From 2D Video Conferencing to Photorealistic Immersive 3D Communication

Gunkel, Simon Norbert Bernhard

2024

DOI (link to publisher)
10.5463/thesis.906

document version
Publisher's PDF, also known as Version of record

VRIJE UNIVERSITEIT

FROM 2D VIDEO CONFERENCING TO PHOTOREALISTIC IMMERSIVE 3D COMMUNICATION

ACADEMISCH PROEFSCHRIFT

ter verkrijging van de graad Doctor of Philosophy aan
de Vrije Universiteit Amsterdam,
op gezag van de rector magnificus
prof.dr. J.J.G. Geurts,
in het openbaar te verdedigen
ten overstaan van de promotiecommissie
van de Faculteit der Bètawetenschappen
op dinsdag 3 december 2024 om 13.45 uur
in een bijeenkomst van de universiteit,
De Boelelaan 1105

door

Simon Norbert Bernhard Gunkel

geboren te Berlijn, Duitsland

promotoren:	prof.dr. D.C.A. Bulterman prof.dr. J.R. van Ossenbruggen
copromotor:	prof.dr. P.S. Cesar
promotiecommissie:	prof.dr.ir. F.F.J. Hermans prof.dr. T. Hartmann prof.dr. P.J. Werkhoven prof.dr.ing. A. Raake dr. L. Toni dr. G.H. Berndtsson

From 2D Video Conferencing to Photorealistic Immersive 3D Communication

Dit proefschrift is goedgekeurd door de

promotoren:	Prof. dr. D.C.A. Bulterman
	Prof. dr. J. van Ossenbruggen
copromotor:	Prof. dr. P.S. César Garcia

Samenstelling promotiecommissie:

Prof. dr. ir. F. Hermans,	Vrije Universiteit Amsterdam
Prof. dr. T. Hartmann,	Vrije Universiteit Amsterdam
Prof. dr. P.J. Werkhoven,	Utrecht University
Prof. Dr.-Ing. A. Raake,	Technische Universität Ilmenau
Dr. L. Toni,	University College London
Dr. G.H. Berndtsson,	Ericsson LM

The work in this dissertation has been carried out under TNO (Netherlands Organisation for Applied Scientific Research) and CWI (Centrum Wiskunde & Informatica). Furthermore, the work in this dissertation was partially funded by the European Commission as part of the H2020 program, under the grant agreement 762111 (VRTogether, http://vrtogether.eu/).

Keywords: Immersive, Communication, Social XR, Video Conferencing, Virtual Reality, VR, Augmented Reality, AR, WebXR, WebRTC, RGBD, 3D

Printed by: Ridderprint | https://www.ridderprint.nl

Cover: Simon Gunkel & Yazo Design (Dragana Travas)

Style: https://github.com/Inventitech/phd-thesis-template

The author set this dissertation in LaTeX using the Libertinus and Inconsolata fonts.

ISBN: 978-94-6506-633-2

DOI: http://doi.org/10.5463/thesis.906.

An electronic version of this dissertation is available at https://research.vu.nl/.

Die meisten Menschen legen ihre Kindheit ab wie einen alten Hut. Sie vergessen sie wie eine Telefonnummer, die nicht mehr gilt. ihr Leben kommt ihnen vor wie eine Dauerwurst, die sie allmählich aufessen, und was gegessen worden ist, existiert nicht mehr. […]
Früher waren sie Kinder, dann wurden sie Erwachsene, aber was sind sie nun?
Nur wer erwachsen wird und Kind bleibt, ist ein Mensch.

Erich Kästner, Zum Schulbeginn (1925)

Most people put their childhood behind as if it was an old hat. They forget it like a telephone number that is no longer valid. Their life appears to them like a salami, that they gradually eat up, and what has been eaten no longer exists. […]
They used to be children, and then they become grown-ups, but what are they now?
Only those who grow up and remain children are human.

