasya Naik, Mohammad Farukh Hashmi, and Neeraj Dhanraj Bokde. 2022. A Comprehensive Review of Computer Vision in Sports: Open Issues, Future Trends and Research Directions. *Appl. Sci.* 12, 9 (April 2022), 4429. https://doi.org/10.3390/app12094429

[27] National Hockey League. 2024. National Hockey League Official Rules 2024-2025. https://media.d3.nhle.com/image/private/t_document/prd/yikcdsxofkmgsrhjl3di.pdf. Accessed: 2025-01-16.

[28] Xiaohan Nie, Shixing Chen, and Raffay Hamid. 2021. A Robust and Efficient Framework for Sports-Field Registration. , 1936-1944 pages.

[29] Harish Prakash, Jia Cheng Shang, Ken M. Nsiempba, Yuhao Chen, David A. Clausi, and John S. Zelek. 2024. Multi Player Tracking in Ice Hockey with Homographic Projections. https://doi.org/10.48550/arXiv.2405.13397 arXiv:2405.13397

[30] David Radke, Jaxin Lu, Jackson Woloschuk, Tin Le, Daniel Radke, Charlie Liu, and Tim Brecht. 2023. Analyzing Passing Metrics in Ice Hockey using Puck and Player Tracking Data. , 25–39 pages.

[31] DT Radke, DL Radke, Tim Brecht, and Alex Pawelczyk. 2021. Passing and pressure metrics in ice hockey.

[32] Onni Siivonen. 2024. Predicting player performance in National Hockey League using data-analytics and advanced metrics. https://lutpub.lut.fi/handle/10024/168197

[33] Mohieddine Amine Silvio Giancola. 2018. SoccerNet: A Scalable Dataset for Action Spotting in Soccer Videos. , 1792–179210 pages. https://doi.org/10.1109/cvprw.2018.00223

[34] Sportality. 2024. Sportality - Web publishing for sports. https://www.sportality.se/.

[35] Petr Stastny, Martin Musalek, Robert Roczniok, Daniel Cleather, Dominik Novak, and Michal Vagner. 2023. Testing distance characteristics and reference values for ice-hockey straight sprint speed and acceleration. A systematic review and meta-analyses. , 899–918 pages. https://doi.org/10.5114/biolsport.2023.122479

[36] Label Studio. 2025. https://labelstud.io/.

[37] Getu Teferi and Dessalew Endalew. 2020. Methods of Biomechanical Performance Analyses in Sport: Systematic Review. , 47–52 pages. https://doi.org/10.12691/ajssm-8-2-2

[38] Graham Thomas, Rikke Gade, Thomas B. Moeslund, Peter Carr, and Adrian Hilton. 2017. Computer vision for sports: Current applications and research topics. *Comput. Vision Image Understanding* 159 (June 2017), 3–18. https://doi.org/10.1016/j.cviu.2017.04.011

[39] Rejin Varghese and M. Sambath. 2024. YOLOv8: A Novel Object Detection Algorithm with Enhanced Performance and Robustness. , 18–19 pages. https://doi.org/10.1109/ADICS58448.2024.10533619

[40] Kanav Vats, Pascale Walters, Mehrnaz Fani, David A. Clausi, and John S. Zelek. 2023. Player tracking and identification in ice hockey. *Expert Syst. Appl.* 213 (March 2023), 119250. https://doi.org/10.1016/j.eswa.2022.119250

[41] Wikipedia contributors. 2024. Sports biomechanics — Wikipedia, The Free Encyclopedia. https://en.wikipedia.org/w/index.php?title=Sports_biomechanics&oldid=1220010153 [Online; accessed 16-January-2025].

[42] Greg Wyshynski. 2022. The NHL's tech-heavy plan to draw in young fans. https://www.espn.com/nhl/story/_/id/33803500/the-nhl-tech-heavy-plan-draw-young-fans [Online; accessed 16. Jan. 2025].

[43] Yifu Zhang, Peize Sun, Yi Jiang, Dongdong Yu, Fucheng Weng, Zehuan Yuan, Ping Luo, Wenyu Liu, and Xinggang Wang. 2022. ByteTrack: Multi-object Tracking by Associating Every Detection Box. In *Computer Vision – ECCV 2022*. Springer, Cham, Switzerland, 1–21. https://doi.org/10.1007/978-3-031-20047-2_1

[44] Yifu Zhang, Peize Sun, Yi Jiang, Dongdong Yu, Zehuan Yuan, Ping Luo, Wenyu Liu, and Xinggang Wang. 2021. ByteTrack: Multi-Object Tracking by Associating Every Detection Box. arXiv:2110.06864 https://arxiv.org/abs/2110.06864

[45] Zhixing Zhou. 2024. Real-Time Decision Modeling of Basketball Game Tactics Based on Video Analytics. *jes* 20, 7s (May 2024), 3304–3315. https://journal.esrgroups.org/jes/article/view/4166



VV-DASH: A Framework for Volumetric Video DASH Streaming

Hadi Heidarirad
Department of Computer Science
University of Calgary
Calgary, Alberta, Canada
hadi.heidarirad@ucalgary.ca

Mea Wang
Department of Computer Science
University of Calgary
Calgary, Alberta, Canada
meawang@ucalgary.ca

Abstract

With the increasing demand for immersive experiences, volumetric video has emerged as a critical technology, offering users six degrees of freedom (6DoF) to fully explore three-dimensional scenes. However, despite significant advancements, there remains a lack of a comprehensive and flexible adaptive streaming framework capable of delivering volumetric video over dynamic network conditions. To address this gap, we present VV-DASH, an end-to-end framework for adaptive volumetric video streaming over DASH (Dynamic Adaptive Streaming over HTTP). Our framework covers the entire streaming pipeline, from video source to video playback. We propose a codec-agnostic DASH Volumetric Video (DVV) segment format that consolidates compressed video content into DASH-ready segments. This segmentation improves achievable streaming throughput by 13.2%, effectively reduces bandwidth demand, and enhances the achievable streaming bitrate by up to 37.8%. In summary, VV-DASH provides a practical, high-performance framework for scalable and adaptive volumetric video streaming.

CCS Concepts

• **Information systems** → **Multimedia streaming**.

Keywords

Volumetric Video, DASH, Point Cloud, Streaming

ACM Reference Format:
Hadi Heidarirad and Mea Wang. 2025. VV-DASH: A Framework for Volumetric Video DASH Streaming. In *Proceedings of the 16th ACM Multimedia Systems Conference (MMSys '25), March 31–April 4, 2025, Stellenbosch, South Africa.* ACM, New York, NY, USA, 7 pages. https://doi.org/10.1145/3712676.3718339

1 Introduction

While video streaming has become the primary form for multimedia entertainment (*e.g.,* YouTube [6] and Netflix [23]), demand for immersive and interactive experiences is growing alongside increasing computing and networking capacity. In volumetric video, dynamic objects and scenes are captured using multiple cameras from different angles to represent them in true 3D, offering users six degrees of freedom (6DoF) within the video scene. Unlike 2D frames in conventional videos, volumetric video frames are formed by 3D Meshes or Point Clouds, significantly increasing size and bitrate. Point clouds are more widely used and studied as they are easier to process and manipulate (*e.g.,* better suited for tiling complex scenes [39]), thus, we resort to point cloud format in this paper.

To accommodate varying network conditions and the wide range of bitrates provided by compression algorithms, Dynamic HTTP Adaptive Streaming (DASH) [33] and HTTP Live Streaming (HLS) [25] are among the widely adopted adaptive streaming protocols, utilized by major streaming service providers such as YouTube and Netflix. They are designed to stream video content segment by segment at appropriate quality levels—characterized by a bitrate ladder—subject to the network dynamics. The demand for volumetric streaming has drawn research attention to making it available over DASH for quality adaptation over general network [12, 18, 26, 40–42]. The research challenges have been primarily concerning the trade-off between fast compression/decompression and efficient bitrate for transmission. These challenges motivate the proposal of VV-DASH, an end-to-end framework for advancing the design and analysis of volumetric streaming systems.

In this paper, we propose VV-DASH[1], a highly-efficient DASH framework for volumetric video streaming. The modular design of VV-DASH spans the entire streaming pipeline, from video source to transmission and rendering. The framework is codec-agnostic, with encoder and decoder wrappers enabling integration of any codec. Along with VV-DASH, we introduce a segmentation method for volumetric video content that enables efficient streaming and decoding. Our evaluation results show that the segmentation method effectively improves the achievable streaming bitrate by 37.8% on average, and the decoding speed by up to 51%.

The rest of this paper is organized as follows. Section 2 provides an overview of the background knowledge on volumetric video streaming and related work. Section 3 introduces the codec-agnostic design of VV-DASH, followed by implementation and experiment setup in Section 4. We then present two case studies to showcase that VV-DASH can support streaming with Draco (Section 5) and V-PCC (Section 6). Finally, Section 7 presents the concluding remarks.

2 Background and Related Work

DASH is known for quality adaptation across various network conditions [32]. End-to-end DASH streaming involves the following steps on the server side: (1) compressing the original video into multiple quality levels based on a predefined bitrate ladder (henceforth, *compression*), (2) packaging the compressed videos into segments and preparing a manifest file (*i.e.,* MPD) containing metadata and segment URLs, and (3) serving these segments from an HTTP server that responds to segment requests via URLs.

[1] https://github.com/Rad6/vv-dash

Permission to make digital or hard copies of all or part of this work for personal or classroom use is granted without fee provided that copies are not made or distributed for profit or commercial advantage and that copies bear this notice and the full citation on the first page. Copyrights for components of this work owned by others than the author(s) must be honored. Abstracting with credit is permitted. To copy otherwise, or republish, to post on servers or to redistribute to lists, requires prior specific permission and/or a fee. Request permissions from permissions@acm.org.
MMSys '25, March 31–April 4, 2025, Stellenbosch, South Africa
© 2025 Copyright held by the owner/author(s). Publication rights licensed to ACM.
ACM ISBN 979-8-4007-1467-2/25/03
https://doi.org/10.1145/3712676.3718339

Though there are server-side quality adaptation proposals [44], the most common DASH approach is client-based adaptation, allowing the use of ordinary HTTP servers ([15, 16, 19, 20] for conventional DASH and [13, 28] for volumetric DASH). Thus, a DASH streaming involves the following steps on client end: (1) Receiving the MPD file from the server and preparing the streaming accordingly. The bitrate ladder can be fetched from the MPD file. (2) Requesting the next segment at a specific quality level selected by the adaptation algorithm (according to observed/estimated bandwidth and buffer health). Quality adaptation may depend on playback buffer state [15], the estimated bandwidth [19], or a combination of factors (including, but not limited to buffer status, bandwidth estimation, codec property [16, 20]). (3) Receiving the segment and queuing it into the buffer, which will be dequeued by the decoder. (4) Decoding the segment and then rendering for playback. (5) Repeat Steps 2-4 until the end of the streaming session (either the user terminates the streaming or the end of the video has reached).

ViVo [13] has been one of the pioneering systems for volumetric video streaming on mobile devices. However, the system lacks support for a DASH implementation and does not support encoders other than Draco, limiting the range of encoded bitrates available to accommodate wider network conditions. Away from the V-PCC and Draco codecs, Kyungjin *et al.* targeted mobile point-cloud streaming and proposed a novel data structure called the Parallel Decodable Tree (PD-Tree) [17], which differs from traditional structures like Octree and KD-Tree. The GROOT framework, proposed based on this data structure, can achieve real-time decoding through parallel processing. These proposals are aiming to transform specific encodings of volumetric video for streaming, which are orthogonal work to VV-DASH.

Closely related to VV-DASH, we found `vvtk` [34], an open-source framework consisting of individual modules for encoding, decoding, rendering, and transmission. However, these modules are not streamlined to form an end-to-end streaming system. Moreover, `vvtk` provides only a simulation of DASH, without using an actual manifest (MPD) file, making it not suitable for conducting realistic research on volumetric DASH streaming. To the best of our knowledge, VV-DASH is the first general end-to-end framework that can work with different codecs and bitrate reduction algorithms, as well as efficient encoding and decoding processes. In particular, we propose a codec-agnostic DVV container format that wraps around volumetric video data (encoded in any format) along with information necessary for decoding and playback of DASH segments. Any existing or new video compression techniques can be incorporated into VV-DASH, as exemplified by the two case studies in the paper.

3 Design of VV-DASH

This section presents the design of VV-DASH. As shown in Figure 1, VV-DASH spans the entire streaming pipeline, from video source to rendering for playback. The video source module supports Video-on-Demand. The Encoder module prepares videos at multiple quality levels based on a bitrate ladder. The DASH Volumetric Video (DVV) Segmenter bundles encoded media content with metadata into DVV segments, akin to video segments in conventional DASH streaming. These segments are packaged by a DASH Packager into DASH segments along with an MPD file and stored on an HTTP server.

On the client side, VV-DASH is adaptable by any DASH client, allowing integration with any bitrate adaptation algorithm to select, request, and receive segments based on buffer status and network conditions. VV-DASH wraps a DASH client with a DVV parser, a decoder module, and a renderer. The DVV parser extracts embedded media and playback information from DVV segments. These are passed to a codec-specific decoder module, which decodes the segments and places them into a playback buffer for the renderer module to dequeue and render. VV-DASH also supports headless delivery of media content to external modules, enabling rendering on diverse devices (e.g., head-mounted displays) or for objective quality assessment tools.

Overall, VV-DASH is a comprehensive end-to-end framework for DASH streaming of VoD voluemtric video. Since bottlenecks are potentially presented in every part of this pipeline, VV-DASH is designed to be lightweight and resource-efficient for optimal performance. Not only is VV-DASH a complete setup that facilitates research on any part of the pipeline individually or as a whole, but it is also designed to be modular and flexible, allowing seamless integration of DASH volumetric video streaming into most existing DASH players. The rest of this section details VV-DASH's server-side (Section 3.1) and client-side (Section 3.2) designs.

3.1 VV-DASH Server-side

The Encoder module in VV-DASH adopts any specific encoder (*e.g.,* Google Draco [7] and MPEG V-PCC [9]), and encodes raw volumetric video into multiple quality levels as specified by the bitrate ladder. The Encoder module can also adopt codec-independent bitrate reduction algorithms such as down-sampling [14, 30] for simpler processing.

Diverse bitrate reduction algorithms impose different DASH implementations. For instance, V-PCC provides its own segmentation format, while Draco encodes content frame by frame, which is inefficient for DASH streaming. For a 30 fps video, each frame represents only 0.033 seconds. Frame-level streaming results in frequent requesting and fetching, leading to high control and communication overhead, as well as inefficient decoding. This would make DASH streaming impractical. VV-DASH introduces a novel DVV Segmenter that transforms any coded volumetric video into DASH-ready segments.

The DVV Segmenter can aggregate discrete frames or codec-specific segments, or wrap only one codec-specific segment, along with the necessary presentation and playback information for DASH. This would not only enable more efficient streaming of independently encoded frames but would also allow the system to support the streaming of both the independently encoded frames (as in Draco or down-sampled raw frames) and regular media segments (such as V-PCC V3C) within the same system, using a unified segment format. To the best of our knowledge, no such segment format currently exists for either raw or encoded point cloud-based media. Therefore, we propose the DVV segment format packing one or more encoded media contents (*e.g.,* frames or segments) for a specific playback duration.

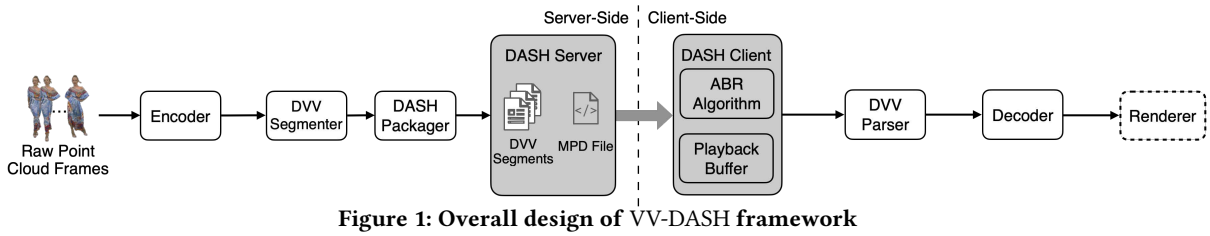


Figure 1: Overall design of VV-DASH framework

As shown in Figure 2, the DVV segment format bundles frames or codec-specific segments and includes essential information for decoding and playback. The bitstream starts with a 3-byte DVV format version and a 4-byte Header Encoding field (e.g., "JSON") to specify header serialization. A 4-byte field defines the header size, separating header and payload. A 2-byte Sequence Number field follows for error-checking, along with a 3-byte Timescale field specifying the segment's time units (e.g., 90,000 ticks per second). Before the header, a 4-byte Codec field identifies the codec (e.g., "DRCO" for Draco or "VPCC" for VPCC).

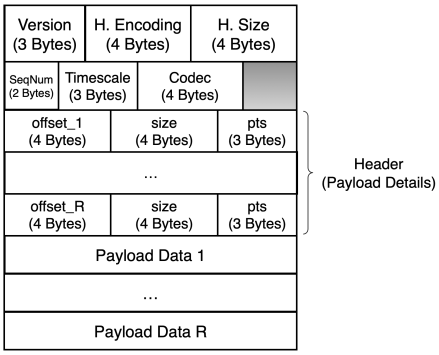


Figure 2: DVV segment format

The header contains metadata for each media payload, including a 4-byte offset for its position in the bitstream, a 4-byte size for its length, and a 3-byte PTS for its presentation timestamp (relative to the timescale). The payload section follows, containing the encoded data. This structured design ensures the DVV bitstream efficiently encapsulates both metadata and frame data, enabling precise access and synchronization during decoding and playback.

3.2 VV-DASH Client-side

The client side of VV-DASH builds on a typical DASH client, which streams DVV-formatted segments at appropriate quality levels. Upon receiving a segment, the DVV Parser identifies the codec and timescale, parses the header, and extracts encoded frames or codec-specific segments. These are passed to the Decoder module, which wraps the appropriate decoder (*e.g.,* Draco for `DRC` frames and V-PCC for `V3C` bitstream segments). The Decoder module uses parallel computing directives to accelerate decoding when possible. Since `DVV` payloads are independently decodable, VV-DASH employs multiprocessing on multi-core systems, with one decoder per payload, boosting the decoding rate. It also minimizes disk I/O by keeping the entire data flow in memory for further optimization.

Decoded frames are passed to the Renderer module for playback. The renderer retrieves frames from the buffer, loads the point cloud data and associated colors, and displays them in an `Open3D` window at the correct presentation timestamp. To the best of our knowledge, there is currently no dedicated rendering library for dynamic sequences of point clouds. Therefore, to ensure seamless playback, we update the points at their exact presentation timestamp in the same renderer window and display each frame for the appropriate duration of $1/fps$ seconds. To enhance usability, a walk-around view on 2D screens allows users to correct the 3D object in any direction using the cursor. Similar to the Decoder module, the Renderer module is designed to ensure efficient playback of video segments by utilizing asynchronous operations, multiprocessing, and concurrent programming techniques. Due to the lack of a real-time renderer for highly dense video datasets, VV-DASH also captures detailed analytical data from the streaming session (*e.g.,* stalls and quality switches from DASH) to aid research and examination regardless of the rendering being real-time. These analytical tools offer a practical playback alternative for dense volumetric video streaming until rendering techniques advance (or alternatively utilizing HMD-based native rendering)

4 Implementation and Experiment Setup

In VV-DASH, we use [2] as the DASH player, that implements a DASH protocol with bandwidth-based, buffer-based, and hybrid Adaptive Bitrate (ABR) algorithms. For our experiments, we utilized the bandwidth-based ABR.

The DASH player is wrapped with VV-DASH modules, as shown in Figure 1, and an NGINX [27] server serves as the HTTP server. To emulate bandwidth scenarios, we employ a network module using the Linux Traffic Control utility on a HAProxy [5] proxy. Positioned between the DASH server and player, the proxy throttles inbound and outbound traffic to accurately emulate desired bandwidth profiles. All components, including the server, client, and network emulator, are containerized and run on a laptop equipped with an Intel Core i9-14900HX CPU, NVIDIA GeForce RTX 4090 GPU, and 64GB of memory.

We evaluate VV-DASH using three point cloud-based volumetric video datasets: On the high quality end, we use the "Ricardo-10" video from the MVUB (Microsoft Voxelized Upper Bodies) dataset [21], featuring dense point clouds with an average of $1M$ points/frame and high spatial resolution of $1024x1024$ voxels per cube (*i.e.,* a spatial depth of 10). On the low quality end, we use the "Phil-9" video from the same dataset [21], with a sparse point cloud (average $334K$ points/frame) and lower spatial resolution ($512x512$ voxels/cube, spatial depth 9). Videos from the MVUB dataset have been used in proposals for volumetric streaming in [11, 24, 41]. In between, we use the "Longdress-10" video from the 8i Voxelized Full Bodies (8i VFB v2) dataset [4], with a relatively dense point cloud (average $834K$ points/frame) and high spatial resolution ($1024x1024$

voxels/cube, depth 10). Videos from the 8i VFB dataset are also widely used in volumetric streaming research [14, 17, 30]. All three videos have 30 fps, a common frame rate for smooth motion and detailed video playback. These datasets represent a good range of point-cloud densities and spatial depths, providing ample room for codec compression. Table 1 compares the average point cloud density and frame size (ranging from 7.43 to 21.05 *MB*). To ensure uniform test durations, each video is concatenated with itself to extend to 30 seconds.

Table 1: Video datasets

Video	Avg # of Points	Avg Frame Size	# of Frames
Ricardo-10	1.002M	21.05 *MB*	216
Longdress-10	834K	18.9 *MB*	300
Phil-9	334K	7.43 *MB*	245

We used the Open3D library [43] as the renderer for volumetric video playback, which is a widely used and powerful tool for 3D graphics visualization. We plan to extend VV-DASH's implementation with a Unity [36, 37] module to enable natural volumetric rendering in head-mounted displays like Meta Quest [22]. With this setup, we will present two case studies demonstrating VV-DASH's capability to support two codec standards: Draco (Section 5) and V-PCC (Section 6). Using bandwidth profiles covering respective bitrate ladders, we will evaluate performance improvements from DVV segmentation. Each section will conclude with results from active DASH sessions under different bandwidth profiles.

5 Case Study #1: Streaming with Draco

We encode all three videos with five rate parameter sets using Draco (via DracoPy [29]) to define a 5-level bitrate ladder, detailed in Table 2. Parameter settings are selected carefully to ensure smooth progression in bitrate and visual quality. Compression level (CL) values span the full range (1–10) with a step size of two. Position and texture quantization parameters (PQP and TQP), which control the number of bits for position and texture quantization, are optimized by analyzing commonly used position and color values across datasets and evaluating decoded frame quality. The bitrate ladder is shown in Table 2. The "Ricardo" video exhibits the widest bitrate range when encoded with Draco, while the "Phil" video achieves high compression with a smaller bitrate range. The "Longdress" video has the highest average bitrate.

Table 2: Draco bitrate ladder

Rate	CL	PQP	TQP	Average Bitrate (Mbps)		
				Longdress	Ricardo	Phil
R1	10	7	6	309.9	141.8	143.4
R2	8	9	8	431.1	276.6	197.5
R3	6	11	10	568.8	454.1	256.4
R4	4	13	12	749.7	838.0	348.4
R5	2	15	14	982.3	1175.7	453.0

Figure 3 illustrates server-side DVV segmentation, showing an example of encapsulating the first 30 frames of a video into a DVV segment. Each frame, independently encoded as a DRC-encoded frame, is placed in an individual payload within the DVV segment. For each DRC frame, the DVV header includes a record with its starting byte offset in the payload bitstream, its size, and its presentation timestamp (PTS) relative to the timescale. The PTS starts from the `SeqNum` (0 for the first frame) and increments by 3,000 per frame, as the timescale for this 1-second segment is 90,000.

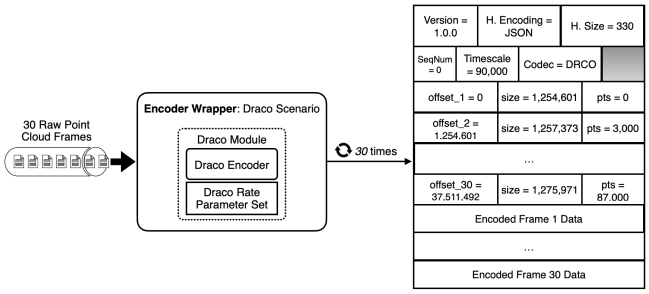


Figure 3: DVV segmentation and parallel encoding/decoding for Draco

Once a DVV segment containing Draco-encoded frames is received on the client side, frames are extracted based on the DVV header. Since each frame is independently decodable, they are decoded in parallel by separate instances of the Draco decoder. The original Draco decoder processes frames one at a time, requiring disk I/O for loading, decoding, and writing outputs, which introduces delays. To address this, we implement an in-memory Draco wrapper using DracoPy [29], allowing received and extracted frames to pass directly to decoders from memory. Decoded frames are then directly pushed into the renderer buffer, eliminating disk I/O delays entirely, as shown in Figure 3.

5.1 DVV Segmentation

The key enabling concept in VV-DASH for volumetric video streaming is DVV segmentation, which significantly enhances streaming performance. To create 1-second segments for testing videos, we package 30 frames into a DVV segment. DVV segmentation primarily improves average frame decoding time and system throughput. We conducted an experiment comparing decoding times for DVV segments and Draco frames. A DASH streaming session without bandwidth limits was run on the "Longdress" video to eliminate transmission bottlenecks and focus solely on decoding. For DVV segments, the DASH client sends all 30 frames in a segment to the Decoder module for parallel decoding. For Draco frames, a new decoding process is forked upon receiving each frame, maximizing parallelism for a fair comparison. As shown in Figure 4, DVV segmentation reduces average decoding time for 1-second of frames by up to 51%. This improvement is due to frames arriving in batches matching the frame rate (30 fps), enabling better CPU utilization. Process creation and scheduling overheads are minimized since 30 processes are launched simultaneously, rather than one every 0.033 seconds in the worst-case scenario for stall-free streaming.

Next, we compare the streaming experience using 1-second DVV segments versus individual frames (Draco default). With DVV segments, the DASH client requests a new segment at most every 1 second, provided segment transmission completes within this time

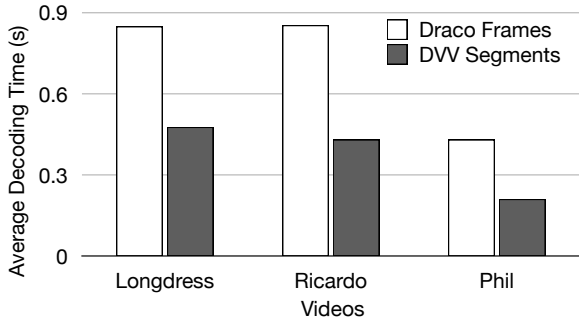


Figure 4: Impact of DVV segmentation on decoding time

for smooth streaming. Without segmentation, the player must request 30 frames consecutively within 1 second (*i.e.,* one request every 0.033 seconds at worst) to maintain the same quality level. To assess the impact of DVV segmentation, we created bandwidth profiles enabling each video to stream at specific quality levels. For example, streaming the "Longdress" video at quality level *R2* (as per Table 2) requires bandwidth between 431.1 *Mbps* and 568.8 *Mbps*. We selected 475 *Mbps*, ensuring the quality adaptation algorithm consistently picks *R2* for DVV segments. Similar bandwidth profiles were created for levels *R2*–*R5*, detailed in Table 3. We measured the average bitrate achieved during streaming for all three videos under these bandwidth profiles. Figure 5 shows that DVV segments consistently achieve the target quality level, whereas Draco frames mix the target and lower quality levels. Bitrates improved by 36.2%, 44.9%, and 32.4% for the "Longdress," "Ricardo," and "Phil" videos, respectively, leading to an overall throughput gain under identical network conditions.

Table 3: Bandwidth profiles for evaluating impact of DVV segmentation

Profiles	Bandwidth Setting (Mbps)		
	Longdress	Ricardo	Phil
L2	475	300	215
L3	610	500	270
L4	850	950	380
L5	1050	1280	490

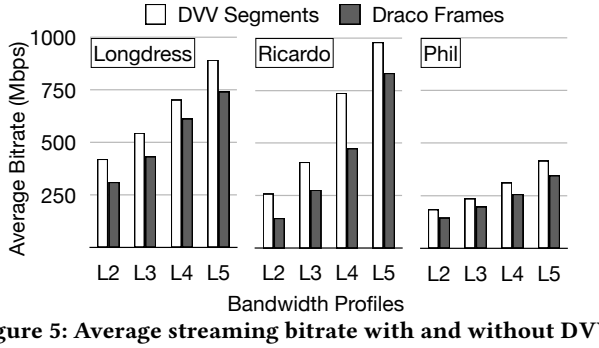


Figure 5: Average streaming bitrate with and without DVV segmentation

The bitrate gain from DVV segmentation is due to reduced protocol overhead. We analyzed the streaming timeline and plotted segment-by-segment and frame-by-frame transmissions for the "Longdress" video under a fixed bandwidth of 900 *Mbps* (maintaining quality level R4). As shown in Figure 6, the average turnaround time for requesting and downloading the first DVV segment is 0.91 seconds, while individually requesting and downloading 30 Draco frames from the same segment takes over 1 second. This pattern is consistent across other DVV segments and Draco frames of this video, explaining the 13.2% throughput gain. The improvement not only allows the player to switch to a higher quality level if close to its boundary—thereby improving the Quality-of-Experience under the same network conditions, but also helps prevent stalls or shorten stall duration under poor network conditions.

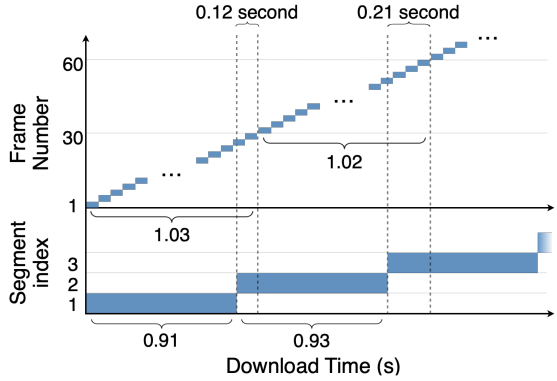


Figure 6: Segment-by-segment download time versus frame-by-frame download timeline from streaming "Longdress" video with 900*Mbps* fixed bandwidth

5.2 DASH with Draco

To demonstrate VV-DASH's adaptive bitrate streaming capability, we conducted experiments on the Draco-encoded "Longdress" video under two bandwidth profiles (as similar to [1]). The first profile, shown in Figure 7 (top), was designed for smooth, stall-free streaming with quality switches across all five quality levels. Bandwidth levels were adjusted every 5 seconds to enable DASH streaming at the marked quality levels. As shown in Figure 7 (top), the DASH session successfully adapted to the available bandwidth and selected appropriate quality levels.

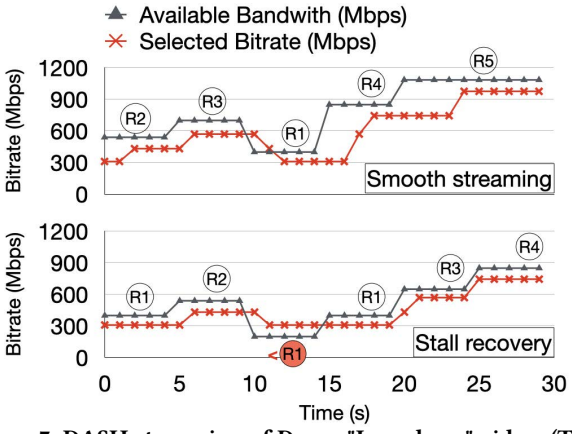


Figure 7: DASH streaming of Draco "Longdress" video: (Top) Smooth playback under sufficient bandwidth supply, (Bottom) Stall recovery after a bandwidth drop at 10 second.

The second profile, shown in Figure 7(bottom), included a stall period. Similar to the first profile, bandwidth levels were adjusted every 5 seconds, but at the 11th second, the bandwidth dropped below level R1. This drop lasted 5 seconds, exceeding the buffer

duration (4 seconds) and causing the buffer to deplete, resulting in playback stalls. As shown in Figure 7(bottom), the DASH session streamed at R1 for 9 consecutive segments following the bandwidth drop, with 1.32 seconds of stall observed during playback.

6 Case Study #2: Streaming with V-PCC

TMC2 is a reference implementation of the MPEG V-PCC codec standard [10], providing both encoder and decoder functionality. The encoder processes raw point cloud frames with a set of configurations, performing intra- and inter-frame 2D encoding, and outputs an encoded V3C bitstream. Key rate parameters include Geometry Quantization (GeoQP) and Attribute Quantization (AttQP), which control compression levels for geometric and attribute data, respectively—higher QP values yield greater compression and lower bitrates. Occupancy Precision (OP) defines the granularity of the occupancy map, with higher precision ensuring more accurate point capture within the voxel grid. For DASH streaming, we define a 5-level bitrate ladder. Rate configurations for V-PCC are based on recommended sample configurations from the TMC2 implementation [10], using the reference HM implementation [31, 35] as the 2D video encoder. As the original TMC2 decoder is slow (*e.g.,* taking over 40 seconds to decode a 1-second "Longdress" segment), we use the faster TMC2-RS [3], which is up to 15 times faster and has been adopted in frameworks like [34]. VV-DASH employs TMC2-RS as the V-PCC decoder. The GeoQP, AttQP, and OP settings for the bitrate ladder are detailed in Table 4.

Table 4: V-PCC bitrate ladder

Rate	GeoQP	AttQP	OP	Average Bitrate (Mbps)		
				Longdress	Ricardo	Phil
R1	32	42	4	4.35	1.84	1.84
R2	28	37	4	7.49	2.48	2.72
R3	24	32	4	13.1	3.68	4.48
R4	20	27	4	24.5	5.84	8.32
R5	16	22	2	44.2	10.4	15.3

Figure 8 illustrates the integration of the V-PCC encoder into VV-DASH and a sample crafted DVV segment. The DVV Segmenter can pack any number of V3C bitstreams (Figure 8 shows one example). In this case, since the only existing V3C bitstream bundles all 30 frames into a segment, the entire timescale is assigned to the segment's PTS, which is 90,000, equivalent to 1 second in length. Utilizing the fast TMC2-RS V-PCC implementation and configuring the DVV segments with 2 frames bundled in each V3C bitstream (adding up to 30 frames in total, which are packed into a payload to form a 30-frame, 1-second segment) with negligible compression ratio loss enables us to conduct a DASH streaming session for the "Phil" video. This segmentation preserves the compression ratio while enabling quick decoding of individual V3C bitstreams, making practical streaming possible.

6.1 DASH with V-PCC

Given the real-time decoding rate achieved for the "Phil" video, we are able to showcase DASH streaming with V-PCC. Similar to the Draco case, we created two bandwidth profiles: one for smooth streaming and one for stall recovery. The bandwidth profile along with the target quality levels is illustrated in Figure 9. Bandwidth

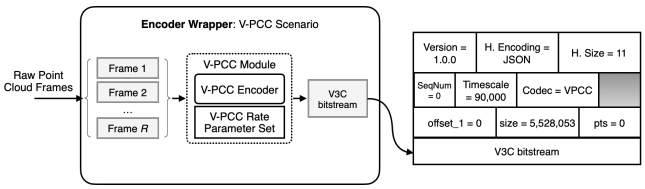


Figure 8: Parallel encoding/decoding and DVV segmentation for V-PCC

levels were adjusted every 5 seconds to enable DASH streaming at the quality levels shown in Figure 9. The DASH session with V-PCC, under the first profile, successfully adapted to the varying bandwidth conditions and selected the appropriate quality level accordingly. With the second profile, the DASH session streamed at the R1 level for four consecutive segments after the bandwidth drop, with stalls during playback.

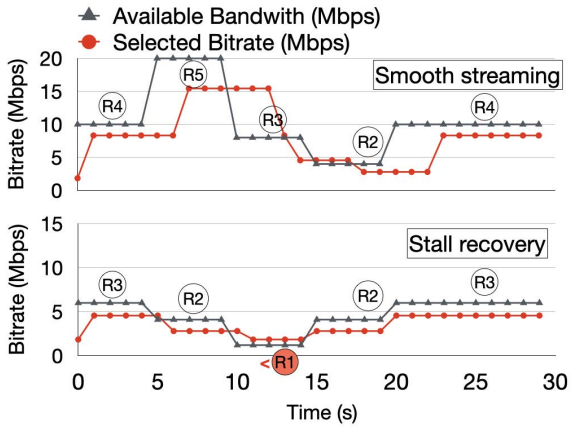


Figure 9: DASH streaming of V-PCC "Phil" video: (Top) Smooth playback under sufficient bandwidth supply, (Bottom) Stall recovery after a bandwidth drop at 10 second.

7 Conclusion

In this study, we proposed VV-DASH, a high-performance end-to-end framework for volumetric video streaming over DASH. VV-DASH improves performance through DVV segmentation, and efficient and parallel computing techniques. We presented two case studies: streaming with Draco and V-PCC, as well as demonstrating support for any bitrate reduction module and uniform packing of encoded/compressed video content. Consistent results in both studies confirm VV-DASH's effectiveness with DASH. We analyzed the impact of DVV segmentation on streaming throughput and decoding rate. Combining DVV segmentation with DASH results in up to 51% faster decoding rate Draco-encoded videos, and a 13.2% system throughput gain under the identical network conditions (*i.e.,* reducing bandwidth demands for smooth playback). In summary, VV-DASH enables practical adaptive volumetric video streaming and lays a foundation for and facilitates future research. While VV-DASH is a complete setup, future work includes containerizing DVV components into an ISOBMFF-compliant segment [8, 38] to facilitate broader modular adoption.

References

[1] Navid Akbari, Reza Hedayati Majdabadi, Akram Ansari, Mea Wang, and Diwakar Krishnamurthy. 2023. iStream: A Flexible Container-Based Testbed for Multimedia Streaming. In *2023 IEEE 6th International Conference on Multimedia Information Processing and Retrieval (MIPR)*. 1–6. https://doi.org/10.1109/MIPR59079.2023.00038

[2] Akram Ansari and Mea Wang. 2023. iStream Player: A Versatile Video Player Framework. In *Proceedings of the 33rd Workshop on Network and Operating System Support for Digital Audio and Video* (Vancouver, BC, Canada) *(NOSSDAV '23)*. Association for Computing Machinery, New York, NY, USA, 65–71. https://doi.org/10.1145/3592473.3592569

[3] B. Clement. 2022. Fast VPCC Point Cloud Decoder. https://github.com/benclmnt/tmc2-rs?tab=readme-ov-file

[4] Eugene d'Eon, Bob Harrison, Taos Myers, and Philip A Chou. 2017. 8i Voxelized Full Bodies-A Voxelized Point Cloud Dataset. *ISO/IEC JTC1/SC29 Joint WG11/WG1 (MPEG/JPEG) input document WG11M40059/WG1M74006* 7, 8 (2017), 11.

[5] HAProxy Enterprise. 2024. HAProxy. https://www.haproxy.com

[6] Google. 2024. https://youtube.com.

[7] Google. 2024. Draco: A Library for Compressing and Decompressing 3D Geometric Meshes and Point Clouds. https://github.com/google/draco

[8] MPEG Group. 2008. *Information Technology – Coding of Audio- Visual Objects – Part 12: ISO base media file format.* standard. ISO/IEC.

[9] MPEG Group. 2023. *Information Technology — Coded Representation of Immersive Media — Part 5: Visual Volumetric Video-based Coding (V3C) and Video-based Point Cloud Compression (V-PCC).* standard. ISO/IEC.

[10] MPEG Group. 2024. TMC2: Video Codec Based Point Cloud Compression (V-PCC) Test Model. https://github.com/MPEGGroup/mpeg-pcc-tmc2

[11] Shuai Gu, Junhui Hou, Huanqiang Zeng, Hui Yuan, and Kai-Kuang Ma. 2020. 3D Point Cloud Attribute Compression Using Geometry-Guided Sparse Representation. *IEEE Transactions on Image Processing* 29 (2020), 796–808. https://doi.org/10.1109/TIP.2019.2936738

[12] Srinivas Gudumasu, Gireg Maury, Ariel Glasroth, and Ahmed Hamza. 2023. Adaptive Streaming of Visual Volumetric Video-based Coding Media. In *Proceedings of the 15th International Workshop on Immersive Mixed and Virtual Environment Systems*. ACM, Vancouver, Canada, 30–33.

[13] Bo Han, Yu Liu, and Feng Qian. 2020. ViVo: Visibility-aware Mobile Volumetric Video Streaming. In *Proceedings of the 26th Annual International Conference on Mobile Computing and Networking* (London, United Kingdom) *(MobiCom '20)*. Association for Computing Machinery, New York, NY, USA, Article 11, 13 pages. https://doi.org/10.1145/3372224.3380888

[14] Mohammad Hosseini and Christian Timmerer. 2018. Dynamic Adaptive Point Cloud Streaming. In *Proceedings of the 23rd Packet Video Workshop*. IEEE, San Jose, CA, USA, 25–30.

[15] Te-Yuan Huang, Ramesh Johari, Nick McKeown, Matthew Trunnell, and Mark Watson. 2014. A Buffer-based Approach to Rate Adaptation: Evidence from a Large Video Streaming Service. In *Proceedings of the 2014 ACM conference on SIGCOMM*. ACM, Chicago, IL, USA, 187–198.

[16] Jonathan Kua, Grenville Armitage, and Philip Branch. 2017. A Survey of Rate Adaptation Techniques for Dynamic Adaptive Streaming over HTTP. *IEEE Communications Surveys & Tutorials* 19, 3 (2017), 1842–1866.

[17] Kyungjin Lee, Juheon Yi, Youngki Lee, Sunghyun Choi, and Young Min Kim. 2020. GROOT: A Real-time Streaming System of High-fidelity Volumetric Videos. In *Proceedings of the 26th Annual International Conference on Mobile Computing and Networking*. MobiCom, Virtual, 1–14.

[18] Jie Li, Cong Zhang, Zhi Liu, Richang Hong, and Han Hu. 2022. Optimal Volumetric Video Streaming With Hybrid Saliency based Tiling. *IEEE Transactions on Multimedia* 25 (2022), 2939–2953.

[19] Chenghao Liu, Imed Bouazizi, and Moncef Gabbouj. 2011. Rate Adaptation for Adaptive HTTP Streaming. In *Proceedings of the second annual ACM conference on Multimedia systems*. ACM, San Jose, CA, USA, 169–174.

[20] Chenghao Liu, Imed Bouazizi, Miska M Hannuksela, and Moncef Gabbouj. 2012. Rate Adaptation for Dynamic Adaptive Streaming Over HTTP in Content Distribution Network. *Signal Processing: Image Communication* 27, 4 (2012), 288–311.

[21] Charles Loop, Qin Cai, Sergio Orts Escolano, and Philip A. Chou. 2016. Microsoft Voxelized Upper Bodies - A Voxelized Point Cloud Dataset. http://plenodb.jpeg.org/pc/microsoft/

[22] "Meta". 2024. Meta Quest 3D. https://www.meta.com/ca/quest/quest-3s/.

[23] Netflix. 2024. https://netflix.com

[24] Dat Thanh Nguyen, Maurice Quach, Giuseppe Valenzise, and Pierre Duhamel. 2021. Lossless Coding of Point Cloud Geometry Using a Deep Generative Model. *IEEE Transactions on Circuits and Systems for Video Technology* 31, 12 (2021), 4617–4629. https://doi.org/10.1109/TCSVT.2021.3100279

[25] Roger Pantos and William May. 2017. HTTP Live Streaming.

[26] Feng Qian, Bo Han, Jarrell Pair, and Vijay Gopalakrishnan. 2019. Toward Practical Volumetric Video Streaming on Commodity Smartphones. In *Proceedings of the 20th International Workshop on Mobile Computing Systems and Applications*. ACM, CA, USA, 135–140.

[27] Will Reese. 2008. Nginx: The High-Performance Web Server and Reverse Proxy. *Linux Journal* 2008, 173 (2008), 2.

[28] Michael Rudolph and Amr Rizk. 2022. View-Adaptive Streaming of Point Cloud Scenes through Combined Decomposition and Video-based Coding. In *Proceedings of the 1st International Workshop on Advances in Point Cloud Compression, Processing and Analysis* (Lisboa, Portugal) *(APCCPA '22)*. Association for Computing Machinery, New York, NY, USA, 41–49. https://doi.org/10.1145/3552457.3555732

[29] seung lab. 2024. Python Wrapper for Google's Draco Mesh Compression Library. https://github.com/seung-lab/DracoPy

[30] Yuang Shi, Pranav Venkatram, Yifan Ding, and Wei Tsang Ooi. 2023. Enabling Low Bit-rate MPEG V-PCC-encoded Volumetric Video Streaming With 3D Subsampling. In *Proceedings of the 14th Conference on ACM Multimedia Systems*. ACM, Toronto, Canada, 108–118.

[31] HEVC Test Model (HM) Reference Software. 2013. The H.265 Reference Software HM. https://github.com/listenlink/HM

[32] Thomas Stockhammer. 2011. Dynamic Adaptive Streaming over HTTP– Standards and Design Principles. In *Proceedings of the second annual ACM conference on Multimedia systems*. ACM, San Jose, CA, USA, 133–144.

[33] T. Stockhammer. 2011. Dynamic Adaptive Streaming over HTTP: Standards and Design Principles. In *in Proc. of the 3rd ACM Multimedia Systems Conference (MMSys)*. ACM, New York, NY, USA, 133–144.

[34] NUS Volumetric Video Streams. 2022. A Toolkit for Volumetric Video Research. https://github.com/nus-vv-streams/vvtk/tree/main

[35] G. J. Sullivan, J. R. Ohm, W. J. Han, and T. Wiegand. 2012. Overview of the Hight Efficiency Video Coding (HEVC) Standard. *IEEE Transactions on Circuit and Systems for Video Technology* 22, 12 (Sept 2012), 1649–1668.

[36] Keijiro Takahashi. 2017. Pcx: Point Cloud Importer & Renderer for Unity. https://github.com/keijiro/Pcx

[37] Unity Technologies. 2024. Unity - Real-time Development Platform. https://unity.com. Accessed: 2024-11-23.

[38] Christian Timmerer and Christopher Müller. 2010. HTTP Streaming of MPEG Media. *Streaming Day* 10 (2010), 1–1.

[39] Jeroen van der Hooft, Hadi Amirpour, Maria Torres Vega, Yago Sanchez, Raimund Schatz, Thomas Schierl, and Christian Timmerer. 2023. A Tutorial on Immersive Video Delivery: From Omnidirectional Video to Holography. *IEEE Communications Surveys & Tutorials* 25, 2 (2023), 1336–1375. https://doi.org/10.1109/COMST.2023.3263252

[40] Yizong Wang, Dong Zhao, Huanhuan Zhang, Teng Gao, Zixuan Guo, Chenghao Huang, and Huadong Ma. 2024. Bandwidth-Efficient Mobile Volumetric Video Streaming by Exploiting Inter-Frame Correlation. *IEEE Transactions on Mobile Computing* 1, 1 (2024), 1–15.

[41] Yizong Wang, Dong Zhao, Huanhuan Zhang, Chenghao Huang, Teng Gao, Zixuan Guo, Liming Pang, and Huadong Ma. 2023. Hermes: Leveraging Implicit Interframe Correlation for Bandwidth-Efficient Mobile Volumetric Video Streaming. In *Proceedings of the 31st ACM International Conference on Multimedia*. ACM, Ottawa, Canada, 9185–9193.

[42] Anlan Zhang, Chendong Wang, Bo Han, and Feng Qian. 2021. Efficient Volumetric Video Streaming Through Super Resolution. In *Proceedings of the 22nd International Workshop on Mobile Computing Systems and Applications* (Virtual, United Kingdom) *(HotMobile '21)*. Association for Computing Machinery, New York, NY, USA, 106–111. https://doi.org/10.1145/3446382.3448663

[43] Qian-Yi Zhou, Jaesik Park, and Vladlen Koltun. 2018. Open3D: A Modern Library for 3D Data Processing. *arXiv preprint arXiv:1801.09847* 1, 1 (2018), 1–10.

[44] Junni Zou, Chenglin Li, Chengming Liu, Qin Yang, Hongkai Xiong, and Eckehard Steinbach. 2019. Probabilistic Tile Visibility-based Server-side Rate Adaptation for Adaptive 360-degree Video Streaming. *IEEE Journal of Selected Topics in Signal Processing* 14, 1 (2019), 161–176.



Nagare Media Engine: Towards Self-Adapting MPEG NBMP Multimedia Workflows

Matthias Neugebauer
matthias.neugebauer@uni-muenster.de
University of Münster
Münster, Germany

Abstract

With ISO/IEC 23090-8 Network-Based Media Processing (NBMP), MPEG published a standard for today's multimedia workflows: complex distributed systems deployed in multi-cloud and multi-edge environments. It defines data models, APIs and a reference architecture for implementing and operating multimedia workflows as well as the corresponding workflow system. Next to the increased architectural complexity, multimedia workflows are subject to changing surroundings or objectives. As such, they are forced to adapt in order to still meet the workflow goals. In this paper, we explore self-adaptability in the context of NBMP to automate this process. We give an overview of adaptation types encountered in multimedia workflows. Moreover, we propose a general design for self-adapting NBMP workflows. Finally, we implement our approach as part of `Nagare Media Engine`, our existing open source NBMP implementation. The evaluation of our prototype demonstrates self-adaptability in an HTTP Adaptive Streaming (HAS) scenario resulting in decreased computing resource usage.

CCS Concepts

• **Networks** → **Network architectures**; • **Information systems** → **Multimedia streaming**.

Keywords

nbmp, network-based media processing, multimedia, workflow, self-adapting, streaming

ACM Reference Format:
Matthias Neugebauer. 2025. Nagare Media Engine: Towards Self-Adapting MPEG NBMP Multimedia Workflows. In *ACM Multimedia Systems Conference 2025 (MMSys '25), March 31–April 4, 2025, Stellenbosch, South Africa.* ACM, New York, NY, USA, 7 pages. https://doi.org/10.1145/3712676.3718340

1 Introduction

The development and operation of multimedia workflows increasingly becomes more sophisticated. Multimedia streams often traverse through numerous networks and data centers before finally reaching viewers. Along this way, different media functions are applied to the streams to cater for business, quality, technical and various other objectives. HTTP Adaptive Streaming (HAS), for instance, encodes an input stream into multiple rendition streams that are packaged and indexed in a manifest. Media players then select the most adequate rendition in order to maximize the quality of experience (QoE). In order to empower today's multimedia workflows, the Moving Picture Experts Group (MPEG) recently published ISO/IEC 23090-8 Network-Based Media Processing (NBMP) [8], a standard that defines data models, APIs and a reference architecture for multimedia workflows. NBMP splits the workflow into network-connected tasks that are deployed in various computing environments. Standardized API definitions enable an interoperable deployment of NBMP workflows over these computing environments.

Simultaneously to the increased architectural complexity, the workflow context as well as defined objectives are subject to change mid-execution. Moreover, not every information is available a priori and might only become known during the workflow run. Changed circumstances thus could require an adaptation of the workflow in order to still satisfy set objectives. For example, an increase in video complexity might necessitate altering encoding parameters in order to reach quality targets. Similarly, cost and energy efficiency goals (green streaming) might require redeploying media functions to other computing environments. Self-adaptive software systems plan and execute adaptations automatically based on defined objectives. This paper explores self-adaptability for multimedia workflows in general and for NBMP workflows specifically. Hereby, we are guided by the following research questions:

RQ1 What types of adaptations can occur in actively running multimedia workflows?

RQ2 How can self-adaptability be brought to NBMP workflows and be defined in NBMP workflow descriptions?

To answer the first question, we look at example workflows from practice and current research. For RQ2, we first design and implement a general approach for self-adapting NBMP workflows as part of `Nagare Media Engine`, an open source NBMP implementation. In a second step, we evaluate our implementation in an example self-adapting workflow. To summarize, this paper makes the following contributions:

(1) An overview of adaptation types for multimedia workflows.
(2) A general design for self-adapting NBMP workflows.
(3) Our extensions to `Nagare Media Engine`, an open source NBMP implementation, that enable self-adapting workflows.

The rest of this paper is structured as follows. In Section 2, we discuss NBMP and `Nagare Media Engine` as background for this paper. The next section gives an overview of related work. Sections 4 and 5 address our research questions by exploring relevant workflow adaptations and designing as well as implementing an approach for self-adaptation in NBMP workflows, respectively. We evaluate our approach in Section 6 and address limitations in Section 7. Finally, Section 8 concludes this paper.



This work is licensed under a Creative Commons Attribution 4.0 International License.
MMSys '25, Stellenbosch, South Africa
© 2025 Copyright held by the owner/author(s).
ACM ISBN 979-8-4007-1467-2/2025/03
https://doi.org/10.1145/3712676.3718340

2 Background

2.1 NBMP

MPEG first published NBMP as ISO/IEC 23090-8 in 2020 [8]. This standard defines APIs and data models for network-distributed multimedia workflows. Figure 1 depicts the components involved.

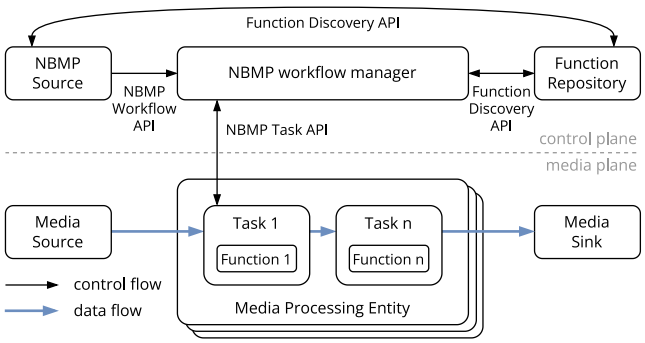


Figure 1: NBMP reference architecture, compare [8].

The NBMP reference architecture is split into a control and a media plane. The *workflow manager* is the central component of the control plane. It implements the *Workflow API* that gives clients, so-called *NBMP sources*, control over the workflows running in the system. To instantiate a workflow, the necessary media functions need to be selected. For this, either the NBMP source or the workflow manager (based on constraints defined by the NBMP source) consult the *function repository* for appropriate media functions. Information about available functions is provided through the *Function Discovery API*.

After the selection process, the workflow manager instantiates the media functions within the media plane in the form of *tasks* that are deployed to *media processing entities* (MPEs). The workflow manager uses the *Task API* to configure and control the lifecycle of a task. Instantiated workflows logically form a directed acyclic graph (DAG). Tasks are represented as nodes with arbitrary many in- and output *ports*. Edges in the DAG represent the connections of ports between different tasks. Ports are then said to be bound to a *stream* established between tasks. NBMP differentiates between metadata and media streams, but is otherwise agnostic to specific formats. The *media sources* and *media sinks* are special nodes in the DAG as they form the inputs and outputs of the whole workflow. Streams are either pulled or pushed from sender to receiver.

The behavior of workflows and tasks is governed by a state machine. First, they start out as *instantiated*. Then, after applying the configuration, the state moves on to *idle* indicating that the component is now ready to receive data. Starting the processing will advance the state to *running*. In case of an error, the component will enter the *error* state. Finally, the lifecycle ends in the *destroyed* state after termination.

Workflows, tasks and media functions are represented as JavaScript Object Notation (JSON) documents within the APIs. NBMP defines create, read, update and delete operations for the Workflow and Task APIs, respectively. The Function Discovery API only contains operations to query for media functions. All APIs follow the Representational State Transfer (REST) style. Some interfaces between components are intentionally left undefined (e.g. between the workflow manager and MPEs). Further, the exact behavior of MPEs, media sources and media sinks is also left open. This allows for competing implementations of the NBMP standard.

2.2 Nagare Media Engine

`Nagare Media Engine`[1] is our ongoing effort to implement NBMP as an open source solution. We previously described our earlier work in detail in [15, 16]. Figure 2 gives an overview of the design.

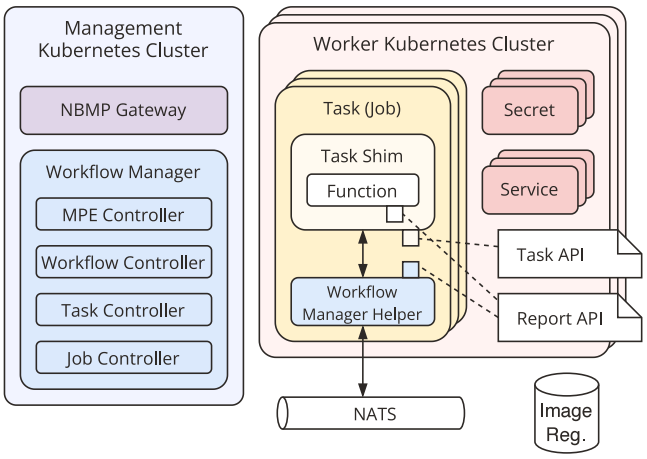


Figure 2: `Nagare Media Engine` architecture, compare [16].

With the management and worker Kubernetes[2] clusters, `Nagare Media Engine` utilizes Kubernetes as platform for both the control and the media plane, respectively. While in simple deployment scenarios these clusters might coincide in a single one, in more complex scenarios multiple clusters are also supported (e.g. cloud-edge deployments).

The *NBMP Gateway* and *Workflow Manager* together with the *Workflow Manager Helper* form the NBMP control plane. The NBMP Gateway implements the Workflow API and translates between the JSON description and Kubernetes resources. Custom Resource Definitions (CRDs) are used to add the necessary entities to Kubernetes. Following the Kubernetes model, the Workflow Manager consists of multiple controllers that reconcile the custom resource specifications with the current cluster state. Finally, the Workflow Manager Helper implements the required interaction of the control plane with the Task API to configure and manage tasks. It further provides an interface for event reporting. All reported events are persisted in a NATS JetStream[3], a messaging and event sourcing system. Tasks can then later on replay earlier events, e.g. after a restart for task error recovery.

For each task in the media plane, a Kubernetes `Job`, `Secret` and `Service` resource is created. The `Secret` contains all necessary information for creating a task through the Task API. The `Service` enables the use of a stable endpoint for streaming data between tasks. Moreover, `Nagare Media Engine` provides a common Task

[1] https://github.com/nagare-media/engine
[2] https://kubernetes.io
[3] https://nats.io

API implementation with the *Task Shim* component. The separation between the Task API implementation and the media function allows media function authors to focus on the core logic instead of dealing with NBMP API interactions. All media functions are packaged as container images that are uploaded to and retrieved from an image registry.

3 Related Work

The software industry aspired to build self-adaptive software systems for some time. Kephart and Chess already painted a picture of future software systems that manage themselves in 2003 [12]. Salehie and Tahvildari give a comprehensive overview of this research field in [19]. More recently, learner-based methods have been applied to self-adaptive software systems. Gheibi et al. conducted a literature review to better understand the use of machine learning in this area [5].

Self-adaptability can be utilized in various domains, workflow systems among them. Baird et al. explored the use in business workflows and propose to adapt workflow steps by choosing between alternative business services based on user directives [3]. In [20], Seiger et al. propose a framework for self-adaptive workflows in cyber-physical systems. However, these approaches are not directly applicable to NBMP workflows which justifies a dedicated look into self-adaptive multimedia workflows.

Prior research has been conducted around NBMP. Wien et al. give a general overview of the standard family that NBMP is part of [22]. NBMP has been proposed as a solution in various use cases [4, 14, 18, 23, 24]. Finally, we previously presented `Nagare Media Engine`, our open source NBMP implementation [15, 16]. To the best of our knowledge, self-adaptability has not yet been explored for NBMP workflows.

4 Multimedia Workflow Adaptations

Multimedia workflows can be adapted in various ways. By looking into the current research and examples from practice, we identified four types of adaptations that need to be supported in a media workflow system that enables self-adaptation. Note that we only considered adaptations that can be applied directly by a workflow manager. There might be simultaneous outside influences affecting the workflow execution, e.g. a cloud system might live-migrate a virtual machine to a different host or a network device might choose a different path through the network. But as this is typically outside of the control of the workflow manager, we did not further explore these adaptations. Instead, we focused on changes that can be represented in the logical workflow DAG. However, as these outside influences still have an effect on the workflow, it might be beneficial to investigate lower-level adaptations in future work.

The simplest adaptation is a **task reconfiguration** as shown in Figure 3. Here, only individual tasks are adapted, otherwise the workflow remains the same. This adaptation is often employed in research and practice in the form of an optimization problem. For example, Mueller et al. demonstrated in [14] how encoding parameters can be continuously optimized. And Tashtarian et al. implemented a workflow that dynamically adapted the encoding ladder of a packager [21].



Figure 3: Task reconfiguration adaptation.

Another adaptation is the switch between **alternative media functions**. This can be beneficial for a variety of technical, business or user oriented reasons. Changed circumstances might make another function more cost-effective, energy-efficient (green streaming) or lead to a higher QoE. Moreover, this could also be a simple version update of an existing media function. Figure 4 depicts this adaptation. One example from practice is HAS where alternative functions for packaging and manifest generation are used depending on the client.



Figure 4: Alternative media function adaptation.

Similar to the previous adaptation and for similar reasons, the next adaptation type switches between **alternative MPEs** as can be seen in Figure 5. Moreover, compliance reasons could prompt redeploying tasks to other MPEs, e.g. when data processing should only occur in certain regions. Oftentimes the workloads are brought nearer to viewers in order to reduce network latencies (edge computing). With changing viewers, another MPE might be more advantageous. Examples in literature include the following. Afzal et al. select computing instances for encoding tasks based on cost and energy ranking [2]. Similarly, Kara and Simon use content steering standards to direct clients to power-efficient Content Delivery Networks (CDNs) [10]. In an extended reality (XR) use case, Heo et al. distribute tasks between a local device and remote servers [6].



Figure 5: Alternative MPE adaptation.

Lastly, Figure 6 depicts a scenario where a **DAG adaptation** has occurred. Here, tasks could be added (`Task D`) or removed (`Task B`) from the DAG. Likewise, edges between the tasks could be altered (`Task A'` and `C'`). A prominent example for this adaptation type is Netflix's encoding workflow. Here, an analysis in combination with trial encoding steps steers subsequent workflow operations and the media content flow allowing for per-title and per-shot encoding optimizations [1, 11, 13].

Note that all previous adaptations can also be described in terms of a DAG adaptation, where the previous task is replaced by an adapted task. However, this might result in a higher overhead and a disruption of the workflow execution. Maintaining an uninterrupted stream during workflow adaptations is a challenge in general.

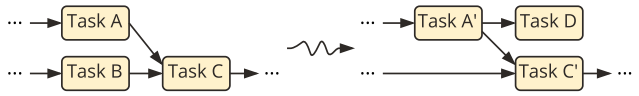


Figure 6: DAG adaptation (add and remove tasks or connections).

Currently, MPEG is working on a proposed standard for redundant encoding and packaging Dynamic Adaptive Streaming over HTTP (DASH) streams, which might alleviate this challenge [9].

5 Self-Adapting NBMP Workflows

In order to enable self-adaptation in NBMP workflows, we first designed a general approach in Subsection 5.1. Next, Subsection 5.2 outlines how we extended `Nagare Media Engine` to implement our proposed design.

5.1 Design Overview

Traditional self-adapting software systems are based on the monitor, analyze, plan, execute and knowledge (MAPE-K) adaptation loop as visualized in Figure 7. The autonomic manager, also called the managing system, continuously collects data about the managed system (monitor). Based on this, it decides whether an adaptation is necessary (analyze). Next it determines the what and how of the adaptation (plan). Finally, the change is applied to the managed system and the loop starts again. Throughout these steps, the autonomic manager might consult a knowledge-base (e.g. for policies, configuration, etc.). The autonomic manager might be internal, i.e. part of, or external, i.e. running aside, to the managed system. [12, 19]

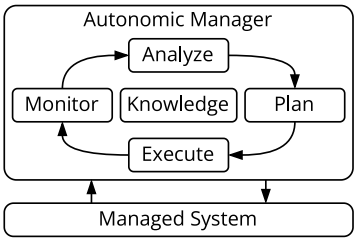


Figure 7: MAPE-K adaptation loop [12].

We applied this design to self-adapting NBMP workflows which we will describe in the following. In our approach, we decided to integrate the MAPE-K loop into the workflow itself (internal autonomic manager). Therefore, instead of a new component next to the NBMP system, workflow authors describe the adaptation logic within the workflow definition in the form of additional tasks. The workflow definition thus contains a complete description of the self-adapting workflow.

We map the monitor step to tasks that gather data about the workflow execution. These can be implemented as new functions with this sole purpose. Alternatively, existing media functions could be extended to additionally collect data. For example, an encoding task might simultaneously calculate quality metrics for the generated media. In both cases, the task should have an output port that provides this data in the form of a metadata stream.

In Section 4, we already highlighted diverse use cases for workflow adaptations. Hence, the adaptation logic necessary for the analyze and plan steps varies widely. The NBMP standard only defines a declarative description of workflows and does not allow expressing arbitrary logic. We therefore propose a new domain-specific language (DSL) that can be used by workflow authors. Programs in that DSL are passed as task configuration to a script execution function. The language should allow easy access to metadata streams of monitoring tasks to enable the analyze and plan steps. Further, workflow authors should be able to apply all possible adaptations to the workflow. Due to its domain-focus, the DSL should allow expressing the logic in the terms used by the NBMP standard. Moreover, it should be easy to understand and write programs.

In the final step, the workflow adaptation is executed. For this, the Workflow API should receive a request with an updated workflow definition. It is then the work of the workflow manager to take the necessary steps. Task reconfigurations only require update requests to the Task API. Other changes demand interactions with MPEs. Figure 8 visualizes our proposed design for self-adapting NBMP workflows.

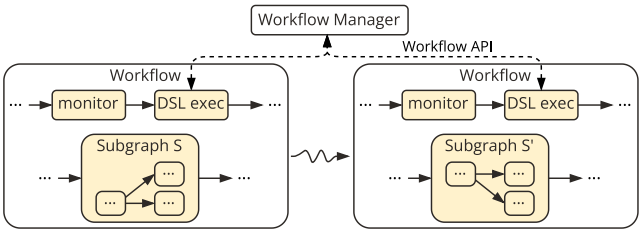


Figure 8: Design of a self-adapting NBMP workflow.

5.2 Implementation

We extended `Nagare Media Engine` to implement our proposed design. All changes are open source[1] and licensed under Apache 2.0. We also included our documented examples in the codebase.

For custom monitoring tasks, we added support for arbitrary metadata streams. Next, the NBMP Gateway now has an improved handling of update operations. We make sure to follow the task lifecycle model. For instance, updates to terminated tasks are not executed. The Workflow Manager component was also refined to handle all possible workflow adaptations. A task reconfiguration will now result in an update to the Kubernetes `Secret` which is then detected by the corresponding Workflow Manager Helper resulting in a Task API update request. This way, a redeployment of the Kubernetes `Pod` and, thus, an even longer disruption can be avoided.

We further added `script-lua` as new function for the analyze and plan steps. It implements an internal DSL in the programming language Lua[4] version 5.1. This scripting language is primarily

[4] https://www.lua.org/

designed to be embedded into applications and has a user-friendly syntax. Listing 1 shows an example Lua script discussed in more detail in Section 6. For a general overview of Lua, see [7]. Our primary goal was to provide a comprehensive but simple to use language to workflow authors. Implementing it as a function allows for quick iterations on future versions of the DSL as other NBMP components can remain unchanged.

We extended Lua with additional types and operations relevant for NBMP. Workflow authors can read metadata input streams of the `script-lua` task through the `Port` type. The `Workflow` type allows access to the workflow the task is currently running in. While `Task` objects describe tasks, `Connection` objects provide information about the connection between them or workflow `Inputs` and `Outputs`. The `Workflow` type also provides methods for adding and removing `Connections`. Adaptations to the workflow are not immediately rolled out. Instead, workflow authors have to deliberately execute the changes by calling the `update` method. Multiple simultaneous adaptations are thus possible. We tried to avoid "boilerplate code" as much as possible, i.e. code that is repeated in every program with little value to the core business logic. All exposed Lua modules are preloaded. Moreover, workflow authors do not have to deal with establishing connections to metadata streams or the Workflow API.

6 Evaluation

We evaluated our proposed design and implementation. Subsection 6.1 describes the evaluation use case while Subsection 6.2 discusses the results.

6.1 HTTP Adaptive Streaming Use Case

We chose HAS as commonly encountered use case for the evaluation. Traditionally, a high quality input stream is encoded into a fixed set of renditions varying in resolution, bitrate, codecs, etc. (the encoding ladder). A manifest lists available renditions and the player selects the most appropriate one, e.g. based on the viewers bandwidth and viewport. Recent research explored the idea of a dynamic encoding ladder where a rendition is only created e.g. if enough viewers justify the additional work [17, 21]. Players that request an unavailable rendition will receive the closest alternative. This way, costs and energy consumption can potentially be reduced.

This idea can be implemented in a self-adapting multimedia workflow that adds or removes encoding tasks. Each encoding task is connected to the same input task (`media-source`) and outputs the encoded stream to the `packager` task. As we focus on self-adaptation, our packager only receives the stream but does not further process it. We also implemented a `monitor` task that outputs simulated viewer statistics in regular intervals through a metadata stream. Figure 9 depicts the resulting workflow DAG.

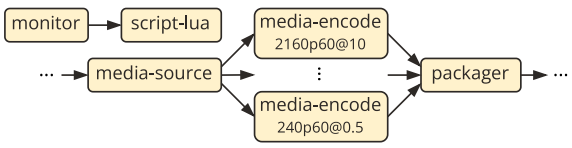


Figure 9: HAS workflow with self-adapting bitrate ladder.

```lua
-- knowledge
wf         = nbmp.Workflow:self()
src_task   = wf:get_task("media-source")
pkg_task   = wf:get_task("packager")
stats_port = nme.get_input_port("stats")
enc_ladder = {
  ["2160p@10"]  = {r="3840x2160"; b="10000"; min=20},
  [ "1440@7"]   = {r="2560x1440"; b= "7000"; min=10},
  [ "1080@6"]   = {r="1920x1080"; b= "6000"; min=10},
  [ "720p@3"]   = {r="1280x720";  b= "3000"; min= 5},
  [  "480@1.5"] = {r= "854x480";  b= "1500"; min= 5},
  [  "360@1"]   = {r= "640x360";  b= "1000"; min= 1},
  [ "240p@0.5"] = {r= "426x240";  b=  "500"; min= 0}
}

while true do
  -- analyze
  stats_data = stats_port:read_line()
  if not stats_data then
    break
  end

  stats = json.parse(stats_data)
  for enc, ladder in pairs(enc_ladder) do
    enc_task = nbmp.Task:new({
      id = "encode-"..enc,
      function_restriction = {
        id = "Function/media-encode",
        config = {
          resolution = ladder.r,
          bitrate    = ladder.b,
        }
      }
    })

    con_media = nbmp.Connection:new({
      id   = src_task.id.."-"..enc_task.id,
      from = {task = src_task; port = "out."..enc},
      to   = {task = enc_task; port = "in"}
    })

    con_pkg = nbmp.Connection:new({
      id   = enc_task.id.."-"..pkg_task.id,
      from = {task = enc_task; port = "out"},
      to   = {task = pkg_task; port = "in-"..enc}
    })

    -- plan
    if ladder.min <= stats[enc] then
      wf:add_connection(con_media)
      wf:add_connection(con_pkg)
    else
      wf:remove_connection(con_media)
      wf:remove_connection(con_pkg)
    end
  end

  -- execute
  wf:update()
end
```

Listing 1: Self-adaptation logic for a HAS encoding ladder.

The self-adaptation loop is implemented in our DSL and executed in an additional task (`script-lua`). Listing 1 shows a slightly abbreviated version of our implementation. We store the knowledge about possible renditions in the `enc_ladder` table object (lines 6–14). Additionally, the `min` field indicates the minimal number of viewers necessary for the rendition to be created. At maximum, the encoding ladder consists of seven steps ranging from 2160p at 10 MBit/s to 240p at 0.5 MBit/s. We also define variables that point to the current workflow, the `media-source` and `packager` tasks as well as the `stats` input port from the `monitor` task (lines 2–5).