Erich Kästner, Start of School (translated)

Contents

Summary

Video conferencing has become a commonly used tool for both private and professional remote communication. Despised yet immensely popular, current video conferencing and telepresence solutions have severe drawbacks in the provided quality of communication (including decreased creativity, engagement, social cues, and "zoom fatigue"). This creates the need for new communication technology that allows more natural communication with greater social presence. One way to potentially increase the communication quality is to extend video conferencing towards immersive 3D communication (also called holographic communication or Social Extended Reality, XR). Creating Social XR systems comes with multiple challenges in both technology developments and impact to the user experience.

In this dissertation, we present a novel and modular capture, processing, transmission, and rendering pipeline as a reference for the development and research of Social XR applications. Our pipeline outlines the basic building blocks needed for Social XR and their individual performance implications. To outline the different components and to execute detailed in-depth performance evaluations, we consider 3 main areas:

First, we present a *3D representation* format based on color and depth data (RGBD) and all the necessary steps to capture, process and render users in photorealistic 3D. Our results show an overhead of 5-16% CPU & GPU processing and 300-400ms capture to render delay. Although this is significantly higher than comparable 2D video conferencing, we expect this to be acceptable in an operational environment.

Second, we present an example web-based system to evaluate the impact on *network performance*. Our results show that our grayscale-based transmission of RGBD data performs well under network constraints similar to existing video conferencing solutions. Furthermore, the web-based client has (CPU & GPU) resource demands suitable for any modern (VR-ready) PC.

Third, we present a novel central composition component to show *optimization of transmission* and scalability of simultaneous participants. Our solution allows to combine different user bitstreams into a single decodable video stream. The result increases bitrate by ~16% with a client performance gain of 20% and a server performance gain of 85-90%.

Furthermore, the concepts presented in the dissertation were implemented in three prototype demonstrators to show the applicability of our solutions. The prototypes follow a multi-user shared experience use case in VR, an AR social care communication use case, and an AR remote training use case.

The results presented in this dissertation can be seen as a new reference framework for the general building blocks of Social XR systems and a point of reference for performance evaluation comparisons. In addition, it opens up many possibilities for controlled user evaluations and Quality of Experience (QoE) studies, as well as further research in technical developments surrounding Social XR (to name a few: 3D user encoding, spatial computing, Social XR metadata, 5G, 6G, and edge compute).

Samenvatting

Videoconferenties zijn een veelgebruikt hulpmiddel geworden voor zowel particuliere als professionele communicatie op afstand. De huidige oplossingen voor videoconferenties en telepresence worden veracht maar zijn immens populair en hebben ernstige nadelen wat betreft de geboden kwaliteit van de communicatie (waaronder verminderde creativiteit, betrokkenheid, sociale signalen en "zoomvermoeidheid"). Dit creëert de behoefte aan nieuwe communicatietechnologie die een meer natuurlijke communicatie met een grotere sociale aanwezigheid mogelijk maakt. Eén manier om de communicatiekwaliteit potentieel te verhogen is door videoconferenties uit te breiden naar meeslepende 3D-communicatie (ook wel holografische communicatie of Social XR, Extended Reality, genoemd). Het creëren van dergelijke nieuwe Social XR-systemen brengt meerdere uitdagingen met zich mee op het gebied van zowel technologische ontwikkelingen als de impact op de gebruikerservaring.

In dit proefschrift presenteren we een nieuwe en modulaire opname-, verwerkings-, transmissie- en weergavepijplijn als referentie voor de ontwikkeling en het onderzoek van Social XR-applicaties. Onze pijplijn schetst de basisbouwstenen die nodig zijn voor Social XR en hun individuele prestatie-implicaties. Om de verschillende componenten te schetsen en gedetailleerde, diepgaande prestatie-evaluaties uit te voeren, beschouwen we 3 hoofdgebieden:

Eerst presenteren we een 3D-weergaveformaat gebaseerd op kleur- en dieptegegevens (RGBD) en alle noodzakelijke stappen om gebruikers in fotorealistisch 3D vast te leggen, te verwerken en weer te geven. Onze resultaten laten een overhead zien van 5-16% CPU- en GPU-verwerking en 300-400 ms capture om vertraging weer te geven. Hoewel dit aanzienlijk hoger is dan vergelijkbare 2D-videoconferenties, verwachten wij dat dit in een operationele omgeving acceptabel zal zijn.

Ten tweede presenteren we een voorbeeld van een web-gebaseerd systeem om de impact op de netwerkprestaties te evalueren. Onze resultaten laten zien dat onze op grijswaarden gebaseerde overdracht van RGBD-gegevens goed presteert onder netwerkbeperkingen die vergelijkbaar zijn met bestaande videoconferentieoplossingen. Bovendien heeft de web-gebaseerde client (CPU & GPU) resourcevereisten die geschikt zijn voor elke moderne (VR-ready) PC.

Ten derde presenteren we een nieuwe centrale compositiecomponent om optimalisatie van de transmissie en schaalbaarheid van gelijktijdige deelnemers te tonen. Met onze oplossing kunt u verschillende gebruikersbitstreams combineren tot één decodeerbare videostream. Het resultaat neemt toede bitrate met ~16% met een prestatiewinst voor de client van 20% en een prestatiewinst voor de server van 85-90%.

Bovendien werden de concepten gepresenteerd in het proefschrift geïmplementeerd in drie prototype-demonstrators om de toepasbaarheid van ons algoritme aan te tonen. De prototypes zijn gebaseerd op een use case voor gedeelde ervaringen met meerdere gebruikers in VR, een use case voor AR-communicatie in de sociale zorg en een use case voor AR-training op afstand.

De resultaten gepresenteerd in dit proefschrift kunnen worden gezien als een nieuw referentiekader voor de algemene bouwstenen van Social XR-systemen en een referentiepunt voor vergelijkingen van prestatie-evaluaties. Bovendien opent het veel mogelijkheden voor gecontroleerd gebruikersevaluaties en QoE-onderzoeken, evenals verder onderzoek naar technische ontwikkelingen rondom Social XR (om er maar een paar te noemen: 3D-gebruikerscodering, ruimtelijke computing, Social XR-metadata, 5G, 6G en edge compute).

Acknowledgments

There truly is a "time and place for everything", and it is absolutely astonishing to me how so many, at the time, random and strange moments in my life connect so well into the final result of my PhD dissertation. The most important aspect in this is that it would not have been possible without the help, support, and challenge of many people. If you know me, you know that I place a lot of emphasis and value on my personal relationships. Thus, it should come as no surprise that a great number of people had a key influence on my personal and professional path. In the following I would like to thank some key people, but this is by far not an exhaustive list.

I would like to express my deepest appreciation to my promoters Dick Bulterman and Jacco van Ossenbruggen. Dick, your vision, sharp mind, and academic perspective were a true blessing in my development as a scientist. Jacco, even though our interaction was short, I am very grateful for your support. Words cannot express my gratitude to Pablo César. Pablo, you have my deepest appreciation for welcoming me into your work family, originating my PhD journey, and continuing to support and advise me throughout all those years. You are an incredible mentor, friend, and role model for me. Further, this PhD would not have been possible without Omar Niamut. Omar, with your guidance, mentorship, and space that you created for me, you have equipped me with so many resources, mindset, and drive that are directly linked to the high-quality research and outcomes of this dissertation.

Many thanks to all of my coauthors and direct collaborators for the dissertation work. Sylvie Dijkstra-Soudarissanane, our discussions and the way you encouraged and challenged me were vital in the completion and quality of this dissertation. Rick Hindriks, you were a true buddy to me in many moments of complex technical issues, and many of your work are key contributing factors for the completion of this work. Hans Stokking, both the discussion on and off topic sparked numerous breakthroughs in my thinking and the development of this dissertation. Martin Prins, you shaped many of the initial concepts of this work and I always enjoyed your calm and goal-oriented attitude. Karim El Assal, your technical contributions to this topic were essential to its success. Emmanuel Thomas, it always amazed me how quickly you pick up on ideas and enhance them, which also relates to many topics surrounding this work. I would like to extend my sincere thanks to Véronne Reinders, Yonatan Shiferaw, Nanda van der Stap, and Frank ter Haar. Thank you for the collaboration and all your contributions. It was a great pleasure to work with you. In addition, I would like to recognize Alexandre Gabriel, Aschwin Brandt, Connec2 B.V. (especially Tim Moelard and Stefan Leushuis), Deborah van Sinttruije, Emmanouil Potetsianakis, Evangelos Alexiou, Fraunhofer EMFT (especially Frank Ansorge and Elias Meltzer), Marina Alvarez, Simardep Singh, Tessa Klunder, and Viaccess-Orca(especially Vincent Lepec, Jean-Baptiste Pigree and Guillaume Debeneix) for their collaboration and contributions to this work. I like to extend these thanks to my whole team and department at TNO (incl. Lucia D'Acunto, Adrian Pais, Annemieke Kips, Vasiliki Georgiadou, Dick van Smirren and Frits Klok); thanks for creating such an inspiring and supportive environment.