We start the self-adaptation loop with the analyze step (lines 17–46). The `monitor` task outputs the viewer statistics for every rendition as JSON data that is parsed to a Lua table object (lines 18–23). We then iterate over every possible rendition and create an in-memory representation of the respective encoding task as well as the necessary connections (lines 24–46).

Next, in the plan step, we compare the viewer count to the minimum to decide if the encoding task should be added to or removed from the workflow (lines 48-55). Note that NBMP requires tasks to have connections. We therefore only add or remove connections to workflows. Additionally, we implemented the `add_connection` and `remove_connection` methods as idempotent functions, i.e. calling them twice (e.g. over multiple loop iterations) will result in the same workflow.

Finally, in line 59, we execute all adaptations by calling the workflow `update` method which will result in an update request to the Workflow API.

We use the Meridian short film[5] as test media. The `media-source` task provides it as a simulated real-time livestream. Meridian is approximately 12 minutes long, has a resolution of 3840x2160 and a framerate of 59.94 fps. It was created as open test media to reflect typical streaming content.

6.2 Results

Without loss of generality, our evaluation was executed on a five node Kubernetes 1.31.2 cluster managed by Scaleway, a European cloud service provider. We used the machine type `POP2-HC-8C-16G` for all nodes (8 dedicated vCPUs and 16 GB RAM). `Nagare Media Engine` and all dependencies were deployed in this Kubernetes cluster. Additionally, we deployed Prometheus[6], a monitoring solution that was configured to collect metrics about the usage of computing resources every five seconds.

In total, we executed three test scenarios. The *minimal-ladder* scenario established the lower bound for the evaluation. As such, it only encoded the 240p rendition of the encoding ladder. Similarly, the *maximal-ladder* scenario established the upper bound of the evaluation by encoding all renditions. Finally, in the *adaptation* scenario the `monitor` task reports varying statistics such that in total all renditions are requested for half the time of the short film. Thus, the self-adaptation logic adapts the workflow DAG to add or remove encoding tasks.

For all test scenarios, we measured the total CPU time of all Kubernetes containers that executed workflow functions (i.e. the `container_cpu_usage_seconds_total` metric in Kubernetes). It includes the time spent in the kernel and user space across all CPU cores, showing how intensively the available computing resources were used. Table 1 summarizes the results.

Table 1: Total measured CPU time of all containers that executed workflow functions in seconds.

Test Scenario	Total CPU Time (Seconds)
minimal-ladder	1125.12
maximal-ladder	14189.40
adaptation	6657.85

As can be seen, there is a large gap between the minimal-ladder and maximal-ladder scenarios. This signifies an opportunity for savings in computing resources and, thus, costs and energy savings as well. Our measurement for the adaptation scenario demonstrates this potential. We therefore see our approach and self-adaptability for NBMP workflows validated.

7 Limitations

Currently, we have not yet considered the influence of learner-based self-adaptation on our design. Gheibi et al. proposed a design process for using machine learning in self-adaptive software systems after conducting a systematic literature review in [5]. Future work might apply this process for NBMP workflows.

Our implementation is not yet able to maintain an uninterrupted stream when adapting the workflow. While we implemented support for task reconfiguration, our media functions still require a restart at this moment. Since this is handled by the Task Shim component within the running Kubernetes `Pod`, a restart is a relatively quick operation without the need for a complete `Pod` rescheduling. Future work should look into the upcoming MPEG standard ISO/IEC 23009-9 to provide redundancy during an adaptation [9].

8 Conclusion

In this paper, we applied self-adaptability in multimedia workflows. Guided by our two research questions, we first identified the four adaptation types "task reconfiguration", "alternative media functions", "alternative MPEs" as well as "DAG adaptations" (RQ1). Next, we proposed a general design for self-adapting NBMP workflows and implemented support for this approach in our NBMP implementation `Nagare Media Engine` (RQ2). We evaluated our approach in a HAS workflow scenario and showed a decreased usage of computing resources when using self-adaptation.

Future work should focus on strategies for maintaining uninterrupted streams even during workflow adaptations. Additionally, our design should be tested with learner-based self-adaptation methods. Finally, future work could explore how automated adaptation decisions could be explained to human operators effectively.

References

[1] Anne Aaron, Zhi Li, Megha Manohara, Jan De Cock, and David Ronca. 2015. Per-Title Encode Optimization. https://netflixtechblog.com/per-title-encode-optimization-7e99442b62a2

[2] Samira Afzal, Narges Mehran, Sandro Linder, Christian Timmerer, and Radu Prodan. 2023. VE-Match: Video Encoding Matching-based Model for Cloud and Edge Computing Instances. In *Proceedings of the First International Workshop*

[5] https://opencontent.netflix.com/

[6] https://prometheus.io/

on Green Multimedia Systems. ACM, Vancouver BC Canada, 1–6. doi:10.1145/3593908.3593943

[3] R. Baird, N. Jorgenson, and R. Gamble. 2011. Self-Adapting Workflow Reconfiguration. *Journal of Systems and Software* 84, 3 (March 2011), 510–524. doi:10.1016/j.jss.2010.10.026

[4] Louay Bassbouss, Mehdi Ben Fadhel, Stefan Pham, Anita Chen, Stephan Steglich, Eric Troudt, Marc Emmelmann, Jesús Gutiérrez, Nebojsa Maletic, Eckhard Grass, Stefan Schinkel, Annette Wilson, Sven Glaser, and Christian Schlehuber. 2021. 5G-VICTORI: Optimizing Media Streaming in Mobile Environments Using mmWave, NBMP and 5G Edge Computing. In *Artificial Intelligence Applications and Innovations. AIAI 2021 IFIP WG 12.5 International Workshops*, Ilias Maglogiannis, John Macintyre, and Lazaros Iliadis (Eds.). Vol. 628. Springer International Publishing, Cham, 31–38. doi:10.1007/978-3-030-79157-5_3

[5] Omid Gheibi, Danny Weyns, and Federico Quin. 2020. Applying Machine Learning in Self-adaptive Systems: A Systematic Literature Review. *ACM Transactions on Autonomous and Adaptive Systems* 15, 3 (Sept. 2020), 1–37. doi:10.1145/3469440

[6] Jin Heo, Ketan Bhardwaj, and Ada Gavrilovska. 2023. FleXR: A System Enabling Flexibly Distributed Extended Reality. In *Proceedings of the 14th ACM Multimedia Systems Conference*. ACM, Vancouver BC Canada, 1–13. doi:10.1145/3587819.3590966

[7] Roberto Ierusalimschy, Luiz Henrique de Figueiredo, and Waldemar Celes. 2006. *Lua 5.1 Reference Manual*. Lua.org. https://www.lua.org/manual/5.1/index.html

[8] ISO/IEC. 2020. *ISO/IEC 23090-8:2020 Information Technology – Coded Representation of Immersive Media – Part 8: Network Based Media Processing*. Standard. International Organization for Standardization, Geneva, CH.

[9] ISO/IEC. 2024. *ISO/IEC FDIS 23009-9 Information Technology – Dynamic Adaptive Streaming over HTTP (DASH) – Part 9: Redundant Encoding and Packaging for Segmented Live Media (REaP)*. Standard. International Organization for Standardization, Geneva, CH.

[10] Burak Kara and Gwendal Simon. 2024. Power Efficient Multi-CDN Communication over Content Steering Server. In *Proceedings of the 15th ACM Multimedia Systems Conference*. ACM, Bari Italy, 478–484. doi:10.1145/3625468.3652196

[11] Ioannis Katsavounidis. 2018. Dynamic Optimizer — a Perceptual Video Encoding Optimization Framework. https://netflixtechblog.com/dynamic-optimizer-a-perceptual-video-encoding-optimization-framework-e19f1e3a277f

[12] J.O. Kephart and D.M. Chess. 2003. The Vision of Autonomic Computing. *Computer* 36, 1 (Jan. 2003), 41–50. doi:10.1109/MC.2003.1160055

[13] Megha Manohara, Anush Moorthy, Jan De Cock, Ioannis Katsavounidis, and Anne Aaron. 2018. Optimized Shot-Based Encodes: Now Streaming! https://netflixtechblog.com/optimized-shot-based-encodes-now-streaming-4b9464204830

[14] Christoph Mueller, Louay Bassbouss, Stefan Pham, Stephan Steglich, Sven Wischnowsky, Peter Pogrzeba, and Thomas Buchholz. 2022. Context-Aware Video Encoding as a Network-Based Media Processing (NBMP) Workflow. In *Proceedings of the 13th ACM Multimedia Systems Conference*. ACM, Athlone Ireland, 293–298. doi:10.1145/3524273.3533250

[15] Matthias Neugebauer. 2023. Nagare Media Engine: Towards an Open-Source Cloud- and Edge-Native NBMP Implementation. In *Proceedings of the 18th International Conference on Software Technologies*. SCITEPRESS - Science and Technology Publications, Rome, Italy, 404–411. doi:10.5220/0012087200003538

[16] Matthias Neugebauer. 2024. Nagare Media Engine: Task Error Recovery in MPEG NBMP Workflows Through Event Sourcing. In *Proceedings of the 15th ACM Multimedia Systems Conference*. ACM, Bari Italy, 257–263. doi:10.1145/3625468.3652167

[17] Jan Outters and Mickael Raulet. 2023. Audience Aware Streaming: New Dynamics in OTT Distribution. In *Proceedings of the First International Workshop on Green Multimedia Systems*. ACM, Vancouver BC Canada, 13–15. doi:10.1145/3593908.3593946

[18] Roberto Ramos-Chavez, Rufael Mekuria, Theo Karagkioules, Dirk Griffioen, Arjen Wagenaar, and Mark Ogle. 2021. MPEG NBMP Testbed for Evaluation of Real-Time Distributed Media Processing Workflows at Scale. In *Proceedings of the 12th ACM Multimedia Systems Conference*. ACM, Istanbul Turkey, 173–185. doi:10.1145/3458305.3463380

[19] Mazeiar Salehie and Ladan Tahvildari. 2009. Self-Adaptive Software: Landscape and Research Challenges. *ACM Transactions on Autonomous and Adaptive Systems* 4, 2 (May 2009), 1–42. doi:10.1145/1516533.1516538

[20] Ronny Seiger, Steffen Huber, Peter Heisig, and Uwe Aßmann. 2019. Toward a Framework for Self-Adaptive Workflows in Cyber-Physical Systems. *Software & Systems Modeling* 18, 2 (April 2019), 1117–1134. doi:10.1007/s10270-017-0639-0

[21] Farzad Tashtarian, Abdelhak Bentaleb, Hadi Amirpour, Sergey Gorinsky, Junchen Jiang, Hermann Hellwagner, and Christian Timmerer. 2024. ARTEMIS: Adaptive Bitrate Ladder Optimization for Live Video Streaming. In *21st USENIX Symposium on Networked Systems Design and Implementation (NSDI 24)*. USENIX Association, Santa Clara, CA, 591–611. https://www.usenix.org/conference/nsdi24/presentation/tashtarian

[22] Mathias Wien, Jill M. Boyce, Thomas Stockhammer, and Wen-Hsiao Peng. 2019. Standardization Status of Immersive Video Coding. *IEEE Journal on Emerging and Selected Topics in Circuits and Systems* 9, 1 (March 2019), 5–17. doi:10.1109/JETCAS.2019.2898948

[23] Yiling Xu, Jiaoyang Yin, Qi Yang, and Le Yang. 2022. Media Production Using Cloud and Edge Computing: Recent Progress and NBMP-Based Implementation. *IEEE Transactions on Broadcasting* 68, 2 (June 2022), 545–558. doi:10.1109/TBC.2022.3140704

[24] Yu You, Ari Hourunranta, and Emre B. Aksu. 2020. OMAF4Cloud: Standards-Enabled 360° Video Creation as a Service. *SMPTE Motion Imaging Journal* 129, 9 (Oct. 2020), 18–23. doi:10.5594/JMI.2020.3010400



A Congestion Control Test Suite for Real-Time Communication

Quanwei Zhang
University of Victoria
Victoria, BC, Canada

Zhiming Huang
University of Victoria, Canada
Victoria, BC, Canada

Jinwei Zhao
University of Victoria, Canada
Victoria, BC, Canada

Jianping Pan
University of Victoria, Canada
Victoria, BC, Canada

Abstract

Real-time communication (RTC) systems, such as video conferencing and cloud gaming, depend on effective *congestion control (CC)* algorithms to manage diverse network conditions and access technologies like Wi-Fi, LTE/5G, and satellite networks. While tools like AlphaRTC and Pandia have significantly advanced CC algorithm development for WebRTC, there is an absence of a unified framework for systematic benchmarking and cross-platform evaluation.

We introduce a practical test suite that offers a unified platform for evaluating algorithms developed with containerized tools such as AlphaRTC and Pandia. Leveraging Mahimahi and Containernet, our test suite supports trace-driven emulations of realistic heterogeneous network environments and incorporates diverse topologies with multiple RTC flows communicating simultaneously, enabling testing under complex and heterogeneous conditions.

Beyond evaluation, our test suite promotes community engagement by enabling easy contribution and integration of algorithms. Researchers can add their own algorithms and test scenarios to the suite, allowing them to seamlessly compare their solutions with archived ones under reproducible evaluation conditions.

CCS Concepts

• **Networks** → **Network experimentation**.

Keywords

Congestion control, real-time communication, WebRTC, testing framework

ACM Reference Format:
Quanwei Zhang, Zhiming Huang, Jinwei Zhao, and Jianping Pan. 2025. A Congestion Control Test Suite for Real-Time Communication. In *ACM Multimedia Systems Conference 2025 (MMSys '25), March 31-April 4, 2025, Stellenbosch, South Africa.* ACM, New York, NY, USA, 7 pages. https://doi.org/10.1145/3712676.3718341

1 Introduction

Real-time communication (RTC) has become a cornerstone of modern networks, powering applications such as video conferencing, cloud gaming, and telemedicine. The increasing reliance on RTC stems from the growing need for seamless, high-quality user experiences across diverse environments. These applications must operate over heterogeneous network access technologies, including LAN, Wi-Fi, LTE/5G, and even satellite networks. The dynamic nature of these networks, coupled with competing traffic demands, poses significant challenges in ensuring low latency, high throughput, and reliable performance. Effective *congestion control (CC)* is therefore essential to optimize the *quality of experience (QoE)* in such complex and varing network scenarios.

To address these challenges, significant efforts have been made to simplify the implementation and evaluation of congestion control mechanisms for WebRTC. Notable tools, such as AlphaRTC [5] and Pandia [13], have emerged as powerful platforms for experimenting with and deploying advanced CC algorithms. These tools provide a streamlined framework for developers to test and implement congestion control strategies, with an emphasis on integrating modern techniques, including deep reinforcement learning. By reducing the barriers to entry, these tools have accelerated innovation in WebRTC congestion control research.

Despite significant advancements provided by tools like AlphaRTC and Pandia, critical gaps persist in the field. First, while these tools offer evaluation cases tailored to their own frameworks, there is no unified benchmarking framework for assessing the effectiveness of CC solutions across different tools. This fragmentation makes it challenging to compare CC algorithms developed using different tools.

Second, the absence of a unified testing platform hampers objective comparisons and the identification of optimal strategies for diverse RTC scenarios. Existing evaluation setups are often limited in scope, primarily focusing on simple scenarios involving a single sender and receiver. They fail to address the complexity of real-world network environments, which involve heterogeneous access technologies such as Wi-Fi, LTE/5G, and satellite communications.

To address these limitations, we introduce a comprehensive test suite to complement tools like AlphaRTC and Pandia that serves as a unified benchmarking platform for evaluating CC algorithms in real-time communication systems. Our test suite features a diverse set of rigorous test cases, supporting both self-defined scenarios and trace-driven experiments, to simulate a broad spectrum of network conditions and application requirements. By enabling systematic evaluation and direct comparisons, our framework empowers researchers and developers to advance the state of RTC congestion control and design solutions that demonstrate robust performance in complex and dynamic real-world scenarios. The main features of our system are summarized as follows:

Permission to make digital or hard copies of all or part of this work for personal or classroom use is granted without fee provided that copies are not made or distributed for profit or commercial advantage and that copies bear this notice and the full citation on the first page. Copyrights for components of this work owned by others than the author(s) must be honored. Abstracting with credit is permitted. To copy otherwise, or republish, to post on servers or to redistribute to lists, requires prior specific permission and/or a fee. Request permissions from permissions@acm.org.
MMSys '25, Stellenbosch, South Africa
© 2025 Copyright held by the owner/author(s). Publication rights licensed to ACM.
ACM ISBN 979-8-4007-1467-2/2025/03
https://doi.org/10.1145/3712676.3718341

- **Realistic Network Emulation with Seamless Integration of Containernet and Mahimahi**: A core contribution of this work is the seamless integration of Containernet [17] and Mahimahi [15] to provide accurate and realistic network emulation. Containernet creates complex network topologies and hosts RTC senders and receivers, while Mahimahi delivers trace-driven emulation of network conditions, including bandwidth fluctuations, delays, and packet losses. This integration ensures that each container experiences real-world network conditions in a coherent and synchronized manner, allowing researchers to evaluate CC algorithms under dynamic network scenarios like wired networks, Wi-Fi, LTE/5G, and satellite communications (e.g., Starlink).
- **Support for Diverse Topologies and Multiple Flows**: The test suite supports a variety of complex network topologies, such as dumbbell and parking lot configurations [7], which are crucial for evaluating CC algorithms under conditions with multiple concurrent RTC flows [10]. These topologies enable researchers to test how algorithms perform when multiple flows share resources and compete for bandwidth, providing insights into scalability, fairness, and coexistence.
- **Comprehensive Evaluation and Metrics**: In addition to traditional QoE metrics such as video quality, latency, and stability (e.g., rebuffering ratio), our system incorporates perceptual metrics like VMAF [14] for video quality and DNSMOS [18] for audio quality. We also include network performance metrics such as goodput, packet loss, one-way and self-inflicted delay, enabling a thorough analysis of how CC algorithms perform under various conditions. The combination of perceptual and network metrics provides a complete understanding of algorithm performance, both from a user experience and a network efficiency perspective.
- **Unified Interface for Archiving and Comparison**: Our framework features a unified interface that allows users to archive their CC algorithm and test scenarios, and compare CC algorithms developed using different dockerized frameworks (e.g., AlphaRTC and Pandia). Researchers can either add their source code to a designated folder or provide custom containers or binary files. The test suite automatically invokes the appropriate container based on configuration, ensuring seamless integration and direct comparison of algorithms. This interface fosters collaboration by creating a shared platform for community contributions and advancing RTC CC research.

By adopting trace-driven simulation capability, we have one unique strength in making possible the use of our large *low-earth-orbit (LEO)* satellite testbed, enabling the collection and integration of LEO communication traces into our emulation suite. This feature ensures that our test suite supports both terrestrial and satellite network scenarios. By incorporating these satellite communication traces, researchers can extend the boundaries of current RTC CC research, allowing for the simulation of algorithm performance in both conventional and emerging communication environments. This is particularly valuable as it addresses the challenges posed by next-generation communication technologies, including satellite and hybrid networks.

2 Related Works

WebRTC CC. CC is a critical component of WebRTC, ensuring the smooth delivery of real-time audio and video data over the Internet. One of the most widely adopted CC algorithms for WebRTC is *Google congestion control* (GCC) [3], which uses bandwidth probing to estimate available network capacity and dynamically adjusts the sending rate to prevent congestion. GCC has proven effective in handling variable network conditions and is the default CC mechanism in many WebRTC applications. Beyond GCC, numerous alternative CC algorithms have been proposed to address specific scenarios, such as wireless networks and high-latency environments. Examples include *self-clocked rate adaptation for multimedia (SCReAM)* [11] and candidate solutions proposed by the IETF RMCAT (RTP Media Congestion Avoidance Techniques) working group. These efforts underscore the importance of CC in optimizing WebRTC performance across diverse network environments.

WebRTC CC Development Tools. Traditional CC approaches often rely on fixed algorithmic parameters, limiting their adaptability to the dynamic and heterogeneous nature of the modern Internet. To overcome this limitation, *reinforcement learning (RL)* has recently gained traction in CC, offering the capability to learn and adapt to diverse network conditions. This paradigm shift has been facilitated by the emergence of several open-source tools and frameworks designed explicitly for developing and evaluating WebRTC CC algorithms.

AlphaRTC [5] provides a robust open-source environment for real-time evaluation of CC algorithms, decoupling the bandwidth estimation modules on the receiver side from WebRTC. This modular design enables users to implement RL-based algorithms in Python, estimating available bandwidth based on packet statistics at the receiver. However, while AlphaRTC supports receiver-side bandwidth estimation, it does not provide RL-based algorithms with direct control over the sending rate at the sender side.

In contrast, Pandia [13], a counterpart of AlphaRTC, offers customizable sender-based modules, allowing CC algorithms to directly control the sending rate based on feedback from the receivers. This sender-side customization facilitates the exploration of data-driven methods for achieving more responsive and efficient congestion control. However, unlike AlphaRTC, Pandia lacks support for controlling receiver-side bandwidth estimation modules, thereby limiting its scope in certain scenarios.

Despite their strengths, there are some limitations. First, their built-in evaluation mechanisms are overly simplistic, failing to account for real-world scenarios involving heterogeneous network environments such as Wi-Fi, LTE/5G, and satellite communications. Second, they lack a unified benchmarking framework to systematically compare CC algorithms across platforms and configurations. For instance, if a researcher develops a sender-side algorithm using Pandia, there is no existing framework to benchmark it against a receiver-side CC algorithm implemented with AlphaRTC.

These gaps present significant challenges for researchers seeking to evaluate and identify optimal solutions under diverse network conditions. They highlight the urgent need for a comprehensive and unified test suite capable of addressing these limitations. Thus, our test suite would enable systematic, cross-platform benchmarking of CC algorithms under realistic and varing scenarios. Importantly, the

goal of this work is not to replace tools like AlphaRTC or Pandia but to complement them by providing a unified test suite that enhances benchmarking and evaluation capabilities, fostering a more holistic approach to advancing RTC congestion control.

Network Emulation Tools. Accurate network emulation is vital for evaluating RTC CC algorithms under realistic conditions. Large-scale testbeds like PlanetLab [4], Emulab [21], and ORBIT [16] provide real-world-like environments for testing networking applications. However, they require researchers to reserve resources, which limits accessibility for continuous and collaborative contributions. To address this, our test suite offers an accessible platform where researchers can seamlessly contribute algorithms and evaluate them in both controlled environments and real-world scenarios.

Localized emulation tools such as Dummynet [2, 19], NetEm [8], Mininet [6], and Containernet [17] are widely adopted for their ability to simulate key network attributes like delay, packet loss, and bandwidth constraints. These tools are instrumental in enabling reproducible network experiments but often require manual configuration and lack flexibility for emulating heterogeneous and dynamic environments, including Wi-Fi, LTE/5G, and satellite networks.

Pantheon [23] has advanced the field with a platform for benchmarking transport protocols, supporting diverse scenarios and providing detailed metrics for TCP/IP congestion control. Building on Pantheon's principles, our test suite extends the focus to RTC CC algorithms, incorporating more complex network conditions and metrics tailored to real-time communication.

To achieve this, we integrate Mahimahi's trace-driven emulation [15], which replicates real-world network conditions using Wi-Fi and cellular traces, with Containernet's containerized topology support. This integration ensures a cohesive and automated workflow, allowing researchers to deploy and test RTC CC algorithms in realistic, reproducible environments. By leveraging the strengths of both tools, our platform overcomes the limitations of manual configuration and restricted scenario support in existing emulators.

Our test suite supports community-driven contributions, allowing researchers to add and compare algorithms under unified evaluation conditions. This collaborative approach accelerates innovation and provides a robust framework for benchmarking RTC CC solutions across a wide range of platforms and scenarios.

3 Design and Implementation

This section outlines the design and implementation of our evaluation system for RTC CC algorithms. The primary goal of the system is to provide a unified test suite that facilitates the archiving, comparison, and evaluation of CC algorithms developed using different WebRTC frameworks. By integrating powerful emulation tools and providing a straightforward interface for algorithm integration, the test suite addresses the limitations of existing frameworks and supports diverse network scenarios and topologies. The code is available at https://github.com/Masshiro/PyRTC-dev.

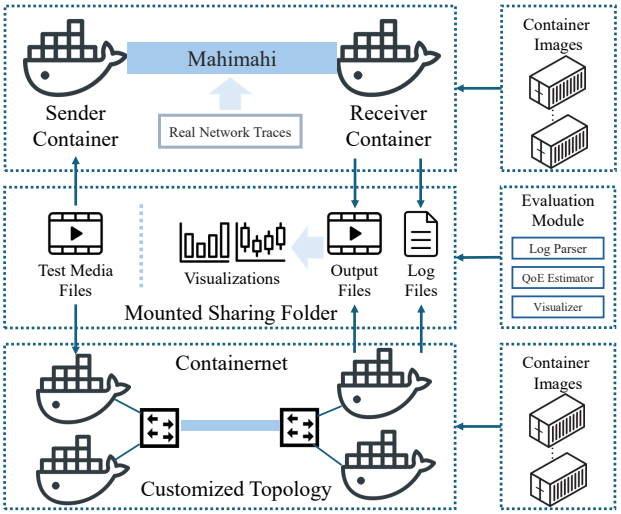


Figure 1: The overview of test suite structure.

3.1 Design Overview

The test suite is designed as a flexible and extensible platform for benchmarking RTC CC algorithms. The core architecture leverages two foundational technologies: Mahimahi and Containernet. Mahimahi provides trace-driven network emulation, simulating real-world network conditions such as Wi-Fi, LTE/5G, and satellite communications. Containernet handles the creation of network topologies and the orchestration of containers running RTC senders and receivers, supporting algorithms developed using various frameworks like AlphaRTC and Pandia. The overview of its structure is illustrated in Figure 1.

To enable seamless integration and comparison, the test suite is designed to support two modes for adding algorithms: direct source code integration and pre-built container execution. It also includes diverse evaluation scenarios, from one-to-one RTC communications to complex topologies with multiple concurrent RTC flows. Metrics are collected for both QoE and detailed network statistics to provide comprehensive insights into algorithm performance.

3.2 Realistic Network Emulation

The system combines the complementary strengths of Containernet [17] and Mahimahi [15] to replicate real-world network environments with high fidelity, forming a robust backend for evaluating RTC CC algorithms.

Containernet provides a versatile environment for creating and emulating complex network topologies. It dynamically generates diverse topologies and establishes communication links between containers, which host RTC senders and receivers running CC algorithms from frameworks like AlphaRTC or Pandia.

While Containernet excels in topology creation, its manual configuration of network parameters such as delay and bandwidth does not fully capture the dynamic nature of real-world conditions. To address this, we integrate Mahimahi into the system, providing trace-driven emulation of network bandwidth. By replaying recorded traces, Mahimahi simulates bandwidth fluctuations by

determining how many MTU-sized packets can leave the queue per millisecond. This approach recreates the characteristics of various network types, including wired connections, Wi-Fi, LTE/5G, and satellite systems like Starlink.

An essential reason for selecting Mahimahi is that its emulation operates at the application level, making it particularly well-suited for container-based architectures. Containers use namespace isolation while sharing the same underlying kernel with the host system. Kernel-based emulation tools, such as Dummynet or NetEm, require complex configurations to achieve the fine-grained per-flow control. However, Mahimahi's application-level approach allows for isolated and accurate emulation within each container, which is an easier way to ensure that RTC senders and receivers in different containers can experience distinct, trace-driven network conditions, enabling fair and realistic algorithm evaluations.

A key innovation of this work lies in achieving seamless synchronization between these tools, enabling each container running within a Mahimahi shell to experience bandwidth fluctuations and delays that accurately reflect the emulated network traces. At the same time, these containers maintain proper connectivity with other nodes in the Containernet topology. This synchronized operation ensures precise and consistent evaluations across a wide range of network scenarios.

3.3 Support for Diverse Topologies and Multiple Flows

Real-world networks are rarely limited to simple one-to-one communication scenarios. They often involve complex topologies and multiple concurrent flows competing for shared resources. Evaluating RTC CC algorithms under these conditions is essential for understanding their scalability, fairness, and ability to coexist with other traffic. Diverse topologies allow researchers to explore how algorithms handle bottlenecks, congestion points, and varying levels of contention, offering a more comprehensive assessment of their performance.

Despite the importance of such evaluations, existing tools are limited in their ability to replicate realistic network scenarios involving complex topologies and multiple RTC flows. Most tools focus on isolated, single-flow testing or require extensive manual configuration to emulate shared bottlenecks or interactions between flows. This gap makes it challenging to systematically assess algorithm behavior in scenarios that closely mirror real-world networking environments.

To address these limitations, our test suite introduces support for diverse topologies and concurrent RTC flows. It includes predefined configurations for commonly studied network setups, such as:

- Dumbbell Topology: Simulating scenarios where multiple senders and receivers share a common bottleneck link, enabling the study of how algorithms distribute resources and handle congestion in shared environments.
- Parking Lot Topology: Creating conditions with varying levels of contention among RTC flows, allowing researchers to evaluate the impact of competing traffic on algorithm performance and the ability to fairly allocate bandwidth.

Recognizing the evolving nature of networking challenges, we are committed to expanding the suite by adding increasingly complex and challenging scenarios. These may include multi-bottleneck setups, hierarchical topologies, or dynamic environments where network conditions change unpredictably. Additionally, we aim to incorporate topologies that reflect emerging technologies, such as 5G ultra-dense networks and satellite-based communication systems.

3.4 Comprehensive Evaluation and Metrics

The test suite enhances traditional evaluation methods by integrating both perceptual quality metrics and network performance metrics, offering a well-rounded assessment of RTC CC algorithm performance. This dual approach balances efficient network utilization with high-quality user experiences.

Perceptual metrics like VMAF [14] and DNSMOS [18], commonly used in frameworks like AlphaRTC, remain core to our evaluation. VMAF, developed by Netflix, measures video quality by combining machine learning models with human subjective ratings, assessing factors like resolution fidelity and compression artifacts. DNSMOS, developed by Microsoft, evaluates audio quality, focusing on background noise suppression and clarity, essential for real-time communication.

Alongside perceptual metrics, we incorporate network performance metrics such as throughput and delay to evaluate how well CC algorithms utilize bandwidth, manage congestion, and maintain stability across various network conditions. This combination provides a comprehensive analysis of algorithmic behavior, capturing both user experience and network efficiency.

By including both types of metrics, the test suite allows researchers to assess trade-offs between video and audio quality versus network performance. For example, an algorithm delivering high video quality but inefficient bandwidth usage may not be optimal, while one that prioritizes throughput over quality may be unsuitable for real-time applications.

Designed to cover a wide range of scenarios, the test suite ensures relevant and comprehensive evaluations. Detailed experimental results and supported metrics are presented in Section 4, illustrating how this dual-metric approach enables a complete understanding of algorithm performance in diverse, realistic conditions.

3.5 Unified Interface for Archiving and Comparison

Another key feature of the test suite is its unified interface, designed to enable users to archive and compare CC algorithms developed using various frameworks. This addresses a critical gap in the research landscape, where frameworks like AlphaRTC and Pandia provide useful tools for algorithm development but lack a unified platform for cross-framework comparison. Without such a platform, researchers face challenges in benchmarking algorithms under consistent conditions, limiting comprehensive and objective performance evaluation. By offering this functionality, the test suite fosters a unified, collaborative research environment.

To facilitate this, the test suite intends to provide two main integration methods. First, users can integrate their algorithms by placing the source code in a dedicated folder within the test

suite. By adhering to a standard configuration format, they specify key details such as the framework used (for example, AlphaRTC). The test suite automatically invokes the appropriate container to execute the algorithm, ensuring smooth and consistent operation across different frameworks. This approach simplifies the process of adding new algorithms and enables systematic benchmarking under comparable conditions.

Second, the test suite allows users to integrate pre-built containers or binary files. Users can provide a container or executable along with a configuration file that specifies execution parameters. The system runs the provided container, executing the sender and receiver components of the algorithm to collect performance metrics for comparative analysis. This flexibility accommodates diverse development workflows and supports a wide range of algorithm implementations, making the suite accessible to researchers with varying technical preferences.

Beyond algorithm archiving, the test suite also allows researchers to contribute and integrate custom test scenarios. This enables users to create new network conditions or topologies and add them to the suite experiment streamline, further enriching the benchmarking process. Researchers can directly compare their algorithms against others under the same conditions, facilitating meaningful comparisons across a range of diverse scenarios.

By offering these integration methods, the test suite not only provides a shared platform for archiving and comparing algorithms but also fosters collaboration by enabling researchers to contribute both their algorithms and test scenarios. This collaborative approach helps overcome fragmentation in the field, encouraging innovation and creating a shared archive of benchmark results that collectively advances research in RTC congestion control.

4 Experiment Flow and Evaluation Metrics

The test suite provides a streamlined and reproducible workflow for benchmarking RTC CC algorithms. Users simply place their algorithm's source code into a designated directory following a standard format. A single Python script automates the entire process, including container deployment, network emulation, and metric evaluation. The results are presented comprehensively, requiring no manual intervention and enabling quick, accurate insights into algorithm performance.

Current Predefined Network Settings. The current test suite supports several predefined network topologies that reflect diverse conditions. These include:

- Simplest Topology: A basic configuration with one sender and one receiver.
- Dumbbell Topology: A shared bottleneck configuration with two senders and two receivers.
- Parking Lot Topology: A setup with three senders and three receivers to test performance under varying levels of contention.

Current Algorithms and Inputs. The repository currently archives three RTC CC algorithms: Dummy, GCC, and HRCC [20]. These serve as baseline implementations for benchmarking, with plans to integrate more open-sourced algorithms as they become available. Dummy is the default bandwidth estimation algorithm provided by AlphaRTC. GCC is Google's congestion control, and HRCC enhances GCC using reinforcement learning.

Experiments are conducted using a publicly available 1280 × 720 conference scene video file [22] for transmission, with results averaged over three independent runs to reduce variability. For the simplest topology, we emulate network conditions using four cellular traces from real-world data [1] and one fixed bandwidth profile. The results shown in Figure 2 indicates that HRCC algorithm balances adaptability and performance better than GCC but exhibits more variability than Dummy. Dummy, while stable, may lack adaptability in fluctuating network conditions, and GCC might be too conservative or inefficient under these scenarios.

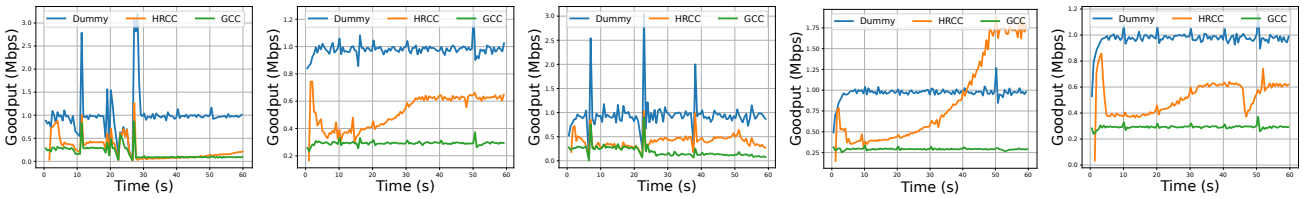
Current Performance Metric. The suite evaluates algorithms using a combination of perceptual and network performance metrics, offering a comprehensive perspective on their effectiveness:

- Perceptual Metrics: These include video quality scores such as VMAF, integrated from AlphaRTC's scoring system. combination of estimation of network and video qualities. The combined score is the weighted summation of S_{net} given by [12] and the S_{video} defined as 100×SSIM [9]. It is denoted as $S_{QoE} = \alpha \cdot S_{net} + \beta \cdot S_{video}$, where α and β are combination weights, and we set $\alpha = \beta = 0.5$ for illustration in Figure 5.
- Network Performance Metrics: These metrics provide additional insights into algorithm behavior under diverse conditions. In addition to the commonly-used metric of goodput over time (Figure 2), we also have two dual-metrics:

(1) Average Self-Inflicted Delay vs. Goodput: This dual-metric examines the trade-off between goodput and the delay introduced by CC algorithms (Figure 3). Goodput refers to the actual data throughput that is successfully transmitted, excluding retransmissions, delays, and other non-productive traffic. It represents the effective utilization of available bandwidth. The Average Self-Inflicted Delay measures the delay caused by the CC algorithm itself, reflecting its efficiency in adapting to network conditions. A good CC algorithm should minimize delay while maximizing goodput. If a high goodput is achieved at the cost of a significant delay, it could indicate that the algorithm sacrifices real-time performance for efficiency. This metric helps in identifying such trade-offs and understanding the algorithm's behavior.

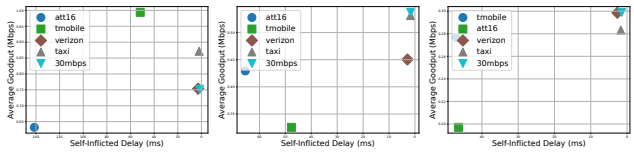
(2) 95th Percentile One-Way Delay and Goodput Comparison: This metric assesses consistency and reliability under peak load conditions (Figure 4). The 95th Percentile One-Way Delay looks at the delay experienced by the slowest 5% of packets, highlighting potential performance degradation during periods of high network congestion. By comparing this delay with goodput, we can determine how well the algorithm sustains throughput while managing congestion spikes. Algorithms that can maintain both low 95th percentile delays and high goodput are better suited for real-time applications, ensuring consistent performance even under challenging network conditions.

For concurrent flows, such as those in dumbbell and parking lot topologies, we plot individual flow goodput to assess fairness



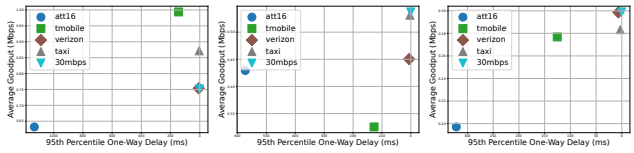
(a) Trace `att16` (b) Trace `taxi` (c) Trace `tmobile` (d) Trace `verizon` (e) Trace `30mbps`

Figure 2: The goodput results for different network traces.



(a) Dummy (b) HRCC (c) GCC

Figure 3: The average self-inflicted delay and goodput comparison between different traces per algorithm.



(a) Dummy (b) HRCC (c) GCC

Figure 4: The average 95th percentile one-way delay and goodput comparison between different traces per algorithm.

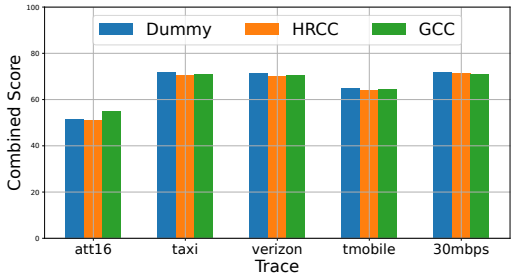
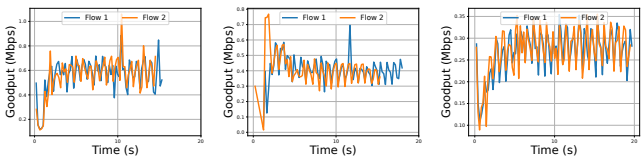
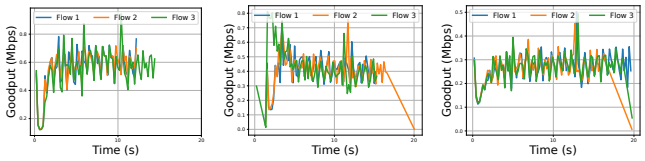


Figure 5: Combined score gained by each algorithm per trace.



(a) Dummy (b) HRCC (c) GCC

Figure 6: The goodput results for multiple flows in dumbbell topology per algorithm.



(a) Dummy (b) HRCC (c) GCC

Figure 7: The goodput results for multiple flows in parking-lot topology per algorithm.

(Figures 6 and 7). Goodput fairness is important because, in multi-flow environments, algorithms should allocate network resources equitably among competing flows without disproportionately favoring some. A result can be regarded as good if the goodput is maximized for all flows while maintaining fairness in their distribution. In a well-performing algorithm, each flow should have relatively equal access to bandwidth, avoiding situations where some flows are starved or excessively prioritized over others. While the results shown feature homogeneous flows, the platform also supports heterogeneous flow evaluations, enabling the comparison of algorithms under more complex, real-world conditions where flows with different bandwidth demands coexist.

5 Discussions and Future Work

Our test suite offers a robust framework for evaluating RTC CC algorithms under diverse network conditions. By integrating trace-driven approaches with container-based topologies, it extends existing emulation tools while providing a unified interface for archiving algorithms and test scenarios. This facilitates unified benchmarking and fosters collaborative research.

Moving forward, we aim to enhance the platform by incorporating more open-source algorithms as they become available and expanding the evaluation metrics to include video freeze rates and receiver-side bit rates. A key focus will be the continued development of satellite communication support. We will leverage our large LEO satellite testbed [24], which provides an extensive resource for studying modern satellite communication networks. We are actively collecting more LEO satellite communication traces to enrich our trace-driven emulation, enabling researchers to evaluate algorithms in emerging satellite network scenarios, as well as to refine our emulation models.

References

[1] Soheil Abbasloo, Chen-Yu Yen, and H Jonathan Chao. 2020. Wanna Make Your TCP Scheme Great for Cellular Networks? Let Machines Do It for You! *IEEE Journal on Selected Areas in Communications (JSAC)* 39, 1 (2020), 265–279.

[2] Marta Carbone and Luigi Rizzo. 2010. Dummynet Revisited. *ACM SIGCOMM Computer Communication Review* 40, 2 (2010), 12–20.

[3] Gaetano Carlucci, Luca De Cicco, Stefan Holmer, and Saverio Mascolo. 2016. Analysis and Design of the Google Congestion Control for Web Real-Time Communication (WebRTC). In *Proc. International Conference on Multimedia Systems (MMSys) (MMSys '16)*. Association for Computing Machinery, New York, NY, USA, Article 13, 12 pages.

[4] Brent Chun, David Culler, Timothy Roscoe, Andy Bavier, Larry Peterson, Mike Wawrzoniak, and Mic Bowman. 2003. Planetlab: An Overlay Testbed for Broad-Coverage Services. *ACM SIGCOMM Computer Communication Review* 33, 3 (2003), 3–12.

[5] Jeongyoon Eo, Zhixiong Niu, Wenxue Cheng, Francis Y Yan, Rui Gao, Jorina Kardhashi, Scott Inglis, Michael Revow, Byung-Gon Chun, Peng Cheng, et al. 2022. OpenNetLab: Open Platform for RL-based Congestion Control for Real-Time Communications. In *Proc. Asia-Pacific Workshop on Networking (APNet)*. Association for Computing Machinery, New York, NY, USA, 70–75.

[6] Nikhil Handigol, Brandon Heller, Vimalkumar Jeyakumar, Bob Lantz, and Nick McKeown. 2012. Reproducible Network Experiments using Container-based Emulation. In *Proc. International Conference on Emerging Networking Experiments and Technologies (CoNEXT)*. Association for Computing Machinery, New York, NY, USA, 253–264.

[7] D. Hayes, D. Ros, L. L. Andrew, and S. Floyd. 2014. *Common TCP Evaluation Suite.* Internet-Draft draft-irtf-iccrg-tcpeval-01. IETF. [Online]. Available: https://datatracker.ietf.org/doc/draft-irtf-iccrg-tcpeval/01/.

[8] Stephen Hemminger et al. 2005. Network Emulation with NetEm. In *linux.conf.au*, Vol. 5. 2005.

[9] Alain Hore and Djemel Ziou. 2010. Image Quality Metrics: PSNR vs. SSIM. In *Proc. International Conference on Pattern Recognition*. IEEE, 2366–2369.

[10] Zhiming Huang, Kaiyang Liu, and Jianping Pan. 2023. End-to-End Congestion Control as Learning for Unknown Games with Bandit Feedback. In *Proc. 2023 IEEE 43rd International Conference on Distributed Computing Systems (ICDCS)*. IEEE, 327–338.

[11] Ingemar Johansson and Roland Bless. 2024. *Self-Clocked Rate Adaptation for Multimedia (SCReAM): Version 2.* Internet-Draft draft-johansson-ccwg-rfc8298bis-screamv2-00. Internet Engineering Task Force (IETF). https://www.ietf.org/archive/id/draft-johansson-ccwg-rfc8298bis-screamv2-00.html Work in Progress.

[12] Sami Khairy, Gabriel Mittag, Vishak Gopal, Francis Y Yan, Zhixiong Niu, Ezra Ameri, Scott Inglis, Mehrsa Golestaneh, and Ross Cutler. 2024. ACM MMSys 2024 Bandwidth Estimation in Real Time Communications Challenge. In *Proc. of the 15th ACM Multimedia Systems Conference*. 339–345.

[13] Xuebing Li, Esa Vikberg, Byungjin Cho, and Yu Xiao. 2024. Pandia: Open-Source Framework for DRL-based Real-Time Video Streaming Control *(MMSys '24)*. Association for Computing Machinery, New York, NY, USA, 299–305.

[14] Zhi Li, Anne Aaron, Ioannis Katsavounidis, Anush Moorthy, and Megha Manohara. 2016. *Toward A Practical Perceptual Video Quality Metric.* Technical Report. Netflix. https://netflixtechblog.com

[15] Ravi Netravali, Anirudh Sivaraman, Somak Das, Ameesh Goyal, Keith Winstein, James Mickens, and Hari Balakrishnan. 2015. Mahimahi: Accurate Record-and-Replay for HTTP. In *Proc. USENIX Annual Technical Conference (ATC)*. USENIX Association, USA, 417–429.

[16] Maximilian Ott, Ivan Seskar, Robert Siraccusa, and Manpreet Singh. 2005. Orbit Testbed Software Architecture: Supporting Experiments as a Service. In *Proc. International Conference on Testbeds and Research Infrastructures for the DEvelopment of NeTworks and COMmunities*. IEEE, USA, 136–145.

[17] M. Peuster, H. Karl, and S. van Rossem. 2016. MeDICINE: Rapid Prototyping of Production-Ready Network Services in Multi-PoP Environments. In *Proc. IEEE Conference on Network Function Virtualization and Software Defined Networks (NFV-SDN)*. IEEE, USA, 148–153. doi:10.1109/NFV-SDN.2016.7919490

[18] Chandan KA Reddy, Vishak Gopal, and Ross Cutler. 2021. DNSMOS: A Non-Intrusive Perceptual Objective Speech Quality Metric to Evaluate Noise Suppressors. In *Proc. IEEE International Conference on Acoustics, Speech and Signal Processing (ICASSP)*. IEEE, Toronto, ON, Canada, 6493–6497.

[19] Luigi Rizzo. 1997. Dummynet: A Simple Approach to the Evaluation of Network Protocols. *ACM Computing Communication Review* 27, 1 (1997), 31–41.

[20] Bo Wang, Yuan Zhang, Size Qian, Zipeng Pan, and Yuhong Xie. 2021. A Hybrid Receiver-Side Congestion Control Scheme for Web Real-Time Communication. In *Proc. ACM Multimedia Systems Conference (MMSys)*. ACM, New York, NY, USA, 332–338.

[21] Brian White, Jay Lepreau, and Shashi Guruprasad. 2003. Lowering the Barrier to Wireless and Mobile Experimentation. *ACM SIGCOMM Computer Communication Review* 33, 1 (2003), 47–52.

[22] Xiph.org Foundation. n.d.. Xiph.org Video Test Media. https://media.xiph.org/video/derf/.

[23] Francis Y Yan, Jestin Ma, Greg D Hill, Deepti Raghavan, Riad S Wahby, Philip Levis, and Keith Winstein. 2018. Pantheon: The Training Ground for Internet Congestion-Control Research. In *Proc. USENIX Annual Technical Conference (ATC)*. USENIX Association, USA, 731–743.

[24] Jinwei Zhao and Jianping Pan. 2024. LENS: A LEO Satellite Network Measurement Dataset. In *Proceedings of the 15th ACM Multimedia Systems Conference*. 278–284.

HockeyOrient: A Dataset for Ice Hockey Player Orientation Classification

Mehdi Houshmand Sarkhoosh
OsloMet & Forzasys
Oslo, Norway

Sushant Gautam
SimulaMet & OsloMet
Oslo, Norway

Cise Midoglu
Forzasys
Oslo, Norway

Saeed Shafiee Sabet
Forzasys
Oslo, Norway

Tomas Kupka
Forzasys
Oslo, Norway

Pål Halvorsen
SimulaMet, OsloMet & Forzasys
Oslo, Norway

Abstract

Understanding player orientation is a critical component of sports analytics, offering insights into gameplay strategies and player behavior. This paper presents *HockeyOrient*, a novel dataset for classifying the orientation of ice hockey players based on their poses. The dataset comprises 9,700 manually annotated frames, selected randomly and non-sequentially, taken from Swedish Hockey League (SHL) games during the 2023 and 2024 seasons. Each player image is cropped from game footage and categorized into one of eight orientation classes: *top*, *top-right*, *right*, *bottom-right*, *bottom*, *bottom-left*, *left*, and *top-left*. The dataset includes diverse scenarios, such as different teams, jersey colors, referees, and goaltenders with unique protective gear. Alongside the dataset, we provide an open-source classification model trained on the dataset using the SqueezeNet architecture, achieving an F1 score of 75% across all the classes. This work addresses a significant gap in ice hockey analytics, enabling advanced player tracking and gameplay analysis. The dataset and the trained model can be accessed publicly on Hugging Face under an open-source license (https://huggingface.co/datasets/SimulaMet-HOST/HockeyOrient).

CCS Concepts

• **Computing methodologies** → **Computer vision**; **Video summarization**; *Activity recognition and understanding*; **Visual inspection**.

Keywords

AI, computer vision, ice hockey, pose estimation, player tracking

ACM Reference Format:
Mehdi Houshmand Sarkhoosh, Sushant Gautam, Cise Midoglu, Saeed Shafiee Sabet, Tomas Kupka, and Pål Halvorsen. 2025. HockeyOrient: A Dataset for Ice Hockey Player Orientation Classification. In *ACM Multimedia Systems Conference 2025 (MMSys '25), March 31-April 4, 2025, Stellenbosch, South Africa.* ACM, New York, NY, USA, 7 pages. https://doi.org/10.1145/3712676.3718342

Permission to make digital or hard copies of all or part of this work for personal or classroom use is granted without fee provided that copies are not made or distributed for profit or commercial advantage and that copies bear this notice and the full citation on the first page. Copyrights for components of this work owned by others than the author(s) must be honored. Abstracting with credit is permitted. To copy otherwise, or republish, to post on servers or to redistribute to lists, requires prior specific permission and/or a fee. Request permissions from permissions@acm.org.
MMSys '25, March 31-April 4, 2025, Stellenbosch, South Africa
© 2025 Copyright held by the owner/author(s). Publication rights licensed to ACM.
ACM ISBN 979-8-4007-1467-2/2025/03
https://doi.org/10.1145/3712676.3718342

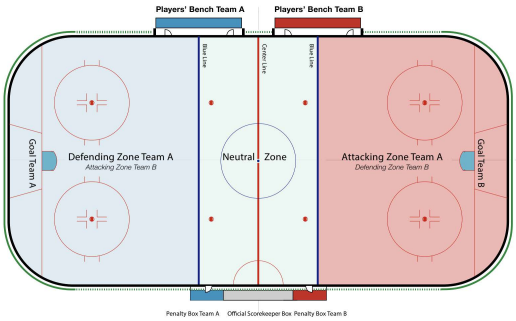


Figure 1: Ice rink zones define player movement, strategy, and specialized areas like the Goalkeeper Restricted Area and face-off circles [11].

1 Introduction

Ice hockey is a fast-paced and dynamic sport that demands advanced analytical tools to capture the intricacies of gameplay [17, 22]. Player orientation, defined as the direction a player is facing, is crucial for understanding on-ice behavior, decision-making, and team strategies [30]. Accurate orientation detection has applications in player tracking, tactical analysis, and event detection, providing valuable insights for coaches, analysts, and broadcasters [16, 18].

Despite the growing interest in computer vision applications for sports, ice hockey remains underexplored due to its unique challenges. The sport combines fast-paced dynamics, complex player interactions, and unique environmental factors. Players move rapidly and unpredictably, navigating multi-directional patterns across a large rink. These frequent directional changes are influenced by puck position, opponent actions, and tactical strategies, making player orientation detection particularly challenging.

Gameplay rules and constraints further complicate analysis. Regulations such as the offside rule force players to adjust their orientation and positioning dynamically. During penalty situations or power plays, movements are restricted to specific zones, directly influencing player orientation. Goalkeepers, confined to the "goalkeeper's restricted area," have distinct pose and movement patterns, adding another layer of complexity.

The ice hockey rink is divided into defensive, neutral, and offensive zones, each shaping player strategies and movements (see Figure 1). In defensive zones, players often orient themselves toward their goal to block shots or intercept passes. In offensive zones,

their orientation is forward-facing, aimed at creating scoring opportunities. The neutral zone is marked by quick directional changes as players transition between offense and defense. These dynamics make player orientation a critical aspect of tactical analysis, requiring datasets and models capable of capturing these nuances.

While existing datasets in other sports, such as SoccerNet [26] and Basketball AI [3, 9], have advanced event detection, player tracking, and action recognition, they fail to address orientation detection and are specific to their respective sports. Ice hockey, with its fast player movements, frequent occlusions, small object sizes, and varying jersey designs, requires domain-specific datasets to overcome these challenges.

To address this gap, we introduce HockeyOrient, a dataset specifically designed for classifying ice hockey player orientation based on their head poses (see Figure 2). As far as we understand, HockeyOrient is the first publicly available dataset of its kind, comprising 9,703 manually annotated frames extracted from SHL [13] games from the 2023 and 2024 seasons. The dataset captures diverse scenarios, ensuring robustness and applicability across various contexts. Alongside HockeyOrient, we provide an open-source classification model trained on the dataset using the SqueezeNet [10] architecture, achieving an F1 score of 75%. Both the dataset and the model are publicly available on Hugging Face to encourage reproducibility and collaboration.

The key contributions of this paper are:

(1) Introducing HockeyOrient, a novel dataset tailored for ice hockey player orientation classification.
(2) Providing a baseline SqueezeNet classification model trained on HockeyOrient to establish benchmark performance.
(3) A publicly available Hugging Face Space that enables real-time inference on HockeyOrient and serves as an interactive demo.
(4) Comprehensive annotation with eight discrete orientation classes, simplifying classification tasks.

2 Background and Related Work

Understanding object orientation is a fundamental challenge in computer vision, closely tied to tasks like pose estimation, spatial reasoning, and action recognition. Accurate orientation estimation is essential for comprehending spatial relationships and behaviors, particularly in dynamic environments. While recent advancements in techniques such as Neural Radiance Fields (NeRFs) [5] have improved pose estimation by leveraging realistic rendering and spatial features, the scarcity of labeled datasets remains a significant bottleneck. Furthermore, challenges such as occlusion and subtle spatial variations emphasize the importance of robust orientation modeling for effective action recognition [31]. The mutual dependency between pose estimation and tasks like action recognition further underscores the need for precise orientation data in domains like sports analytics.

In general contexts, orientation estimation has benefited from synthetic datasets generated by rendering 3D models from diverse viewpoints [32]. However, these datasets and pipelines often lack specificity for domain-specific applications, such as sports. Ice hockey, for example, presents unique challenges like rapid player movement, frequent occlusions, and interactions in crowded environments, which are not adequately addressed by general-purpose solutions. To illustrate this limitation, we ran sample images from HockeyOrient through the OrientAnything pipeline [32], a state-of-the-art model trained on synthetic and general-purpose orientation data. As shown in Figure 3, the predicted orientations were often misaligned with the player's actual pose, as evidenced by the incorrect XYZ coordinate overlays. These results highlight the challenges of applying general-purpose orientation models to ice hockey and reinforce the need for domain-specific datasets like HockeyOrient.

2.1 Existing Datasets and Their Limitations

Several datasets have been developed for player tracking and identification in ice hockey. For instance, the McGill Hockey Player Tracking Dataset (MHPTD) provides annotated NHL broadcast videos for player tracking tasks, following the format of the MOT challenge dataset [34]. Similarly, the VIP-HTD dataset offers a public benchmark for multi-player tracking but lacks orientation annotations [20]. Another notable resource is the Player Identification Tracklet Dataset, which includes annotated bounding boxes and identities but does not cater to orientation detection [29]. These datasets, while invaluable for tracking and identification, fail to address the specific needs of orientation classification.

As mentioned earlier, existing datasets in other sports, such as SoccerNet have advanced event detection, player tracking, and action recognition. However, these resources primarily emphasize positional data and gameplay events, with limited focus on directional orientation. Unlike soccer or basketball, where movements are often linear or predictable, ice hockey players navigate multi-directional patterns. This necessitates orientation estimation models that account for rapid direction changes and nuanced visual cues like jersey designs, referee uniforms, and goaltender equipment, making general-purpose datasets unsuitable for ice hockey and highlighting the need for domain-specific datasets like HockeyOrient.

2.2 Challenges in Ice Hockey Orientation Classification

Ice hockey presents specific challenges for orientation classification:

- **Rapid Movements:** Players exhibit swift and unpredictable movements, making real-time orientation detection complex.
- **Occlusions:** Frequent occlusions occur due to player proximity and equipment, complicating visual analysis.
- **Uniform Similarities:** Players on the same team have similar uniforms, making individual identification and orientation detection more difficult.

These factors underscore the need for specialized datasets and methodologies tailored to this sport's nuances.

2.3 Computer Vision for Sports

Recent advancements in computer vision, including pose estimation and optical flow analysis, have demonstrated potential for action recognition and pose analysis in ice hockey [6, 14]. Techniques such as convolutional neural networks (CNNs) and transformers



Figure 2: Examples of the eight orientation classes in HockeyOrient.

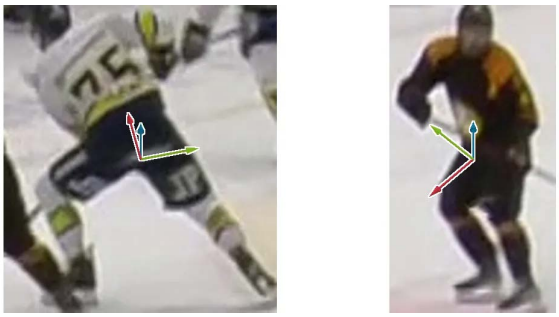


Figure 3: Examples of inaccuracies in the Orient Anything pipeline for ice hockey.

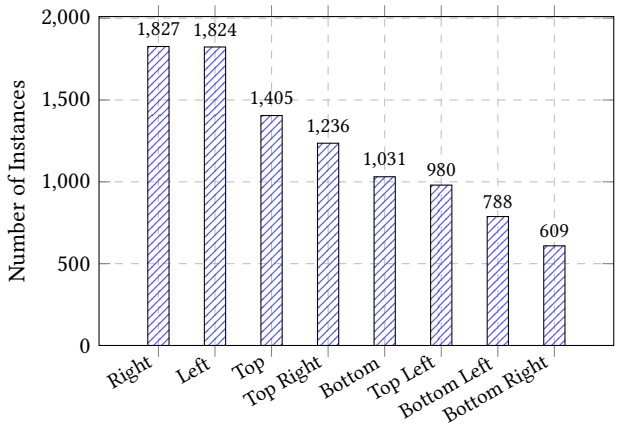


Figure 4: Direction distribution in HockeyOrient.

have been applied to player tracking in sports. For example, Arbués-Sangüesa et al. [1] combined pose estimation with contextual data (e.g., ball position) to estimate player orientation in soccer. Similarly, Vats et al. [29] leveraged weakly supervised learning for player identification in hockey. However, applying these methods to ice hockey requires overcoming challenges posed by the sport's dynamics and environmental factors.

Trajectory prediction models in sports [35] often rely on spatial data to analyze player movement but frequently overlook orientation, a critical element for understanding decision-making and gameplay strategies. By focusing on player orientation, HockeyOrient offers a complementary perspective that enhances the analysis of player behavior and team dynamics, addressing gaps in traditional sports analytics.

HockeyOrient simplifies the task of player orientation by framing it as a classification problem with eight discrete classes, diverging from the continuous angle predictions common in general-purpose datasets [32].

3 The HockeyOrient Dataset

HockeyOrient is, to the best of our knowledge, the first publicly available dataset tailored for classifying the orientation of ice hockey players. It was created by extracting and annotating high-resolution frames from SHL games, resulting in 9,700 images of players, goalkeepers, and referees under diverse gameplay scenarios. The dataset emphasizes diversity in poses, jersey designs, referees, and goaltenders, ensuring robustness across different situations. Key features of HockeyOrient include:

- **Diverse gameplay scenarios:** The frames include key moments such as attacking, defending, neutral play, faceoffs, and varied player poses.
- **Class diversity:** Annotations cover eight orientation classes, providing comprehensive coverage for orientation direction prediction.
- **Diverse jersey colors:** The dataset captures a wide range of jersey colors, accommodating variability across teams, referees, and goalkeepers, thereby ensuring robustness under different lighting and environmental conditions.

3.1 Sources and Collection

In collaboration with SHL and Sportality [27], the dataset was sourced using APIs provided by Forzasys [4] to access M3U8 URLs for SHL games from the 2023 and 2024 seasons. Original videos were encoded in H.264 format with a resolution of 1920×1080 pixels at 50 fps. Frames were extracted and stored as high-quality JPEG files (quality setting of 100) to maintain fidelity.

The data collection process prioritized games tagged with "Shot" events, which often feature wide-angle views encompassing multiple objects such as players, referees, and goalkeepers. From each event, frames were extracted at one-second intervals, complemented by additional frames at regular intervals (e.g., every 500 frames)

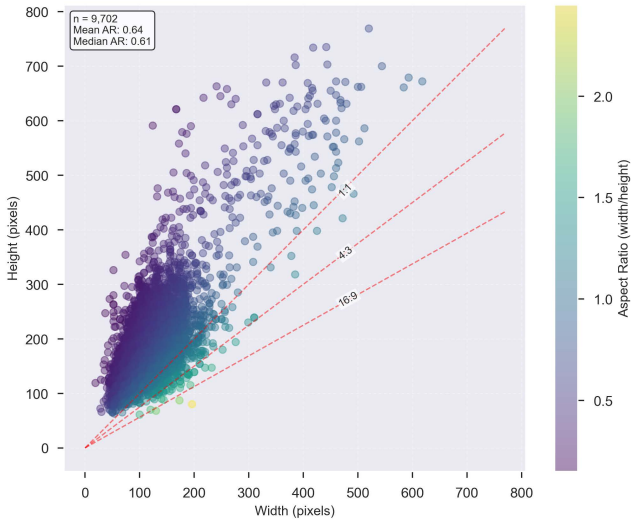


Figure 5: Image dimension distribution in HockeyOrient, with aspect ratios and standard ratios marked.

to capture diverse poses and actions. Manual inspection ensured high-quality frames suitable for cropping.

To isolate individual objects, a custom-trained YOLO-based object detection model, *HockeyAI* [8], was employed. The model, trained on an ice hockey dataset with seven classes (including player, referee, and goalkeeper), provided bounding boxes for cropping. The cropped objects, stored in JPEG format, varied in size depending on their position and proximity to the camera, reflecting real-world gameplay conditions (see Figure 5).

3.2 Annotations

The orientation of each player in HockeyOrient is categorized into one of eight discrete classes, as depicted in Figures 6 and 2. Orientation is determined using the angular displacement of the player's head with respect to a fixed coordinate system aligned with the camera view. The reference point for annotation is the central axis of the player's head, estimated based on helmet alignment and facial orientation. The classification follows a quantized discretization scheme, dividing the 360-degree field of view into eight 45-degree sectors (see Table 1). Although the bounding boxes used for annotation, as shown in Figure 2, encompass the entire body, orientation classification is based solely on the player's head direction. These full-body bounding boxes are generated using an automated object detection model [8] for player localization and provide spatial context for downstream tasks such as player pose estimation and action recognition. However, the orientation labels remain independent of body posture; if a player's head and torso face different directions (e.g., a player skating left while looking right), the annotation is assigned based strictly on head direction to maintain consistency.

The annotations were carried out manually with the aid of Label Studio [28] and subsequently reviewed to ensure consistency throughout the dataset. Figure 4 illustrates the class distribution in HockeyOrient, showing the frequency of each orientation class across the dataset.

Table 1: Orientation classes and corresponding angular ranges in HockeyOrient.

Class	Angle Range	Description
Top	$0^\circ \pm 22.5^\circ$	Facing directly upward
Top Right	$45^\circ \pm 22.5^\circ$	Head rotated toward the top-right
Right	$90^\circ \pm 22.5^\circ$	Facing directly right
Bottom Right	$135^\circ \pm 22.5^\circ$	Oriented downward-right
Bottom	$180^\circ \pm 22.5^\circ$	Facing directly downward
Bottom Left	$225^\circ \pm 22.5^\circ$	Oriented downward-left
Left	$270^\circ \pm 22.5^\circ$	Facing directly left
Top Left	$315^\circ \pm 22.5^\circ$	Head rotated toward the top-left

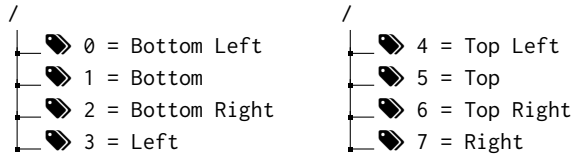


Figure 6: Class index mapping for the eight canonical directions.

3.3 Public Dataset

HockeyOrient is hosted on Hugging Face under an open-source license. The dataset is organized with a clear directory structure, facilitating ease of use for researchers and developers. Its structure is illustrated in Figure 7.

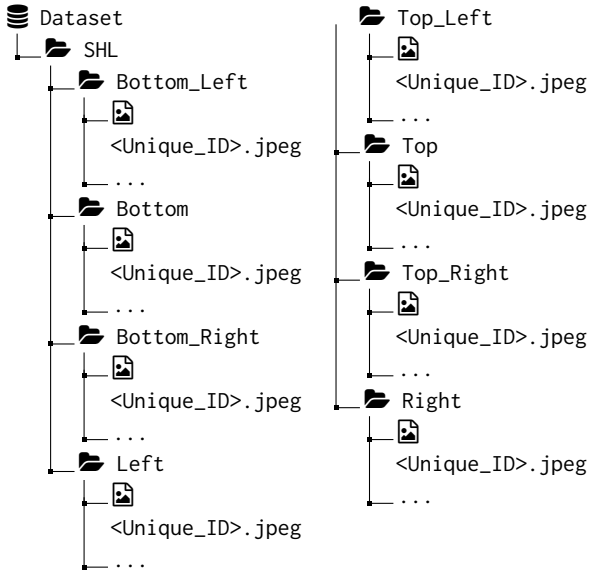
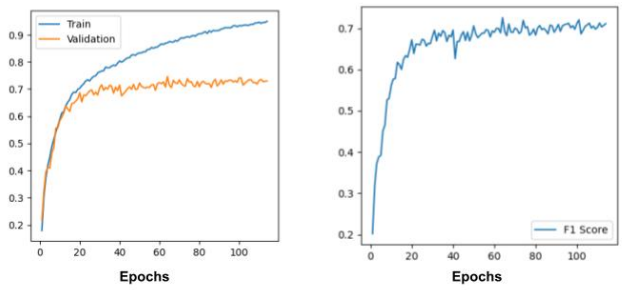


Figure 7: HockeyOrient directory structure.

3.4 Model Performance

A baseline classification model, based on the SqueezeNet architecture, achieved a 75% F1 score on the test set, demonstrating the



(a) Training Accuracy (b) F1 Score

Figure 8: Training accuracy and F1 score over epochs. (a) Accuracy progression, (b) F1 score balance.

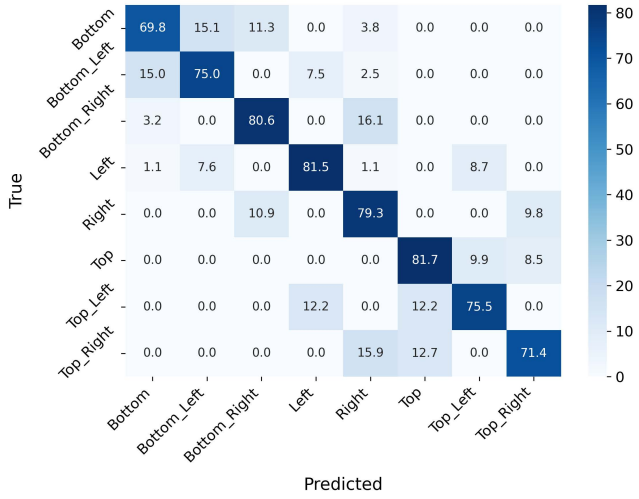


Figure 9: Confusion matrix for player directions. Rows sum to 100%, with darker blue indicating higher accuracy.

dataset's effectiveness for player orientation classification. Figure 8 illustrates training accuracy (a) and F1 score (b) across epochs, highlighting the model's learning progression and balance between precision and recall. The confusion matrix (Figure 9) shows an 81.7% accuracy for *Top* class, while underrepresented classes, such as *Bottom* reached 69.8%, indicating the need for better class balance in future dataset expansions. A sample classification prediction is shown in Figure 2. The model is available on Hugging Face under an open-source license[1].

4 Dataset Applications

The HockeyOrient dataset enables a wide range of applications in sports analytics and computer vision. Its unique focus on player orientation detection facilitates novel use cases across gameplay analysis, tactical planning, and broader research in pose estimation and activity recognition. This section highlights key areas where HockeyOrient can make significant contributions.

[1]https://huggingface.co/SimulaMet-HOST/HockeyOrient

Gameplay Analysis and Tactical Insights: Player orientation plays a critical role in understanding gameplay dynamics [21, 24, 25]. By analyzing player orientations during key moments such as shots, passes, or defensive maneuvers, HockeyOrient enables fine-grained analysis of decision-making and strategy [12]. For example, orientation data can help identify patterns in offensive and defensive formations, shedding light on team strategies and individual player behavior [2]. Additionally, tracking player orientations over time can reveal trends in reaction times, situational awareness, and adaptation to in-game conditions.

Pose Estimation and Tracking Enhancements: HockeyOrient provides a robust foundation for developing pose estimation models tailored to the unique dynamics of ice hockey [19]. Annotated examples of players, referees, and goalkeepers enable algorithms to better understand player movements and interactions. For instance, analyzing player poses when hitting the puck with a stick can offer valuable insights into how different stances contribute to puck speed and trajectory. Referee tracking can also benefit from pose data, enabling automated detection of fouls and ensuring fair gameplay (see Figure 10).

Game Clipping and Highlight Generation: An application of HockeyOrient is automating game clipping and highlights for social media [22, 23]. Using player orientation from the HockeyOrient model, cone projections show attention directions for players, referees, and goalkeepers, overlaid on video to create heatmaps of shared focus. Overlapping cones highlight critical gameplay areas (see Figure 11), like when players focus on the puck or a goalkeeper aligns with an attacker. These high-action zones guide dynamic video reframing, centering key moments for platforms like Instagram, TikTok, or Twitter. Utilizing HockeyOrient and related models, content creators can produce engaging highlights that capture the game's essence [7].

Broader Applications in Computer Vision: Beyond sports, HockeyOrient can contribute to general-purpose research in orientation detection and activity recognition [15]. The dataset's structured annotations and diverse scenarios provide a foundation for exploring cross-domain transfer learning, particularly for tasks involving small objects or occlusions [33]. It also complements existing datasets in robotics, surveillance, and autonomous systems, where orientation is a key factor in understanding human behavior or object dynamics.

5 Discussion

The HockeyOrient dataset marks a significant advancement in the domain of sports analytics by addressing a previously underserved area: ice hockey player orientation classification. Through its robust annotation framework, diverse scenarios, and tailored design, HockeyOrient provides a foundation for applications ranging from tactical analysis to automated game clipping. However, the scope of the dataset also reveals opportunities for improvement, particularly in terms of size and contextual diversity.

One of the primary challenges encountered during the dataset's development was its initial size. Early iterations of the dataset comprised approximately 2,000 player images, each cropped from relatively small regions of the original video frames. The limited



Clipping Puck in the net Penalty shot Boarding Charging

Figure 10: Referee poses signaling fouls, stoppages, and game events, useful for automated officiating [11].

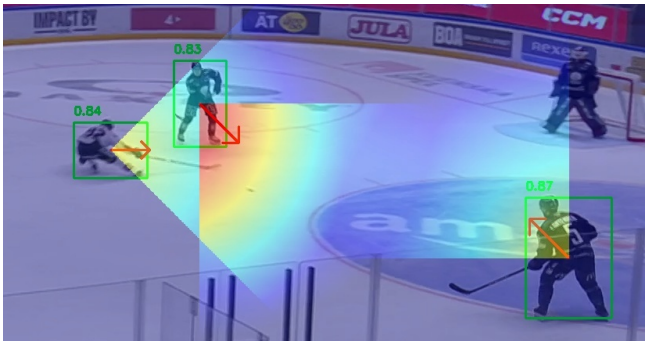


Figure 11: Automated game clipping in HockeyOrient using cone-based projections and heatmaps for key moments.

size and lack of contextual information in these single-player images made it difficult for machine learning models to effectively learn meaningful features. As a result, the dataset underwent three iterative expansions, ultimately reaching its current size of 9,700 annotated images. This expansion not only increased the diversity of scenarios but also significantly improved model performance, achieving an F1 score of 75%. These results underscore the importance of dataset size and diversity in enhancing model learning. Future efforts could focus on further expanding the dataset to include more diverse scenarios and gameplay elements, which could yield even better results.

Another challenge in this task is selecting an appropriate model for classifying small, cropped images. We evaluated multiple architectures, including ResNet-18, ResNet-50, MobileNetV1, and SqueezeNet, on HockeyOrient. Table 2 shows that lightweight models outperform deeper architectures, with SqueezeNet achieving the highest F1 score at 75%. In contrast, ResNet-50 performed the worst at 26%, highlighting the inefficiency of deeper models for this dataset. These results indicate that compact architectures like SqueezeNet and MobileNetV1 are better suited for HockeyOrient, where image resolution and feature complexity are limited. The performance gap suggests that deeper networks struggle to extract meaningful features from the dataset, reinforcing the need for efficient models in this context. While HockeyOrient captures player orientation effectively, it does not incorporate additional factors such as interactions between players in close proximity. These factors could provide deeper insights into team dynamics and individual player behavior. For example, body posture beyond head orientation could reveal more nuanced movements, such as preparatory actions before a pass or shot. Additionally, analyzing interactions among players could facilitate a better understanding of team strategies and coordination. Future work could explore the development of multi-modal datasets that integrate orientation data with other gameplay features, such as puck trajectory and team formations. Such datasets would enable richer analytical capabilities, bridging the gap between individual player analysis and broader tactical insights.

Table 2: Baseline classification performance on HockeyOrient.

Model	F1 Score (%)
ResNet-50	26.0
ResNet-18	38.0
MobileNetV1	57.0
SqueezeNet	75.0

6 Conclusion

This paper presents HockeyOrient, the first publicly available dataset dedicated to ice hockey player orientation classification. By offering high-quality annotations across eight orientation classes and emphasizing diversity in scenarios, HockeyOrient establishes a robust foundation for advancing computer vision applications in sports. The dataset has demonstrated its utility through a baseline classification model, achieving an F1 score of 75%, which validates its quality and relevance for orientation-based tasks.

The contributions of HockeyOrient extend beyond ice hockey, offering methodologies and insights that are adaptable to other sports and general-purpose pose estimation problems. Its release as an open-source resource fosters collaboration and reproducibility, encouraging further innovation in sports analytics, video summarization, and automated officiating. Future work will focus on expanding the dataset, exploring more advanced models, and enabling real-time applications to bridge the gap between analytical insights and practical deployment in live scenarios.

The introduction of HockeyOrient represents a significant step forward in leveraging AI for enhancing understanding, engagement, and performance in sports, setting the stage for broader impacts in computer vision and beyond.

Acknowledgments

This research was funded by the Research Council of Norway, projects #354154 (Smart Sports Studio) and #346671 (AI-Storyteller).

References

[1] Adrià Arbués-Sangüesa, Adrián Martín, Javier Fernández, Coloma Ballester, and Gloria Haro. 2020. Using Player's Body-Orientation to Model Pass Feasibility in Soccer. https://doi.org/10.48550/arXiv.2004.07209 arXiv:2004.07209

[2] Adrià Arbués-Sangüesa, Adrián Martín, Javier Fernández, Coloma Ballester, and Gloria Haro. 2020. Using Player's Body-Orientation to Model Pass Feasibility in Soccer. https://doi.org/10.48550/arXiv.2004.07209 arXiv:2004.07209

[3] Adrià Arbués-Sangüesa, Gloria Haro, and Coloma Ballester. 2019. Multi-Person tracking by multi-scale detection in Basketball scenarios. arXiv:1907.04637 [cs.CV] https://arxiv.org/abs/1907.04637

[4] Forzasys AS. 2025. Forzasys DIGITALIZING SPORTS. https://forzasys.com/.

[5] Varun Burde, Artem Moroz, Vit Zeman, and Pavel Burget. 2024. Object Pose Estimation Using Implicit Representation For Transparent Objects. arXiv:2410.13465 [cs.CV] https://arxiv.org/abs/2410.13465

[6] Zixi Cai, Helmut Neher, Kanav Vats, David Clausi, and John Zelek. 2018. Temporal Hockey Action Recognition via Pose and Optical Flows. https://doi.org/10.48550/arXiv.1812.09533 arXiv:1812.09533

[7] Sayed Mohammad Majidi Dorcheh, Mehdi Houshmand Sarkhoosh, Cise Midoglu, Saeed Shafiee Sabet, Tomas Kupka, and Michael A. Riegler. 2024. SmartCrop: AI-Based Cropping of Soccer Videos. , 11–13 pages. https://doi.org/10.1109/ISM59092.2023.00009

[8] Mehdi Houshmand Sarkhoosh, Sushant Gautam, Cise Midoglu, Saeed Shafiee Sabet, Tomas Kupka, and Pål Halvorsen. 2025. HockeyAI: A Multi-Class Ice Hockey Dataset for Object Detection. https://huggingface.co/SimulaMet-HOST/HockeyAI. In *Proceedings of the 16th ACM Multimedia Systems Conference* (Stellenbosch, South Africa). Association for Computing Machinery, New York, NY, USA. https://doi.org/10.1145/3712676.3718335

[9] Q. Hu, A. Scott, C. Yeung, et al. 2024. Basketball-SORT: an association method for complex multi-object occlusion problems in basketball multi-object tracking. , 86281–86297 pages. https://doi.org/10.1007/s11042-024-20360-2

[10] Forrest N. Iandola, Song Han, Matthew W. Moskewicz, Khalid Ashraf, William J. Dally, and Kurt Keutzer. 2016. SqueezeNet: AlexNet-level accuracy with 50x fewer parameters and <0.5MB model size. arXiv:1602.07360 [cs.CV] https://arxiv.org/abs/1602.07360

[11] International Ice Hockey Federation. 2024. IIHF Official Rulebook 2024/25 – Version 1.0, May 2024. https://blob.iihf.com/iihf-media/iihfmvc/media/contentimages/4_sport/officiating/rule_book/24_25/2024_iihf_rulebook_24052024_v1.pdf. Accessed: January 8, 2025.

[12] Leonardo Lamas, Rene Drezner, Guilherme Otranto, and Junior Barrera. 2018. Analytic method for evaluating players' decisions in team sports: Applications to the soccer goalkeeper. *PLoS One* 13, 2 (Feb. 2018), e0191431. https://doi.org/10.1371/journal.pone.0191431

[13] Professional Swedish Hockey Leauge. 2025. https://www.shl.se/.

[14] Jamie Menjay Lin, Jisoo Jeong, Hong Cai, Risheek Garrepalli, Kai Wang, and Fatih Porikli. 2024. SciFlow: Empowering Lightweight Optical Flow Models with Self-Cleaning Iterations. , 2162–2171 pages.

[15] Yixin Liu, Yiyang Han, Huayi Zhou, and Jiawen Li. 2024. Joint Multi-person Body Detection and Orientation Estimation Via One Unified Embedding. In *Pattern Recognition and Computer Vision*. Springer, Singapore, 467–480. https://doi.org/10.1007/978-981-97-8692-3_33

[16] Adrien Maglo, Astrid Orcesi, Julien Denize, and Quoc Cuong Pham. 2023. Individual Locating of Soccer Players from a Single Moving View. https://doi.org/10.3390/s23187938

[17] Mohammad Majidi, Mehdi Houshmand Sarkhoosh, Cise Midoglu, Saeed S. Sabet, Tomas Kupka, Dag Johansen, and Pål Halvorsen. 2024. SmartCrop-H: AI-Based Cropping of Ice Hockey Videos. In *Proceedings of the 15th ACM Multimedia Systems Conference* (Bari, Italy) *(MMSys '24)*. Association for Computing Machinery, New York, NY, USA, 471–477. https://doi.org/10.1145/3625468.3652195

[18] Amir M. Mansourian, Vladimir Somers, Christophe De Vleeschouwer, and Shohreh Kasaei. 2023. Multi-task Learning for Joint Re-identification, Team Affiliation, and Role Classification for Sports Visual Tracking. In *Proceedings of the 6th International Workshop on Multimedia Content Analysis in Sports* (Ottawa ON, Canada) *(MMSports '23)*. Association for Computing Machinery, New York, NY, USA, 103–112. https://doi.org/10.1145/3606038.3616172

[19] Guilherme de Sousa Pinheiro, Xing Jin, Varley Teoldo Da Costa, and Martin Lames. 2022. Body Pose Estimation Integrated With Notational Analysis: A New Approach to Analyze Penalty Kicks Strategy in Elite Football. *Front. Sports Act. Living* 4 (March 2022), 818556. https://doi.org/10.3389/fspor.2022.818556

[20] Harish Prakash, Yuhao Chen, Sirisha Rambhatla, David Clausi, and John Zelek. 2024. VIP-HTD: A Public Benchmark for Multi-Player Tracking in Ice Hockey.

[21] Zhiqiang Pu, Yi Pan, Shijie Wang, Boyin Liu, Min Chen, Hao Ma, and Yixiong Cui. 2024. Orientation and Decision-Making for Soccer Based on Sports Analytics and AI: A Systematic Review. *IEEE/CAA J. Autom. Sin.* 11, 1 (2024), 37–57. https://doi.org/10.1109/JAS.2023.123807

[22] Mehdi Houshmand Sarkhoosh, Sayed Mohammad Majidi Dorcheh, Cise Midoglu, Saeed Shafiee Sabet, Tomas Kupka, Dag Johansen, Michael A. Riegler, and Pål Halvorsen. 2024. AI-Based Cropping of Ice Hockey Videos for Different Social Media Representations. *IEEE Access* 12 (2024), 118227–118249. https://doi.org/10.1109/ACCESS.2024.3449152

[23] Mehdi Houshmand Sarkhoosh, Sayed Mohammad Majidi Dorcheh, Cise Midoglu, Saeed Shafiee Sabet, Tomas Kupka, Dag Johansen, Michael A. Riegler, and Pål Halvorsen. 2024. AI-Based Cropping of Soccer Videos for Different Social Media Representations. In *MultiMedia Modeling*. Springer, Cham, Switzerland, 279–287. https://doi.org/10.1007/978-3-031-53302-0_22

[24] Mehdi Houshmand Sarkhoosh, Sushant Gautam, Cise Midoglu, Saeed Shafiee Sabet, and Pål Halvorsen. 2024. Multimodal AI-Based Summarization and Storytelling for Soccer on Social Media. In *Proceedings of the 15th ACM Multimedia Systems Conference* (Bari, Italy) *(MMSys '24)*. Association for Computing Machinery, New York, NY, USA, 485–491. https://doi.org/10.1145/3625468.3652197

[25] Mehdi Houshmand Sarkhoosh, Sushant Gautam, Cise Midoglu, Saeed Shafiee Sabet, Thomas Torjusen, and Pål Halvorsen. 2024. The SoccerSum Dataset for Automated Detection, Segmentation, and Tracking of Objects on the Soccer Pitch. In *Proceedings of the 15th ACM Multimedia Systems Conference* (Bari, Italy) *(MMSys '24)*. Association for Computing Machinery, New York, NY, USA, 353–359. https://doi.org/10.1145/3625468.3652180

[26] Mohieddine Amine Silvio Giancola. 2018. SoccerNet: A Scalable Dataset for Action Spotting in Soccer Videos. , 1792–179210 pages. https://doi.org/10.1109/cvprw.2018.00223

[27] Sportality. 2025. Sportality - Web publishing for sports. https://www.sportality.se/.

[28] Label Studio. 2025. https://labelstud.io/.

[29] Kanav Vats, William McNally, Pascale Walters, David A. Clausi, and John S. Zelek. 2021. Ice hockey player identification via transformers and weakly supervised learning. https://doi.org/10.48550/arXiv.2111.11535 arXiv:2111.11535

[30] Kanav Vats, Pascale Walters, Mehrnaz Fani, David A. Clausi, and John Zelek. 2021. Player Tracking and Identification in Ice Hockey. https://doi.org/10.48550/arXiv.2110.03090 arXiv:2110.03090 [cs.CV]

[31] Bo Wan, Desen Zhou, Yongfei Liu, Rongjie Li, and Xuming He. 2019. Pose-Aware Multi-Level Feature Network for Human Object Interaction Detection. , 9468-9477 pages. https://doi.org/10.1109/ICCV.2019.00956

[32] Zehan Wang, Ziang Zhang, Tianyu Pang, Chao Du, Hengshuang Zhao, and Zhou Zhao. 2024. Orient Anything: Learning Robust Object Orientation Estimation from Rendering 3D Models. arXiv:2412.18605 [cs.CV] https://arxiv.org/abs/2412.18605

[33] Xinkai Xu, Hailan Zhang, Yan Ma, Kang Liu, Hong Bao, and Xu Qian. 2023. TranSDet: Toward Effective Transfer Learning for Small-Object Detection. *Remote Sens.* 15, 14 (July 2023), 3525. https://doi.org/10.3390/rs15143525

[34] Kua Chen Yingnan Zhao, Zihui Li. 2020. A Method for Tracking Hockey Players by Exploiting Multiple Detections and Omni-Scale Appearance Features.

[35] Stephan Zheng, Yisong Yue, and Patrick Lucey. 2017. Generating Long-term Trajectories Using Deep Hierarchical Networks. arXiv:1706.07138 [cs.LG] https://arxiv.org/abs/1706.07138



PCVD: A Dataset of Point Cloud Video for Dynamic Human Interaction

Jie Li
Hefei University of Technology
Hefei, Anhui, China
lijie@hfut.edu.cn

Shujiao Chen, Qiyue Li
Hefei University of Technology
Hefei, China
liqiyue@mail.ustc.edu.cn

Zhi Liu
The University of Electro-Communications
Tokyo, Japan
liu@ieee.org



Figure 1: Sample frames from the video sequence in the PCVD dataset.

Abstract

Point cloud is widely used in computer vision and augmented reality for representing 3D information of real-world scenes. However, challenges such as noise, incompleteness, and quality variations, particularly in dynamic environments, hinder effective processing and analysis. These issues are further complicated in human activity scenarios due to motion and changing lighting. To address these challenges, this paper introduces a point cloud video dataset *PCVD* captured with synchronized Azure Kinect cameras, designed to support tasks like denoising, segmentation, and motion recognition in single and multi-person scenes. It provides high-quality depth and color data from diverse real-world scenes with human actions. We compare it with existing datasets, and the results show its superiority in uniformity and completeness, making it ideal for dynamic environments. We also evaluate the state-of-the-art denoising schemes on this dataset to demonstrate the practicality and sophistication of the dataset. The dataset is publicly available at https://github.com/Atlantichan/PCVD-A-Dataset-of-Point-Cloud-Video-for-Dynamic-Human-Interaction.

CCS Concepts

• **Information systems** → **Multimedia information systems**;
• **Networks** → *Network performance evaluation.*

Permission to make digital or hard copies of all or part of this work for personal or classroom use is granted without fee provided that copies are not made or distributed for profit or commercial advantage and that copies bear this notice and the full citation on the first page. Copyrights for components of this work owned by others than the author(s) must be honored. Abstracting with credit is permitted. To copy otherwise, or republish, to post on servers or to redistribute to lists, requires prior specific permission and/or a fee. Request permissions from permissions@acm.org.
MMSys '25, March 31st–April 4th, 2025, Stellenbosch, South Africa
© 2025 Copyright held by the owner/author(s). Publication rights licensed to ACM.
ACM ISBN 979-8-4007-1467-2/25/03
https://doi.org/10.1145/3712676.3718343

Keywords

Point Cloud, Dataset, Volumetric Video, Interactive Media

ACM Reference Format:
Jie Li, Shujiao Chen, Qiyue Li, and Zhi Liu. 2025. PCVD: A Dataset of Point Cloud Video for Dynamic Human Interaction. In *Proceedings of MMSys '25*. ACM, New York, NY, USA, 7 pages. https://doi.org/10.1145/3712676.3718343

1 Introduction

Point cloud video, an emerging media format, captures and reconstructs dynamic 3D objects from real-world scenes, making it ideal for immersive applications like augmented and virtual reality [1, 2]. Its simple structure and efficient processing enable real-time applications, and ongoing advancements in resolution and processing speed position it as a key technology in various fields.

However, challenges such as large data volumes and high computational demands persist, especially in denoising, segmentation, and real-time processing [3, 4]. While simulation data offers synthetic scenes, it lacks the complexity and real-world characteristics (e.g., lighting changes, occlusions) that make real-world data indispensable for research. The limited availability of high-quality real-world datasets underscores the need for more comprehensive datasets to advance research [5–7].

Current dynamic point cloud video datasets, such as FSVVD [8], HUMAN4D [9], and CWIPC-SXR [10], have notable limitations. In particular, FSVVD suffers from low point cloud density and incomplete data; HUMAN4D has motion blur and limited diversity; and CWIPC-SXR contains missing regions, affecting visual completeness. These issues restrict the applicability of these datasets in high-precision tasks.

To address these limitations, we introduce *PCVD*, a new point cloud video dataset captured with six synchronized Azure Kinect

cameras in a 4m × 4m space for single and multiple-person scenes. The cameras are connected via a sync cable, and a specialized tool is then employed to ensure precise synchronization and spatial calibration. Six volunteers perform various activities to ensure diverse data. The raw data is processed, optimized, and converted into .obj files, producing synchronized depth maps, RGB images, and 3D point clouds for various purposes.

Our dataset offers key advantages: higher point cloud density, integration of RGB and depth data for richer spatial and visual information, and optimal lighting for accurate color reproduction. It considers multi-person interactions and complex scenes, making it valuable for social behavior analysis and other advanced use cases as well. Note that the addition of three-person interactions increases the complexity of point cloud processing, introducing challenges such as occlusion and overlap. However, these are particularly critical for tasks like pose estimation, denoising, and segmentation, as multi-person interactions are more complex than two-person scenarios. Our dataset offers high-density point cloud data, enabling more precise capture and modeling of these interactions. Each sequence is 20-30 seconds long, balancing research needs with manageable data processing. Our dataset is publicly available, and can be accessed through the following link[1].

The rest of the paper is organized as follows: Section 2 reviews related work and compares existing datasets; Section 3 details the dataset creation process; Section 4 discusses dataset characteristics and comparisons; Section 5 demonstrates its applications; Section 6 concludes the work.

2 Related Work

Point cloud video datasets have gained attention for advancing technologies like compression, reconstruction and streaming [14–17], particularly in human activity analysis for tasks such as pose estimation, action recognition, and interaction analysis. They can be classified into three categories, as shown in Table 1.

Single-person Point Cloud Video Datasets: Microsoft Voxelized Upper Bodies [11] is an early dataset focusing on upper-body motion using RGB-D sensors. It includes various poses and actions for pose estimation and action recognition but is limited to upper-body movements, reducing its applicability in dynamic interaction scenarios.

Large-scale Point Cloud Video Datasets: Datasets like 8iVFB v2 [13] and Owlii [12] cater to large-scale point cloud processing, offering high-resolution human pose data, action sequences, multi-view captures, and fine details. However, their focus on single-person actions and the limited number of sequences restrict their applicability in collaborative and interactive tasks.

Multi-person Interaction and Complex Scene Point Cloud Video Datasets: FSVVD [8], HUMAN4D [9], and CWIPC-SXR [10] capture multi-person interactions in complex scenes, making them more suitable for analyzing behaviors and human performance. FSVVD offers 26 sequences recorded with six Azure Kinect devices, while HUMAN4D features 19 actions from four subjects captured with 24 Vicon cameras and 4 Intel RealSense sensors. CWIPC-SXR records 21 sequences with seven Azure Kinect cameras in an SXR environment.

Compared to existing datasets such as FSVVD, HUMAN4D, and CWIPC, our dataset offers significant advantages in key areas. By using synchronized depth cameras to capture both RGB and depth information, our dataset improves the precision of spatial and visual data, enabling more accurate 3D reconstruction and scene depth perception. This provides high-quality foundational data for advanced learning models and computer vision tasks [18–21]. In contrast to FSVVD, which suffers from low-density point clouds and incomplete details, HUMAN4D, which is constrained by a limited number of participants and motion blur in fast movements, and CWIPC, which exhibits low visual quality and blocky appearance due to missing patches, our dataset excels in point cloud integrity, detail representation, and visual clarity. Additionally, the high-density point cloud data allows for finer 3D reconstruction of scenes and objects, capturing subtle details of actions and expressions, thereby enhancing behavior recognition accuracy. The advantages of our dataset in these aspects are shown in Fig 2.

3 Data Collection and Processing

During data acquisition, we design a custom capture tool to synchronize multiple Azure Kinect devices for simultaneous depth and RGB capture, ensuring consistent data with unified timestamps. A data processing tool converts raw .obj data to .ply format. Six Azure Kinect cameras are deployed, each connected to a powerful PC for real-time processing. The workflow is shown in Fig. 3.

3.1 Software Setup

To facilitate the processing and manipulation of point cloud data, we have adopted a customized suite of tools for synchronizing multiple Azure Kinect devices and processing the captured data. The suite ensures accurate and consistent capture of depth maps and RGB images with synchronized timestamps for high-quality 3D reconstruction.

The captured raw data undergoes a series of transformations. Initially, spatial calibration and registration align the depth data from multiple devices into a unified 3D coordinate system, correcting for any spatial discrepancies. Following calibration, the depth data is combined with RGB images for segmentation and boundary detection. The depth and color information are then merged into point cloud data, where further optimization is applied, including noise reduction and artifact removal.

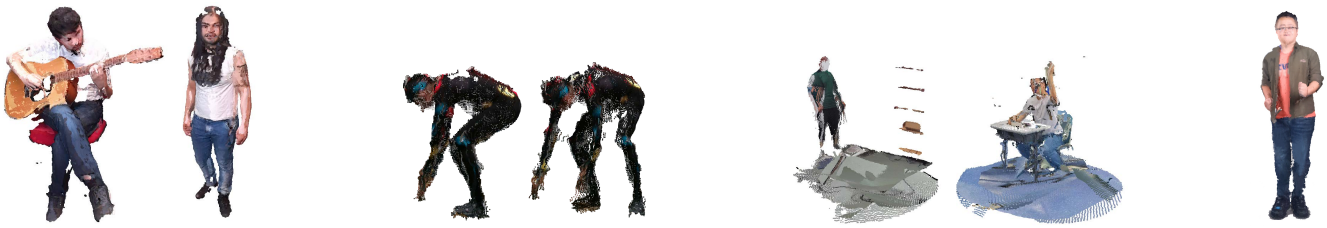
For performance-critical tasks such as data acquisition, synchronization, and calibration, the core of the suite is implemented in C++. For data processing, optimization, and analysis, we utilize Python due to its simplicity and powerful libraries. After optimization, the processed point cloud data is converted into a standardized .obj file format, containing essential information such as vertices, normals, and texture coordinates. These files are suitable for subsequent analysis, rendering, and various applications, providing high-quality 3D models for further use.

Using this processing tool, the point cloud data from all depth maps and images were merged, optimized, and filtered to remove noise and artifacts, thereby improving the details and clarity of the

[1]Dataset URL: https://github.com/Atlantichan/PCVD-A-Dataset-of-Point-Cloud-Video-for-Dynamic-Human-Interaction

Table 1: Characteristics of related datasets

	3D Mesh	Density	Full Human Body	Interactive Objects	Single-person scene	Two-person scene	Three-person scene	Videos	Categories
MVUB [11]	-	Medium	-	-	✓	-	-	10	5
Owlii [12]	-	High	✓	✓	✓	-	-	4	4
8iVFB v2 [13]	-	High	✓	✓	✓	-	-	4	4
FSVVD [8]	✓	Low	✓	✓	✓	✓	-	26	4
HUMAN4D [9]	✓	Medium	✓	✓	✓	✓	-	19	3
CWIPC-SXR [10]	-	Medium	✓	✓	✓	✓	-	21	4
Ours	✓	Medium	✓	✓	✓	✓	✓	20	3



(a) CWIPC-SXR (b) HUMAN4D (c) FSVVD (d) Ours

Figure 2: Sample frames from the CWIPC-SXR, HUMAN4D, FSVVD, and our dataset.

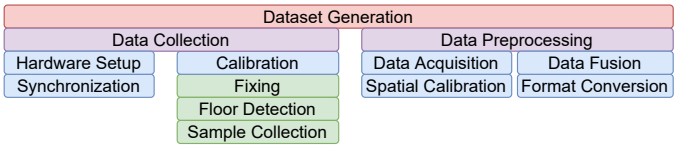


Figure 3: Illustration of the dataset generation.

3D models. The processed data were then converted into standardized .obj format files, which include information such as vertices, normals, and texture coordinates of the 3D mesh model, supporting subsequent analysis, rendering, and applications.

3.2 Shooting Process

We design and configure the hardware and software platform to capture high-quality 3D point cloud data, addressing key aspects such as software deployment, hardware setup, device synchronization, and sensor calibration. Multiple synchronized Azure Kinect cameras are employed, and a rigorous calibration process is implemented to ensure accurate spatial registration between the sensors. The following sections detail the implementation of the hardware setup, synchronization mechanism, and calibration process. The shooting process is shown in Figure 4.

3.2.1 Hardware Setup We use six Azure Kinect cameras to capture color and depth data, each connected to the host via a 10-meter Logitech C930 USB 3.0 extension cable. The host is a high-performance PC with an Intel Core i9-13900K CPU, an NVIDIA RTX 4080 16GB GPU, ASUS Z790 DDR5 motherboard, 128GB Corsair memory, and a 1TB Samsung M.2 NVMe SSD.



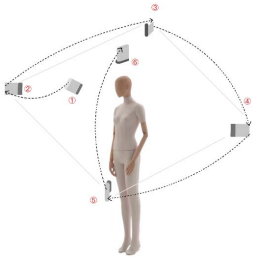
Figure 4: Shooting Process.

Due to the shadowing characteristics of the Azure Kinect depth cameras, they are positioned vertically to capture full-body poses. Each camera is mounted on a tripod for adjustable height and angle. To capture point cloud videos, we use four 200W DEEP high-definition supplementary lights placed strategically to minimize shadows and enhance detail. Light angles, intensity, and color temperature are adjusted based on our custom tool to ensure uniform lighting and reduce overexposure.

The capture space is approximately 4m × 4m, with six Azure Kinect cameras arranged as follows: three cameras in a semi-circle at the front, 1 meter apart, with the center camera near the subject's face. The other two cameras are placed at the back, 3 meters apart, symmetrically around the center of the capture area. An additional camera is mounted centrally on top, 2.5 meters above the ground. This configuration ensures comprehensive coverage, minimizing occlusions and enhancing point cloud accuracy. The front cameras, along with the rear and overhead cameras, provide multi-angle capture and redundancy, promoting even lighting and reducing shadows and brightness inconsistencies.



(a) Hardware Setup



(b) Capture Setup Diagram

Figure 5: Equipment and Setup. (a) Actual setup for capturing point cloud video with synchronized Azure Kinect cameras, computers, and lighting. (b) Schematic showing Kinect camera positioning and synchronization via a daisy-chain configuration from Kinect 1 to Kinect 6.

3.2.2 Synchronization We use six Azure Kinect cameras, requiring synchronization to ensure consistent data. This is achieved by connecting the cameras with a 3.5mm audio sync cable in a daisy-chain topology. In this setup, one sensor acts as the generator, providing a synchronization reference to the others through their sync ports in succession. The Kinect camera facing the subject's face is selected as the primary sensor, with its sync input unconnected. The sync output of the primary sensor is then connected to the sync input of the next sensor, and this process is repeated until the sixth sensor, which has no device connected to its sync output.

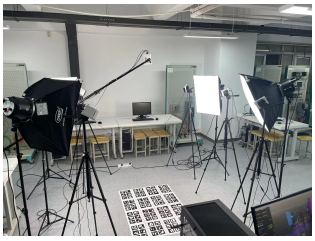
3.2.3 Calibration In multi-sensor systems, spatial calibration is essential to align all sensors within a unified coordinate system. Minor sensor displacements, sagging, or disturbances from cables can cause positional offsets, affecting data fusion and system accuracy. To address this, calibration markers are used to align each sensor's view, ensuring consistency during data registration.

Fixing: Any small sensor movements can distort point cloud data. Therefore, before calibration, the sensor support apparatus is fixed in place, cables are connected, and zip ties are used to secure them. This ensures stable sensor positioning, minimizing geometric misalignments and ensuring accurate spatial representation.

Floor Detection: Floor detection establishes the 3D origin and orientation of the scene. A marker board is placed on the floor, visible to at least one Azure Kinect camera. The system captures depth data to identify the ground plane, then calculates the ground's position and orientation relative to the sensors. This ensures the scene is aligned correctly in 3D space, preventing errors from sensor misalignment.

The ground plane is detected across all sensors, ensuring consistent data fusion. This step also prevents geometric distortion in the 3D scene, facilitating accurate data registration for subsequent processes like scene reconstruction. The specific floor detection process is shown in Figure 6a. By accurately establishing the 3D origin and orientation, this process enhances spatial accuracy and consistency, improving the quality of data fusion and ensuring precise 3D scene representations for applications such as behavior analysis, motion capture, and spatial modeling.

Sample Collection: To keep the marker board stationary, markers are placed on cubes of varying heights (using different-height cardboard boxes for convenience) and stable supports, as shown in Figure 6b. Markers are moved in different directions within the capture area, with each collection lasting about 10 seconds. Greater variation in height and angles ensures more effective capture by all Azure Kinect cameras. This step improves sensor alignment by providing markers at different depths and angles, ensuring accurate point registration across all sensor views. The result is a higher-quality point cloud with reduced errors from occlusions and misalignments.



(a) Floor inspection



(b) Calibrate with bracket

Figure 6: Calibration process: (a) Floor detection to establish the 3D origin and orientation; (b) Sample collection setup with markers on cubes of varying heights for Azure Kinect capture.

4 Dataset Description and Analysis

We systematically analyze our dataset, describing its characteristics, acquisition methods, and file structure in detail. Subsequently, we present the evaluation indicators for comparison with other datasets.

Table 2: Description of the Dataset

Themes	Screenplay	Actors	Frame Number
Physical exercise	Sitting and kicking	1 Male	707
	Lifting a dumbbell	1 Male	814
	Kicking	1 Female	789
	Cantering	1 Male	807
	High-fiving	1 Female	769
	Performing Tai Chi for two	2 Female	663
	Chest expansion exercises	1 Female	847
	Rotational motion	1 Female	810
	Stepping	1 Female	786
Daily activities	Drinking water	1 Female	757
	Throwing a ball	1 Female	671
	Sweeping the floor	1 Female	857
Social interaction	Three people sitting down	3 Male	844
	Three people reading books	3 Female	747
	Three people playing different instruments	3 Female	687
	Two people playing different instruments	2 Female	689
	Two people talking	2 Male	909
	Shaking hands and embracing	2 Female	738
	Taking photos	2 Female	641
	Finger guessing game	2 Female	764

4.1 Dataset Description

To create a point cloud video dataset reflecting real-life scenarios, we invite five volunteers (three females, two males) to ensure authenticity and data quality. We design three types of sequences: single-person, two-person, and three-person action scenes, including physical exercises, daily activities, and social interactions. This diversity aims to capture a broad range of real-world scenarios for behavior analysis and human action research.

Each sequence lasts 20 to 30 seconds, with a frame rate of 30 frames per second, totaling 600 to 900 frames. To enhance clarity, we ensure the volunteers' clothing contrasts with the background. Some scenes also include interactions with objects, such as playing instruments or reading together. In total, we capture 20 sequences: 14 single-person, 3 two-person, and 3 three-person. A detailed description of the actions is in Table 2.

The dataset's file structure is designed for efficient access and use. Each sequence is organized into a folder containing mesh data files (.obj), point cloud data files (.ply), RGB image files (.png), and corresponding timestamps. Sequence folders are named by their sequence number for easy retrieval.

Figure 8 illustrates the "throwing a ball" sequence. Each sequence folder contains "mesh" and "point cloud" subfolders. In the "mesh" folder, files are named as "mesh-fXXXXX," where "f" is a fixed prefix, and "XXXXX" is a sequential number starting from 00001. For example, "throwing a ball_mesh" contains "mesh-f00001.mtl," "mesh-f00001.obj," and "mesh-f00001.png" for material, 3D model, and image data. Similarly, the "point cloud" folder follows the same naming convention, with files like "mesh-f00001_processed.ply" (processed point cloud data) and "mesh-f00001.png" (image frame).

4.2 Dataset Comparison

We compare our dataset with common point cloud video datasets using four metrics: Spatial-Color Consistency [22], Shadow Ratio, Missing Ratio, and Temporal Distribution Uniformity. Spatial-Color Consistency measures point cloud alignment over time, reflecting registration stability and temporal consistency; Shadow Ratio assesses shadow distribution, indicating color uniformity; Missing Ratio quantifies missing regions, revealing dataset completeness; Temporal Distribution Uniformity evaluates frame balance across subjects, reflecting temporal consistency. These metrics highlight our dataset's clear advantages in quality, stability, and applicability. The specific calculation methods for these metrics are as follows:

Spatial-Color Consistency: This metric evaluates the alignment accuracy of point clouds over time by comparing both spatial and color differences between consecutive frames. The spatial difference is measured by calculating the Euclidean distance between each point in the target frame and its closest point in the source frame. The color difference is evaluated by comparing the color values of corresponding points. The combined spatial and color differences are normalized by the maximum distance and color variations in the dataset. If the calculated value falls below a predefined threshold, the two points are considered to be spatially and color-consistent, indicating good alignment.

Shadow Ratio: The shadow ratio quantifies the distribution of shadow points in the point cloud. To detect shadow points, we calculate the color difference between each point and the global mean color of the dataset, as well as the local brightness of the point. A point is considered a shadow if its color difference exceeds a certain threshold and its brightness is lower than the local mean brightness. The ratio of detected shadow points to the total number of points in the cloud gives the shadow ratio, which reflects the distribution of shadows within the dataset.

Missing Ratio: This metric measures the extent of missing regions in the point cloud, often due to occlusions or sensor limitations. To identify missing regions, we first detect boundary points, which are characterized by a higher distance to neighboring points, large angles in the normal vector, and higher curvature. These points indicate areas where the point cloud may be incomplete. Poisson reconstruction is then applied to estimate the missing regions, and the ratio of the missing area to the total area gives the missing ratio.

Uniformity: This metric evaluates the uniformity of frame distribution by comparing the average frame count per subject with the overall average. Smaller differences indicate a more balanced distribution.

Figure 9 compares the average frame counts per subject across datasets, using each dataset's average frame count as the benchmark. The results show that our dataset exhibits more uniform frame distribution, reducing biases from varying sequence lengths, which enhances training stability and efficiency [23]. Radar charts (Figure 10) compare dataset performance in terms of Spatial-Color Consistency Index, Missing Ratio, Shadow Ratio, Temporal Distribution Uniformity, and Maximum Number of People. Our dataset excels in uniformity, completeness, and shadow suppression.

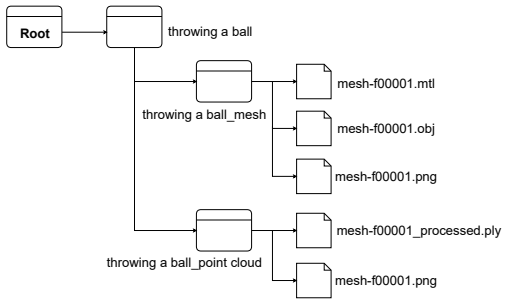


Figure 8: Dataset structure

5 Experiment

Our dataset supports various point cloud tasks. To validate its practical applicability, we select the task of denoising and employed several state-of-the-art methods, laying the foundation for its further use in various tasks.

5.1 Overview of the Denoising Method

In the experiment, we used IterativePFN [24] as the main denoising scheme. This graph convolution-based network iteratively filters noise through multiple IterationModules, using a dual-path structure with dynamic EdgeConv layers and generalized patch stitching. It leverages local geometric relationships to predict displacements, refining the noisy point cloud into a denoised result.

However, the output lacked color information. To address this, we proposed a method to recolor the point clouds by interpolating and mapping the original RGB data, enhancing both visual quality and the data's utility for further analysis.

We compare IterativePFN with three denoising methods in this experiment: IterativePFN, PDFlow [25], and ScoreDenoise [26], to assess their effectiveness in improving point cloud quality.

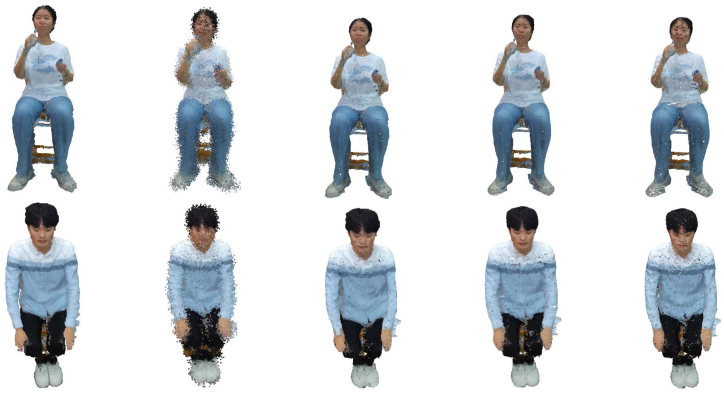


Figure 7: Denoising Performance Comparison on Our Dataset. (a) Original clean point cloud. (b) Noisy point cloud with 1.5% Gaussian noise. (c) Denoised by IterativePFN. (d) Denoised by PDFlow. (e) Denoised by ScoreDenoise.

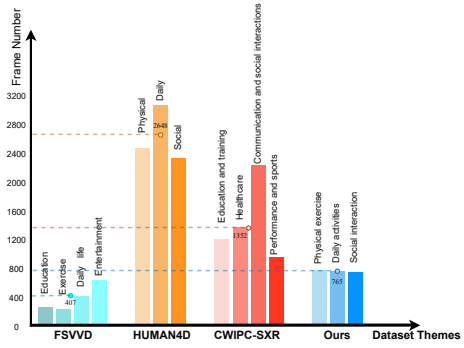


Figure 9: Frame count distribution across different datasets.

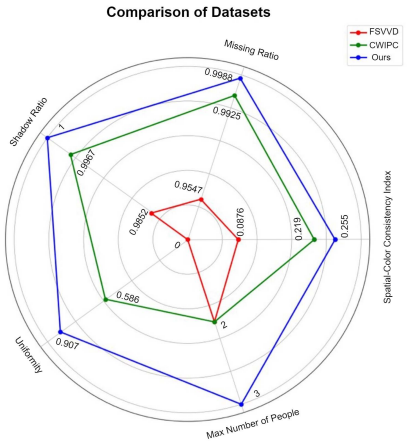


Figure 10: Comparison of Dataset Performances.

5.2 Setup

In this study, the experimental setup is based on PyTorch 1.11.0 as the deep learning framework, utilizing CUDA 11.3 for GPU acceleration, an NVIDIA GeForce GTX 1650 GPU for computation, and Ubuntu 20.04 as the operating system.

To use this dataset, we first sample our point cloud data at resolutions of 30K, 50K, and 70K points, converting these .ply formatted point cloud data into .xyz files. Then we introduce Gaussian noise with a standard deviation of 1%, 1.5%, and 2% of the boundary sphere radius into each point cloud as input. After the denoising, we enhance the visual representation by adding color information to the denoised point cloud data.

5.3 Results

Figure 7 shows the denoising performance comparison. The results indicate that IterativePFN better preserves details, removes noise, and avoids artifacts and distortions. In contrast, PDFlow and ScoreDenoise retain noise or affect surface details in high noise or complex shape scenarios, leading to less clean denoising. These findings align with the results shown in original paper [24], which highlights its iterative process for superior denoising, especially in complex and noisy cases.

Despite its strengths, IterativePFN has some minor drawbacks. While it effectively suppresses noise and fills gaps, improving data completeness, small residual noise points still cause minor gaps in certain areas. These limitations suggest that, although IterativePFN removes most noise, there is room for improvement in handling intricate details and restoring the full geometric structure.

6 Conclusion

In this paper, we introduce a novel point cloud video dataset designed to support a wide range of 3D point cloud tasks, including denoising, segmentation, motion recognition, and streaming. The dataset consists of 20 point cloud sequences captured from dynamic real-world scenes using synchronized Azure Kinect cameras, providing high-quality depth and color data. It includes both single-person and multi-person interactions. We compare it with existing datasets, and the results show its superiority in uniformity and completeness, making it ideal for dynamic environments. We also apply a state-of-the-art denoising network on this dataset as one of its practical applications, demonstrating its effectiveness.

References

[1] S. Laato, N. Xi, V. Spors, M. Thibault, and J. Hamari, "Making sense of reality: a mapping of terminology related to virtual reality, augmented reality, mixed reality, xr and the metaverse," in *Annual Hawaii International Conference on System Sciences*, 2024, pp. 6625–6634.

[2] Z. Wang, Y. Deng, and A. H. Aghvami, "Goal-oriented semantic communications for avatar-centric augmented reality," *IEEE Transactions on Communications*, 2024.

[3] Z. Liu, Q. Li, X. Chen, C. Wu, S. Ishihara, J. Li, and Y. Ji, "Point cloud video streaming: Challenges and solutions," *IEEE Network*, vol. 35, no. 5, pp. 202–209, 2021.

[4] J. Li, C. Zhang, Z. Liu, R. Hong, and H. Hu, "Optimal volumetric video streaming with hybrid saliency based tiling," *IEEE Transactions on Multimedia*, vol. 25, pp. 2939–2953, 2022.

[5] M. Yujian, Y. Wu, J. Zhao, H. Yinghao, J. Wang, J. Yan *et al.*, "Sparse query dense: Enhancing 3d object detection with pseudo points," in *ACM Multimedia 2024*.

[6] W. Gao and G. Li, "Point cloud compression, enhancement and applications: From 3d perception to large models," in *Proceedings of the 32nd ACM International Conference on Multimedia*, 2024, pp. 11 292–11 293.

[7] A. Akhtar, Z. Li, and G. Van der Auwera, "Inter-frame compression for dynamic point cloud geometry coding," *IEEE Transactions on Image Processing*, 2024.

[8] K. Hu, Y. Jin, H. Yang, J. Liu, and F. Wang, "Fsvvd: A dataset of full scene volumetric video," in *Proceedings of the 14th Conference on ACM Multimedia Systems*, 2023, pp. 410–415.

[9] A. Chatzitofis, L. Saroglou, P. Boutis, P. Drakoulis, N. Zioulis, S. Subramanyam, B. Kevelham, C. Charbonnier, P. Cesar, D. Zarpalas *et al.*, "Human4d: A human-centric multimodal dataset for motions and immersive media," *IEEE Access*, vol. 8, pp. 176 241–176 262, 2020.

[10] I. Reimat, E. Alexiou, J. Jansen, I. Viola, S. Subramanyam, and P. Cesar, "Cwipc-sxr: Point cloud dynamic human dataset for social xr," in *Proceedings of the 12th ACM Multimedia Systems Conference*, 2021, pp. 300–306.

[11] C. Loop, Q. Cai, S. O. Escolano, and P. A. Chou, "Microsoft voxelized upper bodies - a voxelized point cloud dataset," ISO/IEC JTC1/SC29 Joint WG11/WG1 (MPEG/JPEG), Input Document m38673/M72012, May 2016.

[12] Y. Xu, Y. Lu, and Z. Wen, "Owlii dynamic human mesh sequence dataset," in *ISO/IEC JTC1/SC29/WG11 m41658, 120th MPEG Meeting*, Macau, October 2017.

[13] E. d'Eon, B. Harrison, T. Myers, and P. A. Chou, "8i voxelized full bodies - a voxelized point cloud dataset," ISO/IEC JTC1/SC29 Joint WG11/WG1 (MPEG/JPEG), Geneva, Input Document WG11M40059/WG1M74006, January 2017.

[14] J. Li, H. Wang, Z. Liu, P. Zhou, X. Chen, Q. Li, and R. Hong, "Towards optimal real-time volumetric video streaming: A rolling optimization and deep reinforcement learning based approach," *IEEE Transactions on Circuits and Systems for Video Technology*, vol. 33, no. 12, pp. 7870–7883, 2023.

[15] J. Li, H. Wang, Q. Li, X. Liu, Z. Li, and Z. Liu, "Horizon: a real-time point cloud video streaming system over wireless networks," in *Proceedings of the 21st Annual International Conference on Mobile Systems, Applications and Services*, 2023, pp. 606–607.

[16] G. Li, W. Gao, and W. Gao, *Point Cloud Compression: Technologies and Standardization*. Springer Nature, 2024.

[17] W. Liu, W. Gao, and X. Mu, "Fast inter-frame motion prediction for compressed dynamic point cloud attribute enhancement," in *Proceedings of the AAAI Conference on Artificial Intelligence*, vol. 38, no. 4, 2024, pp. 3720–3728.

[18] J. Zhang and J. Zhou, "Point cloud data construction based on vslam image processing integrating cnn and transformer," in *AIP Conference Proceedings*, vol. 3194, no. 1. AIP Publishing, 2024.

[19] D. Ouyang, Y. Zhang, J. Guo, and G. Zhou, "Mlc-net: A sparse reconstruction network for tomosar imaging based on multi-label classification neural network," *ISPRS Journal of Photogrammetry and Remote Sensing*, vol. 220, pp. 85–99, 2025.

[20] S. Li, Y. Zhao, C. Li, B. Guo, J. Zhang, W. Zhu, Z. Ye, C. Wan, and Y. C. Lin, "Fusion-3d: Integrated acceleration for instant 3d reconstruction and real-time rendering," in *2024 57th IEEE/ACM International Symposium on Microarchitecture (MICRO)*. IEEE, 2024, pp. 78–91.

[21] F. Zhao, H. Huang, and W. Hu, "An optimized hierarchical point cloud registration algorithm," *Multimedia Systems*, vol. 31, no. 1, pp. 1–13, 2025.

[22] J. Li, Z. Li, Q. Li, W. Sun, W. Li, H. Wang, and Z. Liu, "Landscape: Saliency and trajectory based viewport prediction in point cloud video streaming," in *Proceedings of the 21st Annual International Conference on Mobile Systems, Applications and Services*, 2023, pp. 604–605.

[23] H. Liu, S. Patwardhan, P. Grasch, S. Agarwal *et al.*, "Model stability with continuous data updates," *arXiv preprint arXiv:2201.05692*, 2022.

[24] D. de Silva Edirimuni, X. Lu, Z. Shao, G. Li, A. Robles-Kelly, and Y. He, "Iterativepfn: True iterative point cloud filtering," in *Proceedings of the IEEE/CVF Conference on Computer Vision and Pattern Recognition*, 2023, pp. 13 530–13 539.

[25] A. Mao, Z. Du, Y.-H. Wen, J. Xuan, and Y.-J. Liu, "Pd-flow: A point cloud denoising framework with normalizing flows," in *The European Conference on Computer Vision (ECCV)*, 2022.

[26] S. Luo and W. Hu, "Score-based point cloud denoising," in *Proceedings of the IEEE/CVF International Conference on Computer Vision (ICCV)*, October 2021, pp. 4583–4592.

AMIS: An Audiovisual Dataset for Multimodal XR Research

Abhinav Bhattacharya[1], Luís Fernando de Souza Cardoso[2], Andy Schleising[2], Gareth Rendle[3], Adrian Kreskowski[3], Felix Immohr[1], Wolfgang Broll[2], Rakesh Rao Ramachandra Rao[1], Alexander Raake[1]
[1]Audiovisual Technology Group, TU Ilmenau, Germany
[2]Virtual Worlds and Digital Games Group, TU Ilmenau, Germany
[3]Virtual Reality and Visualization Research Group, Bauhaus-Universität Weimar, Germany
abhinav.bhattacharya@tu-ilmenau.de,luis.cardoso@tu-ilmenau.de,andy.schleising@tu-ilmenau.de,gareth.rendle@uni-weimar.de,adrian.kreskowski@uni-weimar.de,felix.immohr@tu-ilmenau.de,rakesh-rao.ramachandra-rao@tu-ilmenau.de,wolfgang.broll@tu-ilmenau.de,alexander.raake@tu-ilmenau.de



Figure 1: Overview of visual modalities per actor: top tile shows talking-head videos; bottom tiles (left-to-right) show personalized avatars, full-body videos, and volumetric avatars.

Abstract

The Audiovisual Multimodal Interaction Suite (AMIS) is an open-source dataset and accompanying Unity-based demo implementation designed to aid research on immersive media communication and social XR environments. AMIS features synchronized audiovisual recordings of three actors performing monologues and participating in dyadic conversations across four modalities: talking-head videos, full-body videos, volumetric avatars, and personalized animated avatars. These recordings can be used to simulate scenarios such as traditional video conferences or XR meetings with 3D avatars in controlled and replicable environments. The limitations of existing datasets, which include a restricted number of audiovisual formats, a narrow application focus, and suboptimal inclusion of verbal and non-verbal cues, are addressed by AMIS. With AMIS Studio, a Unity-based demonstrator, researchers can explore the recordings and compare the different audiovisual formats in VR scenes. This paper outlines the creation of AMIS, its design considerations, and how it may be applied in interdisciplinary domains, including cognitive psychology, audiovisual quality assessment, and social XR research.

CCS Concepts

• **Human-centered computing** → **Human computer interaction (HCI)**; **Interaction design**; **Collaborative and social computing**; • **Social and professional topics** → *User characteristics.*

Keywords

Audiovisual Dataset, Multimodal Interaction, Talking-Head Videos, Volumetric Avatars, Personalized Avatars, Immersive Media, Virtual Reality, Quality of Experience, Social XR

ACM Reference Format:
Abhinav Bhattacharya[1], Luís Fernando de Souza Cardoso[2], Andy Schleising[2], Gareth Rendle[3], Adrian Kreskowski[3], Felix Immohr[1], Wolfgang Broll[2], Rakesh Rao Ramachandra Rao[1], Alexander Raake[1]. 2025. AMIS: An Audiovisual Dataset for Multimodal XR Research. In *ACM Multimedia Systems Conference 2025 (MMSys '25), March 31-April 4, 2025, Stellenbosch, South Africa.* ACM, New York, NY, USA, 7 pages. https://doi.org/10.1145/3712676.3718344

This work is licensed under a Creative Commons Attribution-NonCommercial-ShareAlike 4.0 International License.
MMSys '25, Stellenbosch, South Africa
© 2025 Copyright held by the owner/author(s).
ACM ISBN 979-8-4007-1467-2/2025/03
https://doi.org/10.1145/3712676.3718344

1 Introduction

High-quality multimodal audiovisual datasets are essential for addressing challenges in immersive media communication and social Extended Reality (XR) environments. Here, different types of representation for human communication partners are conceivable, and many variants are currently implemented in several social XR and multimedia communication applications. These may range from traditional 2D videos with a head-and-shoulder view [7, 34, 43] and non-spatial audio to volumetric videos [10] or AI-animated, rigged Computer-Generated Imagery (CGI) realizations with location-specific spatial audio [17]. Evaluation of Quality of Experience (QoE) in 'live' mediated communication is hindered by evolving content and differing stimuli between subjects, which undermines the reliability of comparative analyses and the derivation of meaningful insights. Multi-modal communication datasets combat this problem by providing consistent and controllable stimuli while reflecting the variability of real communication.

Reviews of existing datasets and technologies derive requirements that would support research in specific contexts or with particular applications in immersive media. Jin et al. [23] observe a progression from small, single-subject volumetric datasets to larger, multi-person interaction-based datasets, pointing towards the need for more diverse and contextually rich volumetric media datasets. Ferrarotti et al. [16] emphasize the need for tests with high-quality multisensory stimuli that facilitate objective data collection to inform a holistic view of QoE in social XR environments. Furthermore, Fels et al. [15] stress the importance of stimuli that support auditory and cognitive research, like memory retention and conversational participation in dynamic audiovisual interactions. For a lifelike and engaging experience reflected in higher perceived social presence, Weidner et al. [48] highlight the superior performance of realistic full-body renderings and the importance of avatar realism in Virtual (VR) and Augmented Reality (AR) applications. A dataset that potentially considers these demands is desirable. Existing datasets often lack the synchronized, multimodal complexity needed to identify design factors that influence the replication of real-world interaction in immersive environments. Limitations such as narrow application focus reduce representational diversity, and inadequate integration of verbal and non-verbal communication cues limit their effectiveness in controlled and replicable studies.

We introduce the **A**udiovisual **M**ultimodal **I**nteraction **S**uite (**AMIS**)[1], an open-source dataset paired with a simple, adaptable Unity-based demonstrator. The novelty of the **AMIS dataset** lies in synchronized audiovisual recordings of monologues and dyadic conversations in English, captured across four modalities: talking-head videos, full-body videos, volumetric avatars, and personalized animated avatars, representing a range of visual and behavioral realism (depicted in Fig. 1). We included talking-head videos to emulate traditional video conferencing, focusing on facial expressions and speech. Full-body videos were added to examine the added role of body language, bridging the gap between conventional formats and 3D avatars. Personalized animated avatars and volumetric avatars offer realistic depictions of remote users in shared virtual spaces realized through social XR. Using studio cameras for 2D videos and advanced systems [3, 25] for 3D avatars, the recordings provide high-quality data for cross-format comparisons and analyses of verbal and non-verbal cues. A German version of the dataset with the same structure and characteristics is under preparation, enabling cross-language studies and research with German-speaking participants.

The dataset is complemented by **AMIS Studio**, a modular, Unity-based demonstrator for researchers to explore, interact with, and integrate the data for experiments. It features interactive scenes offering co-location with the dataset content, for example in VR. It can be adapted to research how different audiovisual representations of conversation partners influence interpersonal relationships, accommodating various playback methods such as AR/VR, screens, projections, and auralizations. AMIS represents a resource for interdisciplinary research with a combination of characteristics that, to the best of our knowledge, no other dataset provides. This paper focuses on the creation of the AMIS dataset, the design decisions, and its potential applications in various research domains.

2 Related works

In this section, contemporary human-centric datasets are categorized into three broad areas based on their application focus.

Affective Computing & Computer Vision. Multimodal datasets, such as IFADV [43] and IEMOCAP [7], highlight the critical role of analyzing communication through both verbal and non-verbal channels such as facial expressions and body gestures. The Real Talk dataset [19] features 692 unscripted dyadic conversations sourced from YouTube, combining large language models and vision language models to analyze associations between verbal and non-verbal cues. Similarly, the M^3ED dataset [51] provides 990 annotated emotional dialogues from 56 Chinese TV series, with 24,449 utterances for cross-cultural emotion recognition and analysis. MELD [35], using 1000 dialogues from the TV show *Friends* with over 130,000 labeled utterances, offers relatable content for emotion recognition frameworks. The CCv2 dataset [34] includes 26,467 monologues with 11 attributes of 5,567 participants for benchmarking fairness and robustness in audio, vision, and speech tasks, with applications like deepfake detection. The OMG-Emotion Dataset [4] focuses on 567 YouTube monologues with 7,371 labeled utterances for emotion modeling, while the OMG-Empathy dataset [5] explores empathy through 80 speaker-listener interactions using original audiovisual stimuli. HEADSET [27] captures volumetric, light field, and RGB images of 6 facial expressions of 27 participants, also addressing challenges like HMD occlusions. The 4D CCDb database [44] contributes to this category by providing 17 2D and active stereo vision recordings of unscripted conversations with detailed non-verbal cue annotations.

Audiovisual Scene Analysis. The AVT-ECoClass-VR dataset [17] was created to evaluate cognitive performance in classroom-style, multi-talker audiovisual settings. It comprises 360° videos and single-channel audio recordings of 20 speakers reading 10 stories each, along with 20 3D scans of speakers compatible with a CGI Unity scene. The Tragic Talker dataset [6] focuses on spatial audio research, aligning audio and video sources in 30 monologues and dyadic conversations inspired by Shakespearean scripts. Its interactive scenes feature 22 views of improvisational speakers who

[1] https://github.com/Telecommunication-Telemedia-Assessment/AMIS

move and occlude one another, posing challenges for 3D audiovisual tracking systems. The QoEVAVE dataset [36] and the subset of HOA-SSR dataset used by [14] present 12 360° audiovisual scenes each with 8K video and 4th-order Ambisonics audio. The scenes provide high-fidelity, real-life settings for evaluating audiovisual attention and underlying saliency [39], and to carry out perceptual quality tests [33] across audio, video, and audiovisual modalities.

Human Motion and Activity. The Nymeria dataset [30] captures 20 natural activities and unscripted interactions from 264 participants using synchronized multimodal egocentric devices. It supports motion synthesis, action recognition, and body tracking, with a focus on linking human motion to contextual actions in everyday scenarios. Similarly, ActionSense [13] offers rich, synchronized wearable sensing data streams in 20 kitchen activities, covering tasks from simple object manipulation to complex sequences like setting tables or cutting vegetables. Both datasets facilitate insights into fine-grained actions and cross-modal learning pipelines.

HUMAN4D [10] provides marker-based motion capture (MoCap), volumetric video, and audio for studying 19 activities like jumping and dancing with 4 actors. It emphasizes spatio-temporal alignment and multimodal fusion for tasks such as activity recognition and scene analysis. MHAD [11] and RELI11D [49] focus on body actions like sports, utilizing RGB, depth, and inertial data to explore fusion approaches between depth cameras and wearable sensors. In contrast, the Personality Dyads Corpus [41] shifts focus to motion and gesture data in dyadic conversations, targeting nonverbal behaviors of different personality traits. Replay [38] captures 68 long-form social interactions in natural indoor settings, such as playing games or exercising. Employing 12 static and egocentric cameras with 8 binaural microphones, it supports reconstruction quality analysis and novel-view synthesis for visual and acoustic modalities.

The discussed datasets represent a broader collection that often focus on narrow application areas [4, 5, 19, 34–36, 38, 41, 43, 44, 51], feature modalities without audio [11, 27, 30, 41, 44, 49], have restricted access [6, 7, 10, 11, 30, 34, 41, 44, 50], or do not reflect the complexity of real-world communication dynamics [6, 17, 34, 51]. AMIS addresses the gaps in existing datasets by offering synchronized multimodal data across diverse formats, including 2D and volumetric videos, animated personalized avatars, and audio recordings. Unlike many datasets that focus on static or predefined interactions, AMIS emphasizes dynamic conversational interactions, role variability, and naturalistic speech to simulate realistic interactions in reproducible conditions. Its comprehensive multimodal representations support research beyond the discussed categories, enabling the exploration of emerging domains and novel research questions (see Section 7).

3 Design Decisions

Setup and Equipment. The recordings were performed in the media lab[2] at the Institute of Media Technology and the Ilmenau Interactive Immersive Technologies Center (I3TC), TU Ilmenau, a high-production facility for academic and professional studio work. The stage and background were evenly lit using a studio lighting setup to eliminate shadows. A unique multi-camera setup, as seen in Fig. 2, enabled simultaneous recording in different formats with consistent and comparable representations across modalities.

An *Autoscript* teleprompter was mounted on the primary camera (Canon C500), and audio was recorded with a *Shure* omnidirectional lavalier microphone. The RED camera was rotated 90° on its axis to capture full-body videos. Facial expressions for personalized avatars were recorded using an iPhone mounted at face level, and four calibrated RGB-D cameras captured image streams for volumetric reconstruction. Table 1 shows the hardware components with the specifications used in the recordings. A clapperboard ensured audiovisual synchronization before every recording.



Figure 2: Studio setup and labeled camera assembly

Actors. The actor pool, consisting of two female actors (EF1, EF2) and one male actor (EM), comprised students at TU Ilmenau from different ethnic backgrounds with a mean age of 27 years. Both female actors were Native English Speakers (NS), while the male actor was a Non-Native English Speaker (NNS) who uses English as a *lingua franca*. They had 3-4 years of acting experience, ensuring familiarity with performing on camera. The actors were instructed to avoid dressing completely in white or black to prevent issues like overexposure or loss of detail. The study received approval from the ethics commission of TU Ilmenau, and all participants provided informed consent to use their likenesses for scientific research.

4 Data Collection

The data collection process consisted of two communication scenarios: monologues, like [4, 6, 10, 17, 27, 34, 40], and dyadic conversations, as in [5–7, 19, 30, 35, 41, 43, 44, 51]. These scenarios enable research to obtain complementary insights into human communication. Monologues offer controlled, replicable performances across modalities and actors, while the recorded dyadic conversations capture the added complexity of interpersonal interactions.

4.1 Monologues

A subset of nine 5-sentence passages from the 40 available in the English transcripts of the EUROM1 Multilingual Speech Corpus [9] was selected and categorized into three thematic groups to ensure variety: formal requests and complaints, personal anecdotes, and narrative accounts. This resource, designed for phonetic comparisons across languages, provides phonetically balanced materials ideal for studies requiring consistent linguistic features. The text includes various phonemes, consonant clusters, and vowel combinations, supporting phonetic coding [45] and prosodic analysis [8].

The selected passages had balanced formal and informal tones, ranging from serious emotions (e.g., frustration in complaints) to

[2]https://www.tu-ilmenau.de/en/mt/labs-and-technical-equipment/media-lab-for-video-production

Table 1: Hardware components used in the recordings

Captures	Hardware	Capture Settings
Talking-head videos	Canon EOS C500 Mark II Zeiss Compact Prime CP.3 85/T2.1 Lens	ISO:1000, Color: 3300K, 59.94 fps Focal length: 85mm, Aperture: T4
Full-body videos	RED SCARLET-W 5K Camera Zeiss Compact Prime CP.3 21/T2.9 Lens	ISO:800, Color: 3322K, 59.94 fps Focal length: 21mm, Aperture: T2.9
RGB-D image streams	4x Femto Bolt Cameras	Color (RGB, 1280 × 720 px) and depth (640 × 576 px) at 30 fps
Face expressions	iPhone 12 pro	Animation using Reallusion iClone8 Software
Audio	Shure MX-150B Microphone with Shure QLX-D wireless system	48 kHz sampling rate

lighter, humorous elements (e.g., personal anecdotes). They included both direct speech (e.g., requests, complaints) and indirect speech (e.g., narrative accounts), offering a spectrum of communication styles and emotional expressions. Each actor rehearsed and performed all nine monologues in separate sessions using a teleprompter.

4.2 Dyadic Conversations

The short dyadic conversations reflect the nature of small talk, which, as noted by Wahab et al. [46], can serve as an icebreaker, silence-filler, or gateway to deeper discussions in social and professional settings. Wei et al. [47] found that sharing positive stories strengthens interpersonal bonds. This study centered conversations on light, relatable topics, such as weekends, movies, and music artists that mirror everyday communication. Each actor pair recorded two conversations, totaling six, using a single turn-taking format. To introduce variability, speaker-listener orders were alternated between recordings of each pair. Of the six conversations, four featured NS-NNS pairs, while two involved NS-NS pairs. Conversations also varied by the gender composition of the pairs.

As only one actor could be recorded at a time, conversation topics and assisting prompts were provided. Adapting approaches in [5, 21], the actors discussed off-camera to synchronize reactions and create meaningful exchanges, with room for improvisation within a structured framework. A countdown timer was used to synchronize their actions and responses.

5 Post-Production

All visual formats discussed in this section are paired with synchronized audio files. The audio recordings were processed in DaVinci Resolve 18.6 [3] to reduce hiss and improve speech clarity. The audio files were then exported in linear PCM format at a 48 kHz sample rate with identical signals in two channels to increase compatibility with standard media players and avoid dichotic representation.

5.1 Talking-Head and Full-Body Videos

In DaVinci Resolve 18.6, raw footage captured with the *Canon* camera was trimmed and color-corrected to produce 4K (3840×2160) Standard Dynamic Range (SDR) talking-head videos. Footage from the *RED* camera was rotated to portrait orientation and processed to generate 4K (2160 × 3840) SDR full-body videos, with skin and clothing colors manually matched to the talking-head videos in DaVinci Resolve. Both video formats were encoded using the H.265 codec with the native encoder and Main encoding profile. The final outputs were exported at the original frame rate of 59.94 fps, with a bitrate of 40 Mbps, and saved in MP4 format. Larger HDR versions are also available upon request.

5.2 Personalized Animated Avatars

The avatars of the actors were created using Character Creator 4 [4] and the Headshot 2 plugin [5], tools commonly used in similar studies for the generation of realistic avatars and accurate facial modeling [3, 31]. The facial features of each actor were based on headshot photographs processed in the plugin, while body morphologies were adjusted by analyzing full-body videos. The workflow included initial creation, iterative improvement, and manual corrections to ensure similarity.

Using the full-body video footage, clothing similar to what the actors wore was also created. Character Creator's base 3D clothing models, as well as freely available models, were utilized and exported to Blender [6], and the meshes were adjusted manually and with the Blender clothing simulation. Some materials were also created with GIMP [7] to accurately represent the actors' outfits. After adjustments in Blender, the clothing was re-imported into Character Creator to apply pose weighting to the avatars and eliminate collisions between the clothing meshes and the avatars.

Animations for facial expressions were captured through iPhone and iClone 8 [8] software, which generated the motion data for the actors' faces as FBX files. Body motions were derived from full-body footage using Rokoko's [9] pose estimation, with further refinements applied in Blender to smoothen movements and reduce jitter for higher animation quality. Animations and avatars were imported into Unity using the Character Creator/iClone Unity Tools 1.6.3 package for avatar integration. Animations were assigned to layers within Unity's Animator and synchronized for minor adjustments such as torso and arm rotation to prevent issues like self-collision. Final avatar animations were exported as a single animation sequence using the FBX Exporter package.

[3] https://www.blackmagicdesign.com/products/davinciresolve/

[4] https://www.reallusion.com/character-creator/
[5] https://www.reallusion.com/character-creator/headshot/
[6] https://www.blender.org/
[7] https://www.gimp.org/
[8] https://www.reallusion.com/iclone/
[9] https://www.rokoko.com/products/vision

5.3 Volumetric Avatars

To generate volumetric representations of the actors, four synchronized and spatially registered *Femto Bolt* RGB-D cameras captured color and depth streams during each performance. The RGB-D streams were processed frame-by-frame through a reconstruction pipeline, applying morphological and bilateral filtering to depth images before integrating them into a Truncated Signed Distance Field (TSDF) [12]. Temporal filtering is applied to the TSDF volume to combat sensor noise. An indexed triangle mesh was then extracted from the TSDF volume using the Marching Cubes isosurface extraction algorithm [28]. Vertex quantization is applied and the mesh is compressed using the *meshoptimizer* library[10]. Color textures for the extracted mesh were generated by blending contributions from multiple cameras in a similar manner to Kreskowski et al. [25]. Color matching between cameras was performed by shifting the mean and standard deviations of each channel in CIELAB color space to match the characteristics of the image from the front-on camera. Encoding of redundant texture data was mitigated by analyzing the occupancy of blended textures at an 8×8 pixel granularity and packing occupied tiles into a dense buffer, which is encoded as a JPEG image. The compressed mesh and texture data are saved as frame-wise volumetric streams of the reconstructed avatars, with each frame containing approximately 50k triangles.

The volumetric streams can be viewed in Unity via the Volumetric Video Player package included with AMIS. The package contains a Unity plugin that loads each frame of the volumetric avatar stream and performs decompression and efficient upload and unpacking of the color and geometry information using compute shaders. Note that the player is written in a subset of *Unity's Managed Code*, which allows efficient decoding and rendering of avatar streams even on integrated graphics processors, such as those found in modern head-mounted displays such as the *Meta Quest 3*. We note that although these streams were processed post-recording, the GPU-based pipeline supports real-time reconstruction for remote communication scenarios.

6 Suite Overview

The AMIS suite comprises the dataset and accompanying tools, all hosted in a repository with long-term availability. The dataset is provided under a Creative Commons (CC BY-NC-SA) license, and the corresponding Unity project is released as open source.

AMIS Dataset. The AMIS dataset features 156 synchronized audiovisual recordings captured across four distinct formats: talking-head videos, full-body videos (both in 4K, 59.94 fps as MP4 files), personalized 3D avatars with animations (FBX files with associated textures and materials), and volumetric avatars (reconstructed byte streams). Each recording is accompanied by audio in WAV format, compatible with all visual formats. The dataset comprises 27 monologues by three actors, each delivering nine monologues, totaling approximately 40 minutes across four visual formats. It also includes six dyadic conversations, each with two synchronized recordings to simulate natural dialogue, totaling 52 minutes across all formats. The monologues and conversations add up to around 2 hours of content across different audiovisual formats. A README guide outlines the dataset's structure, contents, and usage. Metadata files include monologues, topic- and speaker-labeled transcripts of conversations[11], and participant information.



Figure 3: Screenshot of a conversation in the personalized animated avatar scene and the hybrid scene showing all visual formats synced with each other and the audio.

AMIS Studio. We present AMIS Studio as a complementary addition to the dataset, a Unity-based XR demonstration platform that enables researchers to explore and interact with the dataset in 6 Degrees of Freedom (6-DoF). The Unity project is paired with implementation instructions for the VR scenes that showcase the actors' representations synchronized with their corresponding audio tracks, simulating multiparty meetings, and allowing observation and passive interaction with avatars in VR (Fig. 3). These minimalistic scenes can be used to build experiments by integrating world-building elements, questionnaire overlay tools such as ROVER [26], or plugins like VAUnity [22] and pyBinSim [32] for spatial audio and dynamic binaural playback. Key features of AMIS Studio include VR menus for scene switching, dedicated player packages for each media format, and scalability for larger datasets.

Additional files. To increase the applicability of talking-head videos, AMIS includes 12 segmented "reaction videos" from conversation recordings, showing the actors listening to their conversation partners and reacting non-verbally. An included Python script creates multiparty videoconferencing-style videos by combining talking-head clips using FFmpeg software[12], replicating the tiled layout of videoconferencing platforms. Additionally, a processing chain is provided to add virtual backgrounds for easy integration into experiments.

7 Potential Applications

While there are likely more applications and research questions, this section highlights an exemplary selection of potential uses of the AMIS dataset, aiming to inspire novel research directions.

Social Presence and Interpersonal communication. Social presence, defined as the perceived ability of a medium to convey contextual cues that create personal connections, like in a face-to-face setting, involves more than just technical capabilities. It is co-created through participant actions, behaviors, and mutual engagement [37]. Research shows that interpersonal coordination with avatars varies across audiovisual modalities [42], with gesture-based interaction in 3D environments enhancing co-presence, while verbal interaction performs better in 2D due to familiarity with video conferencing systems [24]. The AMIS dataset offers a foundation to investigate how different formats (2D videos to 3D avatars) affect social and co-presence. It can potentially inform the design of

[10] https://github.com/zeux/meshoptimizer

[11] https://restream.io/tools/transcribe-audio-to-text

[12] https://www.ffmpeg.org/

more naturalistic virtual agents for VR conferences by identifying which design factors most effectively support natural communication under varying technical constraints.

Affective Computing and Emotion Recognition. The AMIS dataset can be used to research key indicators of affective states, emotion recognition, and emotional and cognitive empathy within immersive VR environments [2, 5]. By leveraging its diverse audiovisual formats, researchers can investigate how these representations influence emotional contagion and empathic responses during interpersonal communication. The variability in tone and context offers an opportunity to validate emotion prediction models [1].

Cognitive Psychology and Memory Studies. The impact of different media on cognitive load, attention and engagement remains a critical area of research. Studies like [39], investigating audiovisual engagement and attention with spatial audio, may leverage AMIS's characteristics. The listener must retain the talker's speech either to actively participate or to recall the content for future use. The AMIS dataset provides a suitable resource for experiments proposed by [15], which investigates how variations in audiovisual characteristics in closer-to-reality listening environments affect memory retention and comprehension of continuous speech.