The same appreciation goes to my team at CWI (incl. Jack Jansen, Thomas Röggla, and Steven Pemberton); thanks you for all our discussions and your support.

I would like to thank my family for always creating a loving and protective environment, as well as supporting me in finding my own path. Eveline Gunkel, you equipped me with so many important values in my life that I don't even know where to start; thank you! Wolfgang Gunkel, your desire to learn and improve is something that you have very much embedded in me as well. Ruth Dunkel, I think you are one of the most important people in my life and you had such a tremendous impact and influence on me with your support, advice, and wonderful time we spend together. Thomas Gunkel, as my older brother, I always felt that you were looking out for me, and this is still to this day. Marcus Gunkel, I truly value you as a person in my life and always appreciate your perspective and insights.

I am also thankful to my friends and companions who supported me in many ways throughout my PhD and in my personal developments. Special thanks to Tobias Kaufhold, Christoph Hildebrand, and Fabian Maximilian Thiele, it is very rare that you find friends that are as close as family to you. Our friendship has been a true blessing in my life. Tobi, as my best and oldest friend, our friendship guided me through many key moments of my life and is still guiding me to this day; thank you. Chris, words cannot describe the gratitude that I feel for our friendship, it is an absolute privilege and blessing to have you in my life; thank you. Max, your technical know-how and keen mind have always been an inspiration to me. Michael Kaisers, our friendship, and our many shared experiences were a key contributor to why I stayed in the Netherlands and continued to work on my PhD; thank you. Dragana Travas, you supported me throughout the majority of time on my PhD and joined me on many unforgettable adventures. Its safe to say that you had a profound impact on shaping me as a human being. Oliver Friedrich, you are one of my role models in leadership and have been crucial in starting my passion for applied research. I like to extend this thanks to the whole Fraunhofer FOKUS, FAME team (incl. Stephan Steglich and Stefan Arbanowski) who taught me the basics of applied research and prototyping that I follow up to this day (not to mention the legendary "clicky-bunty" presentation style). Marwin Schmitt, we started this journey together, and it would not have been the same without the struggles, hardship, and friendship that I experienced together with you.

The acknowledgments would not be complete without mentioning multiple open-source projects that proved vital in the development of the technical basis presented in this dissertation: Python, Numpy, Numba, pyk4a, OpenCV, Node.js, AFrame, Three.js, Resonance Audio SDK for Web, OBS Studio, GStreamer, pyvirtualcam, Unity Capture, MediaPipe, FFmpeg) System Informer, VideoLAT), VQEG siti-tools, RCM-UV), Mingw-w64, Janus WebRTC Server , and libvpx. I am thankful to everyone involved and contributing to these projects, all your work is greatly appreciated.

As a final remark on the acknowledgments, I like to say that the personal aspects are probably what make me the most proud of my PhD dissertation and research. The past 12 years have brought me together with extremely intelligent, smart, driven, and passionate people. That also included a lot of laughter and joy. Clearly, it also included many bumps, setbacks, and challenges that I probably would not have overcome without all the support I received. Finally, at this point, I am ever more excited to spend more time with these incredible people, find new challenges together, and hopefully build a better world together with the help of new technical innovations.

1

Introduction

The research presented in this dissertation creates a holistic view of the complete end-to-end chain of holographic 3D communication, its minimal (low complexity) building blocks, and its technical (performance) limitations. This work presents a novel Social XR system that allows many-to-many communication via photorealistic human representations based on RGBD video. The main body of the dissertation and its scientific contribution can be divided into 3 parts: a) 3D representation, b) system, and c) transmission optimization. In addition, we present an application chapter to outline the applicability of the Social XR approach. This chapter introduces the dissertation topic, research questions, contributions, and outline.