Multimedia System Testing and Validation. The Storytime dataset [40], employed to assess quality perception using audiovisual stories with realistic videoconferencing degradations, revealed that content differences did not affect perceived QoE. With AMIS, researchers may assess the impact of system constraints, such as streaming bandwidth, through subjective and objective data to explore trade-offs in usability and performance of the 3D avatars [23]. Analyzing how technical limitations affect overall perceived quality could provide insights for optimizing future immersive conferencing systems. Additionally, the 2D videos in the dataset hold independent value as a resource for testing and validating systems such as text-based editing of talking head videos [18, 20] or frameworks for generating realistic face-to-face conversations between virtual avatars [50].

Source Localization and Speaker Identification. The AMIS dataset can provide stimuli for auditory scene analysis and speaker recognition tests in multi-talker or noisy environments, complementing works like [29], which focus on monologues with general topics. AMIS includes monologues featuring direct and indirect speech with contextually relevant information that supports tasks such as source localization and story-to-speaker mapping [17].

8 Conclusion and Outlook

This paper introduced AMIS, a comprehensive audiovisual dataset, to support research in areas such as cognitive psychology, affective computing, audiovisual quality, and QoE modeling within social XR environments. The AMIS dataset features synchronized audiovisual recordings captured using a multi-camera setup. It includes four visual formats paired with synchronized audio tracks: talking-head videos, full-body videos, volumetric avatars, and personalized animated avatars. To increase accessibility, AMIS Studio, a Unity-based VR exploration environment, is provided with the dataset.

Expanding the actor pool in future iterations would allow for greater variability in age, ethnicity, and cultural background, enabling the exploration of presence, proxemics, and agency in virtual environments. One volumetric avatar experienced a partial loss of stream information from the cameras, as documented in the repository, and technical limitations in capture and processing may also influence the visual fidelity and realism of the 3D avatars. Subjective tests are planned to establish baseline metrics for the perceived quality of these modalities. Furthermore, AMIS Studio can be integrated with AR compatibility and live interaction capabilities with existing transmission network frameworks. The German version of the dataset, along with the baseline tests, is planned for future updates, which will also feature further improvements in the overall quality of the data.

Acknowledgments

This work is part of the MULTIPARTIES project, funded by the Bundesministerium für Bildung und Forschung (BMBF) through KMU-innovativ. The authors thank Consensive GmbH, William Menz, Tatiana Surdu, Min Ni, Stephanie Arevalo, Dominik Keller, Steve Goering, Georg Stolz, and Jakob Hartbirch from the Institute of Media Technology (IMT), TU Ilmenau, and the actors for their contributions.

References

[1] Enas Altarawneh, Ameeta Agrawal, Michael Jenkin, and Manos Papagelis. 2023. Predicting Evoked Emotions in Conversations. arXiv:2401.00383 [cs.CL]

[2] Jorge Bacca-Acosta, Cecilia Avila-Garzon, and Myriam Sierra-Puentes. 2023. Insights into the Predictors of Empathy in Virtual Reality Environments. *Information* 14, 8 (2023).

[3] Kwame Agyemang Baffour and Oyewole Oyekoya. 2024. Generating Look-Alike Avatars: Perception of Head Shape, Texture Fidelity and Head Orientation of Another User's Look-Alike Avatar. In *2024 IEEE Conference on Virtual Reality and 3D User Interfaces Abstracts and Workshops (VRW)*. 983–984.

[4] Pablo Barros, Nikhil Churamani, Egor Lakomkin, Henrique Siqueira, Alexander Sutherland, and Stefan Wermter. 2018. The OMG-Emotion Behavior Dataset. In *2018 International Joint Conference on Neural Networks (IJCNN)*. 1–7.

[5] Pablo Barros, Nikhil Churamani, Angelica Lim, and Stefan Wermter. 2019. The OMG-Empathy Dataset: Evaluating the Impact of Affective Behavior in Storytelling. In *2019 8th International Conference on Affective Computing and Intelligent Interaction (ACII)*. 1–7.

[6] Davide Berghi, Marco Volino, and Philip J. B. Jackson. 2022. Tragic Talkers: A Shakespearean Sound- and Light-Field Dataset for Audio-Visual Machine Learning Research. In *Proc. of the 19th ACM SIGGRAPH European Conf. on Visual Media Production* (London, United Kingdom) *(CVMP '22)*. Association for Computing Machinery, New York, NY, USA, Article 5, 8 pages.

[7] Carlos Busso, Murtaza Bulut, Chi-Chun Lee, Abe Kazemzadeh, Emily Mower Provost, Samuel Kim, Jeannette Chang, Sungbok Lee, and Shrikanth Narayanan. 2008. IEMOCAP: Interactive emotional dyadic motion capture database. *Language Resources and Evaluation* 42 (12 2008), 335–359.

[8] Estelle Campione and Jean Véronis. 1998. A multilingual prosodic database. In *Proceedings of the 5th International Conference on Spoken Language Processing (ICSLP 1998)*. 0844. https://doi.org/10.21437/ICSLP.1998-609

[9] Dominic Chan, Adrian Fourcin, Dafydd Gibbon, Björn Granström, Mark Huckvale, Kokkinakis George, Knut Kvale, Lori Lamel, Børge Lindberg, Asunción Moreno, Jiannis Mouropoulos, Francesco Senia, Isabel Trancoso, Corin Veld, and Jerome Zeiliger. 1995. EUROM - a spoken language resource for the EU - the SAM projects. In *4th European Conference on Speech Communication and Technology*.

[10] Anargyros Chatzitofis, Leonidas Saroglou, Prodromos Boutis, Petros Drakoulis, Nikolaos Zioulis, Shishir Subramanyam, Bart Kevelham, Caecilia Charbonnier, Pablo Cesar, Dimitrios Zarpalas, Stefanos Kollias, and Petros Daras. 2020. HUMAN4D: A Human-Centric Multimodal Dataset for Motions and Immersive Media. *IEEE Access* 8 (2020), 176241–176262.

[11] Chen Chen, Roozbeh Jafari, and Nasser Kehtarnavaz. 2015. UTD-MHAD: A multimodal dataset for human action recognition utilizing a depth camera and a wearable inertial sensor. In *2015 IEEE International Conference on Image Processing (ICIP)*. 168–172.

[12] Brian Curless and Marc Levoy. 1996. A volumetric method for building complex models from range images. In *Proceedings of the 23rd Annual Conference on Computer Graphics and Interactive Techniques (SIGGRAPH '96)*. Association for Computing Machinery, New York, NY, USA, 303–312.

[13] Joseph DelPreto, Chao Liu, Yiyue Luo, Michael Foshey, Yunzhu Li, Antonio Torralba, Wojciech Matusik, and Daniela Rus. 2022. ActionSense: A Multimodal Dataset and Recording Framework for Human Activities Using Wearable Sensors in a Kitchen Environment. In *Advances in Neural Information Processing Systems*, S. Koyejo, S. Mohamed, A. Agarwal, D. Belgrave, K. Cho, and A. Oh (Eds.), Vol. 35. Curran Associates, Inc., 13800–13813.

[14] Randy F Fela, Andréas Pastor, Patrick Le Callet, Nick Zacharov, Toinon Vigier, and Soren Forchhammer. 2022. Perceptual evaluation on audio-visual dataset of 360 content. In *2022 IEEE International Conference on Multimedia and Expo Workshops (ICMEW)*. IEEE, 1–6.

[15] Janina Fels, Cosima A. Ermert, Jonathan Ehret, Chinthusa Mohanathasan, Andrea Bönsch, Torsten W. Kuhlen, and Sabine J. Schlittmeier. 2021. Listening to, and Remembering Conversations between Two Talkers: Cognitive Research using Embodied Conversational Agents in Audiovisual Virtual Environments. In *[Fortschritte der Akustik - DAGA 2021, Wien, Austria]* (2021-08-15). 47. Jahrestagung für Akustik, Wien (Austria), 15 Aug 2021 - 18 Aug 2021, Deutsche Gesellschaft für Akustik e.V. (DEGA), Berlin, 1328–1331.

[16] Anna Ferrarotti and Marco Carli. 2023. Objective Metrics Definition for QoE Assessment for Extended Reality Applications. In *Proc. of the 2023 ACM Int. Conf. on Interactive Media Experiences* (Nantes, France) *(IMX '23)*. Association for Computing Machinery, New York, NY, USA, 439–442.

[17] Stephan Fremerey, Carolin Breuer, Larissa Leist, Maria Klatte, Janina Fels, and Alexander Raake. 2024. AVT-ECoClass-VR: An open-source audiovisual 360° video and immersive CGI multi-talker dataset to evaluate cognitive performance. In *2024 16th International Conference on Quality of Multimedia Experience (QoMEX)*. 207–213.

[18] Ohad Fried, Ayush Tewari, Michael Zollhöfer, Adam Finkelstein, Eli Shechtman, Dan B Goldman, Kyle Genova, Zeyu Jin, Christian Theobalt, and Maneesh Agrawala. 2019. Text-based Editing of Talking-head Video. arXiv:1906.01524 [cs.CV]

[19] Scott Geng, Revant Teotia, Purva Tendulkar, Sachit Menon, and Carl Vondrick. 2023. Affective Faces for Goal-Driven Dyadic Communication. arXiv:2301.10939 [cs.CV]

[20] Bo Han, Heqing Zou, Haoyang Li, Guangcong Wang, and Chng Eng Siong. 2024. Text-based Talking Video Editing with Cascaded Conditional Diffusion. arXiv:2407.14841 [cs.CV]

[21] Louise Heron, Jaebok Kim, Minha Lee, Kevin El Haddad, Stephane Dupont, Thierry Dutoit, and Khiet Truong. 2018. A Dyadic Conversation Dataset on Moral Emotions. In *2018 13th IEEE International Conference on Automatic Face & Gesture Recognition (FG 2018)*. 687–691.

[22] Institute for Hearing Technology and Acoustics, RWTH Aachen University. 2020. Virtual Acoustics – A real-time auralization framework for scientific research. http://www.virtualacoustics.org/. Accessed on 2020-04-21.

[23] Yili Jin, Kaiyuan Hu, Junhua Liu, Fangxin Wang, and Xue Liu. 2024. From Capture to Display: A Survey on Volumetric Video. arXiv:2309.05658 [cs.MM]

[24] Daehwan Kim and Dongsik Jo. 2022. Effects on Co-Presence of a Virtual Human: A Comparison of Display and Interaction Types. *Electronics* 11, 3 (2022).

[25] Adrian Kreskowski, Stephan Beck, and Bernd Froehlich. 2022. Output-Sensitive Avatar Representations for Immersive Telepresence. *IEEE Transactions on Visualization and Computer Graphics* 28, 7 (2022), 2697–2709.

[26] Lucas Küntzer, Sandra U. Schwab, Heike Spaderna, and Georg Rock. 2024. ROVER: A Standalone Overlay Tool for Questionnaires in Virtual Reality. In *Companion Proc. of the 16th ACM SIGCHI Symp. on Engineering Interactive Computing Systems* (Cagliari, Italy) *(EICS '24 Companion)*. Association for Computing Machinery, New York, NY, USA, 31–39.

[27] Fatemeh Ghorbani Lohesara, Davi Rabbouni Freitas, Christine Guillemot, Karen Eguiazarian, and Sebastian Knorr. 2023. HEADSET: Human Emotion Awareness under Partial Occlusions Multimodal DataSET. *IEEE Transactions on Visualization and Computer Graphics* 29, 11 (2023), 4686–4696.

[28] William E. Lorensen and Harvey E. Cline. 1987. Marching cubes: A high resolution 3D surface construction algorithm. In *Proceedings of the 14th Annual Conference on Computer Graphics and Interactive Techniques (SIGGRAPH '87)*. Association for Computing Machinery, New York, NY, USA, 163–169.

[29] Kasper Duemose Lund, Axel Ahrens, and Torsten Dau. 2019. A method for evaluating audio-visual scene analysis in multi-talker environments. NA.

[30] Lingni Ma, Yuting Ye, Fangzhou Hong, Vladimir Guzov, Yifeng Jiang, Rowan Postyeni, Luis Pesqueira, Alexander Gamino, Vijay Baiyya, Hyo Jin Kim, Kevin Bailey, David Soriano Fosas, C. Karen Liu, Ziwei Liu, Jakob Engel, Renzo De Nardi, and Richard Newcombe. 2024. Nymeria: A Massive Collection of Multimodal Egocentric Daily Motion in the Wild. arXiv:2406.09905 [cs.CV]

[31] Maryam Nadeem, Raza Imam, Rouqaiah Al-Refai, Meriem Chkir, Mohamad Hoda, and Abdulmotaleb El Saddik. 2024. EVOKE: Emotion Enabled Virtual Avatar Mapping Using Optimized Knowledge Distillation. In *2024 IEEE International Conference on Consumer Electronics (ICCE)*. 1–6.

[32] Annika Neidhardt, Florian Klein, Niklas Knoop, and Thomas Köllmer. 2017. Flexible python tool for dynamic binaural synthesis applications. https://github.com/tuil-emt/pybinsim. In *Audio Engineering Society Convention 142*. Audio Engineering Society.

[33] Andreas Pastor, Pierre Lebreton, Toinon Vigier, and Patrick Le Callet. 2024. Comparison of conditions for omnidirectional video with spatial audio in terms of subjective quality and impacts on objective metrics resolving power. In *ICASSP 2024-2024 IEEE International Conference on Acoustics, Speech and Signal Processing (ICASSP)*. IEEE, 8210–8214.

[34] Bilal Porgali, Vítor Albiero, Jordan Ryda, Cristian Canton Ferrer, and Caner Hazirbas. 2023. The Casual Conversations v2 Dataset. arXiv:2303.04838 [cs.CV]

[35] Soujanya Poria, Devamanyu Hazarika, Navonil Majumder, Gautam Naik, Erik Cambria, and Rada Mihalcea. 2019. MELD: A Multimodal Multi-Party Dataset for Emotion Recognition in Conversations. , 527–536 pages.

[36] Thomas Robotham, Ashutosh Singla, Olli S. Rummukainen, Alexander Raake, and Emanuël A. P. Habets. 2022. Audiovisual Database with 360° Video and Higher-Order Ambisonics Audio for Perception, Cognition, Behavior, and QoE Evaluation Research. In *2022 14th International Conference on Quality of Multimedia Experience (QoMEX)*. 1–6.

[37] Ulrike Schultze and Jo Ann Brooks. 2018. An interactional view of social presence: Making the virtual other "real". *Information Systems Journal* 29 (12 2018).

[38] Roman Shapovalov, Yanir Kleiman, Ignacio Rocco, David Novotny, Andrea Vedaldi, Changan Chen, Filippos Kokkinos, Ben Graham, and Natalia Neverova. 2023. Replay: Multi-modal Multi-view Acted Videos for Casual Holography. arXiv:2307.12067 [cs.CV]

[39] Ashutosh Singla, Thomas Robotham, Abhinav Bhattacharya, William Menz, Emanuël A. P. Habets, and Alexander Raake. 2023. Saliency of Omnidirectional Videos with Different Audio Presentations: Analyses and Dataset. In *2023 15th International Conference on Quality of Multimedia Experience (QoMEX)*. 264–269.

[40] Robert P. Spang, Jan-Niklas Voigt-Antons, and Sebastian Möller. 2022. The Story time Dataset: Simulated Videotelephony Clips for Quality Perception Research. In *2022 14th International Conference on Quality of Multimedia Experience (QoMEX)*. 1–6.

[41] Jackson Tolins, Kris Liu, Yingying Wang, Jean E. Fox Tree, Marilyn A. Walker, and Michael Neff. 2016. A Multimodal Motion-Captured Corpus of Matched and Mismatched Extravert-Introvert Conversational Pairs. In *Proceedings of the Tenth International Conference on Language Resources and Evaluation (LREC'16)*. European Language Resources Association (ELRA), Portorož, Slovenia, 3469–3476.

[42] Bavo Van Kerrebroeck, Giusy Caruso, and Pieter-Jan Maes. 2021. A Methodological Framework for Assessing Social Presence in Music Interactions in Virtual Reality. *Frontiers in Psychology* 12 (2021).

[43] Rob van Son, Wieneke Wesseling, Eric Sanders, and Henk van den Heuvel. 2008. The IFADV Corpus: a Free Dialog Video Corpus. In *Proceedings of the Sixth International Conference on Language Resources and Evaluation (LREC'08)*, Nicoletta Calzolari, Khalid Choukri, Bente Maegaard, Joseph Mariani, Jan Odijk, Stelios Piperidis, and Daniel Tapias (Eds.). European Language Resources Association (ELRA), Marrakech, Morocco.

[44] Jason Vandeventer, Andrew J. Aubrey, Paul L. Rosin, and David Marshall. 2015. 4D Cardiff Conversation Database (4D CCDb): a 4D database of natural, dyadic conversations. In *Auditory-Visual Speech Processing*. 157–162.

[45] M. Vianaz, Ciro Martins, M. Mascarenhasz, Hugo Meinedoy, Jo ao Netoy, Carlos Ribeiro, and Isabel Trancoso. 2002. Spoken Language Corpora for Speech Recognition and Synthesis in European Portuguese. (02 2002).

[46] Isnaeni Wahab, Zul Astri, Novalia Tanasy, and Nurul Fachrunnisa. 2021. A Conversation Analysis: The Use of Small-Talk. *Seltics Journal: Scope of English Language Teaching Literature and Linguistics* 4, 1 (Jun. 2021), 53–62.

[47] Yanqiu Wei, Ling Zhang, Tao Lian, Ying Chen, Sihua Liao, Qi Guo, and Ping Hu. 2024. Happy storytelling promotes emotional contagion and interpersonal closeness. *Current Psychology* 43, 4 (2024), 2993–3001.

[48] Florian Weidner, Gerd Boettcher, Stephanie Arevalo Arboleda, Chenyao Diao, Luljeta Sinani, Christian Kunert, Christoph Gerhardt, Wolfgang Broll, and Alexander Raake. 2023. A Systematic Review on the Visualization of Avatars and Agents in AR & VR displayed using Head-Mounted Displays. *IEEE Transactions on Visualization and Computer Graphics* 29, 5 (2023), 2596–2606.

[49] Ming Yan, Yan Zhang, Shuqiang Cai, Shuqi Fan, Xincheng Lin, Yudi Dai, Siqi Shen, Chenglu Wen, Lan Xu, Yuexin Ma, and Cheng Wang. 2024. RELI11D: A Comprehensive Multimodal Human Motion Dataset and Method. arXiv:2403.19501 [cs.CV]

[50] Yichao Yan, Zanwei Zhou, Zi Wang, Jingnan Gao, and Xiaokang Yang. 2023. DialogueNeRF: Towards Realistic Avatar Face-to-Face Conversation Video Generation. arXiv:2203.07931 [cs.CV]

[51] Jinming Zhao, Tenggan Zhang, Jingwen Hu, Yuchen Liu, Qin Jin, Xinchao Wang, and Haizhou Li. 2022. M3ED: Multi-modal Multi-scene Multi-label Emotional Dialogue Database. , 5699–5710 pages.

MazeLab: A Large-Scale Dynamic Volumetric Point Cloud Video Dataset With User Behavior Traces

Jérémy Ouellette
Concordia University
Canada

Jashanjot Singh Sidhu
Concordia University
Canada

Abdelhak Bentaleb
Concordia University
Canada

ABSTRACT

Point cloud video datasets enriched with user behavior traces are critical for advancing tile-based HTTP adaptive streaming (HAS) systems, particularly those reliant on large training data for player modules like Field of View (FoV) prediction. This paper presents MazeLab, a dynamic volumetric video dataset comprising a feature-rich point cloud representation of a large maze environment. The dataset captures navigation traces from 15 participants interacting with 15 distinct maze variants, categorized into seven classes designed to elicit specific behavioral characteristics such as navigation patterns, attention hotspots, and interaction dynamics. By incorporating diverse environmental designs, each targeting unique user responses, MazeLab provides a comprehensive behavioral repository, enabling robust testing and optimization of HAS-driven volumetric video systems in complex scenarios. The dataset is publicly available here.

CCS CONCEPTS

• **Information systems** → **Multimedia streaming**.

KEYWORDS

6-DoF Dataset, Point Cloud, FoV Prediction.

ACM Reference Format:
Jérémy Ouellette, Jashanjot Singh Sidhu, and Abdelhak Bentaleb. 2025. MazeLab: A Large-Scale Dynamic Volumetric Point Cloud Video Dataset With User Behavior Traces. In *ACM Multimedia Systems Conference 2025 (MMSys '25), March 31–April 4, 2025, Stellenbosch, South Africa.* ACM, New York, NY, USA, 7 pages. https://doi.org/10.1145/3712676.3718345

1 INTRODUCTION

The evolution of multimedia streaming has led to the rise of volumetric videos, which provide immersive experiences by enabling users to navigate 3D environments with six degrees of freedom (6-DoF) [6]. A key approach to delivering such content is through point cloud data (PCD) transmission, which represents 3D scenes as collections of discrete points [4, 20]. While PCDs can offer a highly detailed representation of complex environments, creating smooth, realistic surfaces requires a large number of points, leading to substantial data sizes that can strain network capacity. As volumetric video becomes more mainstream, efficiently managing these data volumes is critical.

To mitigate the challenges of delivering large-scale volumetric content, tile-based HTTP adaptive streaming (HAS) has emerged as an effective solution [1, 20]. This method selectively pre-fetches tiles based on the viewer's predicted field of view (FoV), delivering higher-quality tiles within the FoV while retrieving out-of-FoV tiles at a lower quality or omitting them altogether [21]. By prioritizing relevant content, tile-based HAS solutions [8] reduce bandwidth consumption without sacrificing user experience. This approach, which adapts to user behavior, is particularly crucial for volumetric video where bandwidth requirements can be immense.

Point cloud video datasets are essential for testing and refining tile-based HAS systems, as without them, we are blind to the real-world effectiveness of our system implementations, making their development a hard task. These datasets provide realistic, high-quality data that simulates immersive 3D environments, enabling the training of machine learning models such as FoV predictors, which are critical for optimizing tile selection and improving system performance. However, to the best of our knowledge, existing public datasets [7, 9, 12, 18, 19] are all either missing a complex 3D environment or lacking detailed user behavior (or consumption) traces. Specifically, they do not capture how participants leverage their immersive experience in navigating the environment, which is essential for informed optimization.

This paper introduces MazeLab, a novel dataset that includes both point cloud video data from a complex giant dynamic maze environment and detailed user behavior information, effectively addressing this gap. By capturing both user behavior and content engagement, MazeLab facilitates the refinement of FoV predictors and other key components of tile-based HAS systems. The paper is organized as follows: Section 2 provides an overview of related work, followed by an introduction to the MazeLab dataset in Section 3. Section 4 outlines the MazeLab data collection methodology, with experimental results on existing FoV solutions presented in Section 5. Finally, Section 6 concludes the paper.

2 RELATED WORK

2.1 Point Cloud Video Datasets and FoV Traces

Over the years, numerous point cloud video datasets have been developed to advance applications such as compression, streaming, and rendering. Prominent examples include the *8i Voxelized Full Bodies (8iVFB v2) dataset* [9], the *Dynamic 3D Point Cloud dataset* [19], the *vsenseVVDB2 dataset* [22], and the *OMMO dataset* [12]. These datasets have contributed significantly to the field, providing valuable data for various applications in immersive environments. However, despite the abundance of datasets focused on point cloud

Permission to make digital or hard copies of all or part of this work for personal or classroom use is granted without fee provided that copies are not made or distributed for profit or commercial advantage and that copies bear this notice and the full citation on the first page. Copyrights for components of this work owned by others than the author(s) must be honored. Abstracting with credit is permitted. To copy otherwise, or republish, to post on servers or to redistribute to lists, requires prior specific permission and/or a fee. Request permissions from permissions@acm.org.
MMSys '25, March 31–April 4, 2025, Stellenbosch, South Africa
© 2025 Copyright held by the owner/author(s). Publication rights licensed to ACM.
ACM ISBN 979-8-4007-1467-2/25/03
https://doi.org/10.1145/3712676.3718345

Table 1: Comparison of dynamic point cloud datasets. ✓: Feature present, ✗: Feature absent.

Dataset	Dynamic Content	Behavior Traces	Selectable Density	Spatial Size	Objects/Scene	Environment Type
8iVFB v2 [9]	✓	✗	✗	Small	Single	Real-World
Dynamic 3D [19]	✓	✗	✗	Small	Single	Real-World
vsenseVVDB2 [22]	✓	✗	✗	Small	Single	Real-World
OMMO [12]	✓	✗	✗	Large	Multiple	Real-World
User Navigation [18]	✓	✓	✗	Small	Single	Real-World
VVWUB [7]	✓	✓	✗	Medium	Multiple	Real-World
MazeLab	✓	✓	✓	**Large**	**Multiple**	**Virtual (UE5)**

videos, there is a noticeable gap in datasets that capture user interaction and perception of these videos. Specifically, there is a lack of data that documents how users consume the content, particularly in terms of their gaze patterns and areas of attention. For example, FoV traces, which record user gaze patterns, are crucial for understanding user attention and optimizing video streaming.

Datasets such as the *Volumetric Video Watching User Behavior (VVWUB) dataset* [7] and the *User Navigation dataset* [18] provide some insights into user navigation and areas of interest. However, their small spatial size and simple environments, where the entire scene often fits within the user's FoV, make them less suitable for developing and testing tiling-based streaming approaches, as these rely on varying visibility across different regions. This limitation is particularly evident in the *User Navigation dataset*, which is partly based on the *8iVFB v2 dataset*, as it features videos with only a single object. This results in trivial gaze and movement patterns, as the user's FoV remains centered on it, offering little challenge for evaluating adaptive streaming or FoV prediction techniques.

This phenomenon stands in stark contrast to the complexities of real-world applications, such as streaming point cloud video of sports events (*e.g.*, hockey), where large, dynamic, multi-object 3D environments are involved. In these scenarios, tiling and adaptive streaming methods would play a critical role in optimizing data transmission, as users may only be interested in specific areas of the environment at any given time. This calls for datasets that provide both a complex 3D environment and detailed user consumption traces, which are currently lacking in existing datasets. Such datasets are essential for refining optimization modules like FoV predictors, and for making the next generation of immersive media streaming systems more efficient and user-centric. Table 1 presents a comparison of key characteristics across the most widely used dynamic point cloud datasets, highlighting features such as dynamic content, user behavior traces, selectable density, spatial size, number of concurrent objects, and environment type.

2.2 6-DoF FoV Prediction

Point cloud video streaming is bandwidth-intensive, requiring significant data for detailed 3D representations. Adaptive streaming mitigates this by prioritizing higher quality in areas within the FoV, making accurate FoV prediction essential for optimizing efficiency. While heuristic-based methods exist, learning-based models are increasingly favored due to their adaptability and potential for higher accuracy when trained on diverse, large-scale datasets.

FoV prediction methods are broadly categorized into Head-Mounted Display Trajectory-Based (HTB) and Salience-Aware (SA) predictors. HTB predictors [3, 10, 13, 16] use HMD position and orientation to forecast future FoVs, effectively capturing user movement patterns but overlooking scene content and attention. In contrast, SA predictors analyze the geometry and texture of point clouds to pinpoint attention-grabbing regions [5, 11, 15, 17], achieving finer predictions at the cost of higher computational demands.

To enhance SA predictors' ability to identify which parts of a point cloud capture a user's attention, mapping the participant's FoV onto the point cloud to determine the visible regions is highly beneficial. However, this task is inherently challenging due to the complexities of 3D environments and occlusion effects, which obscure parts of the scene that are within the geometric FoV but not actually visible to the user. To our knowledge, no existing dataset that provides user navigation traces also includes explicit per-point visibility information, alongside other features like material color.

2.3 Unreal Engine 5

MazeLab leveraged Unreal Engine 5 (UE5)[1] for designing intricate maze environments and recording participant behavior. UE5 supports the creation of interactive menus, flexible scripting capabilities for incorporating animations like opening and closing doors within the maze, and seamless integration with a wide range of VR headsets. Additionally, its efficient C++ implementation allows for smooth performance in large, complex environments. UE5 also provides the capability to determine which objects in the environment are rendered in each frame, under the assumption that only visible objects are rendered—an assumption that generally holds true, despite occasional manageable noise. These features make UE5 a versatile and powerful tool for creating and studying immersive 3D experiences.

3 MAZELAB DATASET

The MazeLab dataset is a dynamic point cloud video dataset with user interaction data from 15 participants navigating 15 maze variants designed to influence behavior. Its primary purpose is to serve as a benchmark for FoV prediction models, assessing their ability to identify environmental cues and forecast user behavior. The environment was captured from UE5 as metadata, storing only essential information to reconstruct each frame as a point cloud (*e.g.*, corner

[1] Online [Available]: https://www.unrealengine.com/

Table 2: Summary of The MazeLab Dataset

Metric	Value	Description
Maze Size	120x150x10m	Dimensions in UE5 meters.
Simulations	15	90-second scenarios capturing diverse conditions.
Participants/Sim.	~14	Total of 209 participant-simulation pairs.
PLY Frame Rate	10 fps	900 frames per participant-simulation pair.
HMD Position Rate	60 Hz	5400 samples per participant-simulation pair.
Points/Frame	~3 million	XYZ, material RGB, nearest light RGBI, and render flag.

coordinates for rectangular prisms). Using this metadata, the original mesh maze can be reconstructed with our provided tool[2] as a point cloud video with the desired density, floating-point precision, point-level features, and more. However, for convenience, we also provide a pre-made point cloud video representation optimized for a balance between features, density, and storage size. In both cases, whether using the conversion tool or the pre-made representation, all static objects in the maze are saved only once per variant and must be appended to every PLY frame of that variant before consumption. This optimization reduces the dataset size by several orders of magnitude, from petabytes to hundreds of gigabytes, making it significantly more accessible to the community. Table 2 provides an overview of the key characteristics of the pre-made dataset representation, the details of which are provided in the subsequent sections.

3.1 Maze Structure

The dataset is centered around a vast, single-story maze deliberately designed to be disorienting and monotonous, making it easy for participants to get lost or inadvertently move in circles. As shown in Figure 1, the maze has a somewhat rectangular layout, such that a rather simple rectangular tiling scheme can be effective. The maze spans 120x150x10 UE5 meters[3] and consists of 14 uniformly sized rooms connected by hallways, with numerous circular paths that closely resemble each other. To introduce some variance in the z-axis, certain hallways are replaced with bridges—structures that ascend, level out, and descend back to the original floor height. The walls and floors are uniformly dark gray, with slightly paler floors, and the lighting is uncolored and evenly distributed throughout, creating a bland and muted environment. No decorations or sound-emitting elements are present, and doors between rooms and hallways are open by default, allowing seamless navigation. This minimalistic default design ensures that any added stimuli or objects in the maze variants stand out prominently, allowing researchers to effectively study their impact on participant behavior. While not an integral part of the maze itself, the participant's body position is also recorded, as illustrated in Figure 2 where three different instances of the participant are shown, each spaced 24 frames apart. This inclusion helps SA prediction models easily determine the participant's location within the maze and estimate what might fall within their FoV. A few images of the maze in its various experimental configurations are showcased in Figure 3.

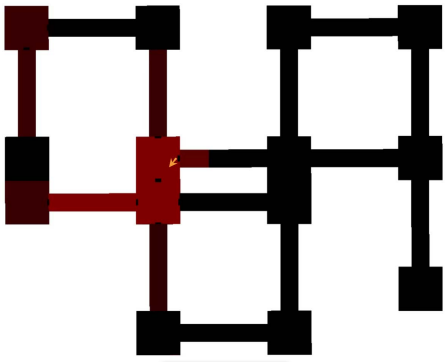


Figure 1: Bottom-up view of a MazeLab dataset sample. The participant is represented as an orange arrow, and the salient regions are highlighted in red.

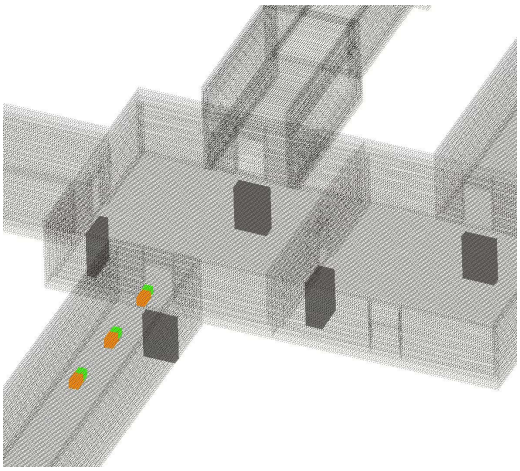
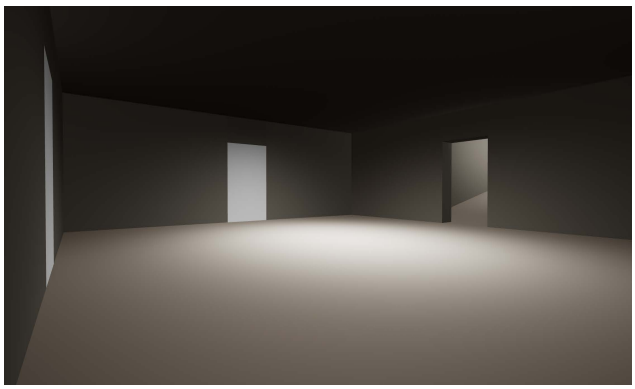


Figure 2: A participant moving through the point cloud maze, entering a room with all doors open.

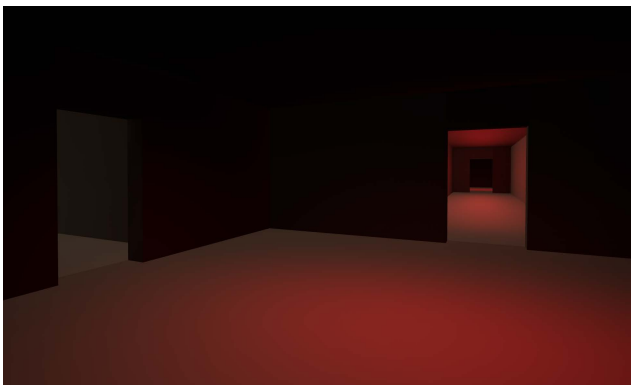
3.2 Simulations

In this paper, the term **simulation** refers to the 15 test scenarios in which participants were recorded. Each simulation is designed to introduce specific stimuli or environmental characteristics intended to influence user behavior in a controlled and observable way. These scenarios serve as benchmarks to evaluate the FoV predictors' ability to effectively utilize these stimuli for more accurate predictions. The MazeLab dataset contains 209 unique simulation-participant pairs, each lasting 90 seconds. These 15 simulations are grouped into 7 categories. Simulations within the same category only differ slightly, such as in the participant's starting location or in doors/lighting properties. These subtle variations help identify patterns in user behavior while increasing the dataset's diversity, enhancing its utility for training machine learning models by enabling them to generalize more effectively to new scenarios once they have been trained. The simulations are described as follow:

[2]GiHub Repository: https://github.com/Thehappydeveloper/MazeLab-Toolset
[3]In UE5, the default unit of measurement is centimeters, where 1 UE5 unit equals 1 cm. Therefore, 1 meter equals 100 UE5 units.



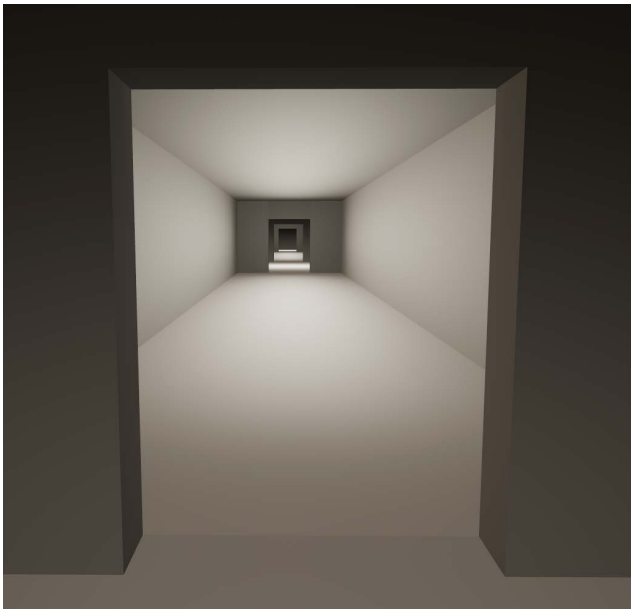
(a) A room with one open door from category 2 simulations.



(b) A red path in simulation $S5$ lit in an otherwise dark maze.



(c) A room from simulation $S11$ featuring the long dress point cloud.



(d) Hallways connecting the various rooms within the maze.

Figure 3: Visual representations of different maze simulations.

(1) **[4 simulations:** $S1$**,** $S2$**,** $S3$**,** $S7$**]** All doors are open, allowing participants to explore the maze freely without restrictions (See Figure 3d). The only difference between the simulations is the player's starting position.
(2) **[4 simulations:** $S4$**,** $S8$**,** $S12$**,** $S15$**]** Certain doors are kept closed, guiding participants along a predefined path through the maze (See Figure 3a). The path and starting location vary between simulations.
(3) **[2 simulations:** $S9$**,** $S13$**]** All doors are open, but a path is marked by either blue lights ($S9$) or red lights ($S13$), with the remaining lights uncolored.
(4) **[2 simulations:** $S5$**,** $S14$**]** All doors are open, with a path marked by either red lights ($S5$) or blue lights ($S14$), while the rest of the maze remains completely dark (See Figure 3b).
(5) **[1 simulation:** $S11$**]** Static PLY objects from the *8iVFBv2* dataset (*Loot*, *Longdress*, *Soldier*, and *Redandblack*) are hidden throughout the maze. Participants are encouraged to find all four objects (See Figure 3c).
(6) **[1 simulation:** $S10$**]** A fast-moving, rapidly flashing sphere alternates between red and blue, guiding participants through the maze.
(7) **[1 simulation:** $S6$**]** Similar to $S10$, but the sphere leads to a closed room containing the *8iVFBv2* PLY objects.

Note: The numbering (e.g., "S1") indicates the order in which the simulations were presented to the participants.

3.3 Point Cloud Video Features

The provided point cloud representation consists of 90-second video sequences, captured at 10 frames per second, resulting in 900 frames per video. The HMD position and angles were recorded at 60 frames per second, accumulating to 5400 samples per video. Each frame contains approximately 3 million points, with each point having thirteen features, all stored as 32-bit floating-point values for precision. These features include:

- **XYZ coordinates**: The 3D spatial location of each point.

- **Material RGBA values**: The color (RGB) and transparency (A) of the material of each point.
- **Light source RGBAI values**: The color (RGB), transparency (A) and intensity (I) of the nearest light source.
- **Render flag**: A boolean value indicating whether the point is visible (rendered) in the current frame. This flag is set to `true` only if the point is within the user's FoV.

While the metadata was recorded at 60 fps and converted to 10 fps for storage efficiency, it can be converted back to a 60 fps point cloud video if higher temporal resolution is needed.

4 COLLECTION METHODOLOGY

This study employed a structured data collection methodology to ensure consistency, maintain participant engagement, and preserve anonymity.

4.1 Participants

Fifteen participants were randomly selected[4], and no personal information, such as age, gender, or name, was recorded to ensure anonymity. Participants were selected regardless of their prior experience with VR headsets. However, if anyone faced significant difficulty navigating the virtual environment, their participation was ended, and no data was collected. This approach was adopted to minimize noise in the dataset and maintain the quality of the collected data.

4.2 Experimental Setup

The movement of the virtual player in the maze was initially tied to the participant's physical movements in the real world. However, replicating the maze's large size on a 1:1 scale was impractical. To address this, a joystick-based movement system was implemented for translational movement (forward, backward, left, right), while rotational movement was achieved by physically turning in place. The system used a non-linear speed model, where acceleration increased proportionally with joystick displacement, ensuring intuitive and responsive control. Participants were seated on a stationary wheeled chair to allow in-place rotation and were instructed to remain seated to avoid collisions with real-world objects. Each participant was equipped with a Meta Quest 2 headset, featuring a horizontal FoV of $\mathbf{97}^{\circ}$ and a vertical FoV of $\mathbf{93}^{\circ}$ [2].

4.3 Quality Control Guidelines

The primary goal during data collection was to capture participants' movements and FoV as they interacted with stimuli in the maze. To ensure high-quality data, a structured protocol was implemented to maximize data quality and minimize noise. The 15 simulations, lasting approximately 30 minutes (including transitions between simulations), were carefully designed to engage participants and maintain consistency.

Step 1. **Movement Test and Familiarization:** Participants enter a menu area to perform movement tests and familiarize themselves with the controls, including joystick-based translation and physical rotation.

Step 2. **Briefing:** Before all simulations, participants are informed that the study aims to capture their movements and behaviors in response to stimuli within the maze. They are briefed on the specific objectives of each task, such as following paths or locating scattered point cloud statues, ensuring clarity and consistency in their actions.

Step 3. **Initial Practice Phase:** Participants are teleported into the maze for three simple practice simulations ($S1$–$S3$), where they are free to explore the maze at their own pace, without explicit objectives, for about five minutes. The goal is to familiarize themselves with the maze and the movement system. While monotonous, these practice simulations serve to help participants gain confidence in navigating the environment and also enhance the contrast and salience of stimuli later in more complex simulations.

Step 4. **Challenge Phase:** Following the learning phase, the simulation sequence ($S4$–$S15$) alternated between different categories. Unlike the learning phase, where participants focused on exploration, most simulations now required participants to complete a specific objective before the timer expired. For path-based simulations, participants were encouraged to retrace their steps in reverse once they reached the end, continuing this loop as many times as possible until the timer ran out. Preliminary trials indicated that consecutive similar simulations often led to disengagement, manifesting as aimless wandering or attempts to exploit the environment (*e.g.*, glitching through walls). To counter this, the alternating simulation types were designed to sustain participant engagement and promote more natural interactions within the maze.

Step 5. **Completion:** After completing all simulations, participants finish the study and exit the maze environment.

By adhering to this structured protocol, we maintained a balance between consistency and variability in participant behavior within the maze, establishing the use of this dataset as a reliable benchmark for FoV prediction models. These models can be evaluated on their ability to recognize environmental cues and predict how participants respond to them, ensuring practical applicability in dynamic scenarios.

5 RESULTS AND ANALYSIS

This section provides an overview of the behavioral data collected and the results of applying FoV prediction to this dataset, focusing on both trajectory prediction and tile salience score prediction. The results presented here are derived from our previous work [14], where we trained multiple predictors on the pre-processed point cloud version of this dataset. These predictors were evaluated based on their ability to accurately predict FoV using the captured participant movements, gaze data, and the point cloud environment. A comprehensive discussion on the nuances of these results and the performance of the predictors can be found in [14].

5.1 Trace Analysis

Figure 4 shows the path taken by a random participant in simulation $S15$ during the first 45 seconds. The movement is represented in blue, with green arrows indicating the viewing direction at regular

[4]This work does not raise any ethical concerns.

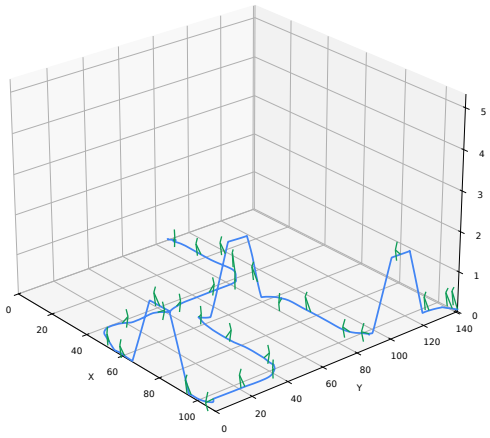


Figure 4: Path of a random user starting from the right corner in simulation $S15$, during the first 45 seconds.

Table 3: Model Performance Metrics Rounded to the Third Decimal. ↓: lower is better (green), lowest performance (red).

Model	0.5s	1s	2s	3s	4s	5s
GRU [17]	**0.021** ↓	**0.021** ↓	0.025	**0.024**↓	0.024	0.024
LSTM [13]	**0.021**↓	0.022	**0.023**↓	**0.024** ↓	**0.022**↓	0.023 ↓
MAMBA [3]	0.026	0.038	0.047	0.052	0.056	0.059
JAMBA [10]	0.025	0.037	0.044	0.047	0.051	0.053
NAIVE	0.041	0.061	0.086	0.106	0.124	0.139
LINEAR [16]	0.078	0.094	0.130	0.177	0.212	0.233

intervals. Participant behavior is highly predictable, with most movement segments being linear, especially when split into 0.5-second intervals. This indicates that predicting user position is relatively easy, as their movements tend to be linear and consistent within short time frames.

5.2 FoV Predictors

We compared multiple FoV predictor baselines: *NAIVE*, which predicts future states as identical to the last observed state; *LINEAR* [16], one linear model per feature for capturing simple trends; *LSTM* [13], leveraging memory cells and gating mechanisms to capture long-term dependencies; *GRU* [17], a lightweight RNN variant with fewer gates than LSTM; *MAMBA* [3], which simplifies Transformers via Selective-State-Spaces (SSM) for linear sequence-length scaling and hardware-aware parallelism; and *JAMBA* [10], combining Transformers with MAMBA's SSM for global dependency capture and efficient processing.

5.3 User Trajectory Prediction

Table 3 shows the performance of the HTB predictors introduced above in predicting user trajectory and gaze direction at different prediction horizons, with performance measured by mean squared error (MSE). As expected, accuracy decreases as the prediction horizon increases, primarily because movements become less linear over time and noise accumulates. However, it is worth noting that the GRU and LSTM models surprisingly maintain relatively high accuracy even at the 5-second mark, outperforming alternatives like MAMBA and JAMBA. These models exhibit more consistent predictions, while the other models show a noticeable increase in MSE, particularly for longer horizons.

Table 4: Latency, VRAM Usage, and Average Validation Loss for Different Models. ↓: lower is better (green), lowest performance (red).

Model	Latency (ms)	VRAM Usage (MB)	Avg. IoU Validation Loss
Naive	**3.39**↓	292.56	**0.34**↓
Linear	5.65	293.62	0.38
GRU	3.61	293.34	0.37
LSTM	3.99	294.55	0.36
MAMBA	5.15	**292.47**↓	0.38
JAMBA	6.89	292.93	0.38
LL-PointNet [14]	67.13	590.23	0.40
LL-SparseConv [14]	60.87	821.71	0.42

5.4 Tile Salience Score Prediction

Table 4 presents a comparison of latency, VRAM usage, and average validation loss across models for tile salience score prediction at a prediction horizon of 0.5 second. All models operated on a voxelized representation of the point cloud, stored as a sparse tensor with 415K points per frame. Simpler models like Naive and Linear are the fastest, with Naive achieving the best validation loss, while GRU and LSTM models strike a balance between accuracy and efficiency.

LL-PointNet, a hybrid model combining GRU and PointNet, processes a fixed 40K downsampled points per frame, resulting in lower VRAM usage compared to LL-SparseConv. LL-SparseConv, a hybrid between a GRU and sparse convolution, processes the full voxelized point cloud. While LL-SparseConv was designed as a proof-of-concept for VRAM-efficient and scalable salience prediction, it is not yet optimized for high accuracy but demonstrates significant potential.

Notably, salience prediction models, such as LL-PointNet and LL-SparseConv, distribute resources across tiles more intelligently and evenly than trajectory prediction models, despite what the validation loss might suggest. A user-based study evaluating the real QoE provided by these models would offer valuable insights into their practical effectiveness.

6 CONCLUSION

This paper introduced the `MazeLab` dataset, a comprehensive, large-scale dynamic point cloud video dataset that integrates high-quality point cloud data with detailed user interaction traces. The dataset addresses a critical need for realistic content in the development and optimization of tile-based HAS systems. By combining point cloud video with user behavior data, `MazeLab` serves as both an effective training platform and a benchmark for evaluating FoV prediction models, emphasizing their ability to detect environmental cues and leverage them to predict user behavior with enhanced accuracy.

Acknowledgments. This research is funded by the Natural Sciences & Engineering Research Council (NSERC)-Discovery Grant RGPIN-2023-04744 and NSERC Alliance (ALLRP 586434-23)-Mitacs Accelerate (IT36792) grant.

REFERENCES

[1] Abdelhak Bentaleb, Bayan Taani, Ali C. Begen, Christian Timmerer, and Roger Zimmermann. 2019. A Survey on Bitrate Adaptation Schemes for Streaming Media Over HTTP. *IEEE Communications Surveys & Tutorials* 21, 1 (2019), 562–585. https://doi.org/10.1109/COMST.2018.2862938 Conference Name: IEEE Communications Surveys & Tutorials.

[2] Rory Brown. [n. d.]. Oculus Quest 2: Full Specification. https://vr-compare.com/headset/oculusquest2 VRcompare.

[3] Albert Gu and Tri Dao. 2024. Mamba: Linear-Time Sequence Modeling with Selective State Spaces. http://arxiv.org/abs/2312.00752 arXiv: 2312.00752 [cs].

[4] Yongjie Guan, Xueyu Hou, Nan Wu, Bo Han, and Tao Han. 2023. MetaStream: Live Volumetric Content Capture, Creation, Delivery, and Rendering in Real Time. In *Proceedings of the 29th Annual International Conference on Mobile Computing and Networking*. ACM, Madrid, Spain, 1–15. https://doi.org/10.1145/3570361.3592530

[5] Xian-Feng Han, Jesse S. Jin, Juan Xie, Mingjie Wang, and Wei Jiang. 2018. A comprehensive review of 3D point cloud descriptors. *ArXiv* (Feb 2018). https://www.semanticscholar.org/paper/A-comprehensive-review-of-3D-point-cloud-Han-Jin/a10980188f8d2903a0c8fc1b849751877199c709

[6] Jeroen van der Hooft, Maria Torres Vega, Tim Wauters, Christian Timmerer, Ali C. Begen, Filip De Turck, and Raimund Schatz. 2020. From Capturing to Rendering: Volumetric Media Delivery with Six Degrees of Freedom. *IEEE Communications Magazine* 58, 10 (2020), 49–55. https://doi.org/10.1109/MCOM.001.2000242

[7] Kaiyuan Hu, Haowen Yang, Yili Jin, Junhua Liu, Yongting Chen, Miao Zhang, and Fangxin Wang. 2023. Understanding User Behavior in Volumetric Video Watching: Dataset, Analysis and Prediction. In *Proceedings of the 31st ACM International Conference on Multimedia*. ACM, Ottawa, ON, Canada, 1108–1116. https://doi.org/10.1145/3581783.3613810

[8] Jong-Beom Jeong, Soonbin Lee, Il-Woong Ryu, Tuan Thanh Le, and Eun-Seok Ryu. 2020. Towards Viewport-dependent 6DoF 360 Video Tiled Streaming for Virtual Reality Systems. In *Proceedings of the 28th ACM International Conference on Multimedia* (Seattle, WA, USA) *(MM '20)*. Association for Computing Machinery, New York, NY, USA, 3687–3695. https://doi.org/10.1145/3394171.3413712

[9] Maja Krivokuća, Philip A. Chou, and Patrick Savill. 2018. 8i Voxelized Surface Light Field (8iVSLF) Dataset. ISO/IEC JTC1/SC29 WG11 (MPEG) input document m42914. https://mpeg-pcc.org/index.php/pcc-content-database/8i-voxelized-surface-light-field-8ivslf-dataset/

[10] Opher Lieber, Barak Lenz, Hofit Bata, Gal Cohen, Jhonathan Osin, Itay Dalmedigos, Erez Safahi, Shaked Meirom, Yonatan Belinkov, Shai Shalev-Shwartz, Omri Abend, Raz Alon, Tomer Asida, Amir Bergman, Roman Glozman, Michael Gokhman, Avashalom Manevich, Nir Ratner, Noam Rozen, Erez Shwartz, Mor Zusman, and Yoav Shoham. 2024. Jamba: A Hybrid Transformer-Mamba Language Model. https://doi.org/10.48550/arXiv.2403.19887 arXiv:2403.19887 [cs].

[11] Junhua Liu, Boxiang Zhu, Fangxin Wang, Yili Jin, Wenyi Zhang, Zihan Xu, and Shuguang Cui. 2023. CaV3: Cache-assisted Viewport Adaptive Volumetric Video Streaming. In *2023 IEEE Conference Virtual Reality and 3D User Interfaces (VR)*. IEEE, Shanghai, China, 173–183. https://doi.org/10.1109/VR55154.2023.00033

[12] Chongshan Lu, Fukun Yin, Xin Chen, Tao Chen, Gang YU, and Jiayuan Fan. 2023. A Large-Scale Outdoor Multi-modal Dataset and Benchmark for Novel View Synthesis and Implicit Scene Reconstruction. arXiv:2301.06782 [cs.CV] https://arxiv.org/abs/2301.06782

[13] Gioacchino Manfredi, Vito Andrea Racanelli, Luca De Cicco, and Saverio Mascolo. 2023. LSTM-based Viewport Prediction for Immersive Video Systems. In *2023 21st Mediterranean Communication and Computer Networking Conference (MedComNet)*. 49–52. https://doi.org/10.1109/MedComNet58619.2023.10168847

[14] Jérémy Ouellette and Abdelhak Bentaleb. 2025. LL-Sparse: Low-Latency 6-DoF Field of View Prediction. In *The 16th ACM Multimedia Systems Conference (MMSys '25)* (March 31–April 4, 2025). ACM, Stellenbosch, South Africa, 11. https://doi.org/10.1145/3712676.3714438

[15] Charles R. Qi, Hao Su, Kaichun Mo, and Leonidas J. Guibas. 2017. PointNet: Deep Learning on Point Sets for 3D Classification and Segmentation. https://doi.org/10.48550/arXiv.1612.00593 arXiv:1612.00593 [cs].

[16] Feng Qian, Lusheng Ji, Bo Han, and Vijay Gopalakrishnan. 2016. Optimizing 360 video delivery over cellular networks. In *Proceedings of the 5th Workshop on All Things Cellular: Operations, Applications and Challenges* (New York City, New York) *(ATC '16)*. Association for Computing Machinery, New York, NY, USA, 1–6. https://doi.org/10.1145/2980055.2980056

[17] Mehdi Setayesh and Vincent W.S. Wong. 2023. A Content-based Viewport Prediction Framework for 360° Video Using Personalized Federated Learning and Fusion Techniques. In *2023 IEEE International Conference on Multimedia and Expo (ICME)*. IEEE, Brisbane, Australia, 654–659. https://doi.org/10.1109/ICME55011.2023.00118

[18] Shishir Subramanyam, Irene Viola, Alan Hanjalic, and Pablo Cesar. 2020. User Centered Adaptive Streaming of Dynamic Point Clouds with Low Complexity Tiling. In *Proceedings of the 28th ACM International Conference on Multimedia (MM '20)*. Association for Computing Machinery, New York, NY, USA, 3669–3677. https://doi.org/10.1145/3394171.3413535

[19] Yuan-Chun Sun, I-Chun Huang, Yuang Shi, Wei Tsang Ooi, Chun-Ying Huang, and Cheng-Hsin Hsu. 2023. A Dynamic 3D Point Cloud Dataset for Immersive Applications. In *Proceedings of the 14th ACM Multimedia Systems Conference* (Vancouver, BC, Canada) *(MMSys '23)*. Association for Computing Machinery, New York, NY, USA, 376–383. https://doi.org/10.1145/3587819.3592546

[20] Jeroen van der Hooft, Hadi Amirpour, Maria Torres Vega, Yago Sanchez, Raimund Schatz, Thomas Schierl, and Christian Timmerer. 2023. A Tutorial on Immersive Video Delivery: From Omnidirectional Video to Holography. *IEEE Communications Surveys & Tutorials* 25, 2 (2023), 1336–1375. https://doi.org/10.1109/COMST.2023.3263252 Conference Name: IEEE Communications Surveys & Tutorials.

[21] Irene Viola and Pablo Cesar. 2022. Volumetric video streaming: Current approaches and implementations. http://arxiv.org/abs/2209.01982 arXiv:2209.01982 [cs, eess].

[22] Emin Zerman, Cagri Ozcinar, Pan Gao, and Aljosa Smolic. 2020. Textured Mesh vs Coloured Point Cloud: A Subjective Study for Volumetric Video Compression. In *2020 Twelfth International Conference on Quality of Multimedia Experience (QoMEX)*. 1–6. https://doi.org/10.1109/QoMEX48832.2020.9123137

Danger Detection and Cloud-Based Vocal Assistance System for Visually Impaired Users Using Meta Quest 3

Fabrizio De Fiore
Technological University of the Shannon: Midlands Midwest
Athlone, Ireland
fabriziodefiore.work@gmail.com

John Barrett
Technological University of the Shannon: Midlands Midwest
Athlone, Ireland
john.barrett@tus.ie

Niall Murray
Technological University of the Shannon: Midlands Midwest
Athlone, Ireland
nmurray@research.ait.ie

Conor Keighrey
Technological University of the Shannon: Midlands Midwest
Athlone, Ireland
c.keighrey@research.ait.ie

ABSTRACT

Individuals who experience a significant loss of vision or blindness often face challenges navigating the world. In such scenarios, there is often a dependence on various types of support or aids to enable individuals to move through their surroundings. Examples of this include guide dogs and assistive white canes. Those who suffer from blindness also rely on a heightened sense of auditory processing. Despite these solutions, navigating the modern world is challenging from a safety perspective.

In recent years, computer vision algorithms have been developed to interpret and report the contents of image streams. Applications of these algorithms have often focused on automated labeling of data (image analysis and audio processing etc.) within vast media databases; however, they can be leveraged to operate with live camera streams. In this research, the authors present an overview of a proof-of-concept technology stack to implement a danger detection system for individuals with significant visual impairment. The system integrates an advanced cloud driven approach employing a fine-tuned YOLO model for hazard detection, performs obstacle avoidance, segmentation processes for scene understanding, text recognition through optical character recognition (OCR), and visual question answering (VQA) for user queries. These technologies collectively address critical issues such as dynamic hazard recognition, spatial awareness, and personalized assistance.

Permission to make digital or hard copies of all or part of this work for personal or classroom use is granted without fee provided that copies are not made or distributed for profit or commercial advantage and that copies bear this notice and the full citation on the first page. Copyrights for third-party components of this work must be honored. For all other uses, contact the owner/author(s).
MMSys '25, March 31–April 4, 2025, Stellenbosch, South Africa
© 2025 Copyright held by the owner/author(s).
ACM ISBN 979-8-4007-1467-2/25/03.
https://doi.org/10.1145/3712676.3719262

CCS CONCEPTS

• **Human-centered computing → Mixed / augmented reality**; *Accessibility systems and tools.*

KEYWORDS

Mixed Reality; Computer Vision; Vocal Assistant; Accessibility

ACM Reference format:

Fabrizio De Fiore, John Barrett, Niall Murray and Conor Keighrey. 2025. Danger Detection and Cloud-Based Vocal Assistance System for Visually Impaired Users Using Meta Quest 3. In *Proceedings of ACM Multimedia Systems conference (MMSys'25). ACM, Stellenbosch, South Africa, 6 pages.* https://doi.org/10.1145/3712676.3719262

1 Introduction

In recent years head mounted display (HMD) technologies have advanced to allow users to consume more realistic augmented reality (AR) or virtual reality (VR) content. Although numerous HMDs have been released to market [1] [2], Meta's Quest [3] series has gained significant attention in recent years due to its value for consumers, ease of use, and social ecosystem. These portable HMDs allow users to experience alternative realities by capturing and interpreting real-world movements and interactions and transporting them to fully simulated virtual environments. VR as a technology combines high-resolution displays, integrated cameras, and sophisticated tracking technologies to enable users to interpret and interact with their surroundings more effectively, paving the way for immersive and practical accessibility applications.

In AR-based systems, a process known as simultaneous localization and mapping (SLAM) [4] is often used to interpret the user's spatial position in real-time. SLAM enables dynamic mapping of the environment. Another similar technology is

visual-inertial odometry (VIO) which combines visual information (from cameras) and inertial sensor data to estimate the motion and position of a device (HMD or controller) in real-time. The combination of visual data from cameras and inertial measurements such as accelerometers, offers greater accuracy and robustness in movement tracking. Recent studies emphasize the importance of incorporating new and emerging deep learning techniques to further refine these methods [5] [6].

Both SLAM and VIO place an emphasis on spatial detection and localization, however there are other algorithms which focus on object prediction within media inputs. Object prediction algorithms often consist of complex convolutional neural networks (CNN) which require a larger computational power to operate. However, in recent years there has been a prevalence of optimized solutions for scene orientated prediction. YOLO (You Only Look Once) [7] is an open-source artificial intelligence (AI) algorithm which has gained popularity since its introduction in 2015 due to its efficiency in image processing. Light weight AI algorithms have also been deployed to understand the fine components within images such as text. This process is known as optical character recognition (OCR) and offers near real-time translation of text. OCR has been deployed on mobile OS to offer insight into images captured by users as well as novel HMD devices which are used within the accessibility domain [8].

Navigating everyday environments poses considerable challenges for visually impaired individuals, who often rely on traditional aids such as white canes or guide dogs. While these tools provide essential support, they are limited in addressing dynamic and unpredictable hazards, such as moving vehicles, uneven terrain, or crowded spaces. These challenges can lead to heightened dependency, reduced mobility, and a diminished quality of life.

In response to these challenges, assistance technologies have emerged as vital tools to improve the quality of life of people with visual disabilities. The current assistance technologies range from tactile devices to advanced software solutions, aimed at improving the quality of life for those who are blind. These visual assistance technologies has significantly transformed the interpretation of the world for individuals with visual impairment, facilitating increased independence and mobility [8] [9] [10]. However, despite their advances, these technologies face significant limitations, regarding their effectiveness in detecting dangers and obstacles in various environments. A main challenge is the variability of the environments themselves, which can introduce complexities that existing technologies can struggle to manage. For example, urban landscapes rich in dynamic elements (moving vehicles, pedestrians and varied surface conditions etc.) request a different understanding of the spatial context that many systems do not have [11]. In addition, limitations in accessibility, affordability and usability need to be tackled to promote greater integration.

Recent studies illustrate that while assistive technologies are very promising, further efforts are crucial to overcome existing barriers and ensuring that people with visual disabilities can navigate their environments more effectively and more independently [12]. In terms of accessibility, smartphones apps often require users to occupy their hands with the device, limiting their ability to navigate freely and efficiently. Problems such as precision in environmental understanding, dependence on the battery life of smartphones and the requirement that users are mobile technology literate make them less accessible to all people [11]. Sensor-based devices often have difficulty with dynamic environments, and their usability can be limited by factors such as the cost and training of users [13]. Moreover, AI glasses rely on static image processing, where the user initiates the capture and waits for feedback, only performant for simple scenarios. This approach introduces significant delays and fails to address immediate hazards, limiting their utility in dynamic and fast-changing environments.

In this context, this research explores the utility of AR, sensor technology, and AI to create advanced accessibility solutions. The proposed system overcomes current assistive technologies limitations by leveraging high-end processing capabilities powered by cloud infrastructure. Unlike static systems, it provides continuous danger detection and spatial awareness, offering immediate and actionable feedback tailored to the user's environment. By utilizing Meta Quest 3's hands-free interface, the system ensures enhanced mobility and usability, delivering feedback through auditory, haptic, and visual modalities. The work aims to deliver enhanced safety for individuals with visual impairments, combining danger detection, spatial awareness, and feedback into one system. By offloading computational tasks to the cloud, the system ensures high precision, low latency, and scalability, while multimodal feedback mechanisms enhance user accessibility and autonomy. This research aims to design a scalable, cloud-based system that performs effective danger detection and vocal assistance tailored for visually impaired users. To achieve this, it needs to provide intuitive multimodal feedback mechanisms, including vocal, haptic, and visual overlays, to enhance situational awareness, and autonomy in diverse, real-world scenarios.

2 Related Work

The development of assistive technologies for visually impaired individuals has evolved significantly, focusing on addressing mobility and safety challenges. Initial systems predominantly relied on tactile and auditory feedback mechanisms to enhance user navigation. Examples of these systems include white canes equipped with ultrasonic sensors or portable obstacle detection devices. They offered limited interaction with the environment [11] [12].

Smartphone applications can be specifically designed for visually impaired and according to [14] these solutions based on smartphones offer different functionalities, such as navigation assistance, recognition of objects and text capabilities to voice. Be My Eyes [10] is a software solution which aims to connect individuals with vision impairments with volunteers around the world to assist them with everyday challenges. It can be deployed on PC or smartphone.

Wearables also represent a significant step in assistance technology [9]. These devices can improve mobility through integrated GPS systems and haptic feedback mechanisms, allowing users to receive navigation information without audio

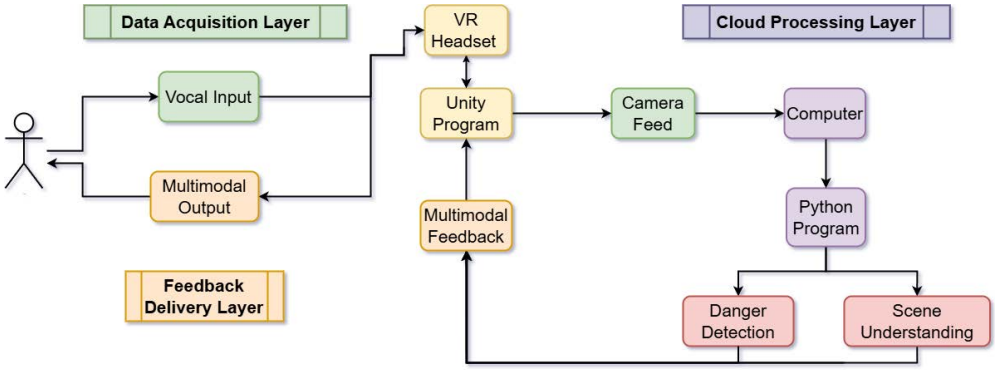


Figure 1: System architecture highlights the data flow between the Data acquisition, Cloud processing, and Feedback layers.

signals, which can be helpful in public environments. Smart glasses have been developed to offer assistive services for individuals with varied levels of visual acuity. Developed by Gentex, eSight [15] offers a smart glasses platform which is targeted at individuals with reduced visual acuity. The product operates by capturing and processing image data captured from the surrounding world and enhancing it for the wearers' needs. Similarly, Envision Glasses [8] have been built on Google Glass, they offer users a range AI enhanced imaged processing to scan and read text, describe a scene, and detect color. More recently, AR and VR technologies have introduced new avenues for assistive systems that provide comprehensive situational awareness. Advanced HMDs like Meta's Quest 3 have enabled the integration of real-time spatial data, creating immersive and intuitive user experiences [16].

AI plays a crucial role in enhancing assistive technologies by enabling real-time object detection, spatial mapping, and semantic scene understanding. Models like YOLO have proven effective in identifying and classifying hazards with high precision, while SLAM has improved spatial navigation by creating dynamic maps of the environment [4] [7]. YOLO offers real-time image recognition, object detection, and object localization. These features have seen significant improvements due to the continued support from the community which has led to improved speed, detection accuracy, and a good level of generalization. Semantic segmentation further enhances environmental understanding by identifying specific objects and their relationships within a scene. Recent findings highlight the effectiveness of AI models which have been tailored for OCR to recognize and interpret the text in various contexts, improving the usability of XR applications [17]. The incorporation of visual question answering (VQA) systems allows for interactive user queries, transforming passive guidance into an interactive, user-centric experience [18]. These AI advancements collectively enable systems that address the unique challenges of visually impaired users by providing timely, context-aware feedback.

Cloud computing has also emerged as a key enabler for assistive technologies by providing scalable resources for computationally intensive tasks. Edge-based solutions often suffer from limitations in processing power and energy consumption, restricting their performance in real-time applications. Cloud-based architecture mitigates these challenges by offloading computational tasks to powerful remote servers, ensuring high-speed processing and low latency [19]. Studies have demonstrated that cloud systems can handle complex AI workflows, such as object detection and scene analysis, with minimal delays, making them ideal for real-time assistive applications [20] [21].

Despite advancements, existing solutions face limitations in achieving the required level of integration, real-time performance, and usability. Many systems rely solely on local processing, which can lead to high latency and reduced scalability. Furthermore, the absence of multimodal feedback mechanisms often restricts accessibility for users with varying degrees of impairment. Privacy concerns and reliance on stable network connections pose additional challenges, emphasizing the need for robust and secure architecture [22]. Most assistance devices are mainly based on sensors, which may find it difficult to differentiate various obstacles and identify potential threats [13]. These challenges highlight the critical need for comprehensive systems that integrate advanced AI capabilities with scalable cloud architectures to provide seamless, reliable, and user-friendly assistive solutions.

3 System Design

The proposed system is designed as a cloud-based framework for danger detection and comprehensive situational assistance. By leveraging the Meta Quest 3's advanced sensors, cameras, and computational capabilities in combination with cloud-based AI processing, the system provides scalable, reliable, and low-latency performance. The architecture focuses on three key processes: data acquisition, AI-powered cloud processing, and multimodal feedback delivery to the user. This distribution aims to deliver a robust, low-latency operation while addressing computational constraints of the HMD device.

3.1 System Architecture

The system architecture (Figure 1) provides an overview of the flow of data throughout operation. This is broken down into three crucial layers of processing (Data acquisition, cloud processing, and feedback delivery layer). Each of these are described in the following sections.

3.1.1 Data Acquisition Layer

The primary objective of the data acquisition layer is to capture and process the cameras streams from the HMD. The Meta Quest 3 captures high-quality RGB video streams through its two 4 mega-pixel cameras located on the front of the device. The cameras have a wide field of view which allows for the capture of images around the user. Due to the technological challenges related to the limited computational processing offered on the device, all data captured is offloaded to the cloud server.

3.1.2 Cloud Processing Layer

Due to the computational requirement of AI models, a cloud server was implemented as part of the design. This server operates as the backbone of the system. Data captured from the Quest 3 cameras is processed using an array of AI models performing object detection, semantic segmentation, and visual question answering. Responses from these systems are driven by user input. Once processed, the system responds to the HMD with data encapsulated as a JSON file. This cloud driven approach provides flexibility in development, allowing for new models to be evaluated and deployed at ease.

3.1.2.1 Danger Detection

YOLO is used to process video frames for object detection and classification. The model is fine-tuned to identify hazards such as moving vehicles, stationary obstacles, uneven surfaces or drop-offs, pedestrians or crowded environments. Advanced semantic or panoptic segmentation models, such as Detectron2, provide a pixel-level analysis of the scene, identifying key objects and their spatial relationships. This enables landmark recognition for obstacle avoidance (e.g., doors, staircases, pathways), contextual understanding of dynamic environments, and enhanced hazard detection through spatial context integration.

Grid based segmentation enhanced by computer vision algorithms, assist users in navigating their environment. The process divides the user's surroundings into a structured, accessible grid, providing a simplified representation of the environment for obstacle avoidance and pathfinding (Figure 2). Objects located towards the cameras left and right field of view are considered low risk as they are not directly within the pathway of the user. Moving inwards objects are considered a medium risk, and lastly objects located within the center of the user are considered high risk. These areas are further segmented into additional top and bottom regions within the system.

This approach enables the system to generate safe and efficient paths for navigation by identifying and marking obstacles within the grid. It also maintains spatial awareness of previously visited locations to avoid redundancy or backtracking and can dynamically adjust routes in response to hazard detection, ensuring adaptability in changing environments.

3.1.3 Scene Understanding

3.1.3.1 Visual Question Answering

Lightweight visual question answering models allow users to interact with the system by asking context-aware questions. For example, users can ask questions such as “What’s in front of me?”, “What direction is the crosswalk?”, “Is the traffic light green or red?”. The models process user queries and respond with concise, actionable information, using Natural Language Processing (NLP) and Text to Speech (TTS).

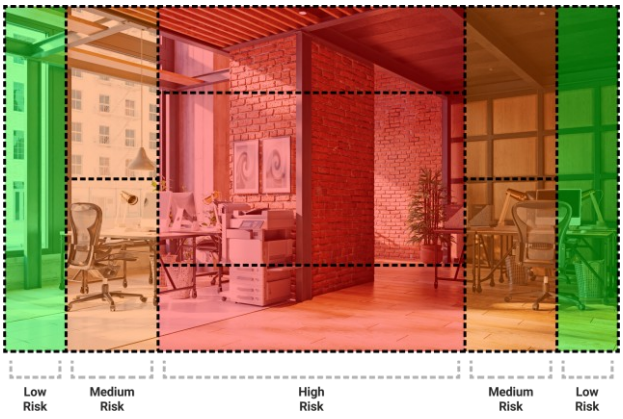


Figure 2: Grid based segmentation

3.1.3.2 Text Recognition

An OCR model works to detect text within the scenario. The user, after the detection of a book, a sign, or any sort of characters, will be able to ask the assistant to read it out loud, using OCR and TTS.

3.1.4 Feedback Delivery Layer

Feedback relayed back from cloud processing layer is presented to the end user. The feedback methods include auditory and haptic feedback and visual overlays. These can be tailored to diverse user needs. Verbal instructions are provided via the HMDs built-in audio system. Some examples include “Obstacle ahead on the left.” or “Chair detected”. Haptic feedback is provided using the Meta Quest 3 controller vibration feature, its proposed users hold these within their pockets. The feedback can convey proximity to obstacles or directional guidance e.g. short pulses indicate nearby hazards, continuous vibrations signal immediate danger.

3.2 Development Environment

The development environment was designed to ensure flexibility, scalability, and compatibility. The key components include Python, C#, PyTorch and OpenCV. Python was used to develop and deploy backend AI models such as YOLO, and semantic segmentation. C# was used with Unity3D for creating the user interface and integrating near real-time feedback on the Meta Quest 3. Unity provided the core framework for developing the user interface and handling feedback delivery. The XR Toolkit streamlined integration with Meta Quest 3’s hardware, including camera streams and interaction mechanisms, enabling feedback and intuitive navigation support. PyTorch enabled efficient training, evaluation, and deployment of machine learning models, including object detection and VQA while OpenCV is used for preprocessing video streams and enhancing input data quality before cloud processing.

3.3 Dataflow Summary

The system's dataflow is designed to ensure efficiency and aims to reduce latency communication between the Meta Quest 3 and server. The process begins with the Meta Quest 3 capturing video streams through its integrated cameras. The data is then securely transmitted to the cloud. In the cloud, the data is distributed across multiple AI processing pipelines. YOLO identifies and classifies hazards, while semantic segmentation models analyze the scene to provide contextual understanding. The grid-based system with state of art algorithms, process spatial data to create dynamic obstacle avoidance maps, and VQA models respond to user queries. These outputs are consolidated into actionable feedback, which include auditory instructions, haptic cues, and visual overlays. The consolidated feedback is transmitted back to the Meta Quest 3 over the network. Unity scripts on the device parse this data to generate user-facing interactions, ensuring that the information is both accessible and actionable. The iterative dataflow design prioritizes speed, accuracy, and user-centric delivery at every stage.

4 Discussion

The proposed system offers several critical advantages that set it apart from traditional assistive technologies. First, its danger detection significantly enhances situational awareness for visually impaired users, enabling them to respond promptly to potential hazards such as moving objects or obstacles. This feature directly contributes to user safety, reducing the likelihood of accidents and fostering confidence during obstacle avoidance. Additionally, the integration of scene understanding and obstacle avoidance aids provides a comprehensive approach to mobility, offering users detailed insights into their surroundings and guiding them through complex environments.

Another key advantage is the system's reliance on cloud-based processing. By offloading computationally intensive tasks to powerful cloud servers, the system ensures scalability and performance while maintaining low latency. This architecture allows for seamless integration of advanced functionalities, such as VQA, which further enriches the user experience by enabling interactive and context-aware responses. The multimodal feedback delivery mechanism, which includes auditory, haptic, and visual cues, ensures inclusivity by catering to a diverse range of impairments and user preferences.

Despite its strengths, the system faces several challenges that warrant attention. One significant limitation is its dependency on stable network connectivity. In low-connectivity environments, users may experience delays or interruptions in data transmission, potentially compromising the system's effectiveness. Addressing this issue may require the exploration of hybrid edge-cloud architectures that allow for limited on-device processing to ensure basic functionality in offline scenarios.

Another challenge is ensuring data privacy and security. Given the sensitive nature of the data being processed, robust encryption and anonymization techniques are essential to protect user information. Furthermore, user adoption may be hindered by the initial cost of the Meta Quest 3 hardware and potential subscription fees for cloud services, highlighting the need for cost-effective solutions to increase accessibility.

5 Conclusion & Future Work

This research presents a transformative cloud-based assistance system for visually impaired individuals, leveraging the capabilities of an off-the-shelf HMD for data acquisition and user interaction. By integrating danger detection, scene understanding, VQA, text recognition, and obstacle avoidance, the system addresses critical challenges in mobility and safety, significantly enhancing user autonomy and confidence.

The system's reliance on cloud-based processing ensures high computational efficiency and scalability while maintaining low latency for feedback. Multimodal feedback mechanisms, including auditory, haptic, and visual cues, cater to diverse user needs, making the solution adaptable and inclusive.

While the system demonstrates significant advancements in accessibility technology, challenges such as network dependency and privacy concerns remain. Future research will focus on hybrid cloud architectures to improve resilience in low-connectivity environments and on enhancing data security measures.

By combining state-of-the-art AI technologies with immersive hardware, this work underscores the potential of integrating AR and AI to improve the quality of life for visually impaired individuals. The research sets a foundation for further exploration and innovation in the field of assistive technologies.

To address these challenges and expand the system's capabilities, several future directions are proposed. First, developing a hybrid architecture that combines edge and cloud processing could mitigate network dependency by enabling basic functionalities, such as local hazard detection, to operate independently of the cloud. Additionally, integrating advanced data compression techniques could further optimize data transmission, reducing latency and bandwidth requirements.

The inclusion of additional sensors, such as ultrasonic or LiDAR, could enhance spatial mapping capabilities. These sensors would complement the visual input from the Meta Quest 3 cameras, providing richer environmental data and improving the system's ability to detect and navigate complex obstacles. This integration could also enable better performance in low-visibility conditions, such as dim lighting or fog.

Finally, expanding the system's capabilities to include additional features, such as weather updates, public transportation assistance, and multilingual support, could further enhance its utility and accessibility. By continuing to innovate and address existing limitations, the system has the potential to transform assistive technologies and significantly improve the quality of life for visually impaired individuals.

REFERENCES

[1] Apple, "Apple Vision Pro - Apple," Apple, [Online]. Available: https://www.apple.com/apple-vision-pro/. [Accessed 15 01 2025].

[2] Microsoft, "HoloLens 2: Find Specs and Features – Microsoft HoloLens 2," Microsoft, [Online]. Available: https://www.microsoft.com/en-ie/d/hololens-2/91pnzzznzwcp?activetab=pivot:overviewtab. [Accessed 15 01 2025].

[3] Meta, "Meta Quest MR, VR headsets and Accessories | Meta Store," Meta, [Online]. Available: https://www.meta.com/ie/quest/. [Accessed 15 01 2025].

[4] H. Durrant-Whyte and T. Bailey, "Simultaneous localization and mapping: part I," *IEEE Robotics & Automation Magazine,* vol. 13, no. 2, pp. 99-110, 2006.

[5] M. Jimenez, I. Becerra and U. Ruiz, "Toward inertial position tracking for head-mounted displays: a dataset and a deep learning approach evaluation.," *Virtual Reality,* vol. 27, pp. 2605-2621, 2023.

[6] C.-L. L. Lu, Z.-Y. Liu, J.-T. Huang, C.-I. Huang, B.-H. Wang, Y. Chen, N.-H. Wu, H.-C. Wang, L. Giarré and P.-Y. Kuo, "Assistive Navigation Using Deep Reinforcement Learning Guiding Robot With UWB/Voice Beacons and Semantic Feedbacks for Blind and Visually Impaired People," *Frontiers in Robotics and AI,* vol. 8, pp. 1-15, 2021.

[7] J. Redmon, S. Divvala, R. Girshick and A. Farhadi, "You Only Look Once: Unified, Real-Time Object Detection," in *IEEE Conference on Computer Vision and Pattern Recognition (CVPR)*, Las Vegas, NV, USA, 2016.

[8] Envision , "Envision Glasses - Smart Glasses for People who are Blind or Low Vision," Envision , [Online]. Available: https://www.letsenvision.com/glasses/home. [Accessed 2025 01 05].

[9] B. Kuriakose, R. Shrestha and F. E. Sandnes, "Tools and technologies for blind and visually impaired navigation support: a review," *IETE Technical Review,* vol. 39, no. 1, pp. 3-18, 2022.

[10] Be My Eyes, "Be My Eyes - See the world together," Be My Eyes, [Online]. Available: https://www.bemyeyes.com/. [Accessed 05 01 2025].

[11] A. Bhowmick and S. M. Hazarika , "An insight into assistive technology for the visually impaired and blind people: state-of-the-art and future trends.," *Journal on Multimodal User Interfaces,* vol. 11, pp. 149-172, 2017.

[12] K. Manjari, M. Verma and G. Singal, "A survey on assistive technology for visually impaired," *Internet of Things,* vol. 11, 2020.

[13] W. Elmannai and K. Elleithy, "Sensor-based assistive devices for visually-impaired people: Current status, challenges, and future directions.," *Sensors,* vol. 17, no. 3, pp. 1 - 42, 2017.

[14] S. Khan, S. Nazir and H. U. Khan, "Analysis of navigation assistants for blind and visually impaired people: A systematic review," *IEEE access,* vol. 9, pp. 26712-26734, 2021.

[15] Gentex, "eSight - Electronic eyewear for the visually impaired," Gentex, [Online]. Available: https://www.esighteyewear.com/. [Accessed 10 01 2025].

[16] X. Yu, Empowering Visually Impaired Individuals With Holistic Assistance Using Real-Time Spatial Awareness System. Doctoral Dissertation, Chicago, Illinois: Graduate College of the Illinois Institute of Technology, 2024.

[17] M. D. Dogan, E. J. Gonzalez, K. Ahuja, R. Du, A. Colaço, J. Lee, M. Gonzalez-Franco and D. Kim, "Augmented Object Intelligence with XR-Objects," in *UIST '24: Proceedings of the 37th Annual ACM Symposium on User Interface Software and Technology*, Pittsburgh, Pennsylvania, USA, 2024.

[18] H. Zhang, N. J. Falletta, J. Xie, R. Yu, S. Lee, S. M. Billah and J. M. Carroll, "Enhancing the Travel Experience for People with Visual Impairments through Multimodal Interaction: NaviGPT, A Real-Time AI-Driven Mobile Navigation System," in *GROUP '25: Companion Proceedings of the 2025 ACM International Conference on Supporting Group Work*, Hilton Head, New Jersey, USA, 2025.

[19] L. R. de Souza, R. Francisco, J. E. da Rosa Tavares and J. L. Victória Barbosa , "Intelligent environments and assistive technologies for assisting visually impaired people: a systematic literature review.," *Universal Access in the Information Society,*, pp. 1 - 28, 2024.

[20] H. Walle, C. De Runz, B. Serres and G. Venturini , "A survey on recent advances in AI and vision-based methods for helping and guiding visually impaired people," *Applied Sciences,* vol. 12, no. 5, pp. 1-16, 2022.

[21] L. Triyono, R. Gernowo, P. S. Rohman Cholil, I. Hestiningsih, W. and S. Fahriah, "Advancing Accessibility: An Artificial Intelligence Framework for Obstacle Detection and Navigation Assistance for the Visually Impaired," in *E3S Web of Conferences*, 2023.

[22] L. Rupasinghe, C. Liyanapathirana, C. H.P.Y.R, H. H.M.T.Y, D. T.O.M and H. T.H.C, "Integrated Assistive System for Precise Indoor Navigation, Object Recognition, and Interaction for Visually Impaired Individuals," *International Research Journal of Innovations in Engineering and Technology,* vol. 7, no. 10, pp. 177-184, 2023.

Real-time Point Cloud Transmission for Immersive Teleoperation of Autonomous Mobile Robots

Nunzio Barone
Politecnico di Bari
Bari, Italy
nunzio.barone@poliba.it

Walter Brescia
Politecnico di Bari
Bari, Italy
walter.brescia@poliba.it

Gabriele Santangelo
Politecnico di Bari
Bari, Italy
g.santangelo@studenti.poliba.it

Antonio Pio Maggio
Politecnico di Bari
Bari, Italy
a.maggio4@studenti.poliba.it

Ivan Cisternino
Politecnico di Bari
Bari, Italy
i.cisternino@studenti.poliba.it

Luca De Cicco
Politecnico di Bari
Bari, Italy
luca.decicco@poliba.it

Saverio Mascolo
Politecnico di Bari
Bari, Italy
saverio.mascolo@poliba.it

ABSTRACT

Autonomous mobile robots (AMRs) are increasingly deployed for remote inspection in various scenarios. While AMRs can perform missions autonomously, human intervention becomes necessary in cases where safety risks or potential damage to the robot arise. Since these platforms are often deployed in remote or potentially dangerous areas, physical access is not always available. In such situations, teleoperation provides an effective solution to resolve issues. The navigation stack of AMRs include obstacle avoidance and mapping functionalities that rely on the environmental feedback, typically acquired with stereo cameras which represent depth and color information using Point Clouds (PCs). In addition to obstacles mapping, PCs provide operators and algorithms with real-time environmental data improving situational awareness and supporting AI-based downstream tasks. However, the large volumes of data inherent to PCs pose challenges for real-time transmission. For this reason, dedicated transmission pipelines and data compression algorithms compatible with the limited computational capabilities of the on-board computer are required. This paper demonstrates such a scenario, equipping a differential drive ground robot with a stereo camera that produces PCs that are streamed to a remote operator wearing a standalone VR Headset. To accomodate the limited computational capabilities of the on-board computer and reduce bandwidth requirements, a distance-based filtering and quantization encoding is applied. Users will step in to assist the robot, with remote real-time teleoperation, when the AMR issues a *"cry for help"* signal during an autonomous mission.

Permission to make digital or hard copies of part or all of this work for personal or classroom use is granted without fee provided that copies are not made or distributed for profit or commercial advantage and that copies bear this notice and the full citation on the first page. Copyrights for third-party components of this work must be honored. For all other uses, contact the owner/author(s).
MMSys '25, March 31-April 4, 2025, Stellenbosch, South Africa
© 2025 Copyright held by the owner/author(s).
ACM ISBN 979-8-4007-1467-2/2025/03
https://doi.org/10.1145/3712676.3719263

CCS CONCEPTS

• **Information systems** → **Multimedia streaming**; • **Human-centered computing** → *Virtual reality*; • **Computer systems organization** → **Robotics**.

KEYWORDS

Teleoperation, Mobile Robots, VR, Point Clouds, WebRTC

ACM Reference Format:
Nunzio Barone, Walter Brescia, Gabriele Santangelo, Antonio Pio Maggio, Ivan Cisternino, Luca De Cicco, and Saverio Mascolo. 2025. Real-time Point Cloud Transmission for Immersive Teleoperation of Autonomous Mobile Robots. In *ACM Multimedia Systems Conference 2025 (MMSys '25), March 31-April 4, 2025, Stellenbosch, South Africa.* ACM, New York, NY, USA, 6 pages. https://doi.org/10.1145/3712676.3719263

1 INTRODUCTION

Autonomous Mobile Robots (AMRs), such as aerial and ground robots, are increasingly deployed in remote or hostile environments to perform several tasks. These robots leverage sensing layers, typically including stereo cameras and LiDARs, to observe and extract environmental information, enabling navigation and situational awareness. Despite their capabilities, the autonomous navigation stacks on board these robots can be faced with difficulties or fail to execute specific operations, often due to inappropriate navigation parameters or the complexity and danger of certain movements. In such scenarios, human intervention is required to mitigate risks, safeguard expensive assets, and ensure mission success. Teleoperation is a practical solution, particularly for robots deployed in inaccessible or dangerous areas. Teleoperation can also be used for managing large fleets of autonomous robots leased to customers globally for industrial applications, enabling remote handling of failures and reducing response times and operational costs.

Teleoperation requires addressing several challenges. First, operators need optimal spatial awareness of the robot's environment to efficiently teleoperate the robot. Head Mounted Displays (HMDs), paired with volumetric video, provide an immersive solution, offering detailed visual representations of the surroundings and distance

information from obstacles. However, volumetric video – typically represented using meshes or Point Clouds (PCs) – poses several challenges due to its high bandwidth requirements and for this reason compression techniques need to be employed [16]. Additionally, the limited computational resources of mobile robots exacerbate the difficulty of providing short glass-to-glass latencies required for efficient teleoperation.

This work demonstrates an immersive platform to allow operators to teleport into remote locations and take control of connected mobile robots equipped with stereo cameras streaming PCs[1]. The operator will use a VR headset, f.i., the Meta Quest 3, to explore the 3D scene received in real-time from the mobile robot through a wireless link. By using dedicated controllers or hand gestures, the operator can both remotely control the mobile robot in different modalities, and navigate and manipulate the 3D environment exploiting different points of view to enhance safety of operations.

To implement the proposed system we make the following contributions. First, we design a capture to rendering pipeline that relies only on the robot's on-board computer and the operator's standalone VR headset. To reduce data latency and meet the real-time requirements we transmit point clouds, telemetry, and commands data using WebRTC. To address the challenge of time-varying bandwidth conditions, we develop a distance-based filtering and quantization encoding methodology that adapts the data stream bitrate dynamically, allowing teleoperation in constrained network environments. On the operator side, we implement a WebXR standalone application for the VR headset. This application includes a graphical user interface (GUI) that allows operators to monitor and interact with the selected robot seamlessly.

The rest of this paper is organized as follows. Section 2 provides background material. Section 3 describes our system in terms of hardware and software components, highlighting some technical difficulties inherited by our robotic and standalone setup. In Section 4 we describe the demo experience presented to the users. Finally, in Section 5, we summarize the key insights provided by the demo, discuss potential future adaptations for improvements, and outline future research directions.

2 BACKGROUND

The sensing stack of AMRs employs stereo cameras and LiDARs to generate point clouds, providing rich environmental data for navigation and situational awareness. This data is employed for autonomous navigation, as it allows robots to map obstacles, plan trajectories, and make decisions in dynamic environments. However, teleoperation scenarios require real-time transmission of this data to a human operator, which poses significant challenges due to the large size of raw point clouds.

In fact, a single point cloud frame at a 480×480 resolution can have a size of up to 29.5 Mbits (see Section 3.3.1), meaning that delivering raw point clouds at a 15 fps framerate would require a bandwidth of approximately 442 Mbps. Such requirements are unrealistic, particularly on mobile networks [2, 15]. To address this, adaptation and compression techniques must be employed to reduce the data size while preserving critical information.

Unlike typical volumetric streaming applications, such as teleconferencing applications [3], robot teleoperation is constrained by the limited computational power of on-board computers due to strict power consumption and mobility requirements. This restriction prevents the use of high-performance computing devices available on high-end workstations, significantly narrowing the range of feasible solutions for adaptation and encoding. Consequently, the algorithms employed must be as lightweight as possible to meet real-time operational constraints.

Additionally, because the robot operates in a real-world environment, the data transmission pipeline must be designed to provide a low-latency communication service. The specific requirements of the communication pipeline depend on the target control application.

In particular, two different scenarios of Human Remote Control (HRC) can be defined [1]:

(1) *Indirect HRC*: The operator provides high level setpoints related to the robot's pose (e.g., position, orientation). This control mode is useful for assisting the robot with global trajectory planning and can tolerate delays of up to 300 ms, as the robot still relies on its navigation stack to carry out missions autonomously [1, 13].
(2) *Direct HRC*: The operator directly controls the robot's actuators (e.g., velocities and accelerations). In this case, the operator acts as a local planner, requiring interaction frequencies that cannot tolerate delays beyond 150 ms [1, 13].

Existing 3D point cloud compression techniques, such as octree representations [6, 10], k-d tree structure [11], geometrical approaches [4], and machine learning approaches [9, 14], have been extensively studied. However, these methods often demand significant computational resources and power, making them unsuitable for the constrained environments of mobile robots.

3 SYSTEM DESCRIPTION

The proposed system is composed of a dedicated hardware and software stack. In this section, descriptions will be provided along with a performance analysis of technologies such as Unity and Draco in our resource-constrained computing units.

3.1 Hardware components

The hardware platform, depicted Figure 1, consists of the robot, the user equipment, a router, and a web server.

The robot is a *Turtlebot3 Waffle Pi*[2], an open-source and open-hardware robot platform designed for education, research, and prototyping. It is widely used in robotics due to its modular design, affordability, and ease of use. The robot's main equipment is composed of a stereo camera and a companion computer. The chosen camera is a *ZED2i* from Stereolabs, an RGB-D camera equipped with internal accelerometer, gyroscope, magnetometer, barometer, and temperature sensor. The companion computer is an *NVIDIA Jetson Xavier NX 16 GB* with a 512 GB SSD storage device which is

[1] In this work, we focus on PCs instead of meshes because such data are inherently used in robotics navigation stacks, making them suitable for integration into teleoperation systems.

[2] https://www.robotis.us/turtlebot-3-waffle-pi/

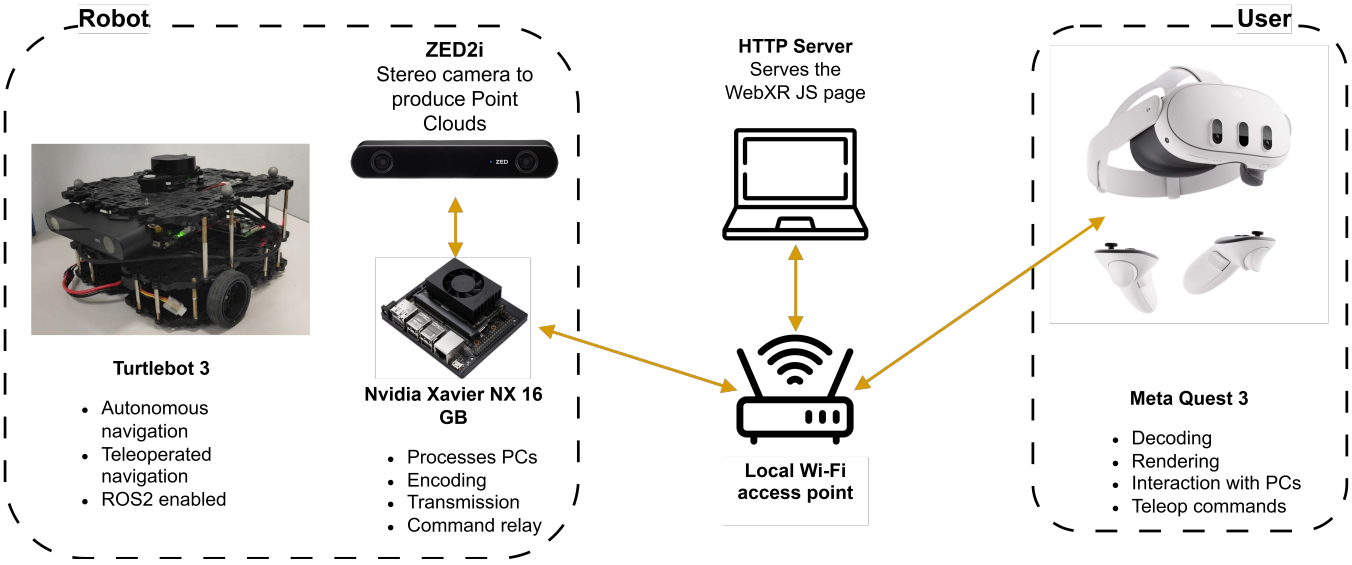


Figure 1: Hardware setup of the proposed system which includes: Robot with all its equipment, Network devices, User equipment consisting in the VR headset, and a server hosting the JS page with the code for the GUI

particularly suitable for mobile platforms due to its compact size and advanced computing performance.

The user equipment mainly consists of the HMD, specifically the *Meta Quest 3*, and the controllers, whose commands are sent to the robot based on the selected teleoperation modality. We deliberately chose this HMD due to its affordability and widespread availability. Moreover, its onboard computation capabilities enable the application to run in standalone mode, unlike other HMDs that required tethering to an external computer.

The web server used in the demo runs on a laptop and serves the WebXR JavaScript page. As a basic web server, it does not perform any computation on the streamed data. Its main function is to provide the JavaScript application to all headsets connected to the network.

3.2 Resource-constrained hardware performance

Having to meet low-power and small-size requirements, the computing capabilities of companion computers on board the robots are significantly limited compared to those of general-purpose laptops or workstations. This reduces performance, limiting the tools and algorithms that can be executed in real-time.

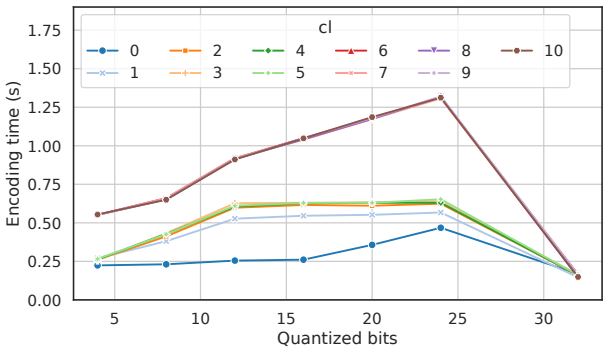
3.2.1 Draco. Among the most popular technologies for point cloud compression, Draco [7] stands out, with the promise of real-time performance for large PCs [5]. In particular, Draco codec has two configurable parameters for PC compression:

- **Quantization parameter (*qp*):** a value ranging from 0 to 30, this parameter sets the number of bits that Draco employs to quantize the positions of the points in the PC. Notice that, according to the Draco documentation when *qp* is set to 0 no quantization is applied, meaning positions are represented using 32 bits. To avoid confusion, instead of reporting the quantization parameter, we show the *quantized bits* (*qb*), which represent the actual number of bits used to encode positions in the compressed PC.
- **Compression level (*cl*):** a value ranging from 0 to 10 that selects different compression features. Typically, a larger value of *cl* should correspond to a lower size of the compressed PC at the expense of a higher encoding latency.

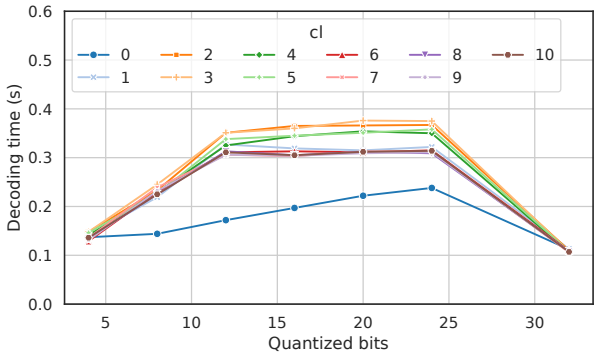
We tested all the possible combinations of these parameters on the reference hardware (*Nvidia Xavier NX*) using a reference point cloud of 810077 points, logging encoding and decoding duration and saved space percentage[3]. Results are presented in Figures 2 and 3. Figure 2(a) shows that the shortest encoding time is 150 ms, achieved when no quantization is applied, i.e., when points are represented using 32 bits ($qb = 32$) regardless of the compression level. Similarly, decoding times, shown in Figure 2, follow the same trend, with a minimum value of approximately ~100 ms, also observed for $qb = 32$. Consequently, the lowest total latency for encoding and decoding is around 250 ms. Adding the point cloud grabbing time, transmission time, and rendering time would result in a total latency exceeding the target maximum for both indirect and direct HRC. Therefore, despite achieving a data size reduction of approximately 68%, Draco is unsuitable for this scenario.

3.2.2 Unity and WebXR. The Meta Quest 3 supports several frameworks for developing immersive interfaces, including Unity [18] and WebXR [8]. Unity is a widely used high-performance game engine, that supports VR and XR applications. On the other hand, WebXR is browser-based framework that employs the `Three.js` library [12]. We experimentally compared Unity and WebXR rendering PCs with increasing sizes. As shown in Figure 4, WebXR with

[3]Notice that, since Draco did not satisfy the maximum latency requirements, we did not measure the distortions in PCs using metrics such as the chamfer distance.



(a) Encoding time



(b) Decoding time

Figure 2: Draco Encoding and Decoding time varying *qb* and *cl* parameters on a Nvidia Xavier NX

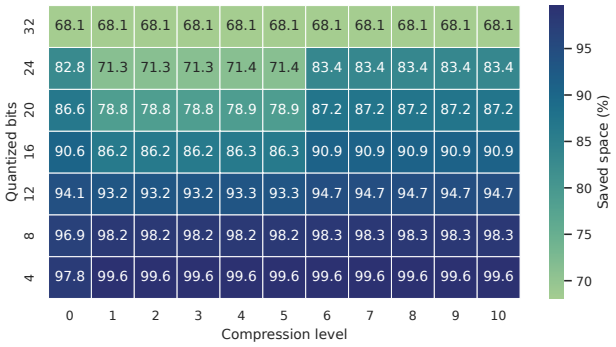


Figure 3: Saved space percentage using Draco codec varying *qb* and *cl* parameters

`Three.js` performs better, maintaining frame rates above 100 fps for point clouds containing up to 2 millions points.

Based on these results, we decided to forgo the use of Draco and Unity in favor of a custom distance-based point cloud filtering approach and the development of the operator GUI using WebXR.

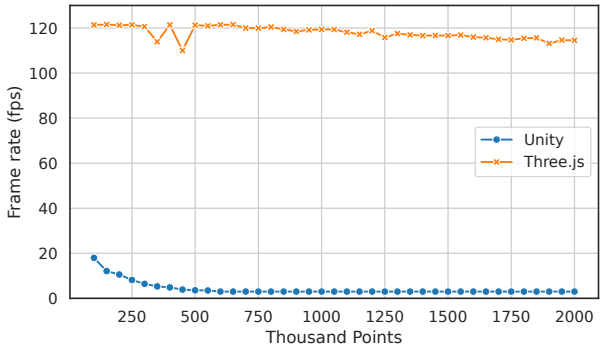


Figure 4: Comparison of rendering frame rate of a test PC with increasing size in Unity and WebXR

3.3 Software stack

The software stack, depicted in Figure 5, was designed and implemented to address the hardware's resource constraints while leveraging its strengths wherever possible.

3.3.1 Robot side. PCs are acquired using the ZED SDK at the desired resolution, which in our case is 480×480. The resulting object is a matrix with dimensions $480 \times 480 \times 4$, containing X, Y, Z coordinates and RGBA in `Float32` for each point, reaching a size of 128 bits per point [17]. Tensors are pre-allocated on GPU memory to further optimize performance. The point cloud matrix is converted to a GPU tensor with a negligible computational overhead exploiting the unified memory of the Nvidia Jetson Xavier NX. The matrix is flattened after outliers are filtered using a mask Quantization: X, Y, Z coordinates converted to `int16` and RBGA to two `uint16` (RGB + alpha). This allows to pass from a size of 128 bits to 80 bits per point. Thus, a compressed point cloud has a size of $480 \times 480 \times 5 \times 16 = 18.4$ Mbits. After this first quantization step, a filter based on distance is applied depending on the available bandwidth. Additional points are uniformly decimated if the distance filter is not enough. The compressed and filtered point cloud is then sent via a WebRTC data-channel to the other peer. Besides the PC transmission pipeline, the robot executes its autonomous routines using ROS2 packages. In particular, the *Nav2 package* is used to carry out the autonomous navigation. This set of tools takes advantage of sensors data to build a representation of the surroundings. This representation, typically a (2D or 3D) map, is used to identify the best trajectories that enable the robot to reach the desired position autonomously while avoiding obstacles. To avoid obstacles, a local cost map is produced. Depending on the selected thresholds the local path planner can make the robot behave either cautiously or recklessly. Then, a dedicated node has the task of bridging ROS2 network with the WebRTC peer, sending telemetry and receiving teleoperation commands.

3.3.2 User side. The received byte stream is decoded into 3D coordinates (positions) and R, G, B color values. The decoded data are used to update the 3D geometry for rendering. The `Three.js` library is employed with `BufferGeometry` for efficient geometry representation. The positions and colors obtained during decoding

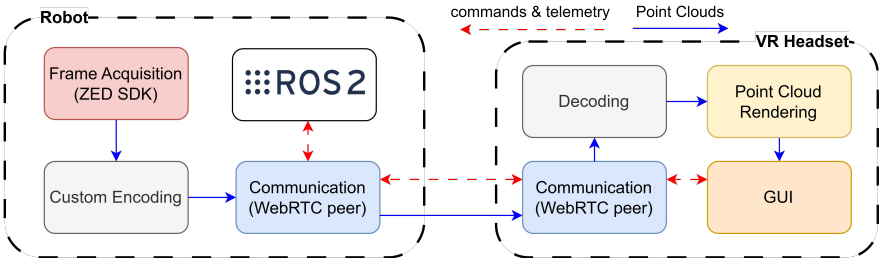


Figure 5: Software stack, representing point clouds (blue arrows) and Telemetry & Control (red dashed arrows) pipelines



Figure 6: Example of navigation scenario. On the left a frame of the volumetric video from robot's PoV. On the right a picture of the navigation environment

are set as geometry attributes. These attributes are then utilized in the WebGL rendering pipeline to render the geometry onto the canvas. A custom `ShaderMaterial` has been implemented in `Three.js`, written in OpenGL Shading Language (GLSL). In addition to PC visualization, the system features a GUI with floating buttons, telemetry banners, and notification flags. This interface enables switching between robots, points of view, and operational modes.

4 DEMO EXECUTION

The proposed demonstration replicates a typical teleoperation scenario, such as the one depicted in Figure 6, involving two robots: the Turtlebot and, for logistic reasons, a stereo camera connected to a laptop acting as a second robot. The user will be able to seamlessly switch between the two robots to experience different situations. These scenarios are designed to showcase the capabilities of the teleoperation system under both ideal and constrained network conditions.

4.1 Unperturbed operational flow

In this scenario, bandwidth is not constrained, allowing the maximum QoE. The following tasks are demonstrated:

(1) **Turtlebot autonomous navigation:** While the robot executes autonomous navigation tasks, the user can view the environment from the robot's PoV or freely explore the mapped area. By manipulating their PoV, users can rotate or zoom into specific parts of the scene using simple hand gestures. The autonomous tasks may be either user-defined (indirect HRC) or autonomously planned by the robot.
(2) **Switching to the other robot:** The system allows the user to monitor different robots. For demonstration purposes, the second "robot" is represented by a stereo camera connected to a laptop. In this case, the user will still be able to move around the mapped scenario, changing the PoV in the same way as the real robot.
(3) **Turtlebot Crying for Help:** During navigation, the Turtlebot may encounter a challenging situation, such as a narrow passage considered too risky to cross based on its navigation stack. When this occurs, the robot will stop and issue a *cry for help*, which is notified to the user via the GUI. This situation is artificially induced by modifying the local cost map, simulating an increase in perceived risk for crossing certain areas. This feature will demonstrate how the teleoperation system can intervene to address limitations in the robot's autonomous capabilities.
(4) **Turtlebot teleoperation:** Upon receiving a "Cry for help" notification, the user will take manual control of the robot, guiding it through the challenging passage. This interaction showcases the system's capability to switch from autonomous navigation to direct teleoperation seamlessly, allowing the robot to complete its mission.

4.2 Perturbed operational flow

In this scenario, different bandwidth constraints are introduced to simulate realistic fluctuations of the available network bandwidth. A dedicated dashboard enable users to adjust system parameters to optimize performance under these conditions. The goal is to demonstrate the system's adaptability and its impact on teleoperation performance and QoE.

Users will be able to adjust the following settings to explore the system's capabilities under constrained conditions:

- **Compression mode selection:** User can choose from the available compression methods and tune their parameters to balance quality and latency.
- **Minimum rendering distance guaranteed:** This parameter ensures that a certain visible range is maintained, regardless of bandwidth limitations, to prioritize critical spatial awareness.
- **Enable/Disable fallback to indirect HRC:** If latency requirements for direct teleoperation are not met, the system can automatically switch to indirect HRC, depending on the user's preference.
- **Enable/Disable maximum velocity adaptation:** If enabled, the robot's maximum velocity is dynamically adjusted

based on the rendering distance to maintain safe and effective operation.

The perturbations that can be applied mainly consist on:

- **Costmap threshold updates**: modifications to the robot's cost map can trigger *Cry for help* messages, requiring user intervention.
- **Bandwidth fluctuations**: The available bandwidth is dynamically varied, impacting the rendering distance and overall latency. The system adapts by modifying the robot's operational mode or maximum velocity as per the user's preferences.

5 CONCLUSIONS

In this paper, we propose a demo of immersive human teleoperation for a wheeled mobile robot, highlighting the limitations imposed by its resource-constrained hardware and exploring alternative solutions. In the demo the users will have the possibility to experience spatial awareness of the area where the robot is located and control it from different PoVs in real-time.

Future research directions include: i) the enhancing the bandwidth adaptation algorithm, by considering semantic information from the scene to allow non-uniform decimation, thus improving visual quality and reducing latency; ii) conducting a QoE assessment campaign to evaluate the impact of visual quality and latency on immersive mobile robot teleoperation.

ACKNOWLEDGMENTS

This work has been funded by the European Union through the DIANE project winner of the Open Call n.1 of the SPIRIT project (101070672).

REFERENCES

[1] 5GAA. 2020. Tele-Operated Driving (ToD): System Requirements, Analysis and Architecture. https://5gaa.org/tele-operated-driving-tod-system-requirements-analysis-and-architecture/ Accessed: 2025-01-18.

[2] Nunzio Barone, Walter Brescia, Saverio Mascolo, and Luca De Cicco. 2024. APEIRON: a Multimodal Drone Dataset Bridging Perception and Network Data in Outdoor Environments. In *Proceedings of the 15th ACM Multimedia Systems Conference* (Bari, Italy) *(MMSys '24)*. Association for Computing Machinery, New York, NY, USA, 401–407. https://doi.org/10.1145/3625468.3652186

[3] Matthias De Fré, Jeroen van der Hooft, Tim Wauters, and Filip De Turck. 2024. Scalable MDC-Based Volumetric Video Delivery for Real-Time One-to-Many WebRTC Conferencing. In *Proceedings of the 15th ACM Multimedia Systems Conference*. Association for Computing Machinery, New York, NY, USA, 121–131.

[4] O. Devillers and P.-M. Gandoin. 2000. Geometric compression for interactive transmission. In *Proceedings Visualization 2000. VIS 2000 (Cat. No.00CH37145)*. IEEE Computer Society, USA, 319–326. https://doi.org/10.1109/VISUAL.2000.885711

[5] M. De Fré, F. De Turck, and J. van der Hooft. 2023. *Low-Latency Volumetric-Video Delivery for Real-Time Conferencing*. Master's thesis. Ghent University. http://lib.ugent.be/catalog/rug01:003150149 Master Thesis Dissertation.

[6] Tim Golla and Reinhard Klein. 2015. Real-time point cloud compression. In *2015 IEEE/RSJ International Conference on Intelligent Robots and Systems (IROS)*. IEEE Computer Society, Hamburg, Germany, 5087–5092. https://doi.org/10.1109/IROS.2015.7354093

[7] Google. n.d.. Draco - A Compression Library for 3D Graphics and Point Clouds. https://google.github.io/draco/

[8] Immersive Web Community Group. n.d.. WebXR Device API. https://github.com/immersive-web/webxr.

[9] Bo Han, Yu Liu, and Feng Qian. 2020. ViVo: visibility-aware mobile volumetric video streaming. In *Proceedings of the 26th Annual International Conference on Mobile Computing and Networking* (London, United Kingdom) *(MobiCom '20)*. Association for Computing Machinery, New York, NY, USA, Article 11, 13 pages. https://doi.org/10.1145/3372224.3380888

[10] Yan Huang, Jingliang Peng, C.-C. Jay Kuo, and M. Gopi. 2008. A Generic Scheme for Progressive Point Cloud Coding. *IEEE Transactions on Visualization and Computer Graphics* 14, 2 (2008), 440–453. https://doi.org/10.1109/TVCG.2007.70441

[11] Erik Hubo, Tom Mertens, Tom Haber, and Philippe Bekaert. 2006. The Quantized kd-Tree: Efficient Ray Tracing of Compressed Point Clouds. In *2006 IEEE Symposium on Interactive Ray Tracing*. IEEE Computer Society, Salt Lake City, UT, USA, 105–113. https://doi.org/10.1109/RT.2006.280221

[12] Ricardo Cabello (mrdoob) and Contributors. n.d.. Three.js - JavaScript 3D Library. https://github.com/mrdoob/three.js.

[13] Oren Musicant, Assaf Botzer, and Shraga Shoval. 2023. Effects of simulated time delay on teleoperators' performance in inter-urban conditions. *Transportation Research Part F: Traffic Psychology and Behaviour* 92 (2023), 220–237. https://doi.org/10.1016/j.trf.2022.11.007

[14] Dat Thanh Nguyen, Maurice Quach, Giuseppe Valenzise, and Pierre Duhamel. 2021. Learning-Based Lossless Compression of 3D Point Cloud Geometry. In *ICASSP 2021 - 2021 IEEE International Conference on Acoustics, Speech and Signal Processing (ICASSP)*. IEEE Computer Society, Toronto, ON, Canada, 4220–4224. https://doi.org/10.1109/ICASSP39728.2021.9414763

[15] Darijo Raca, Dylan Leahy, Cormac J. Sreenan, and Jason J. Quinlan. 2020. Beyond throughput, the next generation: a 5G dataset with channel and context metrics. In *Proceedings of the 11th ACM Multimedia Systems Conference* (Istanbul, Turkey) *(MMSys '20)*. Association for Computing Machinery, New York, NY, USA, 303–308. https://doi.org/10.1145/3339825.3394938

[16] Sebastian Schwarz, Marius Preda, Vittorio Baroncini, Madhukar Budagavi, Pablo Cesar, Philip A Chou, Robert A Cohen, Maja Krivokuća, Sébastien Lasserre, Zhu Li, et al. 2018. Emerging MPEG standards for point cloud compression. *IEEE Journal on Emerging and Selected Topics in Circuits and Systems* 9, 1 (2018), 133–148.

[17] Stereolabs. n.d.. Stereolab ZED SDK - Using the Depth Sensing API. https://www.stereolabs.com/docs/depth-sensing/using-depth#getting-point-cloud-data

[18] Unity Technologies. n.d.. Unity - Game Engine and Development Platform. https://unity.com/.

Streaming Face-Off: A Testbed Analysis of Media-over-QUIC and Low-Latency DASH

Minh Nguyen
minh.nguyen@fokus.fraunhofer.de
Fraunhofer FOKUS Institute
Berlin, Germany

Philip Nys
philip.nys@fokus.fraunhofer.de
Fraunhofer FOKUS Institute
Berlin, Germany

Stefan Pham
stefan.pham@fokus.fraunhofer.de
Fraunhofer FOKUS Institute
Berlin, Germany

Daniel Silhavy
daniel.silhavy@fokus.fraunhofer.de
Fraunhofer FOKUS Institute
Berlin, Germany

Stefan Arbanowski
stefan.arbanowski@fokus.fraunhofer.de
Fraunhofer FOKUS Institute
Berlin, Germany

Stephan Steglich
stephan.steglich@fokus.fraunhofer.de
Fraunhofer FOKUS Institute
Berlin, Germany

Abstract

Recent years have observed a drastic improvement in video streaming. However, there are still non-trivial open issues that need to be solved, including latency and throughput estimation. A common approach is Low-Latency Dynamic Adaptive Streaming over HTTP (LL-DASH) that utilises Common Media Application Format (CMAF) to reduce the latency. Currently, a new solution is being developed by the Internet Engineering Task Force (IETF) that leverages QUIC protocol, called Meida-over-QUIC (MoQ) Transport. MoQ Transport is supposed to achieve even lower latency for large-scale systems. In this paper, we present a testbed to compare the performance of the up-to-date version of MoQ Transport and LL-DASH under various network conditions.

CCS Concepts

• **Networks** → **Application layer protocols**; • **Information systems** → **Multimedia streaming**.

Keywords

QUIC, Media-over-QUIC, HTTP Adaptive Streaming, dash.js, DASH

ACM Reference Format:
Minh Nguyen, Philip Nys, Stefan Pham, Daniel Silhavy, Stefan Arbanowski, and Stephan Steglich. 2025. Streaming Face-Off: A Testbed Analysis of Media-over-QUIC and Low-Latency DASH. In *ACM Multimedia Systems Conference 2025 (MMSys '25), March 31–April 4, 2025, Stellenbosch, South Africa.* ACM, New York, NY, USA, 5 pages. https://doi.org/10.1145/3712676.3719264

1 Introduction

Video streaming services are now prevalent in today's digital landscape and are compatible with a range of devices that have different display features. According to the Sandvine Phenomena report [12], video streaming – encompassing both on-demand and live content—represents 68% of overall downstream traffic.

Permission to make digital or hard copies of all or part of this work for personal or classroom use is granted without fee provided that copies are not made or distributed for profit or commercial advantage and that copies bear this notice and the full citation on the first page. Copyrights for third-party components of this work must be honored. For all other uses, contact the owner/author(s).
MMSys '25, March 31–April 4, 2025, Stellenbosch, South Africa
© 2025 Copyright held by the owner/author(s).
ACM ISBN 979-8-4007-1467-2/25/03
https://doi.org/10.1145/3712676.3719264

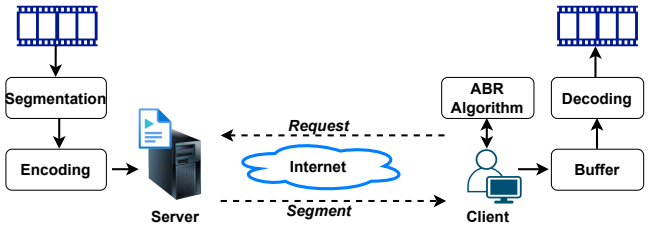


Figure 1: The principle architecture of a HAS system.

HTTP Adaptive Streaming (HAS) is a de-facto technology for video streaming. In HAS, the original video is divided into segments of equal duration. Each segment is encoded into multiple quality versions, which vary by bitrate and resolution, and stored on the server. All quality version information (e.g., bitrate, resolution, and segment duration) is included in a manifest file known as the Media Presentation Description (MPD) in Dynamic Adaptive Streaming over HTTP (DASH) [1]. At the start of each streaming session, the client retrieves the manifest file to extract the quality version details. It then requests an appropriate version for each segment by sending an HTTP GET request, which is saved to the buffer. The chosen quality version is typically determined by an Adaptive Bitrate (ABR) algorithm that considers the buffer status and/or throughput estimation. Figure 1 depicts the main architecture of a HAS system.

Dynamic Adaptive Streaming over HTTP (DASH) is an international standard defined by ISO/IEC (ISO/IEC 23009) [1], that follows the HAS technique. Low-latency DASH (LL-DASH) is an extension of the DASH protocol that is specifically designed to minimize the latency of video streaming. This is particularly important for applications such as live streaming, online gaming, and real-time communication, where delays can significantly affect the user experience.

However, LL-DASH is running on top of TCP, which has significant issues that prevent it from effectively meeting the current demands of media streaming and its growing requirements [9, 11].

The emerging transport protocol QUIC [7] provides solutions to the problems associated with TCP and, in theory, has the potential to exceed TCP's advantages. As previously noted, TCP faces challenges that prevent it from delivering the optimal viewing experience for consumers. It employs an outdated handshake method

for connection establishment, resulting in slower performance than necessary. Furthermore, TCP does not inherently support encryption, which not only raises security concerns but also adds an extra layer to the handshake process when encryption is implemented, further delaying it. Additionally, TCP is affected by Head-of-Line (HoL) blocking, a situation during congestion control where a delayed data packet causes subsequent packets to also be held up. The QUIC protocol effectively addresses many of these issues by streamlining the handshake process, incorporating security from the outset, and implementing significantly improved congestion control algorithms that completely eliminate HoL blocking.

Currently, an active working group in the Internet Engineering Task Force (IETF) called Media over QUIC (MoQ) is proposing a new standard, MoQ Transport, to leverage QUIC protocol for video streaming. While LL-DASH runs on top of TCP with the use case as one-way, low-latency live streaming [5], MoQ Transport built on top of QUIC and UDP, can cover interactive use cases such as online gambling and cloud gaming.

This paper will provide a testbed to analyse the performance of MoQ and compare it with LL-DASH in the live streaming scenario. Our testbed also reports the latency, current buffer level, measured bandwidth and downloaded bitrate.

The remainder of this paper is structured as follows. Section 2 provides an overview of the background about MoQ , followed by our system implementation in Section 3.Section 4 concludes the paper and draws some future directions.

2 Background

The development of QUIC began at Google in 2012 with the SPDY protocol, which was a TCP-based transport protocol designed to address TCP's weaknesses and already showed some similarities to today's QUIC. SPDY integrated TLS for fast connection establishment, improved the multiplexing of multiple transmissions, allowed prioritization of specific requests to prevent head-of-line (HoL) blocking, and enabled the server to initiate transmissions without the client needing to request them. Ultimately, SPDY's features were integrated into HTTP/2 in 2015. Shortly after the release of HTTP/2, Google started developing QUIC and submitted the first draft for standardization in 2016, with the final release occurring in 2021. This led to the release of the next version of HTTP, known as HTTP/3. QUIC is a transport protocol based on the User Datagram Protocol (UDP), but it significantly expands upon it. Unlike TCP, which is a reliable protocol that guarantees data delivery without errors and in order, UDP is inherently unreliable and does not establish a persistent connection between the client and the server; it simply sends data with the expectation that it will arrive. Since there is no additional handshake or encryption step required, QUIC offers excellent latency potential. However, the lack of a reliable connection can result in data arriving incorrectly, in the wrong order, multiple times, or not at all.

This is where QUIC enhances UDP by first incorporating a QUIC handshake that negotiates not only the transport parameters but also the cryptographic parameters. As a result, QUIC reduces the initial round-trip time (RTT) from 2 (TCP + TLS) to just one RTT. Moreover, there is an option for the client to send data immediately in a 0-RTT manner if it has been previously configured. An abstract model illustrating how data, such as a video stream, may be transmitted is shown in Fig. 2.

A single QUIC connection can communicate with multiple clients simultaneously over the same UDP port.

QUIC provides two methods for transmitting data between the client and the server, and it also enables the server to send data to the client without needing specific requests.

The first method is datagrams which are essentially normal UDP payloads with the added benefit of being encrypted and congestion-controlled. Similar to UDP, these datagrams are still unreliable [2].

The other method is the so-called QUIC streams. Streams are abstract transport tunnels, allowing the server and the client to send data between each other reliably. They can be initiated by both client and server and come in two varieties:

- Unidirectional: only the initiator is able to send data and, the recipient can only receive data.
- Bidirectional: both participants can send data as well as receive data on the same stream.

Sending data on these streams is reliable, meaning the data will arrive correctly and in order. This is achieved by concurrently opening streams in the background, which will monitor the correctness of the payloads, and in case of errors or packet loss, they will retransmit the affected packets [2].

Media over QUIC (MoQ) is a new streaming protocol currently in the works by the IETF. As its name suggests, MoQ uses QUIC, and in extension WebTransport, as transport protocol, rather than HTTP (TCP) as current state-of-the-art streaming protocols, namely DASH and HLS. MoQ is based on a Publisher/Subscriber model, where Publishers are responsible for ingesting data, structured as Tracks, Groups, and Objects, to the network. The network consists of any number of Relays, whose tasks are the distribution of the published data. Subscribers are clients that subscribe to Tracks to signal the network they want to get that data forwarded to them.

MoQ is highly configurable in terms of the target latency and it can transport any kind of data. MoQ Tracks are the entities in MoQ, that can be subscribed to, i.e. one quality representation of media, chat messages of a video conference, keystrokes for cloud gaming, etc. Each Track is a sequential stream of MoQ Groups, i.e. a Group of Pictures (GoP) of media, chat messages, or keystrokes within a certain time frame, etc. Each Group again is an ordered sequence of MoQ Objects, the smallest addressable entity in MoQ, they contain the actual data payloads, i.e. video frames, individual messages or keystrokes, etc.

3 System Implementation

To test the capabilities of MoQ and to compare it to dash.js, we implemented a proof-of-concept testbed, which is shown in Fig.4. In our experiments, we evaluate two kinds of servers: LL-DASH over HTTP/1.1 and MoQ. We deployed a Rust Axum server [8] as an HTTP/1.1 server, and Rust moq-rs and moq-js for all MoQ interfaces. To shape the network environment, we deployed a QUIC and HTTP proxy running in a Docker Container[1] routing all traffic through the proxy and limiting the outgoing traffic using traffic control (`tc`)[6]. The tc program has the ability to set so-called `qdics`,

[1] https://www.docker.com/resources/what-container/. Accessed 15 January 2025

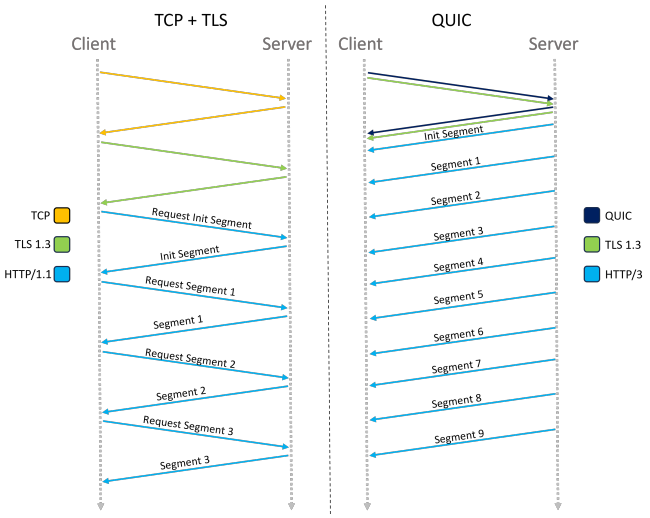


Figure 2: Sending Media Segments using TCP + TLS and QUIC [10]

which can be applied to individual network interfaces to throttle the throughput and introduce latency.

We deployed two clients on a webpage: a dash.js player (for LL-DASH) – the most common HAS player and a MoQ client using moq-js to receive the media streaming and then play it back in a Media Source Extensions (MSE) player to give a fair comparison to dash.js, which also uses MSE for playback, in terms of low delay. The player dash.js is used when evaluating the throughput estimation techniques.

The experiments were conducted on a Lenovo ThinkPad P1 Gen 7 on Ubuntu 24.04.1 LTS with Intel Core Ultra 9 185H CPU, 32GB RAM and NVIDIA RTX4070 GPU. We used Google Chrome v131[2] to operate players at the client side.

The screenshot of our testbed is illustrated in Fig. 3

3.1 MoQ Setup

We used moq-rs as baseline implementation and extended it for our experiments, specifically an older branch based on IEFT draft04 (*i.e.*, draft-ietf-moq-transport-04[3]). We created a new Publisher that is able to publish a DASH-compliant media stream (CMAF). To simulate a live stream, we are using FFmpeg[4] to endlessly loop through a Big Buck Bunny[4] video file. Since FFmpeg does not support outputting to MoQ, we build the Publisher as an HTTP server, using Rust Rocket [3], to receive the output of FFmpeg. The Publisher will, similar to the original moq-rs publisher, publish the segments repackaged to the MoQ relay and also save the raw files to the local disk. The repackaging maps each segment into one Group with each Object being the individual frames. We have configured the FFmpeg command to make each segment exactly one GoP.

The MoQ Relay has been left untouched.

[2]https://support.google.com/chrome/answer/95346. Accessed 18 January 2025.
[3]https://datatracker.ietf.org/doc/draft-ietf-moq-transport/04/. Accessed 24 January 2025
[4]https://peach.blender.org/. Accessed: 6 January 2025

3.2 HTTP Setup

The MoQ Relay is running a simple Axum HTTP server, originally just to host the TLS certificate fingerprint, which is currently required to use self-signed certificates in Google Chrome. We have added the functionality of serving static files, specifically the DASH segments generated by FFmpeg and saved to disk by the MoQ Publisher.

3.3 MoQ Client

The client is an HTML webpage, which on loading starts playing the same live streaming using both MoQ and dash.js concurrently. The MoQ implementation conforms to the IEFT draft04. While streams are playing we are measuring certain metrics: estimated throughput, bitrate being played, and latency. Dash.js reports these metrics out of the box using the appropriate event callbacks.

3.3.1 Throughput Estimation. We are estimating the throughput (ETP) of MoQ by detecting idling gaps in the data transmissions to predict the bandwidth [5]. Once the first new chunk of data of a new Group stream has been received, we assume that the remaining chunk of the Group will follow directly. After the first chunk we can now take the starting timestamp t_1, once the next chunk has been received, we can take the next timestamp t_2 and also download size as the number of bytes of the newly received chunk B. With these values, we can estimate the throughput as follows.

$$ETP_i = \frac{B * 8}{t_2 - t_1} \tag{1}$$

This will give us the throughput of a single chunk in kilobits per second. This will be repeated for all chunks in a Group. After each chunk timestamp t_2 will become t_1 for the next chunk. Once the full Group has been received and we have collected all N ETP values we average all values to get an arithmetic mean of the throughput of that Group using

$$ETP = \frac{\sum_{i=1}^{N} ETP_i}{N} \tag{2}$$

3.3.2 Latency. For both players, we are determining the latency as the sum of the current buffer level bl and the network delay, being the difference between the time of arrival at the client t and the NTP timestamp found in the `Producer Reference Time` (prft) box embedded in the media chunks

$$latency = bl + t - prft \tag{3}$$

giving us the latency in milliseconds.

3.3.3 Playback Bitrate. The MoQ player is using a basic ABR algorithm, which simply chooses the highest available quality Track that is below 90% of the ETP calculated from the previous Group. The player switches Tracks by unsubscribing from the current Track and subscribing to the new Track.

3.4 Traffic Control

All traffic through the proxy Docker container is limited by `tc` using the following commands

(1) set: `tc qdisc add dev <netName> root tbf rate <serverBandwidthStr latency <latencyString> burst <tokenBucketSize>`
(2) delete: `tc qdisc delete dev <netName> root`

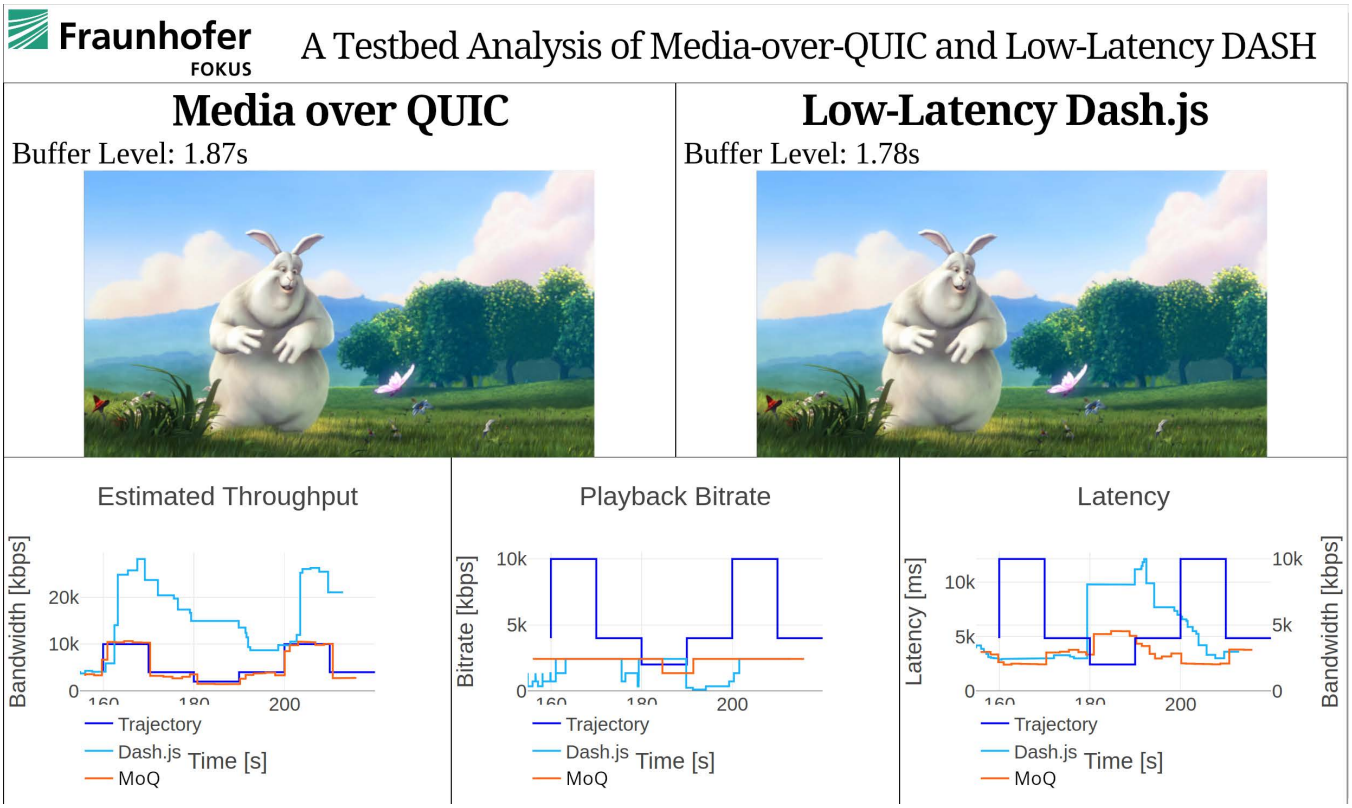


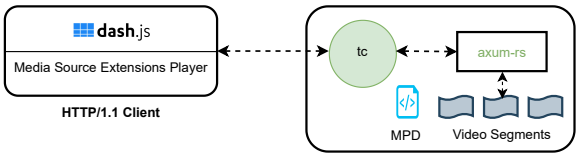
Figure 3: Screenshot from our testbed

Table 1: Bitrate ladder used in our experiments

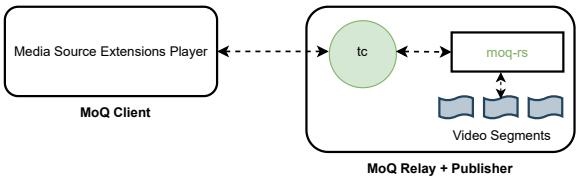
ID	Bitrate (kbps)	Resolution (p)
1	107	144
2	240	240
3	346	360
4	715	480
5	1347	720
6	2426	1080

with

(1) `netName`: the name of the network interface, excluding loopback (lo) (on Ubuntu they are the directories found at `/sys/class/net/`)
(2) `serverBandwidthString`: the string representing the desired limit, e.g. `5000kbit` (*i.e.*, 5,000 kilobits per second), `1000bps` (*i.e.*, 1,000 Bytes per second), etc. [6]
(3) `latencyString`: the string representing the simulated latency, across all test `50ms` has been used
(4) `tokenBucketSize`: buffer size, set to the same as bandwidth to prevent caching bottlenecks



(a) The implemented LL-DASH system for video streaming.



(b) The implemented MoQ for video streaming.

Figure 4: The overview of the architecture of our demo.

4 Conclusion and Future Work

In this paper, we introduce a testbed to analyse the performance of MoQ and compare it with LL-DASH. MoQ is still being developed both the standardization and implementation. The current implementation already shows some potential benefits in video

streaming. It is expected that future versions will provide better performance.

Our future work will focus on the cloud gaming scenario where the latency is more sensitive and the client also needs to send some data to the server. Cloud gaming is used to investigate the performance of MoQ with bi-directional data delivery.

References

[1] Dash.js iso/iec 23009-1:2012 release. [Online] Available: https://www.iso.org/standard/57623.html. Accessed: 10 February 2024.

[2] User Datagram Protocol. RFC 768, August 1980. Accessed: 10 February 2024.

[3] Sergio Benitez. crates.io - rocket. [Online] Available: https://crates.io/crates/rocket. Accessed: 24 Janurary 2025.

[4] FFmpeg. FFmpeg. [Online] Available: https://ffmpeg.org/. Accessed: 15 January, 2024.

[5] Zafer Gurel, Tugce Erkilic Civelek, Deniz Ugur, Yigit K Erinc, and Ali C Begen. Media-over-QUIC Transport vs. Low-Latency DASH: a Deathmatch Testbed. In *Proceedings of the 15th ACM Multimedia Systems Conference*, pages 448–452, 2024.

[6] Bert Hubert. tc(8) Linux Manual Page. [Online] Available: https://man7.org/linux/man-pages/man8/tc.8.html. Accessed: 19 January 2024.

[7] Jana Iyengar and Martin Thomson. QUIC: A UDP-Based Multiplexed and Secure Transport. RFC 9000, May 2021. Accessed: 10 February 2024.

[8] Carl Lerche and David Pedersen. crates.io - axum. [Online] Available: https://crates.io/crates/axum. Accessed: 24 Janurary 2025.

[9] Minh Nguyen, Hadi Amirpour, Christian Timmerer, and Hermann Hellwagner. Scalable High Efficiency Video Coding Based HTTP Adaptive Streaming over QUIC. In *Proceedings of the Workshop on the Evolution, Performance, and Interoperability of QUIC*, pages 28–34, 2020.

[10] Minh Nguyen, Philip Nys, Stefan Pham, Daniel Silhavy, Stefan Arbanowski, and Stephan Steglich. Toward webtransport support in http adaptive streaming. In *2024 Tenth International Conference on Communications and Electronics (ICCE)*, pages 96–101. IEEE, 2024.

[11] Minh Nguyen, Christian Timmerer, Stefan Pham, Daniel Silhavy, and Ali C Begen. Take the Red Pill for H3 and See How Deep the Rabbit Hole Goes. In *Proceedings of the 1st Mile-High Video Conference*, pages 7–12, 2022.

[12] Sandvine. Global Internet Phenomena Report 2024. [Online] Available: https://www.sandvine.com/phenomena. Accessed: 02 January 2025.

No code XR-CAx experience creation workflows for conducting Equipment Design Reviews

Sahir Sharma
Technological University of the Shannon: Midlands Midwest
Athlone, Ireland
a00304794@student.tus.ie

Conor Keighrey
Technological University of the Shannon: Midlands Midwest
Athlone, Ireland
c.keighrey@research.ait.ie

James Lardner
Boston Scientific Ireland
Galway, Ireland
james.lardner@bsci.com

Shane Gilligan
Boston Scientific Ireland
Galway, Ireland
shane.gilligan@bsci.com

Niall Murray
Technological University of the Shannon: Midlands Midwest
Athlone, Ireland
nmurray@research.ait.ie

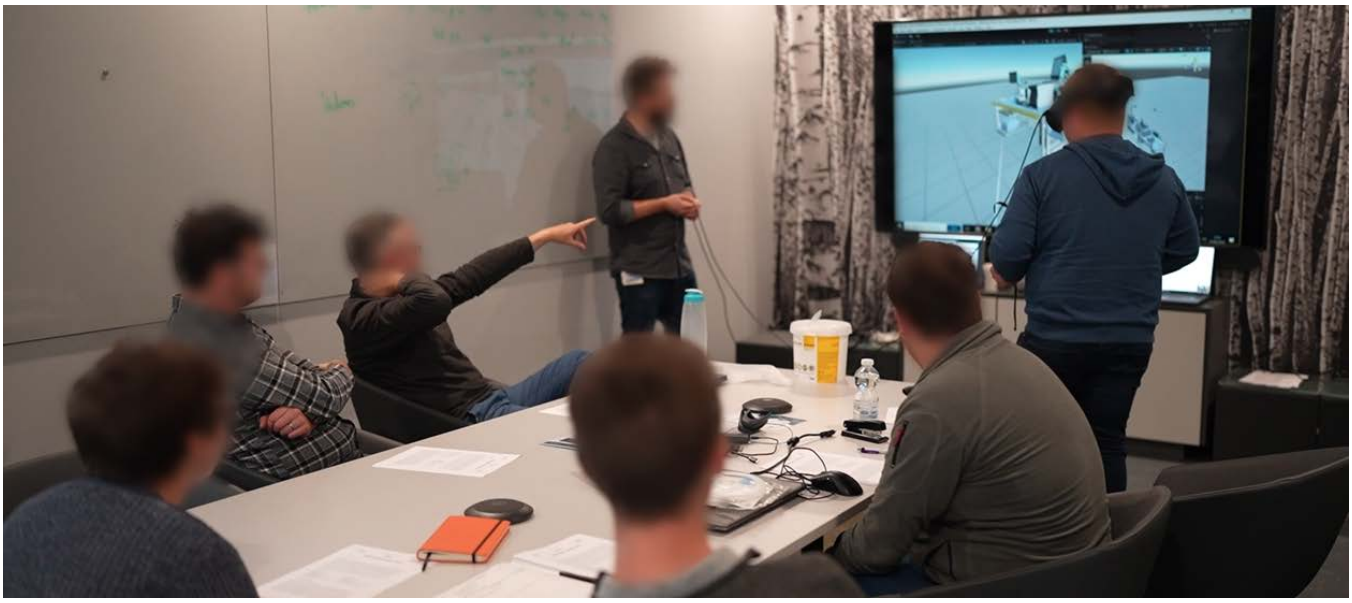


Figure 1: Various industry members participating in a VR-based Equipment Design Review experiment

Abstract

This paper presents a no-code eXtended Reality (XR) workflow that allows non technical people to create XR experiences as part of equipment design review (EDR) processes. The demo showcases how the workflow can enable seamless visualization, interaction, and evaluation of complex CAD models in XR. By combining PiXYZ for CAD-to-mesh translation with Unity's XR Interaction Toolkit and Meta XR SDK, the system delivers immersive experiences that enhance spatial perception and reduce cognitive load for mechanical engineers. The demo also incorporates a stylus based approach for adding hand-drawn notes in real-time on top of imported CAD models. Attendees will witness the complete XR creation pipeline, from CAD import to a live Mixed Reality based design review, demonstrating efficiency, scalability, and potential to revolutionize XR-driven engineering workflows.

Permission to make digital or hard copies of all or part of this work for personal or classroom use is granted without fee provided that copies are not made or distributed for profit or commercial advantage and that copies bear this notice and the full citation on the first page. Copyrights for third-party components of this work must be honored. For all other uses, contact the owner/author(s).
MMSys '25, Stellenbosch, South Africa
© 2025 Copyright held by the owner/author(s).
ACM ISBN 979-8-4007-1467-2/2025/03
https://doi.org/10.1145/3712676.3719265

CCS Concepts

• **Applied computing → Enterprise application integration**; • **Hardware → Emerging simulation**; • **Computing methodologies → Mixed / augmented reality**.

Keywords

Mixed Reality, Equipment, Design aspects, Reviews, Experience Creation, workflow, XR Stylus, Enterprise Adoption

ACM Reference Format:
Sahir Sharma, Conor Keighrey, James Lardner, Shane Gilligan, and Niall Murray. 2025. No code XR-CAx experience creation workflows for conducting Equipment Design Reviews. In *ACM Multimedia Systems Conference 2025 (MMSys '25), March 31-April 4, 2025, Stellenbosch, South Africa.* ACM, Stellenbosch, South Africa, 6 pages. https://doi.org/10.1145/3712676.3719265

1 Introduction

As part of a design review process, working with 2D CAD can hinder spatial reasoning, as it requires users to mentally reconstruct 3D models from flat images eventually increasing cognitive load [13, 14]. The reliance on such representations can inhibit intuitive spatial understanding, which is critical for creative design and problem-solving. Furthermore, 2D CAD's single-plane view often leads to misinterpretations of spatial relationships, complicating the design and review processes in AEC fields [14]. Traditional CAD tools can stifle creativity and innovation, as designers may find themselves constrained by the limitations of the software systems rather than being empowered to explore new design possibilities [30]. Such systems are rarely optimized for real-time and multi-user collaborations. The result is complicating data sharing and communication among geographically dispersed teams [14].

Developing in-house solutions for enabling interoperability among 3D Data can be resource-intensive and often impractical [2]. Standards [1] have been proposed to unify 3D data formats but there has been an emergence of complex and proprietary files [21] that pose a significant learning curve. The specific process of "Translating CAD data to VR-ready" has been identified by many previous researchers [5, 11] as the single most important step in the visualization of the models. Thus, utilizing third-party software for importing CAD models to XR systems has been observed as a good practice recently in the industry.

Virtual and Mixed Reality are the two industry ready technologies within XR that can be applied to different manufacturing contexts [29]. In this work, Unity Engine [35] was chosen as the choice of software development ecosystem. Two no code templates and respective workflows were created: one each for VR and MR.

2 System Design and Methodology

A common philosophy mentioned in Section 2.1 is followed across both the templates mentioned in Section 2.2 by specific individuals defined in Section 2.3.

2.1 Containerizing CAD models within XR

This research was carried out in collaboration with Boston Scientific Ireland (BSCI). After a significant requirement and co-creation process between the researchers and various Equipment Engineering stakeholders at BSCI, a significant impediment in the adoption of XR within Manufacturing was identified as the "dependence on programming expertise to develop XR apps". The desire to eliminate (or significantly reduce) the need for programming to create XR experiences was identified as a key research challenge. Addressing this challenge also has the advantage of empowering mechanical engineers to create their own context aware immersive experiences. The aim of the work is to position XR as a workforce transformation tool with a minimum barrier of adoption. The PiXYZ Plugin [28] is

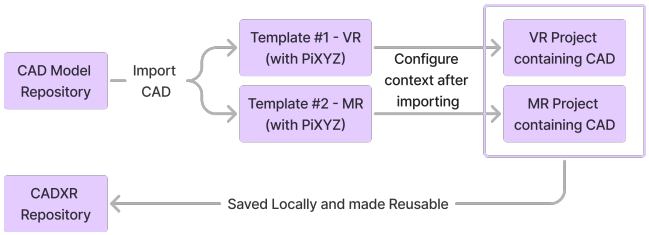


Figure 2: Containerized XR experience creation approach

Unity's own offering (only a part of the Industry license) that takes care of the import process which is key to containerization of CAD files. After containerization, an experience creator can configure position and interaction of the imported equipment CAD model(s).

2.2 Architecture and Operations

Fig 3 presents the two sandbox templates along with their defined workflows (in separate ReadMe files). These were circulated among BSCI mechanical engineers in early 2024 in an attempt to initiate adoption of such technologies in the company:

- **Template #1 for creating a VR-EDR Experience:** A Unity Industry License based project was created with various dependencies like Unity XR Interaction Toolkit [36] and OpenXR Plugin [34], along with adequate configuration of presets, a transparent material and some relevant GameObjects. Post-import, CAD models can be repositioned and simulated in a Virtual Environment for a user just by playing it off the PC. The simulation, termed as a "VR EDR experience" has shown to be a popular choice of modality among EDR stakeholders.
- **Template #2 for creating a MR-EDR Experience:** A separate Unity Industry License based project was created with pre-installed Meta XR All-In-One SDK [27] and Oculus XR Plugin [33]. After importing and repositioning relevant CAD models in physical space according to a set anchor, an in-situ review can be simulated. Such reviews (called "MR-EDR experiences") are often beneficial to showcase equipment upgrades on already existing physical designs.

Each of the template projects (#1 and #2) were compressed and circulated as reusable sandbox templates that could be forked, each time a new XR design review experience was required to be worked upon. All developed VR and MR experiences (instances of design review) can be saved locally on the system, for repeated playback.

2.3 Roles of Stakeholders using XR

Every XR-EDR instance has the following stakeholders involved:

- XR Experience BUILDERS or XR Experience CREATORS: Trained stakeholders in the Equipment Engineering Group that aim to use the XR templates as sandboxes to create immersive experiences from pre-existing CAD models.
- XR Experience USER or PRIMARY VISUALIZER: After creation by XR experience builders, various other users can participate in XR EDR activities as a user of the technology. They get to wear the HMD during the EDR. Example: Subject

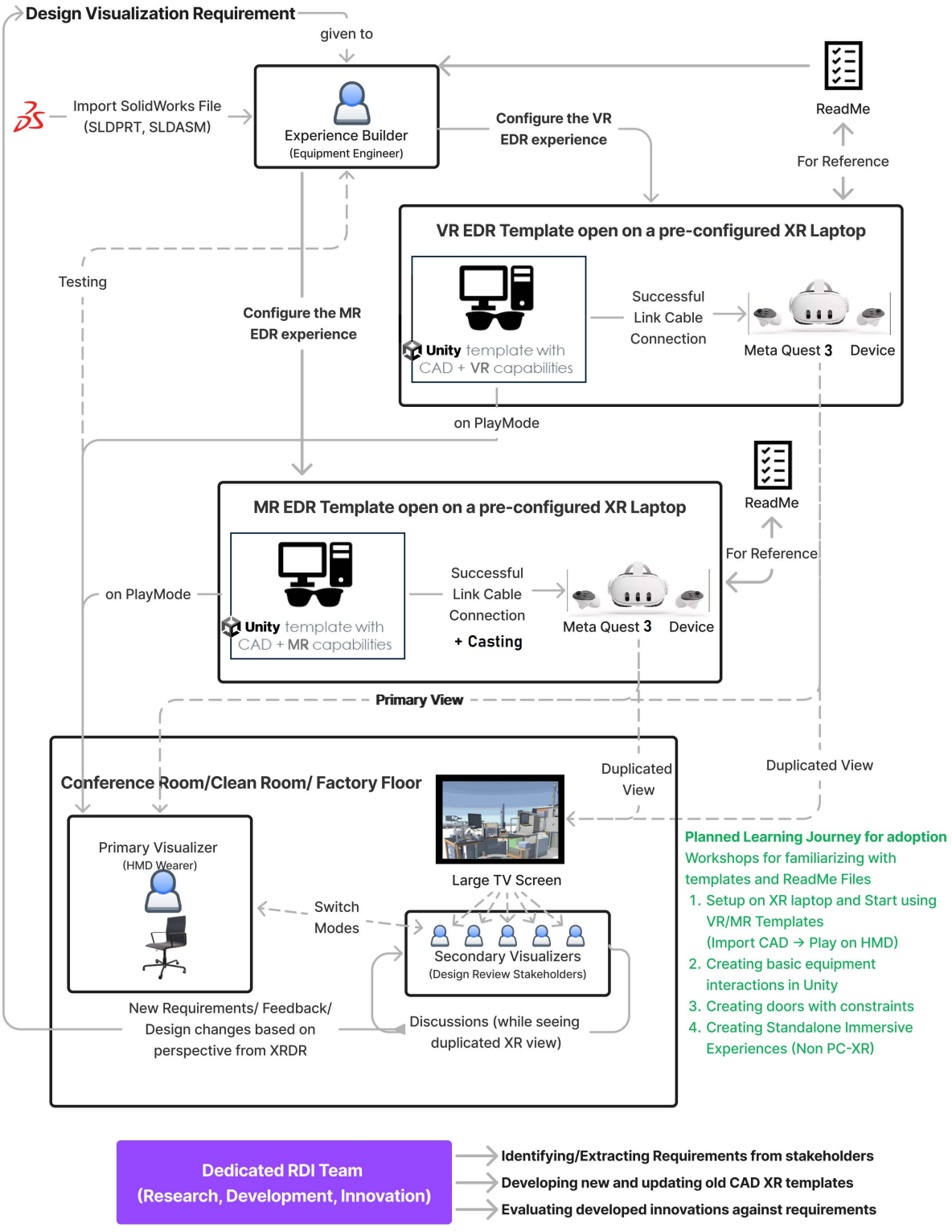


Figure 3: High Level operations for visualizing CAD through XR Templates

Component	Quantity	Cost (EUR)
Unity Industry (with PiXYZ)	1	5000
XR-ready PC	1	3700
Meta quest 3 Device	1	650
Meta quest 3 Carry Case	1	75
Meta quest 3 Elite Strap	1	75
Meta Quest Link Cable	1	94
Logitech MX Ink Stylus + Dock	1	175
Bespoke Mat for MX Ink	1	55
Total		9824

Table 1: Cost involved for XR demo implementation

D_{21} within Fig 1 can be considered as the Primary Visualizer in that scenario.

- XR Experience REVIEWER or SECONDARY VISUALIZERS: Various members of design teams benefit from collaborative discussions while experiencing CAD in both VR and MR. They sit in the same room as the XR Experience User and see a duplicated view on a separate screen, to infer design aspects together. Example: Subjects D_2, D_4, D_6, D_{10} and D_{20} within Fig 1 can be considered Secondary Visualizers.
- XR Experience DEMO MODERATOR: A dedicated member of the team is in-charge of moderating the session. They are required to be an expert at both CAD and XR technologies so they could minimize issues that arise during an EDR. Example: Subject A within Fig 1 is the Demo Moderator.

2.3.1 Ease of Use. Both VR and MR Templates were designed in a manner to accommodate all users (Primary and Secondary Visualizers) with different accessibility needs among controllers, hands and the XR stylus. Demo Moderators prompt all first-time users to develop the intuition towards such interactivity, eventually improving system usability with increase in experience. More experimentation is required to improve usability for Experience Creators and Moderators.

2.3.2 Key Features of the XR system for involved EDR stakeholders.

- CAD file compatibility:
  - More than 50 models were tested for compatibility and importing data with 3 extensions: SLDASM, SLDPRT and STEP files
  - Not tested, but compatible with 20+ other 3D file formats.
  - Not compatible with: EASM, Unity Scene Files etc
- Interactive features:
  - Real-scale 3D visualization or superimposition of CAD within VR or MR experiences respectively.
  - Drawing in VR/MR using the XR Stylus.
- Cost split for the involved Software and Hardware (Fig 4) components is described in Table 1. Please note that a fixed hardware cost is upfront, whereas Unity Industry software license bares an ever-increasing annual operating cost.

3 Demonstration of the XR-EDR Workflow

The XR-ready PC with pre-saved template project is connected to a Meta Quest HMD via a Link Cable along with a larger screen

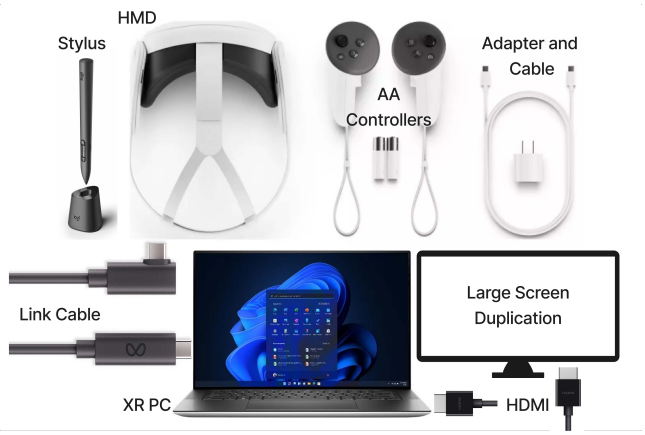


Figure 4: XR EDR Hardware



Figure 5: First-person POV of the MR-EDR Demonstration

(for duplication) via an HDMI. A live demonstration of the no-code MR-EDR experience creation workflow starts by showcasing the template and it's CAD import process. This is followed by some quick configuration steps e.g. repositioning and adding behavior to objects, so an MR-EDR review on the device could be simulated. Such XR simulations not only leads to a valuable experience, but also enables creation of collateral that can be taken out as feedback from the simulation. As a primary visualizer of such a demonstration, an attendee will be able to interact with the system in the following manner:

- Try on the HMD and perceive an Equipment CAD situated in their real physical environment.
- Use their physical hands in the experience to gather a sense of ergonomics and for simulating "Operator Reach".
- Move physically in the demo space to gather different stereoscopic perspective views of the anchored CAD model.
- Pick up and try using a stylus within the real world.
- Intuitively use Stylus and controller to explore options and draw different 3D free flowing lines relevant to the EDR.
- Take Screenshots, to be shared with non-VR team members.

Alternatively, as the secondary visualizer, they can see a duplicated Primary Visualizer view (with all the interactive behavior the user engages in) on a large screen. This creates a reference for discussion



Figure 6: Common words by stakeholders depicting richness among collected themes around XR process transformation

and enables collaboration on the proposed CAD design, leading to quick identifying ergonomic design issues in a proposed design.

4 Evaluation of VR EDR transformation

A three-phased mixed method qualitative data collection was performed with 31 industry stakeholders across a six month period in the context of evaluating the proposed VREDR template. They were interviewed twice (before and after) to gather nuanced perceptions on the VR process transformation in the company (check Fig 6 for commonly mentioned words). After speech to text via Deepgram SDK [15], Sublime Text [31] was used to redact all sensitive information. The anonymized data was then imported into Taguette [32] for further analysis. Inductive thematic analysis was performed on the rich interview data, that drove clarity on industry context specific definitions of EDR and related terminology. In an attempt to publish a dataset in future, videos and CSV files were also captured for each user during these VR-EDRs. Stakeholders reported a significant time reduction in VR experience creation, earlier quantified in "days" to "minutes" by using the proposed workflow.

4.0.1 Phase 1: Gathering clarity about traditional EDRs. Themes were derived from 31 stakeholder interviews that explored various dimensions associated with traditional EDR practices in BSCI. Topics adjacent to EDRs like Objective, Infrastructure, Roles of stakeholders, communication, skills and dependencies were discussed by stakeholders in a descriptive manner.

4.0.2 Phase 2: Implementation and capture of VR EDR. Each VR-EDR instance featured behavioral data collection in the form of CSV files along with video recordings of the VR EDR experience. All Unity projects were saved as a proof of the experimental instance. Captured quantities correspond to certain subjective qualities which in itself exposes various opportunities of exploration. Much of the analysis on this collected data is yet to be performed.

4.0.3 Phase 3: Post-VR EDR clarity. 28 engineers who interacted with the proposed VR offering either as a Primary or a Secondary Visualizer, brought out their individual perceptions during another interview. They described various nuances of VR-EDRs like impact on the objective, emotions felt, comments on usability, comparison with traditional EDRs, impact on communication, spatial perception, ability to focus on sub assemblies and overall decision making. Lastly, stakeholders gave feedback towards future role of VREDR in the company.

5 Discussion

A thorough literature review was conducted on XR based design reviews studied by academic-industrial partnerships. Systems proposed at Penn State University - USA [13, 23–25], University of Applied Sciences - Upper Austria [38–41], Tampere University - Finland [10–12], Karlsruhe University of Applied Sciences - Germany [3–9] and University of Zagreb [17–19, 26] were some of the works already done in this domain. Their limitations and attempts steered various assumptions and decisions in the scope of this research.

5.0.1 Relevance towards adjacent domains: Although "x" in CAx fundamentally stands for "Design", it extends to overlapping concepts like "Process Planning" [16], "Architecture Design" [20], "Engineering Design" [22] or even "Industrial Design" [37]. Besides adding a spatial (3D perceptive) interface to CAD, XR transformations add a layer of enhancement to various connected CAx technologies. XR-CAx is an emerging and expansive area of study for industries, emphasizing the need for a well-defined ontology informed by community-driven reviews of unique contributions. Such systems have a direct impact on bridging communication among teams that use separate informational views linked to the same Equipment Design. For example: Electrical and Mechanical Teams working on their own schematics but collaborating on a single assembly before it is ready for production.

5.0.2 Future Studies: Besides evaluating the MR-EDR workflow, the following requirements were proposed for upgrading XR-EDR Process Requests:

- Procurement and issuance of XR tech to be made easier.
- Training new stakeholders on XR-EDR usage.
- Updating workflows according to ecosystem updates.

XR Workflow Feature Requests:

- Ability to isolate and manipulate sub assemblies in VR.
- Simulate basic movement of equipment parts in XR.
- Synchronous multiplayer implementation for simultaneous stereoscopic views of Equipment CAD to support remote collaboration.
- Ability to play and pause animated CAD assemblies.
- Buffer save/delete to implement undo of drawing lines.
- Record data locally when drawn in 3D using stylus.
- Automatic text recognition of stylus drawn characters.
- Integration of VR/MR experiences with real world context by using tracked or anchored physical objects.

6 Conclusion

This paper exposes an infrastructural philosophy for manufacturing enterprises to deal with CAD data and XR tech in a modular manner. Pre-programmed and prefabricated assets that could be frequently required in experiences are saved inside reusable VR and MR projects so that experience creation comes with minimum overhead. For equipment designers, such container-oriented approaches lower the effort in creating context driven XR experiences that have further shown to reduce cognitive load for mechanical design reviewers. The proposed XR adoption strategy can be implemented

in many overlapping and adjacent domains, although it needs to be evaluated with context specific users in a qualitative manner to sustain itself in each scenario. As observed from stakeholders in this research, by initiating and evolving XR capabilities in teams, a utility driven culture could be built. This aims to position XR as a task enhancing tool instead of a threat to manual task operators.

Acknowledgments

Science Foundation Ireland (SFI) grant number SFI/16/RC/3918, co-funded by CONFIRM Smart Manufacturing and Boston Scientific Galway.

References

[1] Igor Agbossou. 2023. Multi-Scalar urban digital twin design: Architecture and OpenUSD standards based methodology. *PriMera Scientific Engineering* 4, 1 (2023), 33–48.

[2] Peter Bajcsy, Rob Kooper, Luigi Marini, Kenton McHenry, and Michal Ondrejcek. 2010. A framework for understanding file format conversions. In *Proceedings of the 2010 Roadmap for Digital Preservation Interoperability Framework Workshop*. ACM, Gaithersburg Maryland USA, 1–7. doi:10.1145/2039274.2039284

[3] Fahmi Bellalouna. 2019. Virtual-Reality-based Approach for Cognitive Design-Review and FMEA in the Industrial and Manufacturing Engineering. In *2019 10th IEEE International Conference on Cognitive Infocommunications (CogInfoCom)*. 41–46. doi:10.1109/CogInfoCom47531.2019.9090002 ISSN: 2380-7350.

[4] Fahmi Bellalouna. 2019. VR Based Design Process of Industrial Products. In *International Conference on Competitive Manufacturing*. 240–245.

[5] Fahmi Bellalouna. 2020. Industrial case studies for digital transformation of engineering processes using the virtual reality technology. *Procedia CIRP* 90 (2020), 636–641.

[6] Fahmi Bellalouna. 2020. New approach for digital factory using virtual reality technology. *Procedia CIRP* 93 (2020), 256–261.

[7] Fahmi Bellalouna. 2021. The augmented reality technology as enabler for the digitization of industrial business processes: case studies. *Procedia CIRP* 98 (2021), 400–405.

[8] Fahmi Bellalouna. 2021. Case study for design optimization using the digital twin approach. *Procedia CIRP* 100 (2021), 595–600.

[9] Fahmi Bellalouna and David Puljiz. 2023. Use case for the application of the industrial metaverse approach for engineering design review. *Procedia CIRP* 119 (2023), 638–643.

[10] Alisa Burova, Hanna Heinonen, Paulina Becerril Palma, Tuuli Keskinen, Jaakko Hakulinen, Viveka Opas, John Mäkelä, Kimmo Ronkainen, Sanni Siltanen, Roope Raisamo, et al. 2021. Toward efficient academia-industry collaboration: A case study of joint VR system development. In *Proceedings of the 24th International Academic Mindtrek Conference*. 176–185.

[11] Alisa Burova, John Mäkelä, Hanna Heinonen, Paulina Becerril Palma, Jaakko Hakulinen, Viveka Opas, Sanni Siltanen, Roope Raisamo, and Markku Turunen. 2022. Asynchronous industrial collaboration: How virtual reality and virtual tools aid the process of maintenance method development and documentation creation. *Computers in Industry* 140 (2022), 103663.

[12] Alisa Burova, Paulina Becerril Palma, Phong Truong, John Mäkelä, Hanna Heinonen, Jaakko Hakulinen, Kimmo Ronkainen, Roope Raisamo, Markku Turunen, and Sanni Siltanen. 2022. Distributed asymmetric virtual reality in industrial context: Enhancing the collaboration of geographically dispersed teams in the pipeline of maintenance method development and technical documentation creation. *Applied Sciences* 12, 8 (2022), 3728.

[13] Fadi Castronovo, Dragana Nikolic, Yifan Liu, and John Messner. 2013. An evaluation of immersive virtual reality systems for design reviews. In *Proceedings of the 13th international conference on construction applications of virtual reality*, Vol. 47. CONVR 2013) London, UK, 22–29.

[14] Fabio Vinicius de Freitas, Marcus Vinicius Mendes Gomes, and Ingrid Winkler. 2022. Benefits and Challenges of Virtual-Reality-Based Industrial Usability Testing and Design Reviews: A Patents Landscape and Literature Review. *Applied Sciences* 12, 3 (Jan. 2022), 1755. doi:10.3390/app12031755

[15] Deepgram Developers. 2024. *Python SDK- The official Python SDK for use with Deepgram APIs*. Deepgram. https://developers.deepgram.com/docs/python-sdk

[16] Hoda A ElMaraghy. 1993. Evolution and future perspectives of CAPP. *CIRP annals* 42, 2 (1993), 739–751.

[17] Nikola Horvat, Tomislav Martinec, Ivan Uremović, and Stanko Škec. 2024. Use it early: The effect of immersion on spatial and design space aspects in team-based mechanical design reviews. *Advanced engineering informatics* 59 (2024), 102270.

[18] Nikola Horvat, Stanko Škec, Tomislav Martinec, Fanika Lukačević, and Marija Majda Perišić. 2019. Comparing virtual reality and desktop interface for reviewing 3D CAD models. In *Proceedings of the design society: international conference on engineering design*, Vol. 1. Cambridge University Press, 1923–1932.

[19] Nikola Horvat, Stanko Škec, Tomislav Martinec, Fanika Lukačević, and Marija Majda Perišić. 2020. IDENTIFYING THE EFFECT OF REVIEWERS'EXPERTISE ON DESIGN REVIEW USING VIRTUAL REALITY AND DESKTOP INTERFACE. In *Proceedings of the Design Society: DESIGN Conference*, Vol. 1. Cambridge University Press, 187–196.

[20] Yehuda E. Kalay. 1999. *The future of CAAD: From computer-aided design to Computer-aided collaboration*. Springer US, Boston, MA, 13–30. doi:10.1007/978-1-4615-5047-1_2

[21] Vasil Kozov, Aleksandar Ivanov, and Galina Ivanova. 2024. Research Result Differences Between Manual Research and Using AI LLMs for Analyzing Popular 3D File Formats. *TEM Journal* 13, 4 (2024), 2637.

[22] Jason R Lemon, SK Tolani, and Albert L Klosterman. 1980. Integration and implementation of computer-aided engineering and related manufacturing capabilities into the mechanical product development process. In *CAD-Fachgespräch: GI—10. Jahrestagung, Saarbrücken, 30. September–2. Oktober 1980*. Springer, 161–183.

[23] Yifan Liu. 2017. *Evaluating design review meetings and the use of virtual reality for post-occupancy analysis*. Ph. D. Dissertation. The Pennsylvania State University.

[24] Yifan Liu, Fadi Castronovo, John Messner, and Robert Leicht. 2020. Evaluating the Impact of Virtual Reality on Design Review Meetings. *Journal of Computing in Civil Engineering* 34, 1 (Jan. 2020), 04019045. doi:10.1061/(ASCE)CP.1943-5487.0000856 Publisher: American Society of Civil Engineers.

[25] Yifan Liu, Jennifer Lather, and John Messner. 2014. Virtual reality to support the integrated design process: A retrofit case study. In *Computing in civil and building engineering (2014)*. 801–808.

[26] Fanika Lukačević, Stanko Škec, Peter Törlind, and Mario Štorga. 2021. Identifying subassemblies and understanding their functions during a design review in immersive and non-immersive virtual environments. *Frontiers of engineering management* 8 (2021), 412–428.

[27] Meta. 2024. *Meta XR All-in-One SDK*. Meta. https://assetstore.unity.com/packages/tools/integration/meta-xr-all-in-one-sdk-269657?srsltid=AfmBOor_npZlOQBia89JeArSxEkIwt90EsvDsYvmXRdDTpUnI6kGNi_j

[28] PiXYZ. 2024. *PiXYZ Plugin - Build powerful visualizations with the data you already have*. Unity Technologies. https://unity.com/products/pixyz-plugin

[29] Vennan Sibanda, Mufaro Kanganga, and Nomagugu Sibanda. 2023. Bridging the Gap between Industry and Academia-The Essence of Virtual Reality in Skills Development and Learning Factories. In *Proceedings of the 13th Conference on Learning Factories (CLF 2023)*.

[30] Sebastian Stadler. 2022. POTENTIAL USAGES OF VIRTUAL REALITY IN DESIGN RESEARCH AND PRACTICE–A REVIEW. *Business Meets Technology* 7 (2022), 193.

[31] Sublime Text. 2024. *Sublime Text - A text and source code editor*. Sublime Text. https://www.sublimetext.com/

[32] Taguette. 2021. *Taguette - a free and open source tool for textual qualitative data analysis*. Taguette. https://www.taguette.org/

[33] Unity. 2024. *Oculus Plugin for Unity*. Unity. https://docs.unity3d.com/Manual/com.unity.xr.oculus.html

[34] Unity. 2024. *OpenXR Plugin for Unity*. Khronos Group. https://docs.unity3d.com/Packages/com.unity.xr.openxr@1.14/manual/index.html

[35] Unity3D. 2024. *Unity - a cross-platform game engine*. Unity Technologies. https://unity.com/

[36] Unity3D. 2024. *XR Interaction Toolkit*. Unity3D. https://docs.unity3d.com/Packages/com.unity.xr.interaction.toolkit@3.0/manual/index.html

[37] Stephen H Westin. 1998. Computer-aided industrial design. *Acm Siggraph Computer Graphics* 32, 1 (1998), 49–52.

[38] Josef Wolfartsberger. 2019. Analyzing the potential of Virtual Reality for engineering design review. *Automation in Construction* 104 (Aug. 2019), 27–37. doi:10.1016/j.autcon.2019.03.018

[39] Josef Wolfartsberger, Jan Zenisek, and Christoph Sievi. 2018. Chances and Limitations of a Virtual Reality-supported Tool for Decision Making in Industrial Engineering. *IFAC-PapersOnLine* 51, 11 (Jan. 2018), 637–642. doi:10.1016/j.ifacol.2018.08.390

[40] Josef Wolfartsberger, Jan Zenisek, Christoph Sievi, and Mathias Silmbroth. 2017. A virtual reality supported 3D environment for engineering design review. In *2017 23rd International Conference on Virtual System & Multimedia (VSMM)*. 1–8. doi:10.1109/VSMM.2017.8346288 ISSN: 2474-1485.

[41] Josef Wolfartsberger, Jan Zenisek, and Norbert Wild. 2020. Supporting Teamwork in Industrial Virtual Reality Applications. *Procedia Manufacturing* 42 (2020), 2–7. doi:10.1016/j.promfg.2020.02.016

Received ; revised ; accepted

Learned Compression in Adaptive Point Cloud Streaming: Opportunities, Challenges and Limitations

Michael Rudolph
michael.rudolph@ikt.uni-hannover.de
Leibniz University Hannover, Germany

Amr Rizk
amr.rizk@ikt.uni-hannover.de
Leibniz University Hannover, Germany

Abstract

Learned point cloud compression methods have achieved rate-distortion performance, which is comparable to or higher than conventional approaches. However, this often comes at the cost of high hardware requirements and thus low throughput during encoding and decoding. In this paper, we present an adaptive bitrate point cloud streaming system utilizing learned compression. While other learned compression techniques require to split geometry and attributes, resulting in high encoding latency, we deploy a unified model to handle both modalities together, which drastically reduces the coding complexity. We explore the capabilities of the learned encoder to derive multiple quality representations with only re-running a fraction of the encoding steps, making it a suitable fit for adaptive bitrate streaming. Furthermore, we ablate the encoding latency of each component in the encoder and decoder stack, identifying bottlenecks in the process.

CCS Concepts

• **Information systems** → **Multimedia streaming**; • **Computing methodologies** → **Point-based models**.

Keywords

Point Cloud Streaming, Adaptive Bitrate Streaming, Learned Point Cloud Compression

ACM Reference Format:
Michael Rudolph and Amr Rizk. 2025. Learned Compression in Adaptive Point Cloud Streaming: Opportunities, Challenges and Limitations. In *ACM Multimedia Systems Conference 2025 (MMSys '25), March 31–April 4, 2025, Stellenbosch, South Africa.* ACM, New York, NY, USA, 7 pages. https://doi.org/10.1145/3712676.3719266

1 Introduction and Problem Statement

Immersive multimedia transforms digital content consumption from two-dimensional videos to three-dimensional reconstructions of the real world, granting users more degrees of freedom to consume the media. Point clouds have emerged as a suitable data format for three-dimensional content. A multimedia point cloud contains a set of coordinates, describing the geometry of the scanned object and a color value, assigned to each point location to allow for textured components.

Permission to make digital or hard copies of all or part of this work for personal or classroom use is granted without fee provided that copies are not made or distributed for profit or commercial advantage and that copies bear this notice and the full citation on the first page. Copyrights for third-party components of this work must be honored. For all other uses, contact the owner/author(s).
MMSys '25, March 31–April 4, 2025, Stellenbosch, South Africa
© 2025 Copyright held by the owner/author(s).
ACM ISBN 979-8-4007-1467-2/25/03
https://doi.org/10.1145/3712676.3719266

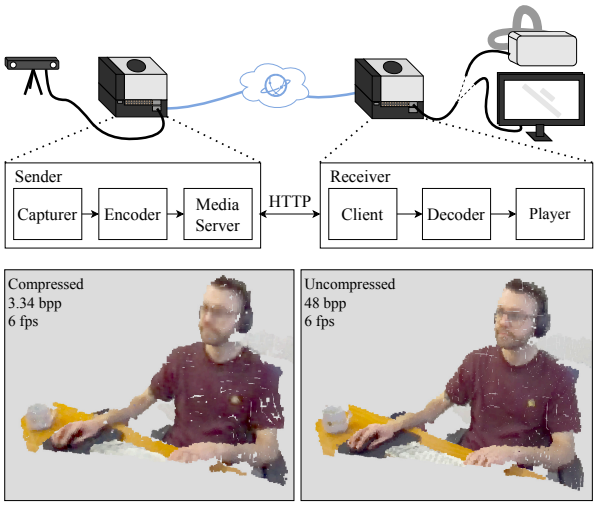


Figure 1: Setup of the presented system, using a NVIDIA Jetson AGX Orin developer kit for encoding a point cloud captured with a ZED2i depth camera. On the receiving side, a client streams the data and decodes them using a second Orin developer kit. Streaming is carried out using DASH.

The flexibility and low processing overhead of point clouds allows for realistic reconstructions of the real world in an immersive scenario. Building on the scalability and simplicity of Adaptive Bitrate Streaming, the de-facto standard for large-scale video delivery, prior work [12] proposed an extension to Dynamic Adaptive Streaming over HTTP (DASH) to handle dynamic point clouds. Further improvements [21] extend the scenes to multiple point clouds (e.g. persons) and evaluate strategies for adaptive bitrate schemes based on the user viewport and bandwidth. Building on this foundation to support point cloud tiling schemes the work in [19, 20] allows for more fine-grained quality adaptation of the content based on a users viewport. Recently, tele-immersive video conferencing systems [13, 22] relying on DASH have been proposed.

As a sufficiently high number of points is required to allow dense, visually pleasing representations of humans [17], compression is essential to reduce the required bitrate, allowing efficient and practical streaming of point cloud content. While elaborate compression techniques from standardization, namely G-PCC [2] and V-PCC [1], exist, their current available implementations possess high encoding and decoding latencies. As a result, most existing point cloud streaming systems [13, 22] resort to fallback coding solutions [15]

or sampling schemes [12], exchanging compression efficiency for required throughput.

So far, there has been an increasing interest in learning-based techniques for point cloud compression [11, 18, 24, 25]. Along the same lines some standardization efforts from the JPEG [10], and increased availability of hardware-support for neural network inference on client devices show that learned point cloud compression is a promising avenue. While the deployment of learned codecs in a video streaming system has been shown before [8], to the best of our knowledge, this work is the first to *demonstrate and profile* learned point cloud compression techniques in an immersive point cloud streaming scenario. In this paper, we present an end-to-end DASH-based multimedia point cloud streaming system leveraging learned compression techniques based on our joint geometry and attribute compression model from [18]. We summarize our contributions in the following:

- We provide a system architecture, describe an implementation and evaluate *learned point cloud compression* techniques for adaptive bitrate point cloud streaming using NVIDIA Jetson AGX Orin devices, including capturing and visualization.
- We show evaluation and profiling results of the proposed point cloud streaming system to highlight challenges and limitations when deploying learned compression models in real-world scenarios.

2 Learned Point Cloud Compression

We base the compression model in the presented system on [18], allowing to jointly compress geometry and attributes in a single encoder-decoder style architecture while covering a wide range of trade-offs between geometry and attribute qualities and thus rates through quality-adaptive layers. Next, we briefly review the motivation of *a single model* in Sec. 2.1 and describe the architectural adaptation of the model from [18] for the presented system in Sec. 2.2. Since the system is focused on inference, we will describe the process for encoding a segment in Sec. 2.3 and decoding in Sec. 2.4.

2.1 Geometry and Attribute Compression in a Single Model

Current learned point cloud compression approaches split geometry and attributes for individual compression [10, 24, 25]. However, this adds increased complexity during encoding. Since attribute compression requires the geometry as side-information for encoding and decoding, it is common practice to reconstruct the geometry at sender-side [10, 24, 25], emulating the expected distortions introduced by geometry compression. As a result, the encoding latency of related approaches is high.

Using a single model for both, the geometry and attributes, allows to significantly simplify the encoding and decoding process. However, traditional rate-distortion optimization [5], which typically optimizes a single trade-off, is unsuitable for balancing multiple distortion terms, leading to the need for grid-searching possible trade-offs [23] or reducing the search-space by fixing trade-offs according to some prior knowledge [3, 11] before training. Addressing this, conditioning an adaptive model for *geometry and attribute* compression on all possible combinations in the search space allows to cover all possible trade-offs at comparable performance [18]. In the presented system, we follow this training procedure as described in [18].

2.2 Architectural Decisions

While the model described in [18] allows to compress geometry and attributes in a single encoding and decoding step, we further simplify the architecture for higher throughput and move the quality adaptive layers into the bottleneck to allow re-using components of the encoder when deriving multiple ABR representations of the same segment.

The architectural changes are mainly restricted to the 3D-CNN encoder and decoder blocks, also termed analysis and synthesis transform, respectively. In contrast to the elaborate 3D-CNN architecture of [18], we follow the significantly simpler structure of image compression models [6, 16], replacing the 2D image convolutions by sparse, 3D convolutions [9]. In the synthesis transform, additional pruning layers [23] are required after each upsampling step to select the $k_{i,j}$ most probable voxels, retaining a sparse representation. Hereby, $k_{i,j}$ is the number of occupied voxels of the ith frame at each level j of the encoder, which will be signaled through the bitstream. Furthermore, we drastically reduce the number of channels in each decoder layer to achieve higher throughput.

Finally, we bundle the rate-adaption of the model entirely in an extension of the Gaussian Bottleneck [16] layer based on [14], significantly simplifying the quality control of the model. This allows to vary the quantization bin sizes before entropy coding the coefficients of the feature representation depending on the desired trade-off between geometry and attribute quality.

2.3 Encoder Components

A schematic overview of the encoding process is depicted in Fig. 2, where each major processing step is denoted by an identifier (e.g. E1 for the Analysis Transform). We denote a point cloud through $\mathcal{P}_i = (\mathbf{g}_i, \mathbf{a}_i)$, where $\mathbf{g}_i$ is the set of coordinates, $\mathbf{a}_i$ are the features associated with $\mathbf{g}_i$ (e.g. colors) and i is the index of a frame in a segment. We omit the subscript for a batched representation, i.e. $\mathbf{g}$ denotes the concatenation of N geometries $\mathbf{g}_1, ..\mathbf{g}_N$ in a batch. We symbolize downsampled versions of a geometry with a superscript, e.g., $\mathbf{g}^j$ represents the geometry $\mathbf{g}$ downsampled to level j, with increasing j indicating coarser levels of resolution.

To increase the throughput of our system, each step is executed by a separate thread and queues are used to pass the output of a step to the next. This allows for pipelined execution of the coding process.

2.3.1 Analysis Transform (E1). In the analysis transform step, the encoder model is used to derive the latent feature representation $(\mathbf{g}^{(3)}, \mathbf{y})$ from the input point cloud frames $\mathcal{P}_1, ..., \mathcal{P}_N$. We combine all input point cloud frames of the point cloud into a batch, allowing to perform the inference using the 3D-CNN only once.

2.3.2 Hyper Analysis Transform (E2). In the next step, the hyper analysis transform is used to derive the hyperprior $(\mathbf{g}^{(5)}, \mathbf{z})$ from the latent representation $(\mathbf{g}^{(3)}, \mathbf{y})$. Similar to the previous step, the

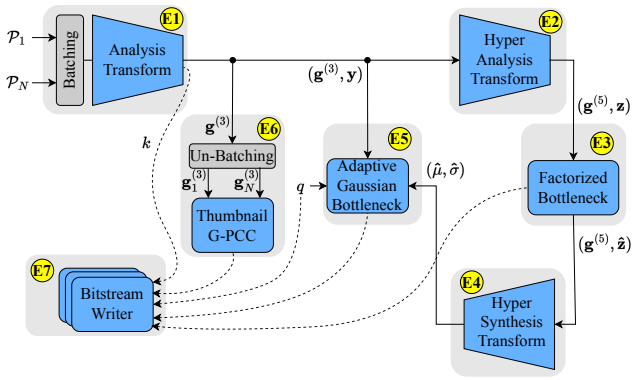


Figure 2: Overview of the encoder steps for compressing a segment of N frames. Thumbnail geometry compression in E6 can be executed in parallel to the steps E3-E5.

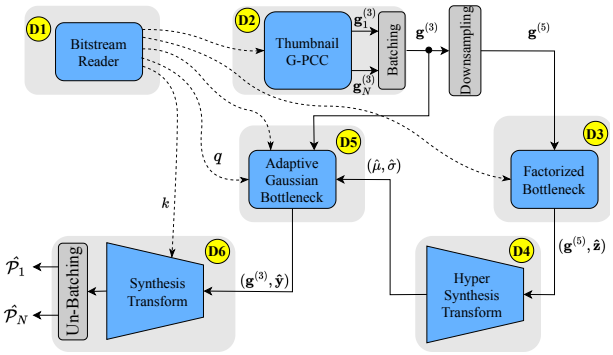


Figure 3: Overview of the decoder steps, required to run in sequential order to decompress a segment into N frames.

hyper analysis transform is run on batched data, requiring only one pass of the 3D-CNN model.

2.3.3 Factorized Bottleneck (E3). The features of the hyperprior $\mathbf{z}$ are compressed using a factorized entropy model [6], receiving a bitstream for the full group of frames. Note that it is not required to compress the geometry $\mathbf{g}^{(5)}$ of the hyperprior, as we can derive it from the latent geometry $\mathbf{g}^{(3)}$ at decoding time. However, it is required to sort the hyperprior features $\mathbf{z}$ before encoding through the factorized bottleneck according to a deterministic order based on their base geometry $\mathbf{g}^{(5)}$, allowing correct reassignment of the decoded hyperprior on the receiver side.

2.3.4 Hyper Synthesis Transform (E4). Next, we emulate the estimation of the parameters for the adaptive Gaussian entropy bottleneck [16]. For this, we infer the means $\hat{\mu}$ and standard deviations $\hat{\sigma}$ through the hyper synthesis transform CNN, which parameterize the local distribution of the latent features $\mathbf{y}$.

2.3.5 Adaptive Gaussian Bottleneck (E5). The adaptive Gaussian Bottleneck quantizes and entropy codes latent features $\mathbf{y}$. Hereby, the probability in the quantization bins is derived from a Gaussian distribution, which is parameterized by estimates of the local mean $\hat{\mu}$ and scale $\hat{\sigma}$ for each feature component [16]. In contrast to the unified compression model in [18], the model here performs *rate-quality adaptation in the adaptive Gaussian bottleneck layer*. For this, a quality vector q, dictating the qualities of the geometry and attributes, is supplied. Per-channel scaling factors are inferred from q through a small neural network, which are then used to vary the per-channel quantization bin sizes of the model, as described by [14]. Similar to the Factorized Bottleneck step (E3), we entropy code the full group of frames into a single bitstream. Supplying multiple quality vectors q, i.e. for deriving multiple quality representations at once, is supported in batched inference and results in a *separate encoded bitstream per quality* tensor.

2.3.6 Geometry Compression (E6). The thumbnail geometry $\mathbf{g}^{(3)}$ is required for decoding, and thus needs to be compressed and transmitted. This is done using G-PCC in lossless octree mode, resulting in a thumbnail point cloud bitstream for each frame. Note that this step is performed in parallel to steps E2-E6.

2.3.7 Bitstream Writing (E7). In the final step of the compression pipeline, we collect the entropy coded sub-streams for the latent features, the hyperprior and the thumbnail geometries to combine them into a self-contained bitstream. Additionally, a bitstream header describes the number of frames N in the segment, the quality configuration q and the length of all sub-streams. Further, for each frame i the list of the number of points at each level j of the encoding process $k_{i,j}$ is written to the bitstream, as it is required for pruning in the synthesis transform on the decoder side.

2.4 Decoder Components

The decoder mirrors the encoder in Sec. 2.3 and is visualized in Fig. 3. We detail the individual steps of the decoder using similar notation as in the preceding section.

2.4.1 Bitstream Reading (D1). First, the bitstream of the segment is parsed and split into the components described in step E7.

2.4.2 Geometry Decompression (D2). Next, the thumbnail geometries $\mathbf{g}_1^{(3)}, ..., \mathbf{g}_N^{(3)}$ of each frame are decompressed using G-PCC and batched for their use in further processing.

2.4.3 Factorized Bottleneck (D3). The factorized bottleneck layer decompresses the hyperprior features $\hat{\mathbf{z}}$. To reassign the decompressed hyperpriors to their geometric location, first, the hyperprior base geometry $\mathbf{g}^{(5)}$ is derived from $\mathbf{g}^{(3)}$ through dyadic downsampling. Then, $\mathbf{g}^{(5)}$ is sorted according to the same criteria as in E3, ensuring correct assignment of the hyperprior features $\hat{\mathbf{z}}$.

2.4.4 Hyper Analysis Transform (D4). The hyper analysis step is identical to step E4 during encoding, described in Sec. 2.3.4.

2.4.5 Adaptive Gaussian Bottleneck (D5). For decoding the coefficients, the reciprocal of the scaling factors used in step E5 is derived from the quality configuration q, allowing to decode and rescale the quantized latent coefficients $\hat{\mathbf{y}}$ [14], once again requiring their respective estimates of mean $\hat{\mu}$ and scale $\hat{\sigma}$ to parameterize the

coding distribution [16]. After decoding the coefficients, they are re-assigned to their respective geometry, using the same sorting procedure as for D3.

2.4.6 Synthesis Transform (D6). In the final step of the decoding process, the batched point cloud is reconstructed from the latent representation $(\mathbf{g}^{(3)}, \hat{\mathbf{y}})$ through the synthesis transform 3D-CNN. The number of points $k_{i,j}$ for each level j per frame i of the reconstruction are supplied as a side-information to perform pruning in a multi-scale manner [23] and reduce the computational complexity of the synthesis transform.

3 System Overview

An overview of our system is given in Fig. 1, which is implemented in Python, leveraging MinkowskiEngine [9] and CompressAI [7] for the learned codec implementation. For deployment, we use Docker for each module on the sender and receiver side, realizing inter-module communication through ZeroMQ. On the sender side, we capture point cloud frames with a ZED 2i depth camera, encode them and publish segments via a media server using DASH. On the receiver side, a client checks for updates and selects and streams segments at a desired quality (i.e. bitrate), decodes them and sends them to a visualizer. While the presented system is using a single receiver, scaling the number of clients is easily possible.

3.1 Sender

In the point cloud capturer, we grab frames from the depth camera and construct a colored point cloud from the color and depth images. We decide to buffer the frames for the duration of one segment before re-sampling to a target frame rate and sending a segment to the encoder. Re-sampling is required to allow a *configurable target frame rate* instead of relying on the source frame rate dictated by the RGB-D camera. For this, we compute the target timestamps for the desired frame rate and select the frames with the closest capturing timestamp to a target timestamp.

As the learned compression model requires a voxelized representation of the point cloud (the model resembles a 3D-CNN operating on sparse voxel data structures), we perform additional pre-processing in the capturing module to derive a sparse voxelized representation for each point cloud frame. Hereby, the selected voxel size allows to adjust the *resolution* of the captured point cloud, and thus the number of points per frame. In contrast to 2D video, the number of points in a frame is not fixed but depends on the content, introducing variable work loads for encoding and decoding. To limit this effect, we provide an option to upper bound the maximum number of points by removing the points furthest away from the camera (background removal).

The encoder module handles the codec for compressing the point cloud, which is detailed in Sec. 2.3. The output of the encoder module is a collection of bitstreams at different, specified quality configurations dictated by the quality vector q, which are forwarded to the media server.

The media server module hosts a HTTP Server and supplies the encoded data representation for adaptive bitrate streaming. The representations of a new segment are published and a Media Presentation Description (MPD) file is dynamically updated with the bandwidth information for every new segment. To counteract fluctuations in the encoding process, a buffer is employed to publish segments with a pre-defined offset to their capturing timestamp.

3.2 Receiver

The client in our presented system emulates an adaptive bitrate streaming client, for which we allow to manually select quality representations to visualize different coding configurations. During streaming the client checks for updates in the MPD and downloads the representation of a segment at the desired quality. Upon arrival, the segment is forwarded to the decoder module. The decoder module manages the decompression of segments using the learned codec as outlined in Sec. 2.4

The player consists of a backend, which receives decoded frames from the decoder and manages the playout queue. On playout, the player forwards the frame to the frontend using websockets. The frontend is implemented in JavaScript, utilizing the three.js package to dynamically render the point cloud in a web browser.

3.3 Demo Setup

For the demo and evaluation, we use two NVIDIA Jetson AGX Orin Developer boards, as they offer GPU support for learned compression. The first Jetson Orin device will serve as the sender and run the modules described in Sec. 3.1. The dynamic point clouds are captured using a ZED 2i depth camera, which is connected to the Jetson device via USB. The second Jetson device is used as a client, running the receiver functionality as described in Sec. 3.2. The code for the demo is available on GitHub[1].

4 Evaluation Results

We present preliminary results for our learned compression-based point cloud streaming system. Our main focus here lies in showing the throughput of the encoder and decoder, and its components which we investigate in detail in Sec. 4.1. Further, we investigate the end-to-end latency of the system at multiple target frame rates.

To fix the baseline, we recorded a test sequence, consisting of a single human in front of the camera, which can be played back from the capturing module.

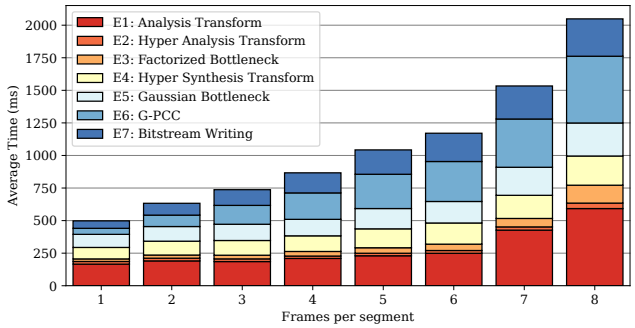
For the evaluation, we select a segment length of 1 second and vary the number of frames per segment, which can be directly translated to the target frame rate. We remove the first 3 segments from the evaluation, as the learned encoder and decoder require a few passes to exclude initialization effects.

4.1 Codec Performance

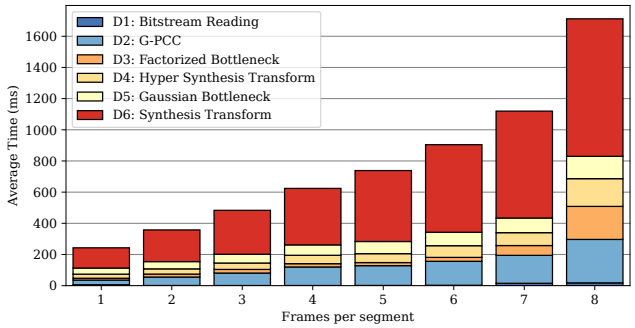
We measure the average latency of each component of the compression model per segment, and visualize them in Fig. 4a and 4b, respectively, while varying the number of frames per segment. The stacked bars in the plot depict the average latency over 69 segments for each component from receiving its input until pushing the result to its output queue. This does not incorporate queuing delays (increasing the overall latency) or effects from parallel execution of step E6 to steps E2-E5 (reducing the overall latency).

We observe an abrupt increase in the latency of components when increasing the number of frames per segment (which directly

[1] https://github.com/ikt-luh/Demo-Learned-Point-Cloud-Compression



(a) Encoder

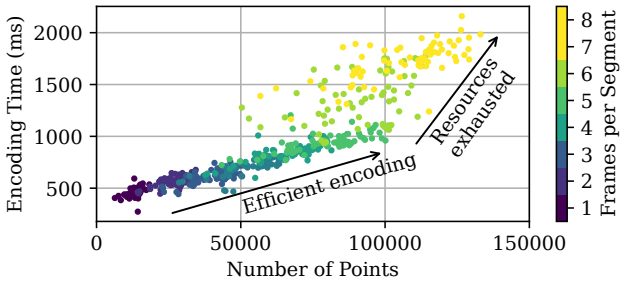


(b) Decoder

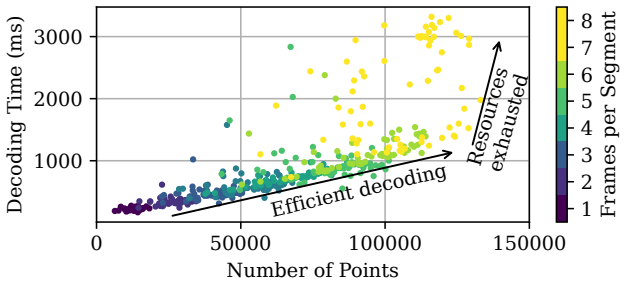
Figure 4: Stacking of the average latencies per component of the encoder and decoder components from Fig. 2 and Fig. 3 to the encoding and decoding latencies, respectively. Measurements are averaged over the full test sequence at different target frame rates with a fixed segment length of 1s.

translates to the number of points to be processed at once) over a certain threshold. At the encoder, simultaneous execution of processing steps that require GPU hardware through pipelined execution of a segment before the current segment has been fully encoded results in the need for context switching. This becomes apparent in the measurement bars for 7 and 8 frames per segment. In detail, context-switching leads to additional overhead and increased latency for GPU intensive steps, i.e. steps E1, E2 and E3. Similar behaviour can be observed for decoding, where we notice an abrupt increase in component latency for 8 frames per segment, which is dominated by the synthesis transform step (D6).

Figure 5 shows the end-to-end encoding and decoding times for segments *with* queuing delays and parallelization effects against the number of points in a segment. With increasing number of points, the encoding and decoding times increase linearly until resources are exhausted and queuing starts which leads to a steeper increase in encoding and decoding times. Overall, we observe an abrupt increase in encoding and decoding latency in Fig. 4 after exceeding a certain threshold of frames per segment. Note that increasing the queue lengths results in overall higher decoder latency compared to the sum of its components.



(a) End-to-end encoding latency



(b) End-to-end decoding latency

Figure 5: End-to-end encoding and decoding latency for a segment in the test sequence vs number of points in the segment. Notice that the encoding and decoding latency increase linearly when increasing the number of points until depletion of computing resources, resulting in exploding latency through queuing delays and context switching.

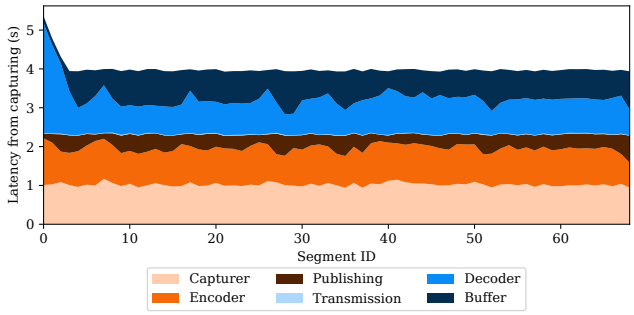
4.2 System Latency

Next, we evaluate the end-to-end latency from capturing to playout of a frame in our system. The overall system latency per segment is visualized in Fig. 6, consisting of the sender side latencies (brownish tones) and receiver side latencies (blueish tones). The top figure shows a case at 6 frames per segment, while the bottom figure visualizes a case for 8 frames per segment that introduces noticeable stalling through high encoding and decoding workloads.

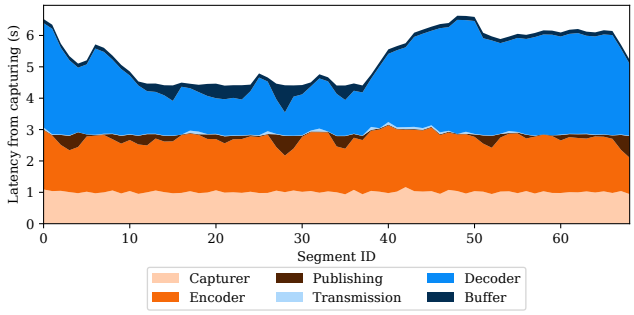
Note that we are still able to deliver stall-free representations for 6 frames per segment, although the stacked individual encoding latency in Fig. 4 exceeds the segment length. This is achieved through pipelined and parallelized execution of the encoder components.

On the sender side, capturing delay incorporates buffering a segment and sampling the frames in the segment, accounting for approximately the segment length. Recall that the encoding latency depends on the number of points in a segment (cf. Fig. 5).

On the receiver side, the transport latency incorporate checking for updates of the MPD, selecting an appropriate quality and requesting the segment via HTTP. Note that our evaluation was performed in a LAN environment, resulting in negligible network delays. Once a segment is received, it is decoded, once again resulting in a decoding latency which varies depending on the number of points within the point cloud.



(a) End-to-end latency for 6 frames per segment



(b) End-to-end latency for 8 frames per segment

Figure 6: End-to-End latency from capturing to playout. While sufficient buffering at 6 frames per segment ensures smooth playback, the high decoding latency results in filling the decoder queue, causing stalling during playback.

5 Discussion

5.1 Opportunities

The adaptivity of the deployed model from [18] is a perfect fit for ABR streaming of multimedia point clouds. On the *receiver* side, having an adaptive model circumvents the need to exchange the current model weights or deploying an ensemble of models for smooth switching, as quality switching can be handled by a single model through supplying quality parameters. On the *sender* side, a majority of the (computationally intensive) processing steps can be re-used when encoding a segment to multiple representations, requiring to only perform batched processing of the Gaussian bottleneck [14] for multiple quality condition parameters q and writing multiple, distinct bitstreams. We observe that GPU processing can be harnessed by processing multiple frames in batched manner which is a promising property of learned compression models. Finally, while the above results are promising, the overall number of processed points and frame rates shown in this paper still require improvement for learned compression to be deployed effectively in real-world applications.

5.2 Challenges and Limitations

The evaluation identifies a number of challenges that need to be addressed to further improve the performance of learned compression techniques. First, on the decoder side, we notice an increase in latency for the Synthesis Transform step (D6) when increasing the number of frames per segment, constituting the majority of decoding latency. This is mainly rooted in the evaluation of all possible voxel locations in each upsampling step, requiring to temporarily materialize locally dense representations which are subsequently pruned [23]. We believe that developing more lightweight upsampling solutions will help alleviating this issue.

A further challenge lies in the requirement for hardware-acceleration. Although the NVIDIA Jetson AGX Orin GPU is light, the requirement of GPU hardware acceleration is a major obstacle in democratizing immersive multimedia experiences that utilize compression with learned codecs. Note that using dedicated hardware on the sender side already allows significantly increased performance.

Deploying learned compression techniques on heterogeneous computing infrastructure will additionally require to change the hyper synthesis step (E4/D4) from floating point to integer computation, as rounding errors caused by different hardware architectures might well result in catastrophic failures during decoding, as shown for related image compression models [4].

Finally, while not specific to learned compression, the variability of the number of points, which depends on the content of the scene, constitutes an issue for compression and streaming. In contrast to image and video compression, that usually processes a given number of pixels, the number of points in a dynamic point cloud is highly dependent on the content and varies over time. This introduces variable workloads to the encoder and decoder, making it more difficult to comply with playout deadlines. This needs to be considered when specifying buffer lengths, possibly resulting in higher overall end-to-end latency.

6 Conclusion

We demonstrated and profiled a deployment of a learned compression model for streaming multimedia point clouds on NVIDIA Jetson AGX Orin devices. Utilizing hardware-acceleration we achieve live streaming of up to 10^5 points per second at 6 fps with an end-to-end latency of roughly 4 seconds.

Conceptually, the learned codec deployed in the presented system offers a perfect fit for adaptive bitrate streaming. Using a single, adaptive model facilitates deployment on the client side by alleviating the need to alternate between different models when switching between quality representations. At the sender, the encoding process allows for elegant derivation of different quality representations by re-using a majority of computationally intensive processing steps.

We find that the overall throughput of the learned compression model does still not match traditional codecs like [15], as demonstrated in [13, 22]. Although learned compression offers strong results in terms of rate-distortion performance [10, 11, 18, 24, 25], the requirement for GPUs hinders practical use of current learning-based solutions and reinforces the need for developing more lightweight learned compression techniques.

References

[1] ISO/IEC JTC 1/SC 29. 2021. *ISO/IEC 23090-5:2021, Information technology — Coded representation of immersive media — Part 5: Visual volumetric video-based coding (V3C) and video-based point cloud compression (V-PCC).* ISO/IEC.

[2] ISO/IEC JTC 1/SC 29. 2023. *ISO/IEC 23090-9:2023, Information technology — Coded representation of immersive media — Part 9: Geometry-based point cloud compression.* ISO/IEC.

[3] Evangelos Alexiou, Kuan Tung, and Touradj Ebrahimi. 2020. Towards neural network approaches for point cloud compression. In *Applications of digital image processing XLIII.* SPIE, 18–37.

[4] Johannes Ballé, Nick Johnston, and David Minnen. 2019. Integer networks for data compression with latent-variable models. In *International Conference on Learning Representations (ICLR).*

[5] Johannes Balle, Valero Laparra, and Eero P. Simoncelli. 2016. End-to-end optimization of nonlinear transform codes for perceptual quality. In *Picture Coding Symposium (PCS).* IEEE, 1–5.

[6] Johannes Ballé, David Minnen, Saurabh Singh, Sung Jin Hwang, and Nick Johnston. 2018. Variational image compression with a scale hyperprior. In *International Conference on Learning Representations (ICLR).*

[7] Jean Bégaint, Fabien Racapé, Simon Feltman, and Akshay Pushparaja. 2020. CompressAI: a PyTorch library and evaluation platform for end-to-end compression research. *arXiv preprint arXiv:2011.03029* (2020).

[8] Bo Chen, Zhisheng Yan, Yinjie Zhang, Zhe Yang, and Klara Nahrstedt. 2024. LiFteR: Unleash Learned Codecs in Video Streaming with Loose Frame Referencing. In *21st USENIX Symposium on Networked Systems Design and Implementation (NSDI 24).* 533–548.

[9] Christopher Choy, JunYoung Gwak, and Silvio Savarese. 2019. 4D Spatio-Temporal ConvNets: Minkowski Convolutional Neural Networks. In *IEEE/CVF Conference on Computer Vision and Pattern Recognition (CVPR).* 3070–3079.

[10] André FR Guarda, Nuno MM Rodrigues, and Fernando Pereira. 2024. The JPEG Pleno Learning-based Point Cloud Coding Standard: Serving Man and Machine. *arXiv preprint arXiv:2409.08130* (2024).

[11] André F. R. Guarda, Manuel Ruivo, Luís Coelho, Abdelrahman Seleem, Nuno M. M. Rodrigues, and Fernando Pereira. 2025. Deep Learning-Based Point Cloud Coding and Super-Resolution: A Joint Geometry and Color Approach. *IEEE Transactions on Multimedia* 27 (2025), 914–926.

[12] Mohammad Hosseini and Christian Timmerer. 2018. Dynamic Adaptive Point Cloud Streaming. In *Proceedings of the 23rd Packet Video Workshop (PV '18).* ACM.

[13] Jack Jansen, Shishir Subramanyam, Romain Bouqueau, Gianluca Cernigliaro, Marc Martos Cabré, Fernando Pérez, and Pablo Cesar. 2020. A pipeline for multiparty volumetric video conferencing: transmission of point clouds over low latency DASH. In *Proceedings of the 11th ACM Multimedia Systems Conference (MMSys '20).* 341–344.

[14] Fatih Kamisli, Fabien Racapé, and Hyomin Choi. 2024. Variable-Rate Learned Image Compression with Multi-Objective Optimization and Quantization-Reconstruction Offsets. In *Data Compression Conference (DCC).* IEEE, 193–202.

[15] Rufael Mekuria, Kees Blom, and Pablo Cesar. 2017. Design, Implementation, and Evaluation of a Point Cloud Codec for Tele-Immersive Video. *IEEE Transactions on Circuits and Systems for Video Technology* 27, 4 (2017), 828–842.

[16] David Minnen, Johannes Ballé, and George D Toderici. 2018. Joint Autoregressive and Hierarchical Priors for Learned Image Compression. *Advances in Neural Information Processing Systems (NeurIPS)* 31 (2018).

[17] Ignacio Reimat, Evangelos Alexiou, Jack Jansen, Irene Viola, Shishir Subramanyam, and Pablo Cesar. 2021. CWIPC-SXR: Point Cloud dynamic human dataset for Social XR. In *Proceedings of the 12th ACM Multimedia Systems Conference (MMSys '21).*

[18] Michael Rudolph, Aron Riemenschneider, and Amr Rizk. 2024. Learned Compression of Point Cloud Geometry and Attributes in a Single Model through Multimodal Rate-Control. *arXiv preprint arXiv:2408.00599* (2024).

[19] Michael Rudolph and Amr Rizk. 2022. View-Adaptive Streaming of Point Cloud Scenes through combined Decomposition and Video-based Coding. In *Proceedings of the 1st International Workshop on Advances in Point Cloud Compression, Processing and Analysis (APCCPA '22).* ACM, 41–49.

[20] Shishir Subramanyam, Irene Viola, Alan Hanjalic, and Pablo Cesar. 2020. User Centered Adaptive Streaming of Dynamic Point Clouds with Low Complexity Tiling. In *Proceedings of the 28th ACM International Conference on Multimedia (MM '20).* 3669–3677.

[21] Jeroen van der Hooft, Tim Wauters, Filip De Turck, Christian Timmerer, and Hermann Hellwagner. 2019. Towards 6DoF HTTP Adaptive Streaming Through Point Cloud Compression. In *Proceedings of the 27th ACM International Conference on Multimedia (MM '19).* ACM, 2405–2413.

[22] Irene Viola, Jack Jansen, Shishir Subramanyam, Ignacio Reimat, and Pablo Cesar. 2023. VR2Gather: A Collaborative, Social Virtual Reality System for Adaptive, Multiparty Real-Time Communication. *IEEE MultiMedia* 30, 2 (2023), 48–59.

[23] Jianqiang Wang, Dandan Ding, Zhu Li, and Zhan Ma. 2021. Multiscale Point Cloud Geometry Compression. In *Data Compression Conference (DCC).* IEEE, 73–82.

[24] Junteng Zhang, Tong Chen, Dandan Ding, and Zhan Ma. 2023. YOGA: Yet Another Geometry-based Point Cloud Compressor. In *Proceedings of the 31st ACM International Conference on Multimedia (MM '23).* 9070–9081.

[25] Junzhe Zhang, Gexin Liu, Junteng Zhang, Dandan Ding, and Zhan Ma. 2024. DeepPCC: Learned Lossy Point Cloud Compression. *IEEE Transactions on Emerging Topics in Computational Intelligence* (2024), 1–13.

Efficient and Accurate Scene Text Recognition with Cascaded-Transformers

Savas Ozkan
savas.ozkan@samsung.com
Samsung Research
United Kingdom

Andrea Maracani
Samsung Research
United Kingdom

Hyowon Kim
Samsung Electronics
South Korea

Sijun Cho
Samsung Electronics
South Korea

Eunchung Noh
Samsung Electronics
South Korea

Jeongwon Min
Samsung Electronics
South Korea

Jung Min Cho
Samsung Electronics
South Korea

Mete Ozay
Samsung Research
United Kingdom

Abstract

In recent years, vision transformers with text decoder have demonstrated remarkable performance on Scene Text Recognition (STR) due to their ability to capture long-range dependencies and contextual relationships with high learning capacity. However, the computational and memory demands of these models are significant, limiting their deployment in resource-constrained applications. To address this challenge, we propose an efficient and accurate STR system. Specifically, we focus on improving the efficiency of encoder models by introducing a cascaded-transformers structure. This structure progressively reduces the vision token size during the encoding step, effectively eliminating redundant tokens and reducing computational cost. Our experimental results confirm that our STR system achieves comparable performance to state-of-the-art baselines while substantially decreasing computational requirements. In particular, for large-models, the accuracy remains same, 92.77 → 92.68, while computational complexity is almost halved with our structure.

CCS Concepts

• **Computing methodologies** → **Artificial intelligence**; *Computer vision.*

Keywords

Scene Text Recognition, Efficient Transformers

ACM Reference Format:
Savas Ozkan, Andrea Maracani, Hyowon Kim, Sijun Cho, Eunchung Noh, Jeongwon Min, Jung Min Cho, and Mete Ozay. 2025. Efficient and Accurate Scene Text Recognition with Cascaded-Transformers. In . ACM, New York, NY, USA, 7 pages. https://doi.org/10.1145/3712676.3719267

Permission to make digital or hard copies of all or part of this work for personal or classroom use is granted without fee provided that copies are not made or distributed for profit or commercial advantage and that copies bear this notice and the full citation on the first page. Copyrights for third-party components of this work must be honored. For all other uses, contact the owner/author(s).
MMSys'25, Stellenbosch, South Africa
© 2025 Copyright held by the owner/author(s).
ACM ISBN 979-8-4007-1467-2/25/03
https://doi.org/10.1145/3712676.3719267

1 Introduction

Scene Text Recognition (STR) is a complex task that involves interpreting and extracting textual content from natural scenes, and converting it into digital character or word sequences. This process enables to estimate high-level semantics, which is crucial for scene understanding and perception. The ability to accurately recognize and extract text from scenes has numerous applications and use cases, including document scanning, navigation, robotics, and many other products and services.

Compared to traditional Optical Character Recognition (OCR) techniques [Islam et al. 2017], which typically operate on scanned or digital documents, STR settings pose significant challenges. The presence of variations in text deformation, occlusions, and cluttered backgrounds can all undermine the performance of STR systems. To address these challenges, STR systems must be designed to be robust and adaptable to various adverse conditions and text formats. This requires the development of advanced algorithms and techniques that can effectively handle the complexities of natural scenes.

Recent breakthroughs in deep learning and computer vision have revolutionized the Scene Text Recognition (STR) domain, enabling the development of more accurate and effective systems. In particular, STR systems have widely adopted deep neural networks, such as convolutional neural networks (CNNs) [Wang et al. 2012], recurrent neural networks (RNNs) [Su and Lu 2015] and Transformers [Atienza 2021], which are trained on large datasets to handle the complexities of text recognition in natural scenes. These deep networks are designed to learn robust features from large training samples, allowing them to effectively tackle challenging conditions.

The unprecedented success of deep learning models in STR can be attributed to the fact that larger datasets and deeper networks tend to yield better performance. One promising research direction in the field of Scene Text Recognition (STR) focuses on utilizing deeper networks based on transformer architectures to effectively capture high-level details of samples in adverse scenarios [Rang et al. 2023]. These transformer-based models have been shown to generalize model predictions better than smaller networks, resulting in lower error rates and improved overall performance.

In particular, the use of transformer-based architectures allows for the effective capture of long-range dependencies and contextual

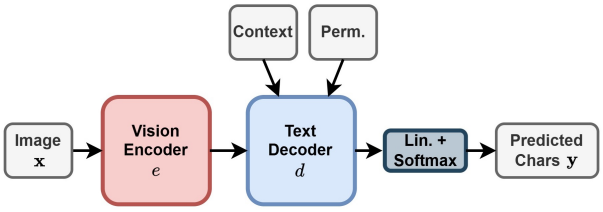


Figure 1: The overall architecture of our STR system. Our system consists of encoder e and decoder d models and predicts text characters y from an input image x.

relationships within the input data, which is critical for accurately recognizing text in complex scenes. One significant bottleneck of these deep learning models is that they require substantial computational and memory resources to operate effectively. As the models grow in complexity and size, their computational and memory demands increase exponentially, making them less practical for deployment on edge devices with limited resources. The high computational requirements of these models can lead to increased power consumption, heat generation, and latency, which can be detrimental to the overall performance and user experience.

Another notable work presents a new paradigm for the STR domain that a single vision token can be sufficient to accurately decode the text, eliminating the need for sequential tokens [Xu et al. 2024]. This proves two significant advantages for STR domain: 1) it avoids the alignment issue between vision tokens and character embeddings during decoding step. 2) it provides computational efficiency during the decoding step by allowing for adjustable vision tokens. By adjusting the number of vision tokens, the model can be optimized for computational efficiency, making it more suitable for resource-constrained devices. This is particularly important for applications where computational resources are limited, such as mobile devices or edge devices. However, it is essential to note that the computational efficiency gained by this approach may be relatively small compared to the overall computational demands of the model.

Moreover, there have been several attempts [Zhang et al. 2020] to search for the optimal Scene Text Recognition (STR) encoder by leveraging the concept of neural architecture search (NAS) [Elsken et al. 2019]. The work in this area has shown that the capacity of encoders can be effectively tuned by considering the constraints imposed by the data and model complexity. This approach enables the automatic design of encoders that are tailored to the specific requirements of the STR task, potentially leading to improved performance and efficiency. Ideally, the goal should be to develop STR models that are not only highly accurate but also efficient, scalable, and practical for real-world application and use-cases.

In this paper, we propose a STR system that aims to increase the efficiency of STR models in terms of computational complexity with a negligible accuracy drop. For this purpose, the model complexity is decreased by hierarchically selecting a subset of vision tokens and learning to discard redundant vision tokens during training step. The motivation of our system is that text content in provided images is typically sparse (i.e., a significant portion of the content covers background rather than text information), therefore there are redundant vision tokens that can be discarded in the early levels of encoder computations. To this end, the efficiency of STR system can be improved. Our contributions can be summarized as follows:

- We propose a novel cascaded-transformers structure that replaces the standard vision-transformer encoder for computer vision problems. This structure consists of a series of sub-transformers that selectively retain the most important vision tokens at each level, reducing computational requirements while preserving essential information. Details of cascaded transformers are presented in Section 3.1.
- We demonstrate its superiority on STR domain that the proposed model improves the efficiency while achieving compatible results with state-of-the-art baselines.
- Moreover, we implement a custom software setup to test our STR system on real-world images. Particularly, our system achieves high true prediction rate under highly distorted, occluded and rotated text scenarios.

This paper is organized as follows: first, we will formulate the STR problem and then we will explain the details of our STR system with novel cascaded transformers structure. Later, we will present our experimental results and conclude our paper.

2 Problem Formulation

Given a natural scene image containing text, the task of scene text recognition (STR) aims to transcribe each individual character within the text present in the image.

Formally, let $\mathcal{X} \subseteq \bigcup_{H,W} \mathbb{R}^{H\times W\times 3}$ be the space of text images and $\mathcal{Y} := \{(y_0, \ldots, y_{L-1}) : L \in \mathbb{N}^+, y_i \in C\}$ be the space of sequences of characters of strictly positive length from a character set C. We assume the existence of an underlying data distribution $\mathcal{D}$ over the set $\mathcal{X}\times\mathcal{Y}$, and we define a loss function $\ell : \mathcal{Y}\times\mathcal{Y} \to \mathbb{R}$ to measure the difference between two sequences. The objective of STR is to find a prediction function $f : \mathcal{X} \to \mathcal{Y}$ that minimizes the expected loss, $\mathbb{E}_{(\mathbf{x},\mathbf{y})\sim\mathcal{D}}[\ell(f(\mathbf{x}), \mathbf{y})]$. The function f is implemented by a deep network ϕ that is parameterized by θ and trained on a dataset $\mathcal{S} = \{(\mathbf{x}^i, \mathbf{y}^i)\}_{i=1}^M$ that contains M i.i.d. samples from $\mathcal{D}$.

3 Our Scene Text Recognition System

In our system, we employ an encoder-decoder architecture. The overall architecture is illustrated in Figure 1. The input image $\mathbf{x} \in \mathcal{X}$ is first projected into a vision tokens $\mathbf{v} \in \mathcal{V}$ comprising a sequence of N K-dimensional vision tokens via an encoder function $e : \mathcal{X} \to \mathcal{V} \subset \mathbb{R}^{N\times K}$. Later, the vision tokens $\mathbf{v}$ is mapped to a character sequence $\mathbf{y} \in \mathcal{Y}$ using a decoder function $d : \mathcal{V} \to \mathcal{Y} \subset \mathbb{R}^L$. To this end, our network is decomposed as $\phi(x) = d \circ e(x)$ and both models (i.e., encoder and decoder models) are based on transformers. Ultimately, our objective is to improve the efficiency of encoder $\mathbf{e}$ while preserving the same representation capacity of latent vision space $\mathcal{V}$ for high true prediction rates.

Decoder Model d: A conditional language model is utilized. Specifically, similar to prior works [Bautista and Atienza 2022], we leverage a Permuted-Language Decoder (PLD), which introduces a positional query stream separated by the key-value stream enabling to decode characters in different orders and incorporating Permutation Language Modeling (PLM) during training. Formally, let $\mathbf{z} = (z_1, z_2, \ldots, z_{L-1})$ be a permutation establishing an ordering among the characters and let $y_{\mathbf{z}<t}$ be the context up to current step

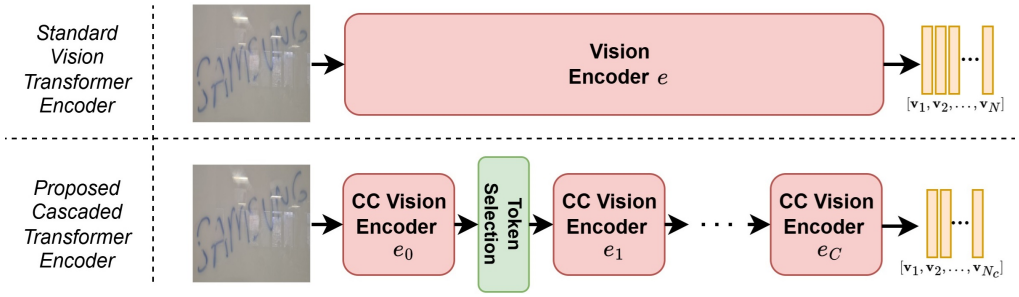


Figure 2: The flows of standard vision transformer and proposed cascaded transformers. Both models take an input image $\mathbf{x}$ and output vision tokens $\mathbf{v}$. The key component of our proposed model is a series of cascaded transformers, which reduces the number of vision tokens at each level by selectively retaining a subset of the most relevant ones for scene text recognition (i.e., $N < N_C$). Details are explain in Section 3.1.

		Blocks	Dim	Heads	Params	GFLOPs $P=8$	GFLOPs $P=16$	GFLOPs $P=32$
ENC	e-tiny	12	192	3	5.5 M	8.8	2.2	0.5
ENC	e-small	12	384	6	21.7 M	34.4	8.6	2.1
ENC	e-base	12	768	12	85.8 M	135.6	33.9	8.5
DEC	d-small	1	768	12	9.6 M	8.7	3.5	2.2
DEC	d-base	2	768	12	19.1 M	17.5	7.0	4.4

Table 1: Details of encoder (ENC) and decoder (DEC) used in our STR system. In particular, the impact of patch size P on the overall computational complexity is demonstrated.

	Acc (%) $P=8$	Acc (%) $P=16$	Acc (%) $P=32$
e-tiny + d-tiny	91.71	91.01	88.53
e-small + d-tiny	92.35	92.24	90.21
e-base + d-tiny	93.34	92.77	91.71

Table 2: Impact of patch resolution P on the word accuracy. Standard encoder and decoder models are utilized and the average word accuracy is reported for different patch resolution P.

t (i.e., already predicted characters), following the order specified by $\mathbf{z}$. Then, the decoder $d(\mathbf{z}_t, \mathbf{v}, \mathbf{y}_{\mathbf{z}<t})$ outputs a probability distribution over the character set C for predicting character at position z_t. The PLD is implemented as a block that contains two cross-attention layers and an MLP. Positional queries are used in the query stream of the decoder to specify the position to predict according to permutation $\mathbf{z}$. In the key-value stream, context is introduced in the first cross-attention layer, which is masked during training to simulate the $\mathbf{z}$ order without actually permuting the characters. Vision tokens $\mathbf{v}$ are then incorporated in the second cross-attention layer. The capacity of decoder model d can be increased using multiple cascaded blocks. We observe that when context and vision tokens are not updated in the multiple block scenario, a clear accuracy improvement can be observed. Indeed, increasing block size negatively affects the computational and memory demands.

Encoder Model e: A vision transformer architecture that is pretrained on ImageNet21k [Deng et al. 2009], is utilized. Similar to the original pipeline [Alexey 2020], an input image $\mathbf{x}$ is converted into a sequence of 2D patches whose resolution is $(P \times P)$. Hence, $N = HW/P^2$ number of patches is extracted per image. Ultimately, when N is high, computations required in self-attentions become high. On the other hand, when N is low, the representation capacity of transformers is expected to be reduced. Later, these patches are projected to patch embeddings using a linear layer and 1D position embeddings are added to retain positional information. Later, T number of multi-headed self-attentions with layer norm is used to compute vision tokens $\mathbf{v}$. In this paper, our objective is to improve the efficiency of encoder e. Unlike the standard architecture, we selectively reduce the number of patches N while computing vision tokens $\mathbf{v}$. To this end, the computational demands can be significantly decreased. Details of our contribution will be explained in 3.1.

To optimize our system (i.e., both encoder and decoder models) in the training step, we minimize the next-permuted-token prediction cross-entropy loss over random permutations as follow:

$$\min_{\theta} \mathbb{E}_{\substack{(\mathbf{x},\mathbf{y})\sim\mathcal{D} \\ \mathbf{z}\sim\mathcal{Z}}} \left[\sum_{t=1}^{L} -1(\mathbf{y}_{\mathbf{z}_t}) \cdot \log d(\mathbf{z}_t, \mathbf{v}, \mathbf{y}_{\mathbf{z}<t}) \right] \tag{1}$$

Here, the parameters of network θ are optimized by minimizing this loss function.

3.1 Efficient Cascaded-Transformers Structure

In our system, we conduct a comprehensive analysis of various encoder and decoder models with different capacities, focusing on their memory and computational complexities. Our analysis reveals that the majority of the computational demands is incurred during the computation of the vision tokens $\mathbf{v}$ from the input image $\mathbf{x}$. This suggests that the encoding process is the primary bottleneck in terms of computational efficiency.

To provide a detailed understanding of these models, we present their hyperparameters related to transformers, parameter size, and GFLOPs in Table 1. The table illustrates the impact of patch resolution P (intuitively patch size N as well) on the overall computational complexity. Also, we report the impact of patch resolution P to the word accuracy in Table 2. It is evident that the patch size is a crucial factor affecting the efficiency and accuracy of the STR system. Specifically, increasing the patch size leads to higher model complexity, while reducing its size adversely affects the word accuracy. This trade-off between patch size and word accuracy highlights the

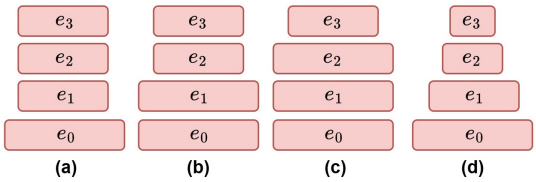


Figure 3: Four different reduction styles for cascaded transformers: (a) early-level reduction (b) mid-level reduction (c) late-level reduction, and (d) multi-level reduction.

need for a balanced approach to achieve efficient and accurate STR system.

Based on this observation, we propose a novel cascaded transformer-based structure for the encoder e. Our approach involves dividing the transformer model into several sub-models, each of which processes a subset of vision tokens in a hierarchical manner. At the output of each sub-model, a selection process is applied to choose a subset of the most relevant vision tokens, which are then passed to the next sub-model. By incrementally refining the representation of the vision tokens through this cascaded process, we aim to reduce the computational complexity of the STR system while preserving its accuracy. The selection of vision tokens at each level allows us to retain the most important information while discarding less relevant tokens, thereby decreasing the dimensionality of the data and the computational requirements. The architecture is depicted in Figure 2, which illustrates the hierarchical processing of vision tokens through the cascaded sub-models.

In our system, instead of using a single encoder model e, we employ a cascaded mechanism consisting of a series of sub-models, denoted as $[e_1, e_2, ..., e_C]$, where C represents the number of cascaded models. This design allows for the adjustment of the number of vision tokens at each level, enabling a more efficient processing pipeline. Formally, each sub-model e_i takes the previously processed vision token $\mathbf{v_{i-1}} \in \mathbb{R}^{N_{i-1} \times K}$ as input and computes the current vision tokens $\mathbf{v_i} \in \mathbb{R}^{N_i \times K}$ by $\mathbf{v_i} = e_i(\mathbf{v_{i-1}})$. Notably, the number of vision tokens is progressively reduced at each level as $N_i < N_{i-1}$ to improve the STR efficiency. This reduction in vision tokens is achieved through the selection of the most relevant information, allowing our system to focus on the most critical aspects of the input data (i.e., text content rather than background). To this end, our encoding step is formulated as follow:

$$e(x) = e_C \circ ... \circ e_2 \circ e_1(x) \tag{2}$$

In our system, we deliberately maintain the total size of the parameters unchanged as $\theta_e = \theta_{e_1} \cup \theta_{e_2} \cup ... \cup \theta_{e_C}$. This ensures that the overall parameter size remains constant, while the model complexity is optimized for efficiency.

For the selection of vision tokens, we investigate various selection schemes, including 1D local average pooling, 1D local max pooling, 2D convolution with strides and selecting the first N_i vision tokens (including the CLS token). Empirically, we find that selecting the first N_i vision tokens yields the best accuracy among the considered schemes.

Moreover, we examine the impact of the way of reducing the number of vision tokens in the cascaded structure. Specifically, we analyze whether it is more effective to reduce the number of vision tokens in the early, middle, or late levels of the cascaded structure. To this end, we conduct experiments with four different reduction styles as illustrated in Figure 3: (a) early-level reduction, (b) mid-level reduction, (c) late-level reduction, and (d) multi-level reduction. The early and mid-level reductions involve starting to reduce the number of vision tokens either in the initial and mid levels of the cascade, while the late-level reduction delays the reduction until the later levels. On the other hand, the multi-level reduction reduces the number of vision tokens uniformly across all levels. Our experiments aim to analyze which approach leads to the best trade-off between accuracy and efficiency.

4 Experiments

4.1 Datasets and Metric

Training large models for the STR domain has historically been a challenging task due to the limited availability of real-world labeled data. However, with the increasing accessibility of real data with corresponding labels, it is possible to train models on both real and synthetic datasets. The training scheme that uses both real and synthetic datasets has been shown to be sample-efficient and can lead to better accuracy, as the models can learn from the diversity of real-world and synthetic data.

Real Datasets (R) (3.3M): To train our STR model with real data, we use COCO-Text [Veit et al. 2016], RCTW17 [Shi et al. 2017], Uber-Text [Zhang et al. 2017], ArT [Chng et al. 2019], LSVT [Sun et al. 2019], MLT19 [Nayef et al. 2019], TextOCR [Singh et al. 2021], ReCTR [Zhang et al. 2019] and OpenVINO [Krylov et al. 2021]. There are several challenging cases such as low-resolution, occluded, curved and rotated text in samples. The detail analysis of these datasets is presented in [Jiang et al. 2023].

Benchmark Datasets (B) (15K): The benchmark datasets used in the training step are IIIT5K [Mishra et al. 2012], Street View Text (SVT) [Wang et al. 2011], ICDAR13 [Karatzas et al. 2013] and ICDAR15 [Karatzas et al. 2015].

Union14M Dataset (U) (3M): We use another real and large dataset to test the impact of data scaling that is collected from 15 publicly available sub-datasets and contains approximately 3M of labeled data [Jiang et al. 2023].

Synthetic Datasets (S) (6M): A subset of MJSynth (MJ) [Jaderberg et al. 2014] and SynthText (ST) [Gupta et al. 2016] is sampled to analyze the scalability of our STR model on synthetic datasets. Experimentally, we observe that a data ratio of 1 : 0.5 between real-synthetic datasets obtains the best performance.

Test Datasets: For evaluation benchmarks, we use IIIT5K [Mishra et al. 2012], CUTE-80 (C80) [Risnumawan et al. 2014], Street View Text (SVT) [Wang et al. 2011], Street View Text-Perspective (SVT-P) [Phan et al. 2013], ICDAR13 (IC13) [Karatzas et al. 2013] and ICDAR15 (IC15) [Karatzas et al. 2015]. Furthermore, we report performance on HOST and WOST [Wang et al. 2021] datasets that mostly have visually occluded examples. Lastly, more recent benchmarks, COCO-Text [Veit et al. 2016], ArT [Chng et al. 2019] and Uber-Text [Zhang et al. 2017], are also utilized.

Evaluation Metric: To evaluate our system, we follow the same procedure as presented in previous works [Rang et al. 2023; Zhao et al. 2023]. Hence, word accuracy metric is used where a predicted character sequence is accepted as correct if all characters match

	d-tiny		*d*-small	
	Acc	**GFLOPs**	**Acc**	**GFLOPs**
e-tiny	91.01	5.7	91.77	9.2
e-cc(3:9)-tiny	90.49	4.0	91.33	5.9
e-cc(6:6)-tiny	90.84	4.2	91.36	6.1
e-cc(9:3)-tiny	90.56	4.5	91.79	6.4
e-small	92.24	12.1	92.49	15.6
e-cc(3:9)-small	91.90	7.5	92.43	9.7
e-cc(6:6)-small	91.94	9.0	92.46	10.9
e-cc(9:3)-small	91.89	9.6	92.56	11.5
e-base	92.77	37.4	93.21	40.9
e-cc(3:9)-base	92.68	23.1	93.09	25.1
e-cc(6:6)-base	92.98	28.0	93.39	29.9
e-cc(9:3)-base	92.72	31.6	93.26	33.6

Table 3: Results for two-level cascaded transformers. Average word accuracy (%) on the test benchmarks is reported with the GFLOPs of overall STR system.

with the ground truth label. Average word accuracy across all 11 test benchmarks is reported. Also, floating-point operations (GFLOPs) are reported using the implementation[1].

4.2 Implementation Details

Input images are augmented by adding random Gaussian blur and noise to provide additional robustness to our system for adverse conditions. Furthermore, all images are resized to 224×224 and patch size P is set to 16 by default. Therefore, 196 vision tokens are initially computed per image. Maximum character length L is 25. During training, a set C of 94 characters is used.

For model optimization, we use AdamW [Loshchilov 2017] with a learning rate of 0.001. The training process involves a batch size of 1024 and a total of 10 epochs. Also, β_1 and β_2 are set to 0.9 and 0.95, respectively. To train Permutation Language Model (PLM), 4 random permutations are sampled in each iteration.

4.3 Experimental Results

In our experimental results, we denote the standard vision transformer encoder by *e*-model$_{id}$, where model$_{id}$ refers to the model capacity, which can be one of three variants: *tiny*, *small*, or *base* (see Table 1 for details). On the other hand, our cascaded-transformers encoder is represented by *e*-cc(i : j : ... : k)-model$_{id}$ where i, j and k specify the block size of each sub-transformer. The notation : indicates that vision token size is halved after the current transformer-block in our implementation.

4.3.1 Two-Level Cascaded Transformers. In this section, we analyze the impact of two-level cascaded transformers in terms of prediction word accuracy and model complexity. Specifically, we divide the encoder model into two cascaded transformers where $C = 2$. In table 3, we present the experimental results for different encoder and decoder model combinations. Intuitively, *e*-cc(3:9)-model$_{id}$, *e*-cc(6:6)-model$_{id}$ and *e*-cc(9:3)-model$_{id}$ represent different reduction styles as early-level, mid-level and late-level, respectively. As expected, *e*-cc(3:9)-model$_{id}$ reduces the model complexity most compared to the standard encoder. In term of accuracy, *e*-cc(6:6)-model$_{id}$ consistently obtains the best results among all configurations. For large-models (i.e., *base*), the cascaded-transformers model even outperforms the standard transformer.

[1]https://github.com/MrYxJ/calculate-flops.pytorch

	d-tiny		*d*-small	
	Acc	**GFLOPs**	**Acc**	**GFLOPs**
e-tiny	91.01	5.7	91.77	9.2
e-cc(4:4:4)-tiny	90.14	3.3	91.17	5.5
e-cc(3:3:3:3)-tiny	89.18	2.9	90.35	4.9
e-small	92.24	12.1	92.49	15.6
e-cc(4:4:4)-small	91.41	7.1	92.38	9.3
e-cc(3:3:3:3)-small	91.63	5.9	92.20	7.9
e-base	92.77	37.4	93.21	40.9
e-cc(4:4:4)-base	92.42	24.2	92.91	19.8
e-cc(3:3:3:3)-base	92.16	21.9	92.78	17.8

Table 4: Results for multi-level cascaded transformers. Average word accuracy (%) on the test benchmarks is reported with the GFLOPs of overall STR system.

	d-tiny		*d*-small	
	Acc	**Params**	**Acc**	**Params**
e-tiny	91.01	15.1 M	91.77	24.6 M
e-cc(3:18)-tiny	91.29	19.1 M	91.93	28.6 M
e-cc(6:12)-tiny	91.22	17.8 M	91.86	27.3 M
e-cc(9:6)-tiny	91.41	16.5 M	91.99	26.0 M
e-small	92.24	31.3 M	92.49	40.8 M
e-cc(3:18)-small	92.62	47.2 M	92.61	56.7 M
e-cc(6:12)-small	92.42	41.9 M	92.54	51.4 M
e-cc(9:6)-small	92.56	36.6 M	92.73	46.1 M
e-base	92.77	95.4 M	93.21	104.9 M
e-cc(3:18)-base	92.96	158.9 M	93.32	168.1 M
e-cc(6:12)-base	92.93	137.6 M	93.23	147.1 M
e-cc(9:6)-base	92.90	116.6 M	93.26	126.1 M

Table 5: Results for cascaded transformers with the same complexity of standard transformer. Average word accuracy (%) on the test benchmarks is reported with the parameter size of overall STR system.

4.3.2 Multi-Level Cascaded Transformers. In this section, we explore the extension of cascaded transformers to multiple levels, where the number of cascaded models is set to $C > 2$. This means that vision token size provided to the decoder also becomes smaller. The experimental results are reported in Table 4. Notably, *e*-cc(3:3:3:3)-model$_{id}$ has the smallest computational complexity. The results show that the performance drop can be substantial for small-capacity transformers, such as *tiny*. In contrast, for larger models, such as *base*, the multi-level cascaded transformers yield comparable results, demonstrating their potential for efficient and accurate scene text recognition.

4.3.3 Cascaded Transformers with the Same Complexity of Standard Transformer. In this section, we specifically investigate the impact of cascaded-transformers when the model complexity is same to that of standard encoders, but with an increased representation capacity. Therefore, we adjust the computational complexity of

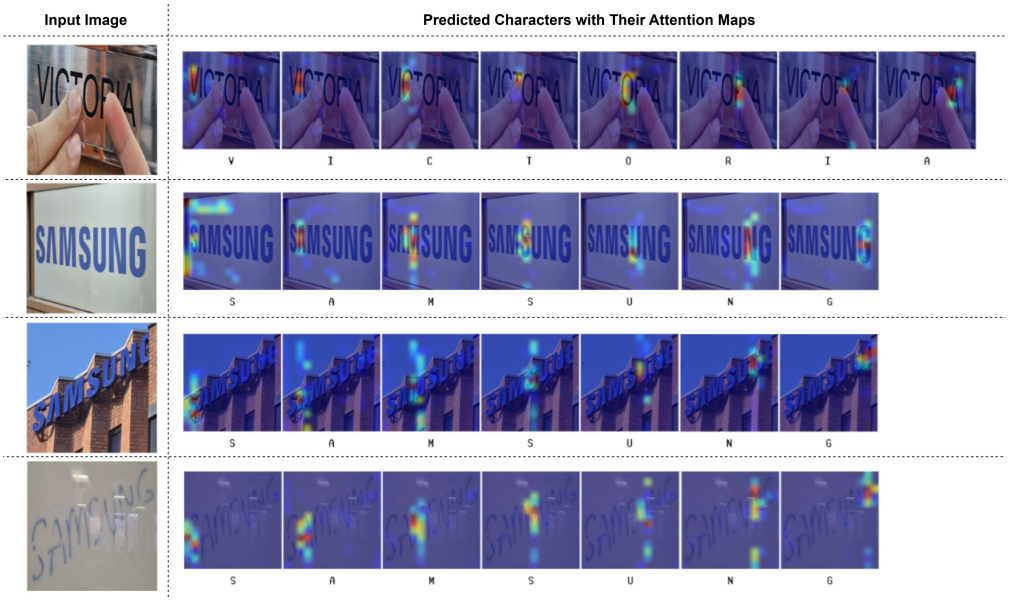


Figure 4: Prediction results of our STR system on real-world examples by visualizing character attention maps separately.

	GFLOPs	ArT	COCO	HOST	IC13 1015	IC15 1811
CLIP4STR-B	39.8	85.8	81.3	79.3	98.6	91.4
CLIP4STR-L	171.9	**86.4**	**82.7**	81.1	**99.0**	91.9
e-cc(6:6)-base+*d*-small	29.9	85.9	82.4	**83.1**	98.8	**92.1**

Table 6: Comparison with a SOTA baseline. Word accuracy (%) for different test benchmarks is reported with the GFLOPs of overall STR system.

cascaded-transformers by increasing the block size of transformers after vision token reduction. The results are reported in Table 5. The results validate that cascaded-transformers improve the accuracy compared to the standard encoders.

4.3.4 Comparison with State-of-the-Art. We compare the accuracy and computational efficiency of our system with two variants of a SOTA model, CLIP4STR, which differ in model capacity. The results are summarized in Table 6. They show that our system outperforms CLIP4STR-B model while requiring fewer computations. Moreover, our system obtains comparable results to CLIP4STR-L, but the computational requirement is reduced approximately 10x less.

4.3.5 Remarks. There are several remarks in the experiments that we need to summarize:

- The key intution behind the negligible drop in accuracy with our cascaded model is that the most crucial layers of VIT for STR are concentrated in the early layers. As a result, progressively decreasing the token sizes has a minimal impact on accuracy. Notably, similar observations have been made in language models [Lan 2019; Men et al. 2024], suggesting that the early layers of transformers are more important to extract robust representations and impact the accuracy more than other layers.
- Our cascaded model is able to learn which tokens are important and which are redundant, allowing it to adaptively focus on the most relevant information.
- Given that our model maintains the same number of layers and representation dimensionality, the memory requirements remain unchanged. Specifically, we conduct an analysis of various encoder and decoder configurations, each can be tailored to specific device requirements. In comparison to our model, which has approximately 100M parameters, a base model, CLIP4STR, has significantly larger parameter sizes, with 160M parameters for its base version and 450M parameters for its large version.

4.3.6 Results on Real-World Examples. We implement a custom software setup in Python, running on a desktop computer, to test our STR system on real-world images. In our system, a user can copy images from the web or a mobile phone into a folder, and then select the images to run the STR algorithm[2]. The outputs of character predictions with separate attention maps for each predicted character are visualized in Figure 4. Notably, test images are deliberately selected to be highly distorted, occluded and rotated, allowing us to demonstrate the robustness of our STR system for challenging conditions.

5 Conclusion

In this work, we present an efficient and accurate scene text recognition (STR) system. To improve the efficiency, we introduce a cascaded-transformers structure, which comprises multiple sub-transformers that selectively retain a subset of vision tokens at each level. Eventually, our system decreases the computational requirements by eliminating the redundancy between vision tokens. Experimental results demonstrate that with only a slight accuracy drop, our system substantially improves the efficiency of STR. Furthermore, we evaluate our STR system on real-world images using a custom software setup. We visualize that our system is robust to various adverse conditions such distortion, occlusion and rotation.

[2]Video link: https://drive.google.com/drive/folders/1OPFWi68iK0gb7EC74Yh9t3S7euwbPawI

References

Dosovitskiy Alexey. 2020. An image is worth 16x16 words: Transformers for image recognition at scale. *arXiv preprint arXiv: 2010.11929* (2020).

Rowel Atienza. 2021. Vision transformer for fast and efficient scene text recognition. In *International conference on document analysis and recognition*. Springer, 319–334.

Darwin Bautista and Rowel Atienza. 2022. Scene text recognition with permuted autoregressive sequence models. In *European conference on computer vision*. Springer, 178–196.

Chee Kheng Chng, Yuliang Liu, Yipeng Sun, Chun Chet Ng, Canjie Luo, Zihan Ni, ChuanMing Fang, Shuaitao Zhang, Junyu Han, Errui Ding, et al. 2019. Icdar2019 robust reading challenge on arbitrary-shaped text-rrc-art. In *2019 International Conference on Document Analysis and Recognition (ICDAR)*. IEEE, 1571–1576.

Jia Deng, Wei Dong, Richard Socher, Li-Jia Li, Kai Li, and Li Fei-Fei. 2009. Imagenet: A large-scale hierarchical image database. In *2009 IEEE conference on computer vision and pattern recognition*. Ieee, 248–255.

Thomas Elsken, Jan Hendrik Metzen, and Frank Hutter. 2019. Neural architecture search: A survey. *Journal of Machine Learning Research* 20, 55 (2019), 1–21.

Ankush Gupta, Andrea Vedaldi, and Andrew Zisserman. 2016. Synthetic data for text localisation in natural images. In *Proceedings of the IEEE conference on computer vision and pattern recognition*. 2315–2324.

Noman Islam, Zeeshan Islam, and Nazia Noor. 2017. A survey on optical character recognition system. *arXiv preprint arXiv:1710.05703* (2017).

Max Jaderberg, Karen Simonyan, Andrea Vedaldi, and Andrew Zisserman. 2014. Synthetic data and artificial neural networks for natural scene text recognition. *arXiv preprint arXiv:1406.2227* (2014).

Qing Jiang, Jiapeng Wang, Dezhi Peng, Chongyu Liu, and Lianwen Jin. 2023. Revisiting scene text recognition: A data perspective. In *Proceedings of the IEEE/CVF international conference on computer vision*. 20543–20554.

Dimosthenis Karatzas, Lluis Gomez-Bigorda, Anguelos Nicolaou, Suman Ghosh, Andrew Bagdanov, Masakazu Iwamura, Jiri Matas, Lukas Neumann, Vijay Ramaseshan Chandrasekhar, Shijian Lu, et al. 2015. ICDAR 2015 competition on robust reading. In *2015 13th international conference on document analysis and recognition (ICDAR)*. IEEE, 1156–1160.

Dimosthenis Karatzas, Faisal Shafait, Seiichi Uchida, Masakazu Iwamura, Lluis Gomez i Bigorda, Sergi Robles Mestre, Joan Mas, David Fernandez Mota, Jon Almazan Almazan, and Lluis Pere De Las Heras. 2013. ICDAR 2013 robust reading competition. In *2013 12th international conference on document analysis and recognition*. IEEE, 1484–1493.

Ilya Krylov, Sergei Nosov, and Vladislav Sovrasov. 2021. Open images v5 text annotation and yet another mask text spotter. In *Asian Conference on Machine Learning*. PMLR, 379–389.

Zhenzhong Lan. 2019. Albert: A lite bert for self-supervised learning of language representations. *arXiv preprint arXiv:1909.11942* (2019).

I Loshchilov. 2017. Decoupled weight decay regularization. *arXiv preprint arXiv:1711.05101* (2017).

Xin Men, Mingyu Xu, Qingyu Zhang, Bingning Wang, Hongyu Lin, Yaojie Lu, Xianpei Han, and Weipeng Chen. 2024. Shortgpt: Layers in large language models are more redundant than you expect. *arXiv preprint arXiv:2403.03853* (2024).

Anand Mishra, Karteek Alahari, and CV Jawahar. 2012. Scene text recognition using higher order language priors. In *BMVC-British machine vision conference*. BMVA.

Nibal Nayef, Yash Patel, Michal Busta, Pinaki Nath Chowdhury, Dimosthenis Karatzas, Wafa Khlif, Jiri Matas, Umapada Pal, Jean-Christophe Burie, Cheng-lin Liu, et al. 2019. Icdar2019 robust reading challenge on multi-lingual scene text detection and recognition—rrc-mlt-2019. In *2019 International conference on document analysis and recognition (ICDAR)*. IEEE, 1582–1587.

Trung Quy Phan, Palaiahnakote Shivakumara, Shangxuan Tian, and Chew Lim Tan. 2013. Recognizing text with perspective distortion in natural scenes. In *Proceedings of the IEEE international conference on computer vision*. 569–576.

Miao Rang, Zhenni Bi, Chuanjian Liu, Yunhe Wang, and Kai Han. 2023. Large OCR Model: An Empirical Study of Scaling Law for OCR. *arXiv preprint arXiv:2401.00028* (2023).

Anhar Risnumawan, Palaiahankote Shivakumara, Chee Seng Chan, and Chew Lim Tan. 2014. A robust arbitrary text detection system for natural scene images. *Expert Systems with Applications* 41, 18 (2014), 8027–8048.

Baoguang Shi, Cong Yao, Minghui Liao, Mingkun Yang, Pei Xu, Linyan Cui, Serge Belongie, Shijian Lu, and Xiang Bai. 2017. Icdar2017 competition on reading chinese text in the wild (rctw-17). In *2017 14th iapr international conference on document analysis and recognition (ICDAR)*, Vol. 1. IEEE, 1429–1434.

Amanpreet Singh, Guan Pang, Mandy Toh, Jing Huang, Wojciech Galuba, and Tal Hassner. 2021. Textocr: Towards large-scale end-to-end reasoning for arbitrary-shaped scene text. In *Proceedings of the IEEE/CVF conference on computer vision and pattern recognition*. 8802–8812.

Bolan Su and Shijian Lu. 2015. Accurate scene text recognition based on recurrent neural network. In *Computer Vision–ACCV 2014: 12th Asian Conference on Computer Vision, Singapore, November 1-5, 2014, Revised Selected Papers, Part I 12*. Springer, 35–48.

Yipeng Sun, Zihan Ni, Chee-Kheng Chng, Yuliang Liu, Canjie Luo, Chun Chet Ng, Junyu Han, Errui Ding, Jingtuo Liu, Dimosthenis Karatzas, et al. 2019. ICDAR 2019 competition on large-scale street view text with partial labeling-RRC-LSVT. In *2019 International Conference on Document Analysis and Recognition (ICDAR)*. IEEE, 1557–1562.

Andreas Veit, Tomas Matera, Lukas Neumann, Jiri Matas, and Serge Belongie. 2016. Coco-text: Dataset and benchmark for text detection and recognition in natural images. *arXiv preprint arXiv:1601.07140* (2016).

Kai Wang, Boris Babenko, and Serge Belongie. 2011. End-to-end scene text recognition. In *2011 International conference on computer vision*. IEEE, 1457–1464.

Tao Wang, David J Wu, Adam Coates, and Andrew Y Ng. 2012. End-to-end text recognition with convolutional neural networks. In *Proceedings of the 21st international conference on pattern recognition (ICPR2012)*. IEEE, 3304–3308.

Yuxin Wang, Hongtao Xie, Shancheng Fang, Jing Wang, Shenggao Zhu, and Yongdong Zhang. 2021. From two to one: A new scene text recognizer with visual language modeling network. In *Proceedings of the IEEE/CVF International Conference on Computer Vision*. 14194–14203.

Jianjun Xu, Yuxin Wang, Hongtao Xie, and Yongdong Zhang. 2024. OTE: Exploring Accurate Scene Text Recognition Using One Token. In *Proceedings of the IEEE/CVF Conference on Computer Vision and Pattern Recognition*. 28327–28336.

Hui Zhang, Quanming Yao, Mingkun Yang, Yongchao Xu, and Xiang Bai. 2020. AutoSTR: efficient backbone search for scene text recognition. In *Computer Vision–ECCV 2020: 16th European Conference, Glasgow, UK, August 23–28, 2020, Proceedings, Part XXIV 16*. Springer, 751–767.

Rui Zhang, Yongsheng Zhou, Qianyi Jiang, Qi Song, Nan Li, Kai Zhou, Lei Wang, Dong Wang, Minghui Liao, Mingkun Yang, et al. 2019. Icdar 2019 robust reading challenge on reading chinese text on signboard. In *2019 international conference on document analysis and recognition (ICDAR)*. IEEE, 1577–1581.

Ying Zhang, Lionel Gueguen, Ilya Zharkov, Peter Zhang, Keith Seifert, and Ben Kadlec. 2017. Uber-text: A large-scale dataset for optical character recognition from street-level imagery. In *SUNw: Scene Understanding Workshop-CVPR*, Vol. 2017. 5.

Shuai Zhao, Ruijie Quan, Linchao Zhu, and Yi Yang. 2023. CLIP4STR: A simple baseline for scene text recognition with pre-trained vision-language model. *arXiv preprint arXiv:2305.14014* (2023).

A Multi-CDN Playground for Dash.js: Enabling Integration of CDN Switching Strategies

Jashanjot Singh Sidhu*
Concordia University
Canada

Chidambar Joshi*
Concordia University
Canada

Abdelhak Bentaleb
Concordia University
Canada

ABSTRACT

The recent introduction of the ETSI TS 103 998 content steering standard marks a significant milestone in the evolution of media delivery for content providers. This standard simplifies the process of utilizing multiple Content Delivery Networks (CDNs), a key requirement for managing large-scale client bases. It offers clear guidelines for real-time CDN state switching, which enhances the user experience by dynamically selecting the most suitable CDN based on changing network conditions. In contrast, current industry solutions often resemble load balancing techniques, relying on heuristic approaches that are manually crafted and unable to adapt effectively to varying network environments. These static solutions frequently fail to optimize the user experience in real-time, especially when faced with diverse and unpredictable network conditions. In this paper, we utilize our previous work `StreamWise`—a DRL-based multi-CDN solution that outperforms existing state-of-the-art methods, offering superior adaptability and efficiency. We integrate this solution into the Dash.js reference player. Through a comprehensive demonstration, we illustrate how `StreamWise`, or any other multi-CDN solution, can be seamlessly incorporated into Dash.js, providing a benchmark for future developments in multi-CDN switching solutions. Experimental results in the wild demonstrate the performance of various multi-CDN solutions for Dash.js in Video on Demand (VoD) streaming mode. On average, `StreamWise` provides a ~78% improvement in VMAF compared to other solutions, while also ensuring smooth quality transitions., while also ensuring smooth quality transitions.

CCS CONCEPTS

• **Information systems → Multimedia streaming**.

KEYWORDS

Multi-CDN, Real-time CDN Selection, QoE, DASH, DRL.

ACM Reference Format:
Jashanjot Singh Sidhu, Chidambar Joshi, and Abdelhak Bentaleb. 2025. A Multi-CDN Playground for Dash.js: Enabling Integration of CDN Switching Strategies. In *ACM Multimedia Systems Conference 2025 (MMSys '25), March 31-April 4, 2025, Stellenbosch, South Africa.* ACM, New York, NY, USA, 7 pages. https://doi.org/10.1145/3712676.3719268

*The first two authors contributed equally to this work.

Permission to make digital or hard copies of part or all of this work for personal or classroom use is granted without fee provided that copies are not made or distributed for profit or commercial advantage and that copies bear this notice and the full citation on the first page. Copyrights for third-party components of this work must be honored. For all other uses, contact the owner/author(s).
MMSys '25, March 31-April 4, 2025, Stellenbosch, South Africa
© 2025 Copyright held by the owner/author(s).
ACM ISBN 979-8-4007-1467-2/2025/03
https://doi.org/10.1145/3712676.3719268

1 INTRODUCTION

The emergence of Content Delivery Networks (CDNs) [28] in 2008 marked a pivotal milestone in the delivery of global internet traffic. This breakthrough played a crucial role in driving the accelerated expansion of video streaming, which today accounts for a substantial 65% share of global internet traffic [19]. Industry leaders such as Meta, YouTube, and Netflix rely heavily on HTTP Adaptive Streaming (HAS) [4] to deliver their content to users. As viewer expectations for Quality of Experience (QoE) continue to grow—demanding higher video quality, reduced rebuffering, and seamless playback—content providers face the dual challenge of optimizing video delivery workflows to ensure a acceptable viewing experience while managing delivery costs effectively.

This is where CDNs play a critical role [27]. By strategically deploying servers across diverse geographic locations, CDNs address these challenges by efficiently managing video quality, scalability, delivery costs, and latency [28]. Their distributed architecture ensures that content is served from locations closer to viewers, reducing delays and enhancing the overall QoE. However, relying on a single CDN can undermine the ability of content providers to deliver uninterrupted playback, as it may encounter issues such as congestion, outages, and server overloads [3, 10, 16]. For example, during peak usage periods or unexpected network failures, a single CDN may struggle to manage the increased load. These disruptions can compromise the user experience and potentially breach the Service Level Agreement (SLA) guarantees of content providers, as the quality of service may fall below the agreed-upon standards.

To tackle these challenges, large content providers often leverage multi-CDN architectures [3, 12, 16, 23], distributing traffic across multiple CDN vendors to balance the load and minimize the risk of service outages. Under the *content steering standard* (ETSI TS 103 998) [1, 16], CDN switching has gained widespread popularity. Yet, existing solutions [25] only rely on heuristics, which struggle to adjust to rapidly changing network conditions and user demands. This limitation paves the way for more advanced approaches, such as `StreamWise` [6], which leverage deep reinforcement learning (DRL) to make real-time decisions, significantly enhancing the viewer QoE.

In this paper, we demonstrate the integration of `StreamWise` as a multi-CDN solution for seamless CDN switching within Dash.js [9], the industry reference player for MPEG-DASH [14]. By embedding `StreamWise` into the core of Dash.js's content steering server, our solution empowers the DASH client to make intelligent, data-driven CDN selections for every segment request, optimizing streaming performance and enhancing the viewer experience. We also introduce a GUI enhancement in Dash.js that features real-time RTT tracking for each CDN, offering valuable insights to facilitate comprehensive performance monitoring. Furthermore, we

Table 1: Comparison of multi-CDN switching solutions.

CDN Switching Type	Complexity	URL Modification	Changes	Limitations
DNS-based	Low	✗	Configure DNS settings	High switching delays
Manifest rewrite	Moderate	✓	Rewrite the manifest dynamically on the server-side	Manifest rewriting errors
Server-side	Moderate	✗	Server-side configuration changes	Delayed CDN switching
Client-side	High	✓	Client-side modifications for segment requests	Scalability
`StreamWise`	**Moderate**	✓	**Client-side modifications for segment requests**	**Single-agent DRL design**

conduct real-world experiments (in the wild) to evaluate different CDN switching strategies within multi-CDN architectures for 4K content, leveraging multiple CDN locations. The `StreamWise` integration demonstration, along with its implementation, is publicly available at [Demo] and [Code]. The main contributions of this paper are threefold:

(1) We demonstrate the seamless integration of `StreamWise` into Dash.js, offering a detailed, step-by-step framework that serves as a comprehensive guide for incorporating future learning-based or heuristic CDN switching solutions. This integration establishes a strong foundation for developing a robust benchmark for performance evaluation and comparison.
(2) We introduce a GUI enhancement for Dash.js featuring real-time RTT tracking for individual CDN locations. This functionality is valuable for performance monitoring, tracking CDN health, and can also assist CDN switching solutions in analyzing and optimizing their CDN selection decisions.
(3) We conduct real-world experiments for serving 4K content through the Dash.js player, evaluating multiple CDN switching solutions by leveraging a variety of geographically distributed CDN locations.

The rest of the paper is structured as follows: Section 2 discusses related CDN switching solutions. Section 3 presents a detailed guide for `StreamWise` integration, followed by performance evaluation in Section 4. Finally, Section 5 concludes the paper.

2 RELATED WORK

The demand for high-quality video streaming has led to the development of multi-CDN strategies [3, 12, 16] that distribute content across multiple CDNs to improve reliability, optimize performance, and reduce failures. However, the solutions for CDN switching differ in complexity and effectiveness:

(1) **DNS-based switching** [22] is simple, using a fixed video URL and switching CDNs via DNS resolution. However, it suffers from significant delays (up to several minutes) in the event of CDN failures [25], consequently degrading QoE.
(2) **Manifest rewriting** [21] enables midstream switching by modifying the content manifest. While this allows for faster switching and avoids session resets, it may introduce errors and is not entirely seamless as the server must detect CDN failures.
(3) **Server-side switching** [17, 23] centralizes decision-making, giving operators more control by analyzing client data. However, this method may add delays and does not always consider individual client conditions, relying on aggregated data.

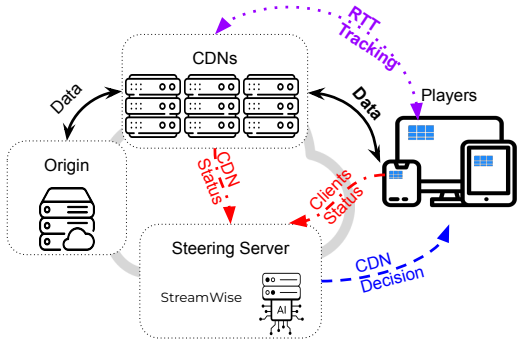


Figure 1: End-to-end workflow for multi-CDN streaming.

(4) **Client-side switching** [3, 5, 11] lets the client make switching decisions based on local performance metrics (*e.g.*, buffer level, throughput). While it offers seamless switching and more accurate QoE data, it is harder to implement at scale, especially for proprietary platforms.

Current CDN strategies often rely on basic fixed heuristics [5, 13], which struggle with fluctuating network conditions. In our work [6], we demonstrated the performance gap between heuristic-based strategies and introduced `StreamWise`—a learning-based solution for real-time CDN selection that works effectively for both on-demand and low-latency live adaptive video streaming. Table 1 summarizes the implementation differences for various multi-CDN solutions.

3 IMPLEMENTATION

Figure 1 illustrates the comprehensive video streaming architecture with a multi-CDN setup, highlighting the integration of `StreamWise` into the steering server and Dash.js reference player. The origin servers employ multiple CDNs at different locations to serve content to a client utilizing the Dash.js player. On the client side, the player parses the Media Presentation Description (MPD) file (*a.k.a. manifest*), where the presence of multiple `baseURLs` indicates that several CDN locations can be utilized for segment requests. The content steering server continuously gathers *real-time statistics* from both the clients (via CMCD [7]) and the CDNs (via CMSD [8]). The deployed CDN switching strategy leverages these metrics for optimal CDN selection. `StreamWise` is integrated at the content-steering server side and outputs the best CDN location to the player for each segment request. During this process, `StreamWise` collects the following CMCD metrics from the player side:

- **Encoded bitrate** *br*: The encoded bitrate of the audio or video segment being requested.

- **Buffer level** *bl*: The current buffer level of the player (seconds).
- **Buffer starvation** *bs*: The stall duration of the player (seconds).
- **Measured throughput** *mtp*: The throughput between the client and the server (Mbps).

These metrics are sent as *CMCD-requests* from the client to the content steering, where they are used by `StreamWise` to make data-driven decisions in selecting the best-performing CDN. We note that `StreamWise` currently leverages only client status in its design. Our aim for future extensions is to incorporate additional factors, such as CDN status, including storage and cost considerations. The working implementation of `StreamWise` is detailed in the subsequent section.

3.1 Dash.js Integration

The integration of `StreamWise` with Dash.js leverages a robust framework for dynamic CDN selection, driven by continuous real-time state monitoring and adaptive decision-making. This architecture incorporates several key algorithms to streamline the process of best CDN selection with `StreamWise`.

Algorithm 1 is the core algorithm that modifies the `baseURL` for each segment request, forming the foundation for best CDN selection and ensuring dynamic, data-driven decisions throughout the streaming process. The `setRequestURL` function (Algorithm 1) is invoked by the Dash.js player during each segment request to ensure efficient content retrieval. This function constructs the `CDNBaseURL` used to fetch content segments, depending on the request and media representation level. To maintain up-to-date CDN states, `setRequestURL` leverages the `trackState` function (defined in Algorithm 2), which refreshes each CDN state every *n* milliseconds. Here, *n* is a configurable parameter that can be adjusted based on application needs. For video-on-demand applications, `StreamWise` sets *n* to 200ms, while for low-latency live streaming, *n* is set to 100ms. Next, the `selectCDN` function (defined in Algorithm 4) is triggered at every UpdateInterval (in seconds), a customizable parameter that can be adjusted based on client, content, or application-specific requirements. This function invokes the DRL model of `StreamWise`—Actor-Critic with PPO [2, 20]—to retrieve the `CDNbaseURL` corresponding to the best CDN selection. Consequently, the media representation path of the selected CDN (i.e CDNDecision) along with the serviceLocation is set as the baseURL to form the CDNBaseURL. The implementation details of `StreamWise` can be found in our previous work [6]. Following this, the `updateRequestAttributes` function (line 6) is executed, which modifies the request URL to reflect the correct CDN URL (line 10). Additionally, it updates the request service location with the best selected CDN service location (line 11), along with any relevant query parameters (line 12). In the context of Dash.js, the CDN URL represents the selected CDN's address, while the service location refers to a mapping of the URL to multiple CDN instances, such as `cdn-a`, `cdn-b`, and `cdn-c`.

Algorithm 2 maintains track of the CDN states in real-time through 3 arrays for *rtt*, *throughput*, and *stall*, respectively. It fetches the current buffer level (line 3) and for only the currently selected CDN, it updates/initializes the arrays (lines 7-14) and for all other CDNs, it triggers the `updateCDNState` function (defined in Algorithm 3). It facilitates real-time updates of CDN states by continuously fetching the RTT, which is directly measured during periodic MPD fetch. Throughput is interpolated based on the RTT and the average segment size, estimated during MPD parsing. These RTT and throughput estimates serve as reliable indicators for estimating the future buffer size to fetch the average segment size, which is used to predict the stall duration for the player based on the current buffer level. If the RTT exceeds 1 second or the request fails,

Algorithm 1 `CDN Selection with StreamWise`

```
1:  Parameters: CDNState, CDNBaseURL, UpdateInterval
    ▷ CDNState tracks the CDN state
    ▷ CDNBaseURL stores the current CDN base URL
    ▷ UpdateTimer tracks update intervals
    ▷ request represents the current segment request, consisting of three components:
    url, which specifies the target URL for the request;
    SL, indicating the service location; and
    QP, containing the query parameters associated with the request.
    ▷ mediaRep, the current media representation level
2:  function setRequestURL(request, mediaRep)
3:      CDNState ← trackState(mediaRep)
    For every UpdateInterval,
4:      selectCDN(mediaRep, CDNState).then(baseURL):
5:      CDNBaseURL ← baseURL
6:      updateRequestAttributes(request, CDNBaseURL)
7:      return true
8:  end function

9:  function updateRequestAttributes(request, CDNBaseURL)
10:     request.url ← CDNBaseURL.url
11:     request.SL ← CDNBaseURL.SL
12:     request.QP ← CDNBaseURL.QP
13: end function
```

Algorithm 2 `StreamWise` State Tracking

```
1:  CDNState: rtt[], thr[], stall[]
    ▷ rtt[] stores RTT values for all CDNs.
    ▷ thr[] stores throughput values for all CDNs.
    ▷ stall[] stores stall times for all CDNs.
2:  function trackState(mediaRep)
3:      currentBuffer ← Get the current buffer from the DASH player
4:      if currentBuffer <= 0 then
5:          stallTime ← |currentBuffer|
6:      end if
    For selected CDN,
7:      thr[] ← [mediaRep.throughput]
8:      stall[] ← [stallTime]
9:      if first request then,
10:         Push initial values:
11:           rtt[] ← [0]
12:     else
13:         rtt[] ← Get the RTT from the DASH player
14:     end if
    For remaining CDNs,
15:     Update CDN state:
16:       updateCDNState(rtt[], thr[], stall[])
17:     return CDNState
18: end function
```

the corresponding CDN is added to a *blacklist* (line 6), preventing further requests to that CDN until it is re-evaluated and removed from the blacklist (line 12). The MPD fetch occurs every n milliseconds, as mentioned earlier. To ensure stability and reliability in decision-making, the metrics are smoothed using an exponentially weighted moving average (EWMA), which leverages the last 10 measurements from segment requests to reduce fluctuations and provide more consistent measurement.

The `CDNState` is leveraged at every `UpdateInterval` (defined in Algorithm 1) to determine the best CDN for fetching the next segment. Algorithm 4 outlines the interfacing between Dash.js (JavaScript) and the `StreamWise` DRL module (Python) at the steering server, detailing how the CDN selection decisions are made by `StreamWise` and how the corresponding request is updated in Dash.js. The `selectCDN` utility (Algorithm 4) leverages WebSockets to establish a seamless connection with the Python-based `StreamWise` DRL model. The Dash.js player initiates a WebSocket connection to the Python server, which continuously listens for incoming requests while running the `StreamWise` model to function as the content steering server. Upon receiving a request, the server evaluates the current `CDNState` and computes the optimal CDN decision. The response is then sent back to the player, which updates the segment request accordingly, as described in Algorithm 1 (lines 10-12). This setup ensures efficient and dynamic CDN selection in real time.

Figure 2 illustrates the Dash.js player streaming the BigBuckBunny [18] video on all four clients, each integrated with `StreamWise` at the content-steering server. On the right, the real-time RTT tracking GUI enhancement is shown for each player, displaying the performance metrics across three CDN locations. These locations are defined in the Dash.js content steering server as `cdn-a`: Cloudfront, `cdn-b`: Fastly, and `cdn-c`: Akamai. As depicted in Figure 2, `StreamWise` guides all four clients to make the best CDN selection decisions (highlighted in green). This demonstrates how `StreamWise` can be seamlessly integrated into Dash.js and scaled to support multiple clients efficiently. Notably, the RTT tracking is part of the GUI enhancement to track each CDN health status and not the sole metric used by `StreamWise` for CDN selection.

Algorithm 3 Real-Time CDN State Update

Require: CDNs: $C = \{C_1, C_2, ..., C_n\}$, MPD file size: s_{MPD}, Avg. segment size: s_{seg}

1: Initialize: $\forall c \in C$; Blacklist $\leftarrow \emptyset$
2: **function** `updateCDNState(rtt[], thr[], stall[])`
3: **for all** $c \in C$ **in parallel do**
4: Send MPD request to c, measure rtt
5: **if** RTT > 1s **or** request fails **then**
6: **if** $c \notin$ Blacklist **then**
7: Blacklist $\leftarrow$ Blacklist $\cup \{c\}$
8: **end if**
9: **else**
10: Interpolate rtt_c, thr_c, and stall_c based on s_{seg} and buffer
11: Update: $\text{rtt}[c] \leftarrow \text{rtt}[c] \cup \{\text{rtt}_c\}$
12: Update: $\text{thr}[c] \leftarrow \text{thr}[c] \cup \{\text{thr}_c\}$
13: Update: $\text{stall}[c] \leftarrow \text{stall}[c] \cup \{\text{stall}_c\}$
14: **if** $c \in$ Blacklist **and** RTT<=1s **then**
15: Blacklist $\leftarrow$ Blacklist $\setminus \{c\}$
16: **end if**
17: **end if**
18: **end for**
19: Smooth: $\text{rtt}_{\text{avg}}[c] \leftarrow \text{Mean}(\text{rtt}[c][-N:])$
20: Smooth: $\text{thr}_{\text{avg}}[c] \leftarrow \text{Mean}(\text{thr}[c][-N:])$
21: Smooth: $\text{stall}_{\text{avg}}[c] \leftarrow \text{Mean}(\text{stall}[c][-N:])$
22: **end function**

Algorithm 4 `CDN Selection with StreamWise`

1: **function** `selectCDN`(mediaRep, CDNState)
2: **Interface:** Initialize WebSocket connection with `StreamWise` model
3: **Python Server:**
4: Load `StreamWise` model
5: **while** connection is active **do**
6: Compute `CDNDecision` $\leftarrow$ `StreamWise` (CDNState)
7: **end while**
8: **return** `CDNDecision`
9: **end function**

10: **Dash.js:**
11: `baseURL` $\leftarrow$ mediaRep.Path
12: `baseURL.SL` $\leftarrow$ `CDNDecision.SL`
13: `baseURL.url` $\leftarrow$ `CDNDecision.baseURL`

4 PERFORMANCE EVALUATION

4.1 Methodology

To evaluate the integration of `StreamWise` with Dash.js in a real-world setup, we incorporated multiple CDN-switching solutions into Dash.js and assessed their performance during a video-on-demand streaming session lasting approximately 150 seconds.

4.1.1 Video Sample and Parameters. The video content used for testing was `3DMark Night Raid [26]` encoded using AVC with a maximum resolution of `3840x2160` and 15 bitrate levels, ranging from 100 kbps to 30 Mbps. These bitrate-resolution pairs were selected based on the methodology in [26].

4.1.2 ABR algorithm. The default Adaptive Bitrate (ABR) algorithm, Dynamic, was used throughout the session. Dynamic switches between Bola [24] and throughput-based ABR strategies.

4.1.3 CDN Locations. Since the multi-CDN setup requires the MPD for 4K content to be distributed across multiple CDN locations, and we do not have access to the default CDN locations used in the Dash.js setup, we configured our own CDN locations at various geographical locations in the testbed:

- **Location-1:** `ftp.itec.aau.at`— CDN node at the Institute of Information Technology (ITEC).
- **Location-2:** `localhost`— CDN node on a university lab computer.
- **Location-3:** `app.3krg.uk`— CDN node hosted on a UK-based server by Cloudflare.
- **Location-4:** `localhost`— CDN node on another university lab computer.

Location-2 and Location-3 are connected to the University's Wi-Fi, with Location-3 further proxying through a tunnel to a device connected to a mobile 4G LTE network. This setup represents a

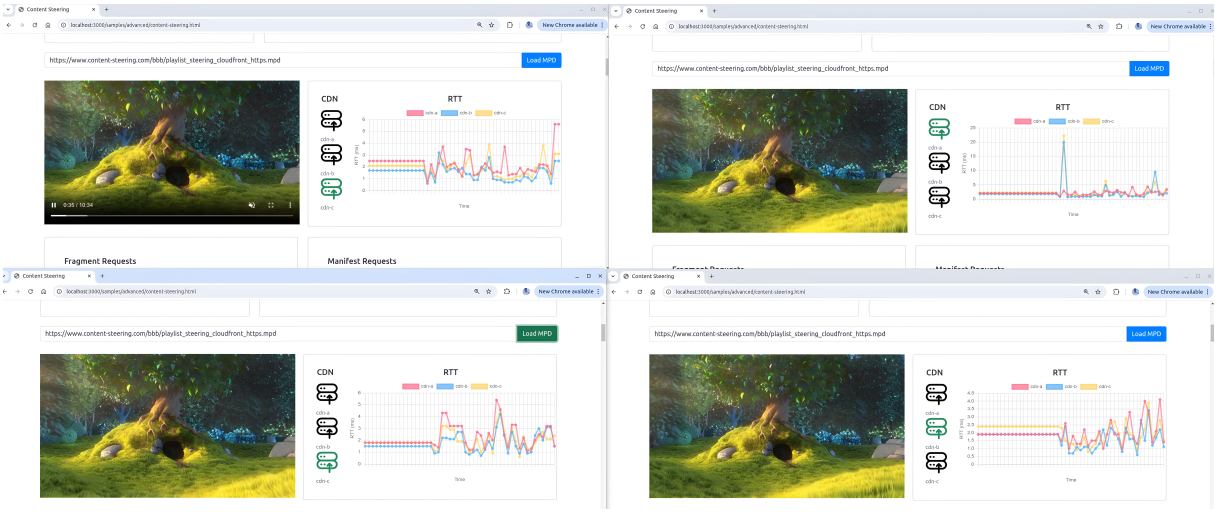


Figure 2: `StreamWise`-Based CDN switching for BigBuckBunny streaming with multiple clients using modified Dash.js players.

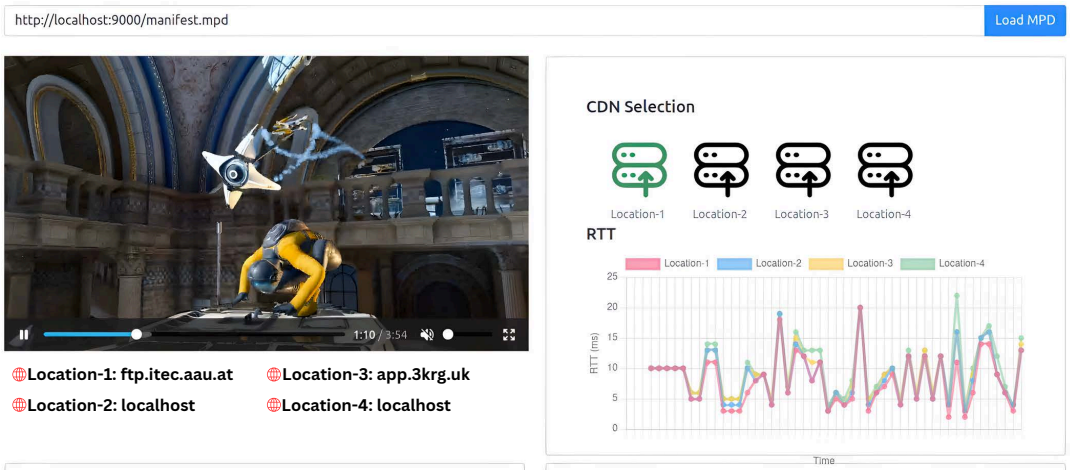


Figure 3: [Experimental setup] One client using modified Dash.js player to stream the 3DMark Night Raid video, with RTT tracking mechanism and `StreamWise` as CDN switching solution at the content steering server.

```
<BaseURL serviceLocation="Location-1">
https://ftp.itec.aau.at/datasets/mmsys22/3DMark_Night_Raid/4sec/avc/</BaseURL>
<BaseURL serviceLocation="Location-2">http://localhost:7000/</BaseURL>
<BaseURL serviceLocation="Location-3">https://app.3krg.uk/</BaseURL>
<BaseURL serviceLocation="Location-4">http://localhost:5000</BaseURL>
<ContentSteering defaultServiceLocation="Location-2"
queryBeforeStart="true">http://localhost:7000/</ContentSteering>
```

Figure 4: MPD snapshot.

real-world test environment, simulating various network conditions across different geographical locations. To make the Dash.js client aware of the updated CDN locations, we must modify the manifest file (MPD) by providing the `baseURL` for each location and specifying the `serviceLocation` name. This `serviceLocation` is used in Dash.js to map the location to the corresponding base URL. A snapshot of the updated MPD is shown in Figure 4. Note that the `ContentSteering` field in the MPD refers to the default service location, which serves as both the primary and fallback location if a segment request from the selected CDN fails.

4.1.4 CDN Switching Strategies. For a comprehensive evaluation, we integrated some of the CDN switching strategies from [6] into Dash.js along with `StreamWise`:

(1) **Default**: The default CDN Service Location in the MPD (Location-2, refer Figure 4). This strategy uses a single CDN.
(2) **DH**: A dynamic heuristic solution that uses the same RTT tracking mechanism as `StreamWise` and selects the CDN with minimal RTT.
(3) **RR**: Selects each CDN once in a round robin manner.
(4) **Det**: Allocates video segments to CDNs by a fixed ratio: 50% to CDN1, 30% to CDN2, 15% to CDN3, and 5% to CDN4..

Note that, for fairness, the location for **Default** was chosen based on the best-performing CDN location determined after multiple runs. It is important to highlight that Dash.js does not implement

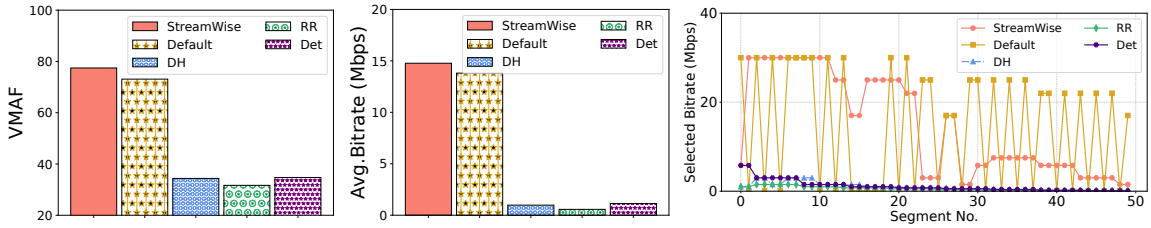


Figure 5: QoE comparison for CDN switching strategies; VMAF (left), average bitrate (center) and quality switches (right).

any CDN switching logic; instead, it consistently selects the default CDN location identified during initialization.

Figure 3 shows a Dash.js player (i.e single client) running the `3DMark Night Raid` video with `StreamWise` integrated into the content-steering server, utilizing the 4 Locations as CDNs.

4.1.5 Comparison Metrics. To evaluate the different CDN switching strategies, we analyzed metrics such as average bitrate, quality switches, and CDN switching frequency. VMAF [15] was used as a perceptual quality metric. Notably, no rebuffering events were observed, as video-on-demand sessions maintain adequate playback buffers.

4.1.6 Testbed Setup. Our testbed setup is illustrated in Figure 1 (end-to-end) and Figure 3 (client-side). Both the client and content steering server were deployed on a physical machine running Ubuntu 22.04.3 LTS, featuring an NVIDIA GA104 (GeForce RTX 3060) GPU, a 24-core CPU, and 64GB of RAM.

4.2 Results

Figure 5 presents a performance comparison of different CDN switching strategies across three key metrics: VMAF (left), average selected bitrate (center), and quality switches (right). `StreamWise` outperforms other approaches by achieving higher VMAF and average bitrate while ensuring smoother quality transitions. Its superior performance can be attributed to the intelligent DRL-based design, which dynamically adapts to changing network conditions and selects the best CDN in real time. Default (only selecting CDN Location-2) lags behind slightly in terms of VMAF and average bitrate as compared to `StreamWise`. However, it experiences frequent and abrupt quality switches, which can negatively impact the user experience by causing noticeable disruptions in playback quality. The DH strategy (dynamic heuristic) makes CDN selection decisions based solely on RTT and always selects the CDN with the least RTT. While this approach can yield favorable QoE outcomes under stable network conditions, our real-world testbed evaluation revealed its limitations. DH struggled to maintain high VMAF scores and average bitrate, resulting in suboptimal performance due to its inability to account for fluctuating network dynamics beyond RTT. RR and Det, being static heuristic approaches, failed to adapt to dynamic network conditions in real-time. As a result, their performance remained similar to DH. `StreamWise` achieves ~78% higher VMAF than the average VMAF of other CDN switching strategies.

Figure 6 presents the CDN switching frequency for `StreamWise`, RR, and Det strategies. Notably, no CDN switching was observed for the Default and DH approaches (Default consistently selected

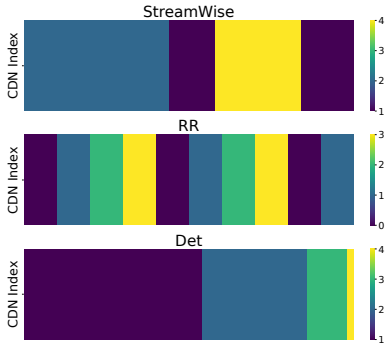


Figure 6: CDN switching frequency.

Location-2, while DH always opted for Location-1). On average `StreamWise` and Det offered ~6% CDN switching while RR had ~25% CDN switching. As seen in Figure 6, `StreamWise`, RR and Det rarely select the same CDN at any given time. This discrepancy contributes to the varied QoE outcomes observed previously.

For the integration of `StreamWise` with Dash.js, `StreamWise` utilized 14.96MB of RAM for state tracking and the inference time for each CDN decision was 0.51ms, ensuring efficient real-time adaptability with minimal resource overhead.

5 CONCLUSION

This paper presented a detailed integration of `StreamWise` with Dash.js, providing a framework through which any multi-CDN switching solution can be incorporated into Dash.js. It aims to establish a multi-CDN benchmarking environment for both heuristic and learning-based solutions. Experimental results in real-world conditions demonstrate the superior performance of `StreamWise`, underscoring that its intelligent design is well-equipped to adapt in real-time to the current network conditions and make optimal CDN selection. However, we acknowledge that further evaluation in controlled environments with more ABR algorithms and additional streaming modes is needed for conclusive results, which we earmark for future exploration.

Acknowledgments. This research is funded by the Natural Sciences & Engineering Research Council (NSERC)-Discovery Grant RGPIN-2023-04744 and NSERC Alliance (ALLRP 586434-23)-Mitacs Accelerate (IT36792) grant.

REFERENCES

[1] ETSI TS 103 998. 2024. ETSI TS 103 998 V1.1.1:DASH-IF: Content Steering for DASH. https://cdn.standards.iteh.ai/samples/67988/93d192067f4d434c9103d1b2317eb4a9/ETSI-TS-103-998-V1-1-1-2024-01-.pdf.

[2] Kai Arulkumaran, Marc Peter Deisenroth, Miles Brundage, and Anil Anthony Bharath. 2017. Deep reinforcement learning: A brief survey. *IEEE Signal Processing Magazine* 34, 6 (2017), 26–38.

[3] Abdelhak Bentaleb, Reza Farahani, Farzad Tashtarian, Hermann Hellwagner, and Roger Zimmermann. 2023. Which CDN to Download From? A Client and Server Strategies. In *Proceedings of the 2nd Mile-High Video Conference*. 135–136.

[4] Abdelhak Bentaleb, Bayan Taani, Ali C Begen, Christian Timmerer, and Roger Zimmermann. 2018. A survey on bitrate adaptation schemes for streaming media over HTTP. *IEEE Communications Surveys & Tutorials* 21, 1 (2018), 562–585.

[5] Abdelhak Bentaleb, Praveen Kumar Yadav, Wei Tsang Ooi, and Roger Zimmermann. 2020. DQ-DASH: A queuing theory approach to distributed adaptive video streaming. *ACM Transactions on Multimedia Computing, Communications, and Applications (TOMM)* 16, 1 (2020), 1–24.

[6] A. Bentaleb C. Joshi, J.S. Sidhu. 2025. StreamWise: An Intelligent Content Steering for DASH. In *Proceedings of the 16th ACM Multimedia Systems Conference (appearing online soon)*.

[7] Consumer Technology Association. 2020. CTA-5004: Web Application Video Ecosystem–Common Media Client Data.

[8] Consumer Technology Association. 2022. CTA-5006: Web Application Video Ecosystem–Common Media Server Data.

[9] DASH-IF. 2020. dash.js. https://reference.dashif.org/dash.js/. Online.

[10] Aditya Ganjam, Faisal Siddiqui, Jibin Zhan, Xi Liu, Ion Stoica, Junchen Jiang, Vyas Sekar, and Hui Zhang. 2015. C3:{internet-scale} control plane for video quality optimization. In *12th USENIX symposium on networked systems design and implementation (NSDI 15)*. 131–144.

[11] Palak Goenka, Kyriakos Zarifis, Arpit Gupta, and Matt Calder. 2022. Towards client-side active measurements without application control. *ACM SIGCOMM Computer Communication Review* 52, 1 (2022), 20–27.

[12] Burak Kara and Gwendal Simon. 2024. Power Efficient Multi-CDN Communication over Content Steering Server. In *Proceedings of the 15th ACM Multimedia Systems Conference*. 478–484.

[13] Chen Ma and Yuhong Chi. 2022. Evaluation test and improvement of load balancing algorithms of nginx. *Ieee Access* 10 (2022), 14311–14324.

[14] MPEG-DASH Standard. 2012. MPEG-DASH: Dynamic Adaptive Streaming over HTTP. https://www.mpeg.org/standards/MPEG-DASH/

[15] Reza Rassool. 2017. VMAF reproducibility: Validating a perceptual practical video quality metric. In *2017 IEEE international symposium on broadband multimedia systems and broadcasting (BMSB)*. IEEE, 1–2.

[16] Yuriy Reznik, Guillem Cabrera, Daniel Silhavy, Stefan Pham, Alex Giladi, Alex Balk, Ali C Begen, and Will Law. 2024. Content Steering: a Standard for Multi-CDN Streaming. In *Proceedings of the 3rd Mile-High Video Conference*. 128–128.

[17] Yuriy Reznik, Adam Waldron, and Guillem Cabrera. 2023. Simplifying multi-CDN delivery with HLS/DASH content steering. In *SMPTE 2023 Media Technology Summit*. SMPTE, 12–15.

[18] Ton Roosendaal. 2008. Big buck bunny. In *ACM SIGGRAPH ASIA 2008 computer animation festival*. 62–62.

[19] Sandvine. 2024. The Global Internet Phenomena Report, White Paper. *Technical Report, Sandvine* (January 2024).

[20] John Schulman, Filip Wolski, Prafulla Dhariwal, Alec Radford, and Oleg Klimov. 2017. Proximal policy optimization algorithms. *arXiv preprint arXiv:1707.06347* (2017).

[21] Robert Seeliger, Daniel Silhavy, and Stefan Arbanowski. 2017. Dynamic ad-insertion and content orchestration workflows through manifest manipulation in HLS and MPEG-DASH. In *2017 IEEE Conference on Communications and Network Security (CNS)*. IEEE, 450–455.

[22] Anees Shaikh, Renu Tewari, and Mukesh Agrawal. 2001. On the effectiveness of DNS-based server selection. In *Proceedings IEEE INFOCOM 2001. Conference on Computer Communications. Twentieth Annual Joint Conference of the IEEE Computer and Communications Society (Cat. No. 01CH37213)*, Vol. 3. IEEE, 1801–1810.

[23] Daniel Silhavy, Will Law, Stefan Pham, Ali C Begen, Alex Giladi, and Alex Balk. 2023. Dynamic CDN switching-DASH-IF content steering in dash. js. In *Proceedings of the 2nd Mile-High Video Conference*. 130–131.

[24] Kevin Spiteri, Rahul Urgaonkar, and Ramesh K Sitaraman. 2020. BOLA: Near-optimal bitrate adaptation for online videos. *IEEE/ACM transactions on networking* 28, 4 (2020), 1698–1711.

[25] Streaming Video Technology Alliance (SVTA). 2023. Investigating Approaches to Multi-CDN Delivery. https://www.svta.org/2023/01/03/investigating-approaches-to-multi-cdn-delivery/ Accessed: 2024-11-22.

[26] Babak Taraghi, Hadi Amirpour, and Christian Timmerer. 2022. Multi-codec ultra high definition 8K MPEG-DASH dataset. In *ACM MM*.

[27] Akamai Technologies. 2025. What is a CDN? https://www.akamai.com/glossary/what-is-a-cdn

[28] Behrouz Zolfaghari, Gautam Srivastava, Swapnoneel Roy, Hamid R Nemati, Fatemeh Afghah, Takeshi Koshiba, Abolfazl Razi, Khodakhast Bibak, Pinaki Mitra, and Brijesh Kumar Rai. 2020. Content delivery networks: State of the art, trends, and future roadmap. *ACM Computing Surveys (CSUR)* 53, 2 (2020), 1–34.

Continual Error Correction on Low-Resource Devices

Kirill Paramonov*
Samsung R&D Institute UK
United Kingdom

Mete Ozay
Samsung R&D Institute UK
United Kingdom

Aristeidis Mystakidis
CERTH
Greece

Nikolaos Tsalikidis
CERTH
Greece

Dimitrios Sotos
CERTH
Greece

Anastasios Drosou
CERTH
Greece

Dimitrios Tzovaras
CERTH
Greece

Hyunjun Kim
Samsung Research
South Korea

Kiseok Chang
Samsung Research
South Korea

Sangdok Mo
Samsung Research
South Korea

Namwoong Kim
Samsung Research
South Korea

Woojong Yoo
Samsung Research
South Korea

Jijoong Moon
Samsung Research
South Korea

Umberto Michieli*†
Samsung R&D Institute UK
United Kingdom

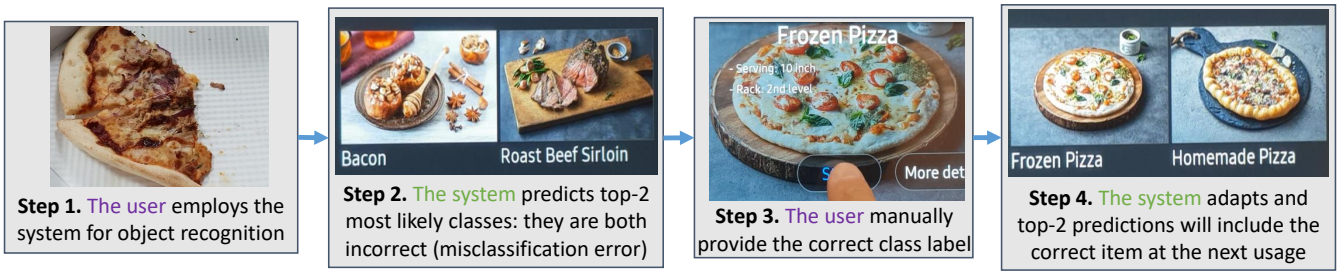


Figure 1: Proposed continual few-shot error correction system and user interactions flow in a food recognition use case.

Abstract

The proliferation of AI models in everyday devices has highlighted a critical challenge: prediction errors that degrade user experience. While existing solutions focus on error detection, they rarely provide efficient correction mechanisms, especially for resource-constrained devices. We present a novel system enabling users to correct AI misclassifications through few-shot learning, requiring minimal computational resources and storage. Our approach combines server-side foundation model training with on-device prototype-based classification, enabling efficient error correction through prototype updates rather than model retraining. The system consists of two key components: (1) a server-side pipeline that leverages knowledge distillation to transfer robust feature representations from foundation models to device-compatible architectures, and (2) a device-side mechanism that enables ultra-efficient error correction through prototype adaptation. We demonstrate our system's effectiveness on both image classification and object detection tasks, achieving over 50% error correction in one-shot scenarios on Food-101 and Flowers-102 datasets while maintaining minimal forgetting (less than 0.02%) and negligible computational overhead. Our implementation, validated through an Android demonstration app, proves the system's practicality in real-world scenarios.

CCS Concepts

• **Computing methodologies** → **Scene understanding**; **Object recognition**; **Object identification**; *Causal reasoning and diagnostics*; • **Human-centered computing** → *Ubiquitous and mobile computing systems and tools.*

Keywords

Error Correction, Continual Learning, AI Mistakes

*Both authors contributed equally to this research.
†Corresponding author: u.michieli@samsung.com.

Permission to make digital or hard copies of all or part of this work for personal or classroom use is granted without fee provided that copies are not made or distributed for profit or commercial advantage and that copies bear this notice and the full citation on the first page. Copyrights for third-party components of this work must be honored. For all other uses, contact the owner/author(s).
MMSys '25, Stellenbosch, South Africa
© 2025 Copyright held by the owner/author(s).
ACM ISBN 979-8-4007-1467-2/2025/03
https://doi.org/10.1145/3712676.3719269

ACM Reference Format:
Kirill Paramonov, Mete Ozay, Aristeidis Mystakidis, Nikolaos Tsalikidis, Dimitrios Sotos, Anastasios Drosou, Dimitrios Tzovaras, Hyunjun Kim, Kiseok Chang, Sangdok Mo, Namwoong Kim, Woojong Yoo, Jijoong Moon, and Umberto Michieli. 2025. Continual Error Correction on Low-Resource Devices. In *ACM Multimedia Systems Conference 2025 (MMSys '25), March 31-April 4, 2025, Stellenbosch, South Africa.* ACM, New York, NY, USA, 7 pages. https://doi.org/10.1145/3712676.3719269

1 Introduction

The integration of AI functionalities into everyday digital devices has grown exponentially, particularly in discriminative applications like image classification and object detection [21]. From smart home appliances to mobile devices, AI models are becoming increasingly prevalent in enhancing user experience through automated recognition, classification, and detection tasks. However, these AI models frequently make prediction errors that frustrate users and diminish product reliability [31]. This is particularly problematic in consumer devices where users expect consistent, accurate performance and have no means to correct persistent errors.

The challenge of AI model errors is compounded by several factors in real-world applications. First, the deployment environment often differs significantly from the training environment, leading to domain shift issues. Second, personal devices frequently encounter user-specific patterns and preferences that were not represented in the training data. Third, resource constraints on edge devices limit the applicability of traditional error correction approaches that require substantial computational power or storage.

Current approaches to handling AI errors primarily focus on detection rather than correction [9, 12, 23]. When correction methods exist, they typically require substantial resources: either collecting additional training data [29], finetuning models [19], or maintaining large storage capacities [7]. Such requirements make these solutions impractical for consumer devices with limited computational and storage resources. Moreover, these approaches often necessitate cloud connectivity for model updates, raising privacy concerns and limiting functionality in offline scenarios.

Furthermore, existing error correction methods often suffer from catastrophic forgetting, where correcting one type of error leads to degraded performance on previously well-handled cases. This creates a frustrating user experience where fixing one issue simply leads to another, ultimately reducing trust in the AI system. The challenge lies in developing a system that can adapt to user corrections while maintaining overall model performance and operating within device constraints.

We present a novel system that enables efficient, on-device error correction for AI models. Our solution allows users to correct model mistakes using just a few labelled examples without requiring significant computational resources or storage. For instance, as shown in Figure 1, users can correct misclassification errors with just one example (1-shot error correction), making the system highly practical for real-world applications.

Our technical approach combines three key innovations that address the challenges of on-device error correction:

- Server-side knowledge distillation from foundation models to create robust yet compact models suitable for resource-constrained devices, leveraging the power of large models while maintaining practical deployability;
- A lightweight, prototype-based classification pipeline provided with an efficient prototype update mechanism that allows continual adaptation to user corrections while preventing catastrophic forgetting;
- A practical demonstration via an Android application that showcases real-time error correction capabilities, validating our approach in practical scenarios.

Our system addresses several key requirements for actual deployment: (i) *resource efficiency*, as the system operates within the computational and storage constraints of typical consumer devices; (ii) *privacy preservation*, as all error corrections happen locally on the device, ensuring user data remains private; (iii) *ease of use*, as users can correct errors through simple interactions; (iv) *stability*, as the system maintains overall performance while adapting to user corrections; (v) *offline operation*, as the system does not require cloud connectivity.

Experimental results demonstrate the effectiveness of our system, achieving over 50% error correction accuracy in one-shot scenarios while maintaining minimal computational overhead. We validate these results through both benchmark evaluations and real-world testing via our Android demonstration app. Our experiments cover a range of scenarios, from standard image classification tasks to more complex object detection applications, showing the system's versatility across different use cases.

The rest of this paper is organized as follows: Section 2 discusses related work and existing approaches to AI error correction. Section 3 formulates the problem setup and introduces our evaluation metrics. Section 4 provides a detailed description of our system architecture, including both server-side and device-side components. Section 5 presents our experimental results and demonstration implementation. Finally, Section 6 concludes the paper.

2 Related Work

Existing work related to AI model errors can be categorized into three main approaches.

1) Error Detection and Confidence Estimation: most current solutions focus solely on analysing [27] or detecting misclassifications [7, 9, 12, 19, 23]. While these methods are effective at identifying errors, they don't provide correction mechanisms. In [17] misclassifications of trained models are reduced thanks to the addition of a regularization term in the optimization process. Related approaches [24, 25] estimate model confidence to implement quality of service policies, such as discarding low-confidence predictions. In contrast, our work assumes users can provide ground truth labels for misclassified samples and focuses on actually correcting these errors with no need for backpropagation.

2) Domain Generalization and Adaptation: several approaches attempt to improve model robustness through domain generalization at pre-training, aiming to handle multiple domains or adversarial samples [2, 6, 10, 34]. Similarly, domain adaptation techniques [3, 8, 30, 33] focus on adapting networks to domains different from

their pre-training distribution. However, these approaches typically require large amounts of labelled data and don't address the correction of individual model mistakes.

3) Distribution Shift Detection: a well-established research direction focuses on out-of-distribution detection, aiming to identify when input samples come from distributions unseen during pre-training [5, 15, 16, 32]. While these methods can identify potentially problematic inputs, they differ fundamentally from our approach as they do not provide mechanisms to adapt models and correct previous misclassifications.

Our work provides a practical and resource-efficient solution for error correction that operates directly on user devices. Unlike previous methods, our system enables immediate error correction with minimal user input while maintaining low computational and storage requirements. By focusing on prototype updates rather than model retraining, we achieve efficient personalization without the need for extensive data collection or computational resources.

3 Problem Formulation

We address the challenging problem of correcting misclassification errors in AI models deployed on resource-constrained devices. While this is a critical issue for practical AI applications, it has received limited attention in existing literature. Our focus is on discriminative AI models for computer vision applications.

3.1 Problem Setup

Our system's deployment process consists of three main phases.

Device Requirements Analysis. The process begins with analyzing the target device's capabilities and constraints. We consider:

- hardware constraints (memory, compute, power, storage);
- maximum allowable inference time;
- minimum required accuracy.

Based on these constraints, we select an appropriate model architecture M_I that offers the optimal accuracy-footprint tradeoff. Additional optimizations, such as quantization to 4-bit precision, may be applied if supported by the target hardware.

Server-side Pre-training. The selected model M_I undergoes pre-training on a training dataset $\mathcal{D}_{train}$ to produce the trained model M_T. The training samples in the dataset are indexed by $i = 1, \ldots, n$ and are composed of input images $\mathbf{X}_i$ and corresponding ground truth label $y_i \in C$, namely $\mathcal{D}_{train} = \{(\mathbf{X}_i, y_i)\}_{i=1}^{n}$. The pre-training techniques employed in our system are described in Section 4.1.

On-device Error Correction. After deployment, the model M_T processes a stream of data $\mathcal{D}_{test} = (\mathbf{X}_i^{test}, y_i^{test})_{i=1}^{n_{test}}$, where labels y_i^{test} are only available when provided by users. The test dataset naturally partitions into: (i) correctly classified samples $\mathcal{D}_C \subset \mathcal{D}_{test}$, and (ii) misclassified ones $\mathcal{D}_E \subset \mathcal{D}_{test}$, where $\mathcal{D}_E \cap \mathcal{D}_C = \emptyset$.

Users can provide ground truth labels for s samples from $\mathcal{D}_E$ for some classes, enabling few-shot error correction. The model M_T is then adapted using these annotations to produce M_A, with improved performance on previously misclassified samples. The adaptation is efficient and backpropagation-free (see Section 4.2).

3.2 Evaluation Metrics

We evaluate our system via three key metrics. We refer to *accuracy* as the percentage of correctly classified images out of the total number of images in the set. We computed:

(1) **Base Recognition Accuracy (↑):** Accuracy of M_T on the test set $\mathcal{D}_{test}$, named Acc_{base}
(2) **Error Correction Accuracy (↑):** Accuracy of M_A on the misclassified set $\mathcal{D}_E$, named Acc_E
(3) **Forgetting Rate Percentage (↓):** we compute the accuracy of the adapted model M_A on the previously correctly-classified samples $\mathcal{D}_C$, named Acc_C. Then, the forgetting rate is computed as: $For := 100 - Acc_C$.

Therefore, a successful error correction system should achieve: (i) high error correction accuracy (Acc_E), (ii) minimal user labelling requirement (small s for low-shot learning), (iii) low forgetting rate (For), (iv) controlled footprints.

Our solution ensures deterministic control over inference time and memory usage increases, with built-in mechanisms to handle resource constraints (detailed in Section 4.2).

4 Our System

Our system consists of two sequential pipelines: (1) server-side pre-training followed by (2) on-device deployment and error correction. We first present the system for image classification and then extend it to object detection in Section 4.3.

4.1 Server-side Pipeline

The server-side pipeline consists of three main stages, as illustrated in Figure 2.

4.1.1 Foundation Model Pre-training. Motivated by the recent success of large foundation vision models, we leverage the transformer-based DINO-v2 [22] model, finetuning it on domain-specific training data $\mathcal{D}_{train}$ at the server side (e.g., food data for food recognition applications). More specifically, starting from pre-trained weights (F_I), we train DINO-v2 without an output classifier (i.e., no output linear layer) using ProtoNet loss [28] to obtain F_T. Intuitively, the ProtoNet loss learns a metric space where classification can be performed by computing distances to class prototypes, providing a robust feature extractor while maintaining flexibility compared to a linear classifier head.

4.1.2 Knowledge Distillation into Small Model. Due to the limited-resource constraints on the device side, the finetuned large foundation model (F_T) may not satisfy the target footprints (e.g., model size and inference time) on the device (see Section 3.1). If this is the case, we select a smaller model M_I (e.g., the convolutional-based MobileNet-V2 [26]). We load the pre-trained weights and continue training with knowledge distillation [11] from the frozen large (teacher) model F_T over the training data $\mathcal{D}_{train}$, obtaining M_T. As the distillation loss, we use an L1 distance between the class tokens of the teacher and student models, namely:

$$\mathcal{L}_{KD} = \frac{1}{n} \sum_{i=1}^{n} |F_T(\mathbf{X}_i) - M(\mathbf{X}_i)|, \tag{1}$$

where M is initialized as M_I and trained to obtain M_T.

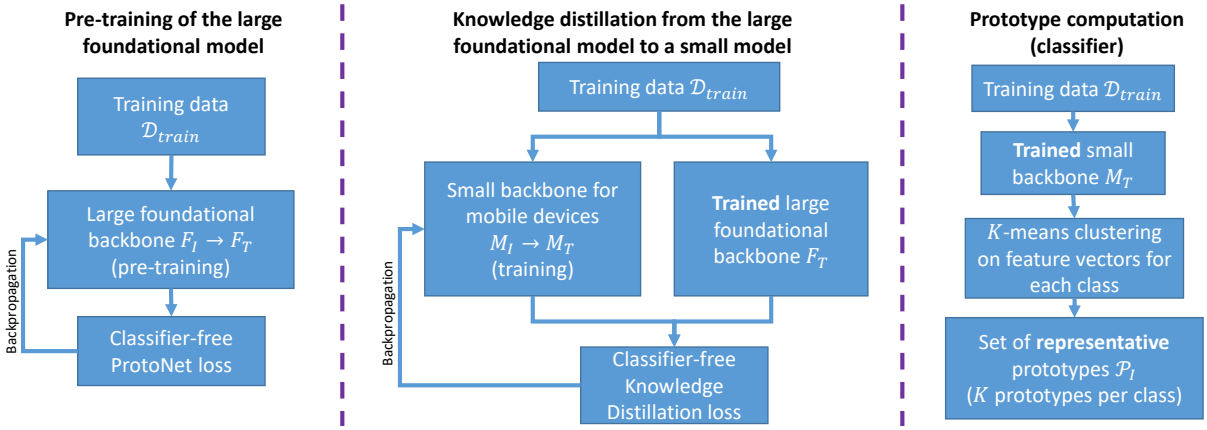


Figure 2: The server-side image classification pipeline is composed of three stages: (i) training of a large foundation model (Section 4.1.1, (ii) knowledge distillation from the trained foundation model to a smaller model (Section 4.1.2), and (iii) computation of prototypes as the classifier (Section 4.1.3).

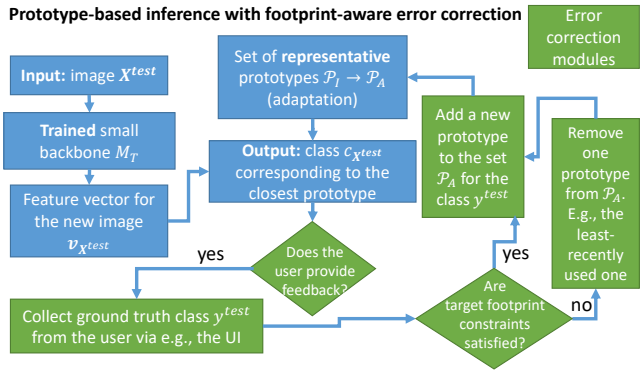


Figure 3: The device-side image classification pipeline with error correction blocks in green.

4.1.3 Prototype Computation. So far, we considered models without any classifier head. Therefore, the final stage at the server side is to compute a set of representative prototypes for each class c in $\mathcal{D}_{train}$ using the trained model M_T.

Specifically, we compute the feature vectors of every sample, $\mathbf{v}_{\mathbf{X}_i} = M_T(\mathbf{X}_i), \forall i = 1, \ldots, n$, and compute a K-means clustering for each class c. Namely, defining as $\mathcal{V}$ the set of features associated to ground truth label c, i.e., $\mathcal{V} = \{\mathbf{v}_{\mathbf{X}_i}\}_{\forall i, \text{s.t.} y_i = c}$, the objective of the K-means for class c is to find:

$$\arg\min_{\mathcal{V}} \sum_{j=1}^{K} \sum_{\mathbf{v} \in \mathcal{V}} ||\mathbf{v} - \mathbf{p}_{c,j}||^2 \tag{2}$$

where the K cluster centroids are denoted as $\mathcal{P}_c = \{\mathbf{p}_{c,j}\}_{j=1}^{K}$ and are the representative prototypes for class c. In other words, $\mathcal{P}_c$ is a set containing the K representative prototypes for class c.

The set of all representative prototypes across all classes together with their corresponding class labels is denoted as $\mathcal{P}_I = \{(\mathcal{P}_c, c)\}, \forall c \in C$.

4.2 Device-side Pipeline

The device-side pipeline is shown in Figure 3.

Classification Process. The system receives an input image $\mathbf{X}^{test}$. The image is passed through the small backbone M_T (pre-trained on the server) to obtain the feature vector $\mathbf{v}_{\mathbf{X}^{test}} = M_T(\mathbf{X}^{test})$. We initialize a new set of adapted prototypes $\mathcal{P}_A = \mathcal{P}_I$. Then, we compute the cosine distance between the feature vector $\mathbf{v}_{\mathbf{X}^{test}}$ and all prototypes $\mathbf{p}_{c,j} \in \mathcal{P}_c, \forall c \in C, \forall j = 1, \ldots, K$ from the set of representative prototypes $\mathcal{P}_A$ (constructed on the server). Finally, we select the class corresponding to the minimum cosine distance as the predicted inferred class $c_{\mathbf{X}^{test}}$. In other words:

$$c_{\mathbf{X}^{test}} := \arg\min_{c}(\cos\text{dist}(\mathbf{v}_{\mathbf{X}^{test}}, \mathcal{P}_c)). \tag{3}$$

We note that cosine distance function cos dist() computes the cosine distance between $\mathbf{v}_{\mathbf{X}^{test}}$ and all elements of $\mathcal{P}_c$.

Error Correction. If the inferred class $c_{X^{test}}$ is incorrect (i.e., the prediction is different from the ground truth y^{test}), the user could optionally decide to manually annotate the image $\mathbf{X}^{test}$ with the correct label y^{test}, via e.g., the user interface (UI). Then, our error correction system updates the array of representative prototypes:

$$\mathcal{P}_A := \{(\mathcal{P}_c, c) \in \mathcal{P}_A, \quad c \neq y^{test}\} \cup \tag{4}$$

$$\{(\text{add}(\mathcal{P}_{y^{test}}, \mathbf{v}_{\mathbf{X}^{test}}), y^{test}), \quad (\mathcal{P}_{y^{test}}, y^{test}) \in \mathcal{P}_A\}. \tag{5}$$

The first term ensures that representative prototypes of classes other than the current one are not modified. The second term adds the current feature vector $\mathbf{v}_{\mathbf{X}^{test}}$ to the current list of representative prototypes for class y^{test}, i.e., $\mathcal{P}_{y^{test}}$.

To maintain resource constraints, the system drops one prototype from the set $\mathcal{P}_A$ whenever the maximum memory or footprint target limits are hit. For example, the least-recently-used prototype can be removed. The system continually updates $\mathcal{P}_A$ based on user feedback by adding incoming user-annotated feature vectors and uses $\mathcal{P}_A$ to predict the class of any follow-up images.

4.3 Extension to Object Detection

Extending the system outlined so far to object detection can be done easily by incorporating an object detection network and image slicing mechanism based on the predicted bounding boxes.

Server-side Modifications. On the server side, we incorporate a state-of-the-art efficient object detection network (e.g., YoloV8 [13]) and finetune it to predict bounding boxes of objects of interest. In the food domain example, the object detector could be trained to identify bounding boxes corresponding to food items in the scene (i.e., one-class classification).

Device-side Implementation. On the device side, we use the finetuned model to extract the bounding boxes of the test image. Then, for each bounding box, we crop the image region and process it through the image classification network.

Implementation Trade-offs. Our current implementation uses separate networks for detection and classification, which offers maximum accuracy but higher computational cost.

To save complexity, one could use detection backbone features directly for classification. This optimization is demonstrated in [1]. The single backbone solution would reduce computational overhead at a cost of less than 5% relative accuracy. While further optimizations are possible by building prototypical inference directly on detection features, we leave this exploration for future work to maintain focus on our core contribution of error correction.

5 Experimental Results

5.1 Experimental Setup

5.1.1 Implementation Details. The server-side pipeline is implemented in Python using PyTorch, and training is conducted on a single NVIDIA GeForce RTX 3090 Ti GPU. Model quantization and initial server-side testing of the quantized models were implemented on an AMD Ryzen 9 5950X CPU. For on-device demonstrations, the model M_T is exported to ONNX and then converted to TFLite, followed by uniform 8-bit quantization. The quantized TFLite model is used with the TFLite C++ interpreter for inference. The device-side pipeline is integrated into NNTrainer [18] for deployment on Android devices. Demonstrations are conducted on a Samsung Galaxy S24 Ultra smartphone.

The complete codebase for both server-side and device-side pipelines will be open-sourced upon acceptance.

5.1.2 Models. For the Image Classification task, the transformer-based foundation model DINO-V2-small [22] is distilled into an efficient convolutional MobileNet-V2 [26]. For the Object Detection task, the YoloV8-nano model [13] is utilized.

5.1.3 Hyperparameters. For the server-side pre-training, we train DINO-v2-small for 100 epochs with a batch size of 256 and a learning rate of 1e-5. Knowledge distillation onto MobileNet-V2 is done for 100 epochs with a batch size of 256 and a learning rate of 0.1. Finally, we used YoloV8-nano [13] as the object detection model and we finetuned it on the OpenImages-V7 [14] dataset reaching 0.347 mAP50. We trained it for 200 epochs with early stopping enabled with patience 20 epochs, using a batch size of 16 and a learning rate of 0.01. For the server-side prototypes computation, we set $K = 3$, i.e., we store the three most representative prototypes per class.

Table 1: Error correction accuracy and forgetting rate of our system on two image classification datasets.

	Food-101		Flowers-102	
# shots s	$Acc_E \uparrow$	$For \downarrow$	$Acc_E \uparrow$	$For \downarrow$
1	51.1%	0.018%	54.3%	0.011%
2	67.3%	0.053%	68.9%	0.038%
3	75.0%	0.118%	78.4%	0.094%
4	79.9%	0.157%	81.1%	0.114%
5	83.2%	0.179%	85.7%	0.162%
7	87.3%	0.202%	89.0%	0.179%
10	91.0%	0.237%	92.6%	0.195%
20	93.2%	0.447%	96.7%	0.364%
50	96.4%	0.870%	98.8%	0.577%

5.1.4 Datasets. We used two datasets: the Food-101 [4] and the Oxford Flowers-102 [20], containing 101 and 102 classes, respectively. Both datasets represent domain-specific applications (e.g., food or flower recognition), where the user may want to provide the correct label to improve the AI model. As these datasets are out-of-domain for DINO-V2, domain finetuning is required. Each dataset is split into 70% training, 15% validation, and 15% testing per class; i.e., we preserve 70% of the images of each class in the training set. Few-shot error correction experiments are conducted with $s \in \{1, 2, 3, 4, 5, 7, 10, 20, 50\}$ annotated samples per class.

5.2 Image Classification Results

Quantitative results of our system are reported in Table 1 in terms of error correction accuracy Acc_E and forgetting rate For for the image classification task. The base recognition accuracy Acc_{base} of the M_T model with initial representation prototypes P_I is 90.6% and 94.3% on the Food-101 and the Flowers-102 datasets, respectively.

Our system allows user to correct most of the misclassification errors with limited feedback. For example, correcting 51.1% of classification errors on Food-101 requires the user to annotate just one sample per class. At the same time, only 0.018% of previously-corrected samples are now misclassified by the adapted model.

5.2.1 Ablation Studies.

Prototypes vs. Linear Classifier Head. First, we checked that using a fully connected linear classifier head does not produce higher base classification accuracy (91.1% on Food-101 and 93.6% on Flowers-102) than prototypical inference (90.6% and 94.3%, respectively). Furthermore, fully-connected layers are less adaptable, and mistakes are harder to correct with efficient updates.

Server-side Pipeline. Table 2 highlights the performance of different server-side pre-training strategies on Food-101. Using only the small model finetuned on domain-specific data, i.e., M_T, results in low Acc_E due to limited feature richness. Using the pre-trained foundation model F_I gives a good balance between error correction accuracy and forgetting, nonetheless the performance on the base classes is quite poor as the model is pre-trained on a large

Table 2: Ablation studies on the server-side pipeline on the Food-101 dataset. Acc_{base} for all approaches is (from left to right): 82.6, 76.4, 92.4, and 90.6%.

	M_T only		F_I		F_T		Ours	
# shots s	$Acc_E \uparrow$	$For \downarrow$	$Acc_E \uparrow$	$For \downarrow$	$Acc_E \uparrow$	$For \downarrow$	$Acc_E \uparrow$	$For \downarrow$
1	0.6%	0.1%	48.7%	0.1%	15.1%	0.1%	51.1%	0.0%
3	7.0%	0.1%	72.3%	0.3%	48.3%	0.1%	75.0%	0.1%
5	13.2%	0.3%	82.9%	0.4%	57.3%	0.1%	83.2%	0.2%
10	21.7%	0.6%	91.2%	0.6%	65.4%	0.2%	91.0%	0.2%

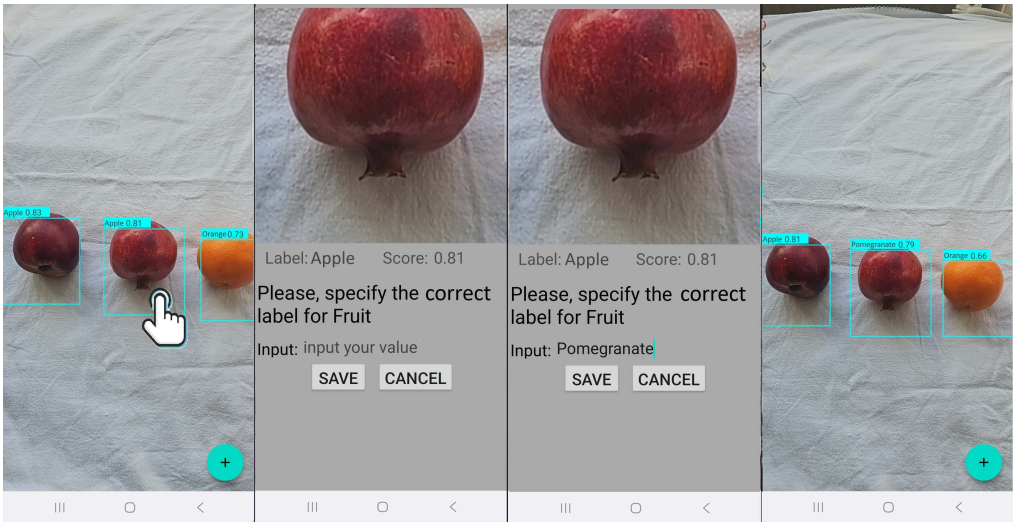


Figure 4: Application workflow of our demo: (1) the user identifies misclassified objects and clicks anywhere within their bounding box, (2-3) the user provides the correct label, and (4) the user observes improved classification in subsequent iterations.

generic dataset and not domain-finetuned. Additionally, the foundation model is typically an order of magnitude larger than the selected small model. The finetuned foundation model F_T overfits to the domain-specific data, achieving high Acc_{base} but poor adaptability to correct mistakes efficiently. Finally, our approach strikes an optimal balance, sacrificing only a small amount of error recognition accuracy (2.2%) to achieve significantly improved error correction capability and minimal forgetting, all while being more computationally efficient than foundation model-based methods.

5.3 On-device Demonstration

In our demo, we focus on the object detection task due to its practical value where users point a camera to target objects. Both MobileNet-V2 and YoloV8-nano are quantized to 8 bits, achieving a $\sim$ 3× inference speedup with negligible accuracy degradation (<1%) compared to the 32-bit models. For instance, on a Samsung Galaxy S24, YoloV8-nano achieves 40 ms inference time (compared to 110 ms for 32-bit) while maintaining a mAP@50 of 0.347 (from 0.350). The 8-bit model has a compact size of 3.2 MB, enabling real-time inference on consumer devices.

Overall, the system runs in real-time on consumer smartphones. Figure 4 illustrates the application workflow. In the first screenshot, the user can benefit from the detection model to identify some food items. In case of a misclassification error, the user may decide to provide the correct ground truth: in our app, the user should click on the bounding box and enter the correct label (second and third screenshots). Finally, the system updates its internal prototypical representations and will correctly identify the item at subsequent iterations (third screenshot).

6 Conclusion

In this work, we introduced a simple yet effective system for few-shot continual error correction in discriminative AI models, such as those used for object recognition. Our approach empowers end users to optionally and seamlessly correct misclassification errors in AI models, enhancing their overall utility and reliability. The proposed system comprises a server-side pre-training stage to develop a robust feature extractor while adhering to the resource constraints of target devices, such as memory, storage, and inference time. Additionally, we designed a novel on-device mechanism for few-shot continual error correction, which efficiently updates the stored prototypical representations. Our system represents a step forward in enabling pervasive AI applications, setting new benchmarks for accuracy, adaptability, and practicality in consumer products.

References

[1] Francesco Barbato, Umberto Michieli, Jijoong Moon, Pietro Zanuttigh, and Mete Ozay. 2024. Cross-architecture auxiliary feature space translation for efficient few-shot personalized object detection. In *2024 IEEE/RSJ International Conference on Intelligent Robots and Systems (IROS)*. IEEE, 8587–8594.

[2] Francesco Barbato, Umberto Michieli, Mehmet Kerim Yucel, Pietro Zanuttigh, and Mete Ozay. 2024. A Modular System for Enhanced Robustness of Multimedia Understanding Networks via Deep Parametric Estimation. In *Proceedings of the 15th ACM Multimedia Systems Conference* (Bari, Italy) *(MMSys '24)*. Association for Computing Machinery, New York, NY, USA, 190âĂŞ201.

[3] Shai Ben-David, John Blitzer, Koby Crammer, and Fernando Pereira. 2006. Analysis of representations for domain adaptation. *Advances in neural information processing systems* 19 (2006).

[4] Lukas Bossard, Matthieu Guillaumin, and Luc Van Gool. 2014. Food-101–mining discriminative components with random forests. In *Computer vision–ECCV 2014: 13th European conference, zurich, Switzerland, September 6-12, 2014, proceedings, part VI 13*. Springer, 446–461.

[5] Elena Camuffo, Umberto Michieli, Simone Milani, Jijoong Moon, and Mete Ozay. 2024. Enhanced Model Robustness to Input Corruptions by Per-corruption Adaptation of Normalization Statistics. In *2024 IEEE/RSJ International Conference on Intelligent Robots and Systems (IROS)*. IEEE, 11558–11565.

[6] Guangyao Chen, Peixi Peng, Li Ma, Jia Li, Lin Du, and Yonghong Tian. 2021. Amplitude-phase recombination: Rethinking robustness of convolutional neural networks in frequency domain. In *Proceedings of the IEEE/CVF International Conference on Computer Vision*. 458–467.

[7] Charles Corbière, Nicolas Thome, Avner Bar-Hen, Matthieu Cord, and Patrick Pérez. 2019. Addressing failure prediction by learning model confidence. *Advances in Neural Information Processing Systems* 32 (2019).

[8] Yaroslav Ganin and Victor Lempitsky. 2015. Unsupervised domain adaptation by backpropagation. In *International conference on machine learning*. PMLR, 1180–1189.

[9] Dan Hendrycks and Kevin Gimpel. 2016. A baseline for detecting misclassified and out-of-distribution examples in neural networks. *arXiv preprint arXiv:1610.02136* (2016).

[10] Dan Hendrycks, Norman Mu, Ekin D Cubuk, Barret Zoph, Justin Gilmer, and Balaji Lakshminarayanan. 2019. Augmix: A simple data processing method to improve robustness and uncertainty. *arXiv preprint arXiv:1912.02781* (2019).

[11] Geoffrey Hinton. 2015. Distilling the Knowledge in a Neural Network. *arXiv preprint arXiv:1503.02531* (2015).

[12] Heinrich Jiang, Been Kim, Melody Guan, and Maya Gupta. 2018. To trust or not to trust a classifier. *Advances in neural information processing systems* 31 (2018).

[13] Glenn Jocher, Ayush Chaurasia, and Jing Qiu. 2023. Ultralytics YOLOv8. https://github.com/ultralytics/ultralytics

[14] Alina Kuznetsova, Hassan Rom, Neil Alldrin, Jasper Uijlings, Ivan Krasin, Jordi Pont-Tuset, Shahab Kamali, Stefan Popov, Matteo Malloci, Alexander Kolesnikov, et al. 2020. The open images dataset v4: Unified image classification, object detection, and visual relationship detection at scale. *International journal of computer vision* 128, 7 (2020), 1956–1981.

[15] Shiyu Liang, Yixuan Li, and Rayadurgam Srikant. 2017. Enhancing the reliability of out-of-distribution image detection in neural networks. *arXiv preprint arXiv:1706.02690* (2017).

[16] Weitang Liu, Xiaoyun Wang, John Owens, and Yixuan Li. 2020. Energy-based out-of-distribution detection. *Advances in neural information processing systems* 33 (2020), 21464–21475.

[17] Elyes Manai, Mohamed Mejri, and Jaouhar Fattahi. 2024. Minimizing Model Misclassification Using Regularized Loss Interpretability. In *2024 16th IIAI International Congress on Advanced Applied Informatics (IIAI-AAI)*. IEEE, 231–236.

[18] Jijoong Moon, Hyun Suk Lee, Jiho Chu, Donghak Park, Seungbaek Hong, Hyungjun Seo, Donghyeon Jeong, Sungsik Kong, and MyungJoo Ham. 2024. A New Frontier of AI: On-Device AI Training and Personalization. In *2024 IEEE/ACM 46th International Conference on Software Engineering: Software Engineering in Practice (ICSE-SEIP)*. IEEE Computer Society, 323–333.

[19] Makaela Nartker, Zhenglong Zhou, and Chaz Firestone. 2023. When will AI misclassify? Intuiting failures on natural images. *Journal of Vision* 23, 4 (2023), 4–4.

[20] Maria-Elena Nilsback and Andrew Zisserman. 2008. Automated flower classification over a large number of classes. In *2008 Sixth Indian conference on computer vision, graphics & image processing*. IEEE, 722–729.

[21] Numalis. 2023. How AI is revolutionizing the home appliance industry. *Communication team* (2023).

[22] Maxime Oquab, Timothée Darcet, Théo Moutakanni, Huy Vo, Marc Szafraniec, Vasil Khalidov, Pierre Fernandez, Daniel Haziza, Francisco Massa, Alaaeldin El-Nouby, et al. 2023. Dinov2: Learning robust visual features without supervision. *arXiv preprint arXiv:2304.07193* (2023).

[23] Xin Qiu and Risto Miikkulainen. 2022. Detecting misclassification errors in neural networks with a gaussian process model. In *Proceedings of the AAAI Conference on Artificial Intelligence*, Vol. 36. 8017–8027.

[24] Haoxuan Qu, Lin Geng Foo, Yanchao Li, and Jun Liu. 2023. Towards more reliable confidence estimation. *IEEE Transactions on Pattern Analysis and Machine Intelligence* (2023).

[25] Haoxuan Qu, Yanchao Li, Lin Geng Foo, Jason Kuen, Jiuxiang Gu, and Jun Liu. 2022. Improving the reliability for confidence estimation. In *European Conference on Computer Vision*. Springer, 391–408.

[26] Mark Sandler, Andrew Howard, Menglong Zhu, Andrey Zhmoginov, and Liang-Chieh Chen. 2018. Mobilenetv2: Inverted residuals and linear bottlenecks. In *Proceedings of the IEEE conference on computer vision and pattern recognition*. 4510–4520.

[27] Daniel Sikar, Artur Garcez, Robin Bloomfield, Tillman Weyde, Kaleem Peeroo, Naman Singh, Maeve Hutchinson, Dany Laksono, and Mirela Reljan-Delaney. 2024. The Misclassification Likelihood Matrix: Some Classes Are More Likely To Be Misclassified Than Others. *arXiv preprint arXiv:2407.07818* (2024).

[28] Jake Snell, Kevin Swersky, and Richard Zemel. 2017. Prototypical networks for few-shot learning. *Advances in neural information processing systems* 30 (2017).

[29] Nathan TeBlunthuis, Valerie Hase, and Chung-Hong Chan. 2024. Misclassification in automated content analysis causes bias in regression. Can we fix it? Yes we can! *Communication Methods and Measures* (2024), 1–22.

[30] Marco Toldo, Andrea Maracani, Umberto Michieli, and Pietro Zanuttigh. 2020. Unsupervised domain adaptation in semantic segmentation: a review. *Technologies* 8, 2 (2020), 35.

[31] Kemal Toprak UĞar. 2023. A Comprehensive Guide to Error Analysis in Machine Learning. *Medium* (2023).

[32] Jingkang Yang, Kaiyang Zhou, Yixuan Li, and Ziwei Liu. 2024. Generalized out-of-distribution detection: A survey. *International Journal of Computer Vision* 132, 12 (2024), 5635–5662.

[33] Kaichao You, Mingsheng Long, Zhangjie Cao, Jianmin Wang, and Michael I Jordan. 2019. Universal domain adaptation. In *Proceedings of the IEEE/CVF conference on computer vision and pattern recognition*. 2720–2729.

[34] Kaiyang Zhou, Ziwei Liu, Yu Qiao, Tao Xiang, and Chen Change Loy. 2022. Domain generalization: A survey. *IEEE Transactions on Pattern Analysis and Machine Intelligence* 45, 4 (2022), 4396–4415.

Received 20 February 2007; revised 12 March 2009; accepted 5 June 2009

3D Gaussian-based Immersive Media Streaming in Networked Extended Reality

Yuang Shi
yuangshi@u.nus.edu
National University of Singapore
IPAL, IRL2955
Singapore

ABSTRACT

3D Gaussian Splatting (3DGS) has emerged as a revolutionary representation for immersive media, offering unprecedented visual quality and efficient rendering. However, streaming 3DGS content presents unique challenges due to its complex parameter space and tight coupling between primitives. This doctoral research aims to address these challenges through three main themes: (i) developing scalable 3DGS representations that enable progressive transmission and flexible rendering, (ii) creating comprehensive quality assessment frameworks that consider both spatial-temporal coherence and multi-attribute distortions, and (iii) designing robust error control strategies for real-time streaming applications. Our research ambitions to advance the state-of-the-art in immersive media representation and delivery, with potential applications in virtual reality, telepresence, and digital twin systems.

CCS CONCEPTS

• **Information systems** → **Multimedia streaming**; • **Computing methodologies** → **Mixed / augmented reality**.

KEYWORDS

Extended Reality, Immersive Media Streaming, 3D Gaussian Splatting

ACM Reference Format:
Yuang Shi. 2025. 3D Gaussian-based Immersive Media Streaming in Networked Extended Reality. In *ACM Multimedia Systems Conference 2025 (MMSys '25), March 31-April 4, 2025, Stellenbosch, South Africa.* ACM, New York, NY, USA, 5 pages. https://doi.org/10.1145/3712676.3719673

1 INTRODUCTION

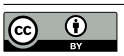
Motivation and Context. Dynamic 3D content allows users to explore environments captured at different locations and times as if they were physically present, offering a level of interaction that surpasses traditional 2D media. The increasing demand for immersive virtual and augmented reality experiences drives the need for more advanced 3D media streaming technologies. However, dynamic 3D content presents significant challenges, requiring high computational power, storage, and bandwidth, which makes real-time streaming over the Internet for remote and interactive use highly complex. This raises a critical question for commercially deployed 3D streaming systems:

> *How can 3D immersive media be streamed efficiently under complex and interleaving constraints without compromising the user's quality of experience?*

Specifically, immersive media streaming systems operate in environments characterized by heterogeneous access conditions and diverse quality requirements. These systems must address several fundamental constraints related to network conditions, device capabilities, and quality demands. First, network constraints represent a primary challenge, as variable bandwidth availability, unpredictable latency, and potential packet loss significantly affect transmission reliability and quality. Second, real-time streaming applications require robust adaptation mechanisms to maintain service continuity under these varying network conditions. Furthermore, device heterogeneity introduces additional complexity, as end-user devices range from mobile phones to high-end VR headsets, each presenting diverse computational capabilities, display resolutions, and power constraints. This heterogeneity requires adaptive content delivery mechanisms to ensure optimal performance across all platforms. Quality requirements encompass both objective Quality of Service (QoS) metrics and subjective Quality of Experience (QoE) parameters. Critical factors include maintaining visual quality, maintaining temporal consistency, and optimizing interactive responsiveness. These requirements must be balanced according to the available resources and system constraints.

To address these challenges, streaming systems must incorporate specific technical capabilities in both representation and delivery. The 3D representation must support multiresolution encoding and progressive transmission while maintaining efficient compression and enabling view-dependent optimization. The system architecture must facilitate dynamic bitrate adaptation, implement error resilience mechanisms, and support scalable delivery while maintaining real-time processing capabilities for interactive experiences. Figure 1 outlines the fundamental stages involved in 3D media streaming. As shown, the 3D content is captured and uploaded to a server which is in charge of delivering it to the client at a given quality and bitrate level, depending on the network and device constraints. Several modules are available to aid in the content delivery. For instance, the content might be pre-processed with 3D tiling or down-sampling to exploit occlusions and be post-processed with super-resolution to improve the visual quality.

3D Representations. Unlike traditional 2D video, which has a standardized and mature format, 3D volumetric video encompasses various representations such as point cloud and 3D mesh. Each format presents unique challenges for streaming. Among these,



This work is licensed under a Creative Commons Attribution 4.0 International License.
MMSys '25, March 31-April 4, 2025, Stellenbosch, South Africa
© 2025 Copyright held by the owner/author(s).
ACM ISBN 979-8-4007-1467-2/2025/03.
https://doi.org/10.1145/3712676.3719673

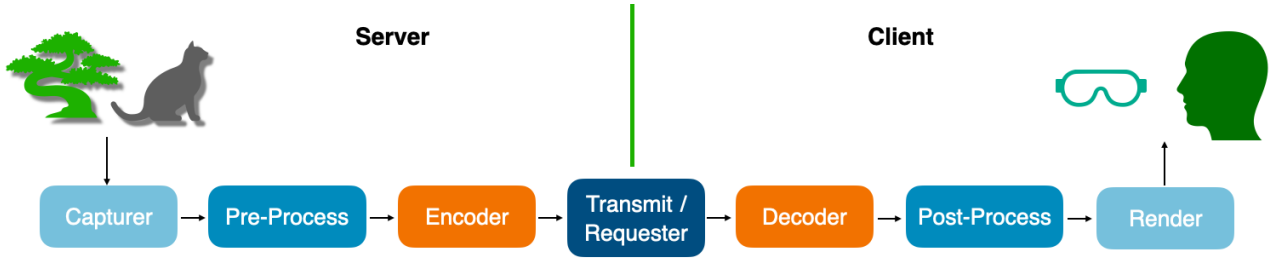


Figure 1: A general framework for 3D media streaming.



(a) Point Cloud. (b) 3DGS Model. (c) 3DGS — Ellipsoids.

Figure 2: Visual quality comparison between point cloud and 3DGS. (a) Point cloud shows inherent visual artifacts due to its discrete nature. (b) 3DGS achieves superior visual quality with fewer primitives. (c) The underlying ellipsoids of 3DGS demonstrate its sophisticated geometric representation.

point cloud, which represents 3D geometry and texture through a collection of 3D points, is a significant representation of volumetric videos. Point cloud is especially suited for creating digital twins and dynamic content due to the characteristics of flexibility and simplicity, the growing availability of capture devices, and the efficient compression standards [9, 12, 13, 17, 20].

Although point cloud offers simplicity and direct capture capability, it often struggles to achieve high-quality results due to the sparse and irregular nature of the data. These limitations often result in visual artifacts, holes in surfaces, and poor handling of view-dependent effects [6, 18, 26]. As shown in Figure 2(a), these artifacts impact the visual experience, requiring dense point sampling and adaptive rendering techniques to achieve acceptable quality.

Meanwhile, recent breakthroughs in learning-based 3D representation have opened exciting new possibilities. In particular, 3D Gaussian Splatting (3DGS) [6] has emerged as a revolutionary learning-based representation that enables unprecedented visual quality and efficient rendering. 3DGS transcends the limitations of point cloud by representing scenes using optimized 3D Gaussians (ellipsoids), each characterized by spatial properties (3D position, covariance matrix) and appearance attributes (opacity, spherical harmonics coefficients). Unlike point cloud where each point carries only position and color information, 3DGS employs sophisticated Gaussian primitives that capture both local geometry and view-dependent appearance through learnable parameters. Figures 2(b) and 2(c) show the example of the rendered images from 3DGS model and its ellipsoids, respectively.

The advantages of 3DGS are compelling for immersive applications. While point cloud often requires millions of points to represent fine details, 3DGS achieves superior visual quality with a more compact representation requiring less storage. The Gaussian-based representation also maintains consistent and photorealistic quality across different viewpoints, which is a crucial feature for immersive experiences where visual artifacts can break user engagement. Furthermore, its real-time rendering capabilities on consumer hardware make it ideal for interactive applications.

Research Domain. While most existing research focuses on the reconstruction quality and rendering speed of 3D Gaussians, little has been done on streaming them under real-world network constraints [22]. Although techniques developed for other 3D representations such as point cloud or 3D mesh have shown success in handling streaming problems, 3D Gaussian introduces unique challenges that make these existing models and techniques insufficient. Specifically, unlike conventional point cloud, 3DGS employs heterogeneous Gaussians parameterized by ellipsoid shape and view-dependent color. These attributes are optimized iteratively to minimize discrepancies between rendered and ground-truth images, leading to a nonuniform and highly interdependent distribution of Gaussians. This tightly coupled representation optimizes a set of Gaussians, thus significantly limiting the effectiveness of 3D downsampling or partitioning techniques, which often result in substantial degradation of the visual quality. Moreover, these unique attributes significantly affect the perceptual quality of Gaussian-based content, leading to different types of artifacts that are not well understood by current perceptual quality models.

Thus, new techniques must be developed to address the specific challenges posed by this novel representation in the context of immersive media streaming. Our research aims to pioneer these solutions, developing AI-enhanced streaming techniques that maintain the revolutionary benefits of 3DGS while enabling efficient transmission over real-world networks. This research combines deep learning, computer graphics, and networking expertise to advance the state-of-the-art in immersive media representation and delivery, with potential applications spanning virtual reality, telepresence, and digital twin systems.

2 BACKGROUND

The 3D Gaussian splatting (3DGS) technique [6] is a method for representing 3D scenes and rendering photorealistic radiance fields in real-time. Here is a summary of the key aspects of 3DGS.

Parameters. A 3DGS model consists of a collection of 3D Gaussians. Each 3D Gaussian in the representation is defined by a center position μ and a 3D covariance matrix $\mathbf{\Sigma}$. The covariance matrix is decomposed into a scaling matrix $\mathbf{S}$ and a rotation matrix $\mathbf{R}$ for optimization purposes. The color of each Gaussian is represented by its opacity α and a set of Spherical Harmonic (SH) coefficients $\mathbf{K}$. SH is a basis for functions defined over the sphere, and it allows for modeling view-dependent colors.

Rendering. The rendering process starts by projecting each 3D Gaussian splat onto a 2D plane using a covariance matrix transformation. The resulting 2D covariance matrix $\mathbf{\Sigma}'$ is calculated in camera coordinates. Anisotropic splats are then α-blended based on their depths to compute the final color of each pixel using an α-compositing technique.

Optimization. To accurately represent a scene, an optimization step is performed to refine the parameters of 3DGS. The optimization is based on differentiable rendering and aims to fit the textures of the given scene. The initialization of the 3D Gaussian set is done using a point cloud generated by Structure-from-Motion (SfM) [19]. Gradient-descent-based optimization, interleaved with adaptive refinement, is used to refine the parameters and control the density of the 3DGS representation. The optimization minimizes the loss between the rendered images and ground truth using a combination of $\mathcal{L}_1$ loss and SSIM loss. Additionally, point densification and pruning techniques are employed to control the density of the 3DGS representation by adaptively increasing the density of splats and removing less impactful splats based on opacity and size.

3 RESEARCH THEMES AND GOALS

RT1: Scalable 3DGS Representation. The 3D representation forms the foundation of immersive media streaming systems, playing a crucial role in determining system performance and capabilities. The current 3DGS representation faces several challenges when considered for streaming applications. These include substantial data sizes for high-quality scenes, a lack of inherent scalability for level-of-detail (LOD) rendering, an all-or-nothing rendering requirement, and complex dependencies between Gaussian primitives. Our research on scalable representation design focuses on developing hierarchical organizations of Gaussian primitives that enable progressive transmission and flexible rendering, which aims to address the fundamental tension between visual quality and streaming efficiency.

Recent efforts have been made to build hierarchical 3DGS representations based on multi-resolution space partitioning to render very large scenes in real time [7, 8, 11]. However, these existing methods are primarily designed for reconstruction quality and rendering speed, without considering the challenges of adaptive streaming [15]. Space-based hierarchies in 3DGS feature discrete quality layers, each independently represented by a set of anchor splats. This approach requires substantial computational resources for maintenance and navigation, especially with frequent level transitions, and limits view-adaptive rendering.

We aim to create layered structures that organize the 3D data into multiple LOD specifically optimized for adaptive streaming. In contrast to existing hierarchical representations, the layered representation must meet specific requirements:

- *Base Layer Completeness*. Given subsets of Gaussian splats, the model should represent complete 3D content with reduced details, allowing for efficient base layer transmission.
- *Progressive Enhancement*. Higher detail levels should inherently share visual information with lower levels to minimize redundancy.
- *Continuous Quality Transitions*. The representation should support smooth quality changes without discrete jumps.
- *View-dependent Optimization*. The layered structure should facilitate view-adaptive streaming by prioritizing visible Gaussians.

By leveraging shared visual characteristics across different levels, this layered representation can reduce transmission overhead while enabling responsive quality adaptation for interactive experiences.

RT2: Quality Assessment and Modelling. Despite the plethora of quality assessment approaches for traditional 2D media, extending these methodologies to immersive 3D content presents unique challenges [1, 2, 4, 14, 16, 25]. This is particularly non-trivial for 3DGS, where the complex relationship between representation parameters and perceived quality requires new assessment approaches.

Current evaluation methods for 3DGS primarily focus on *reconstruction quality* and *rendering speed*, neglecting the crucial aspects of immersive experience from the *consumption perspective*. The dominant evaluation approach relies on naive image-based metrics. Specifically, the 3DGS content is projected into 2D images based on the testing viewpoints which are uniformly selected from the image set. The quality of 3DGS content is then assessed by averaging the scores across viewpoints which are measured with 2D metrics such as PSNR, SSIM, and LPIPS.

This naive image-based evaluation has several limitations:

- *Spatial Inconsistency*. The local 2D-based optimization of 3DGS construction introduces unique artifacts, including inconsistencies of 3D geometry and discontinuities of view-dependent appearance. Such global inconsistency can significantly degrade the immersive user experience.
- *View Dependency*. Quality scores highly depend on the choice of evaluation viewpoints, including the selection of camera positions, camera parameters, number of viewpoints, and pooling applied across different views.
- *Interaction Complexity*. The increased DoF in 3D interaction introduces new quality factors: *e.g.*, mode of interaction (free navigation vs. guided viewing), impact of rendering parameters, or attribute-specific distortions.

We plan to explore the impact of these unique aspects through two primary research directions. First, we will study the interplay among the distortions of different attributes of 3DGS on visual quality, and how they affect the rate-distortion relationship. Second, we will focus on spatial and temporal coherence analysis. This research direction aims to develop comprehensive metrics that capture consistency across viewpoint transitions while analyzing the intricate relationship between Gaussian parameters and temporal stability. We will quantify how different navigation patterns impact perceived quality and create novel assessment methods that account for realistic view trajectories. This analysis will provide

crucial insights into maintaining quality consistency during the interactive exploration of 3DGS content.

Building on these findings, we will develop a comprehensive QoE model that accounts for both individual and combined effects of attribute distortions, view-dependent quality variations, and temporal aspects of quality perception. The culmination of this work will be the creation of objective metrics that effectively correlate with subjective quality assessments, providing a robust framework for evaluating 3DGS streaming quality.

RT3: Error Control for 3D Gaussian Streams. Recent works on rapid and high-fidelity human avatar modeling with 3DGS [3, 23, 24] have demonstrated the potential of applying 3DGS for telepresence, virtual meetings, and interactive VR applications. For these real-time communications scenarios, dynamic 3DGS content is typically streamed over Real-Time Transport Protocol (RTP) to achieve smooth interactions with low latency. Unlike streaming over Dynamic Adaptive HTTP Streaming (DASH), which operates over reliable TCP/QUIC transport protocols, RTP prioritizes low latency over reliability, making the streams susceptible to information losses ranging from 5% to 15% in typical internet conditions [5].

The impact of information losses on dynamic 3DGS quality differs significantly from traditional media formats due to the unique characteristics of the 3DGS representation. When packets containing Gaussian parameters are lost, the effects can propagate both spatially and temporally due to the interdependent nature of the representation. Therefore, understanding these complex error patterns is crucial for developing effective error control strategies.

We will conduct systematic experiments to quantify how information losses affect different attributes of 3DGS content under various network conditions and viewing scenarios. Based on these experimental insights, we will then develop tailored error concealment approaches for dynamic 3DGS streaming. Our strategy leverages both traditional error concealment techniques and modern learning-based methods. For spatial error concealment, we will explore interpolation/inpainting methods that consider the geometric relationships between Gaussian primitives. Temporal concealment will utilize motion prediction and parameter tracking to maintain consistency across frames.

4 CURRENT PROGRESS

RT1: Scalable 3DGS Representation. Recently, we propose a layered progressive 3D Gaussian representation LapisGS [15], which is the first solution to adapt static 3D Gaussian contents for adaptive streaming and progressive rendering.

Figure 3 shows the overall framework. The core of our approach is a layered structure for cumulative representation, where each layer adds additional details to the existing base layers, progressively refining the representation. To force coherence among layers while avoiding re-encoding lower layers, we incorporate dynamic opacity optimization during training, allowing for selective adjustment of layer contributions for optimal visual fidelity. We then utilize an occupancy map to track and exclude less important Gaussian splats during streaming and rendering, improving computational and storage efficiency. Our approach reduces data size and additionally enhances computational efficiency by managing splats dynamically with occupancy maps, achieving high-quality

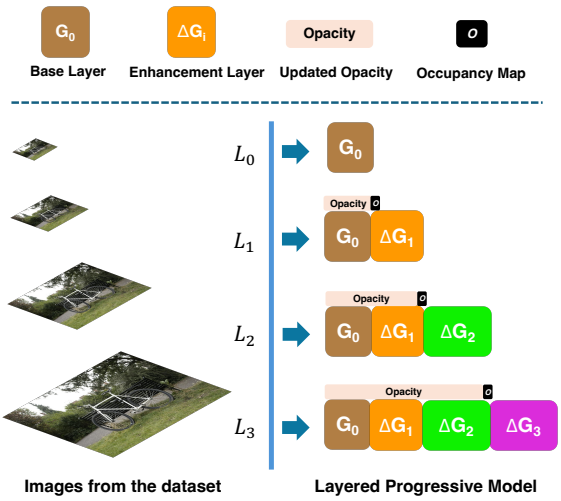


Figure 3: The overview of LapisGS.

rendering and low resource cost, with up to 50.71% improvement in SSIM, 286.53% improvement in LPIPS, and 318.41% reduction in model size.

As our approach is optimized for static content, we further extended it to handle dynamic elements, following the work from Luiten et al. [10]. We then integrate it into an end-to-end system based on the DASH standard for streaming live and on-demand dynamic 3DGS scenes over diverse and dynamic networks [21], which is the first of its kind. In live streaming, our system could achieve up to 99.70% reduction in missing frames on average and deliver a maximum PSNR improvement of 10.08 dB. In on-demand streaming, our system could reduce the freeze time by up to 92.01%, and increase the synthesized view quality by up to 5.14 dB in PSNR and 0.11 in SSIM.

5 CONCLUSION

In this doctoral proposal, we outline our vision for 3DGS-based immersive media streaming. To better understand and address key challenges in streaming static and dynamic 3DGS content under real-world network conditions, we target three research themes including scalable representation, quality assessment, and error control. The proposed work explores practical solutions that consider both the unique characteristics of 3DGS and the constraints of streaming applications. The current progress on scalable representation is also presented. We expect our findings to provide useful insights for the multimedia systems community in developing more efficient streaming solutions for emerging 3D representations.

ACKNOWLEDGMENTS

This work is supervised by Prof. Wei Tsang Ooi from the National University of Singapore, and collaborated with Prof. Cheng-Hsin Hsu from the National Tsing Hua University, and Prof. Géraldine Morin and Prof. Simone Gasparini from the IRIT — Université de Toulouse. I appreciate their significant guidance and support for my research.

REFERENCES

[1] Evangelos Alexiou, Yana Nehmé, Emin Zerman, Irene Viola, Guillaume Lavoué, Ali Ak, Aljosa Smolic, Patrick Le Callet, and Pablo Cesar. Chapter 18 - subjective and objective quality assessment for volumetric video. In *Immersive Video Technologies*, pages 501–552. Academic Press, 2023.

[2] Evangelos Alexiou, Irene Viola, Tomás M Borges, Tiago A Fonseca, Ricardo L De Queiroz, and Touradj Ebrahimi. A comprehensive study of the rate-distortion performance in MPEG point cloud compression. *APSIPA Transactions on Signal and Information Processing*, 8:e27, Nov 2019.

[3] Yufan Chen, Lizhen Wang, Qijing Li, Hongjiang Xiao, Shengping Zhang, Hongxun Yao, and Yebin Liu. MonoGaussianAvatar: Monocular Gaussian point-based head avatar. In *ACM SIGGRAPH 2024 Conference Papers, SIGGRAPH 2024, Denver, CO, USA, 27 July 2024- 1 August 2024*, page 58. ACM, 2024.

[4] Samuel Rhys Cox, May Lim, and Wei Tsang Ooi. VOLVQAD: an MPEG V-PCC volumetric video quality assessment dataset. In *Proceedings of the 14th Conference on ACM Multimedia Systems, MMSys 2023, Vancouver, BC, Canada, June 7-10, 2023*, pages 357–362. ACM, 2023.

[5] Tzu-Kuan Hung, I-Chun Huang, Samuel Rhys Cox, Wei Tsang Ooi, and Cheng-Hsin Hsu. Error concealment of dynamic 3D point cloud streaming. In *MM '22: The 30th ACM International Conference on Multimedia, Lisboa, Portugal, October 10 - 14, 2022*, pages 3134–3142. ACM, 2022.

[6] Bernhard Kerbl, Georgios Kopanas, Thomas Leimkühler, and George Drettakis. 3d gaussian splatting for real-time radiance field rendering. *ACM Transactions on Graphics*, 42(4):139:1–139:14, 2023.

[7] Bernhard Kerbl, Andreas Meuleman, Georgios Kopanas, Michael Wimmer, Alexandre Lanvin, and George Drettakis. A hierarchical 3d gaussian representation for real-time rendering of very large datasets. *ACM Transactions on Graphics*, 43(4):62:1–62:15, 2024.

[8] Yang Liu, Chuanchen Luo, Lue Fan, Naiyan Wang, Junran Peng, and Zhaoxiang Zhang. CityGaussian: Real-time high-quality large-scale scene rendering with Gaussians. In *Computer Vision - ECCV 2024 - 18th European Conference, Milan, Italy, September 29-October 4, 2024, Proceedings, Part XVI*, volume 15074 of *Lecture Notes in Computer Science*, pages 265–282. Springer, 2024.

[9] Zhi Liu, Qiyue Li, Xianfu Chen, Celimuge Wu, Susumu Ishihara, Jie Li, and Yusheng Ji. Point cloud video streaming: Challenges and solutions. *IEEE Network*, 35(5):202–209, 2021.

[10] Jonathon Luiten, Georgios Kopanas, Bastian Leibe, and Deva Ramanan. Dynamic 3D Gaussians: Tracking by persistent dynamic view synthesis. In *The 11st International Conference on 3D Vision, 3DV 2024, Davos, Switzerland, March 18-21, 2024*, pages 800–809. IEEE, 2024.

[11] Kerui Ren, Lihan Jiang, Tao Lu, Mulin Yu, Linning Xu, Zhangkai Ni, and Bo Dai. Octree-GS: Towards consistent real-time rendering with LOD-structured 3D Gaussians. *CoRR*, abs/2403.17898, 2024.

[12] Sebastian Schwarz, Marius Preda, Vittorio Baroncini, Madhukar Budagavi, Pablo César, Philip A. Chou, Robert A. Cohen, Maja Krivokuca, Sebastien Lasserre, Zhu Li, Joan Llach, Khaled Mammou, Rufael Mekuria, Ohji Nakagami, Ernestasia Siahaan, Ali J. Tabatabai, Alexis M. Tourapis, and Vladyslav Zakharchenko. Emerging MPEG standards for point cloud compression. *IEEE Journal on Emerging and Selected Topics in Circuits and Systems*, 9(1):133–148, 2019.

[13] Yuang Shi, Bennett Clement, and Wei Tsang Ooi. QV4: QoE-based viewpoint-aware V-PCC-encoded volumetric video streaming. In *Proceedings of the 15th ACM Multimedia Systems Conference, MMSys 2024, Bari, Italy, April 15-18, 2024*, pages 144–154. ACM, 2024.

[14] Yuang Shi, Samuel Rhys Cox, and Wei Tsang Ooi. Quality assessment and modeling for MPEG V-PCC volumetric video. In *Proceedings of the 16th International Workshop on Immersive Mixed and Virtual Environment Systems, MMVE 2024, Bari, Italy, April 15-18, 2024*, pages 64–70. ACM, 2024.

[15] Yuang Shi, Géraldine Morin, Simone Gasparini, and Wei Tsang Ooi. LapisGS: Layered Progressive 3D Gaussian Splatting for Adaptive Streaming. In *The 12th International Conference on 3D Vision, 3DV 2025, Singapore, March 25-28, 2025*. IEEE, 2025.

[16] Yuang Shi and Wei Tsang Ooi. Perceptual impact of facial quality in MPEG V-PCC-encoded volumetric videos. In *Proceedings of the 16th International Workshop on Immersive Mixed and Virtual Environment Systems, MMVE 2024, Bari, Italy, April 15-18, 2024*, pages 71–74. ACM, 2024.

[17] Yuang Shi, Pranav Venkatram, Yifan Ding, and Wei Tsang Ooi. Enabling low bit-rate MPEG v-pcc-encoded volumetric video streaming with 3d sub-sampling. In *Proceedings of the 14th Conference on ACM Multimedia Systems, MMSys 2023, Vancouver, BC, Canada, June 7-10, 2023*, pages 108–118. ACM, 2023.

[18] Yuang Shi, Ruoyu Zhao, Simone Gasparini, Géraldine Morin, and Wei Tsang Ooi. Volumetric video compression through neural-based representation. In *Proceedings of the 16th International Workshop on Immersive Mixed and Virtual Environment Systems, MMVE 2024, Bari, Italy, April 15-18, 2024*, pages 85–91. ACM, 2024.

[19] Noah Snavely, Steven M. Seitz, and Richard Szeliski. Photo tourism: exploring photo collections in 3D. *ACM Transactions on Graphics*, 25(3):835–846, 2006.

[20] Shishir Subramanyam, Irene Viola, Alan Hanjalic, and Pablo César. User centered adaptive streaming of dynamic point clouds with low complexity tiling. In *MM '20: The 28th ACM International Conference on Multimedia, Virtual Event / Seattle, WA, USA, October 12-16, 2020*, pages 3669–3677. ACM, 2020.

[21] Yuan-Chun Sun, Yuang Shi, Cheng-Tse Lee, Mufeng Zhu, Wei Tsang Ooi, Yao Liu, Chun-Ying Huang, and Cheng-Hsin Hsu. LTS: A DASH streaming system for dynamic multi-layer 3D Gaussian splatting scenes. In *Proceedings of the 16th ACM Multimedia Systems Conference, MMSys 2025, Stellenbosch, South Africa, March 31-April 4, 2025*. ACM, 2025.

[22] Yuan-Chun Sun, Yuang Shi, Wei Tsang Ooi, Chun-Ying Huang, and Cheng-Hsin Hsu. Multi-frame bitrate allocation of dynamic 3D Gaussian splatting streaming over dynamic networks. In *Proceedings of the 2024 SIGCOMM Workshop on Emerging Multimedia Systems, EMS 2024, Sydney, NSW, Australia, August 4-8, 2024*. ACM, 2024.

[23] Jing Wen, Xiaoming Zhao, Zhongzheng Ren, Alexander G. Schwing, and Shenlong Wang. GoMAvatar: Efficient animatable human modeling from monocular video using Gaussians-on-Mesh. In *IEEE/CVF Conference on Computer Vision and Pattern Recognition, CVPR 2024, Seattle, WA, USA, June 16-22, 2024*, pages 2059–2069. IEEE, 2024.

[24] Ye Yuan, Xueting Li, Yangyi Huang, Shalini De Mello, Koki Nagano, Jan Kautz, and Umar Iqbal. GAvatar: Animatable 3D Gaussian avatars with implicit mesh learning. In *IEEE/CVF Conference on Computer Vision and Pattern Recognition, CVPR 2024, Seattle, WA, USA, June 16-22, 2024*, pages 896–905. IEEE, 2024.

[25] Xuemei Zhou, Irene Viola, Yunlu Chen, Jiahuan Pei, and Pablo César. Deciphering perceptual quality in colored point cloud: Prioritizing geometry or texture distortion? In *Proceedings of the 32nd ACM International Conference on Multimedia, MM 2024, Melbourne, VIC, Australia, 28 October 2024 - 1 November 2024*, pages 7813–7822. ACM, 2024.

[26] Matthias Zwicker, Hanspeter Pfister, Jeroen van Baar, and Markus H. Gross. EWA volume splatting. In *The 12th IEEE Visualization Conference, IEEE Vis 2001, San Diego, CA, USA, October 24-26, 2001, Proceedings*, pages 29–36. IEEE Computer Society, 2001.

Author index

Abadie, Laurence, 214
Abdulwahab, Saddam, 214
Adve, Sarita, 101, 124
Alface, Patrice Rondao, 113
Amer, Ihab, 68
Arbanowski, Stefan, 242, 317

Barone, Nunzio, 311
Barrett, John, 305
Bentaleb, Abdelhak, 35, 57, 221, 298, 342
Bhattacharya, Abhinav, 291
Bhattacharya, Arani, 201
Brescia, Walter, 311
Brocard, Christian, 214
Broll, Wolfgang, 291

Cardoso, Luís Fernando de Souza, 291
Chambon, Sylvie, 214
Chang, Chia-Yu, 113
Chang, Kiseok, 349
Chaudhary, Shubham, 201
Chen, Guodong, 79
Chen, Shujiao, 284
Chen, Xinlei, 12
Cho, Jung Min, 335
Cho, Sijun, 335
Choi, Yi King, 148
Cisternino, Ivan, 311
Coquillard, Charles-Edouard, 214
Crouzil, Alain, 214

Dang, Fan, 12
Das, Dibakar, 101
Dasari, Mallesham, 79
De Cicco, Luca, 311
De Fiore, Fabrizio, 305
De Fré, Matthias, 113
De Schepper, Koen, 113
De Turck, Filip, 113
De Vleeschauwer, Danny, 113
Dóka, János, 170
Dougherty, Alan William, 148
Drosou, Anastasios, 349

Fischer, Alexander, 242
Formanek, Bence, 170
Fréneau, Louis, 235

Gambhir, Keshav, 201
Gao, Steven, 101, 124
Gautam, Sushant, 228, 249, 277
Gautier, Guillaume, 235
Ghaddar, Moustafa, 242
Gilligan, Shane, 322
Godfrey, Brighten, 101, 124
Gomez-Barquero, David, 170
Gonzalez-Aguirre, David, 101
Güclü, Görkem, 242

Hácha, Filip, 79
Halepovic, Emir, 90
Halvorsen, Pål, 228, 249, 277
Hartmann, Thierry, 214
Hefeeda, Mohamed, 68
Heidarirad, Hadi, 256
Hooft, Jeroen van der, 113
Hossfeld, Tobias, 90
Hsu, Cheng-Hsin, 46, 136
Hu, Yueyu, 24
Hu, Zhiming, 159
Huang, Chun-Ying, 136
Huang, Zhiming, 270
Huzaifa, Muhammad, 101

Immohr, Felix, 291

Jalabert, Grégory, 214
Jiang, Qinjun, 101, 124
Jin, Xinqi, 12
Jocha, Dávid, 170
Joshi, Chidambar, 35, 342

Karagioules, Theo, 90
Keighrey, Conor, 305, 322
Khorasgani, Salar Hosseini, 159
Kim, Hyowon, 335
Kim, Hyunjun, 349
Kim, Namwoong, 349
Koch, Raphael, 242
Kreskowski, Adrian, 291
Kupka, Tomas, 228, 249, 277

Lardner, James, 322
Lasak, Martin, 242
Le Bras, Marjorie, 214
Lechelek, Loubna, 214
Lee, Cheng-Tse, 136
Li, Chen, 24
Li, Jie, 284
Li, Qiyue, 284
Lim, May, 221
Liu, Jeffrey, 124
Liu, Jiangchuan, 12
Liu, Kebin, 12
Liu, Mingju, 46
Liu, Yao, 46, 136
Liu, Yong, 24
Liu, Yunhao, 12
Liu, Zhi, 284
Luo, Zhenxiao, 148

Maggio, Antonio Pio, 311
Maity, Mukulika, 201
Maracani, Andrea, 335
Mascolo, Saverio, 311
Mercat, Alexandre, 235
Metwaly, Mohammed, 1
Michieli, Umberto, 349
Midoglu, Cise, 228, 249, 277
Min, Jeongwon, 335
Mishra, Navneet, 201
Mo, Sangdok, 349
Mohomed, Iqbal, 159
Moon, Ji Joong, 349
Murray, Niall, 305, 322
Mystakidis, Aristeidis, 349

Nagy, Bálint György, 170
Nandi, Arnab, 181

Neugebauer, Matthias, 263
Nguyen, Minh, 242, 317
Noh, Eunchung, 335
Nys, Philip, 317

Ooi, Wei Tsang, 136
Ouellette, Jérémy, 57, 298
Ozay, Mete, 335, 349
Ozkan, Savas, 335

Pan, Jianping, 270
Pang, Yihan, 101
Paramonov, Kirill, 349
Perez-Ramirez, Javier, 101
Pham, Stefan, 317

Quinn, Alexander J., 1

Raake, Alexander, 291
Rajore, Tanmay, 201
Rao, Rakesh Rao Ramachandra, 291
Ren, Weiming, 159
Rendle, Gareth, 291
Rizk, Amr, 328
Rodrigo, Adrian, 170
Rudolph, Michael, 328

Sabet, Saeed Shafiee, 228, 249, 277
Santangelo, Gabriele, 311
Sarkhoosh, Mehdi Houshmand, 228, 249, 277
Schleising, Andy, 291
Seeliger, Robert, 242
Sentosa, William, 101, 124
Sera, Bruno, 214
Seufert, Michael, 90
Shannigrahi, Susmit, 190
Sharma, Sahir, 322
Shi, Yuang, 136, 356
Sidhu, Jashanjot Singh, 35, 298, 342
Silhavy, Daniel, 317
Simonovski, Filip, 90
Sinclair, Finn, 124
Sonkoly, Balázs, 170
Sotos, Dimitrios, 349
Steenkiste, Peter, 113
Steglich, Stephan, 242, 317
Sun, Xikai, 12
Sun, Yuan-Chun, 136
Sun, Yuran, 148

Tsalikidis, Nikolaos, 349
Tzovaras, Dimitrios, 349

Ugur, Deniz, 68
Usmani, Mohammad Waquas, 190

Vanne, Jarno, 235
Váša, Libor, 79
Viciedo, Iván, 170

Wang, Mea, 256
Wang, Yao, 24
Wauters, Tim, 113
Wehner, Nikolas, 90
Winecki, Dominik, 181
Wu, Chuan, 148

Xu, Jingao, 12

Yip, Man Ho, 148
Yoo, Woojong, 349
Yu, Cunxi, 46

Zhang, Jeffrey, 101
Zhang, Quanwei, 270
Zhang, Zhuoying, 148
Zhao, Jinwei, 270
Zhu, Mufeng, 46, 136
Zhu, Zhui, 12
Zimmermann, Roger, 221
Zink, Michael, 190
Zong, Tongyu, 24