1

Human communication is complex and personal. When it comes to remote communication systems, Skowronek et al. [1] gives a useful overview of the different factors that influence telemeetings, including aspects of the human, context, and system. Furthermore, it provides an outlook on future communication technologies and concludes that "Social XR will bring about new interaction interfaces, modalities, and types, which will require Quality of Experience (QoE) evaluation methods beyond the current standards." Thus, while moving towards immersive communication systems (ICS) on the premise of higher immersion and social presence, we need new technical solutions to implement ICS in a way that they are comparable and allow dedicated user testing. This means that we need to understand technical limitations and gaps, performance implications, and ultimately test real-world use cases. Our research aims to develop a comprehensive understanding of the entire process of Social XR (Extended Reality), encompassing all stages of the holographic 3D communication pipeline. In this chapter, we give an overview of the motivation, research questions, contributions, and outline of this dissertation.

1.1 Background & Motivation

The work presented in this dissertation is motivated by two lines of development:

i The increased importance and use of remote real-time communication software (i.e. video conferencing)

ii An increase in Extended Reality technology and immersive experience applications

When looking at remote real-time communication, video conferencing has become the standard for remote meetings and has increased in popularity during the Covid pandemic [2]. This increase can be observed on a global scale [3]. Furthermore, this trend can be particularly observed in the workplace with hybrid meetings [4] and a "new way of working" [5]. Current videoconferencing systems still have significant limitations, for example, in terms of quality of communication. That is, communication over traditional video conferencing systems may suffer from: decreased creativity [6], decreased engagement [7], decreased social cues [8], and zoom fatigue [8, 9]. Another limitation can be seen in the naturalness of the interaction between people and the lack of social presence [2]. Social presence is "the feeling of being in the presence of, and having an affective and intellectual connection with, other persons"[10]. In general, the increased usage of remote communication software and the limitations in video conferencing raise the need for new communication systems that allow more natural remote interaction with an increased social presence. One possible solution to increase the quality of communication is to move towards 3D immersive communication solutions in Extended Reality (XR)[11].

Extended Reality (XR) is a superset of Virtual Reality (VR), Augmented Reality (AR), and Mixed Reality (MR). It thus covers the whole spectrum of immersive solutions with the idea of projecting "virtual" objects and 3D content into a "virtual" environment or blend it with the real world (explained in the concept of the reality-virtuality continuum [12, 13]). What is important to note is that many of those XR applications are currently a single-user experience. Although this is good in its own right, it is generally human nature to share and participate in experiences together (e.g., with friends or family) [14]. Further, it is important to have a consistent experience in terms of rendering, this means that for a

photorealistic experience (or the real world), we generally can consider it as beneficial to depict users in photorealistic quality. This may not hold true for all use cases; in general, however, the rendering must look real and believable [15]. The effects of realism for the representation of others (avatar realism) have also been widely studied [16–18].

The scope of this dissertation is to look at the system aspects of immersive communication for photorealistic multi-user experiences. Current research indicates multiple possible improvements when using immersive communication systems (ICS) for remote communication, such as

- ICS increases creative quality and productivity, countering some limitations observed in traditional 2D meeting platforms [19]
- ICS increases meeting effectiveness (mainly when it comes to and relationships, communication on feelings or emotions, and when giving or receiving feedback) [2]
- Communication in ICS is more natural and in some cases communication in VR is similar to face-to-face communication [20].

Further research is required to fully understand and improve the benefits of photorealistic ICS (also known as holographic communication or Social XR). Better systems are needed to test, clarify technical constraints and create a generic understanding of its fundamental building blocks. At present, technical constraints, generic building blocks, and performance indicators for photorealistic ICS are not always well defined. Consequently, not all of the advantages and limitations of photorealistic XR communication have been thoroughly investigated, highlighting the importance of additional research. In the following, we describe the key enabling technology and background of remote communication (Section 1.1.1), ICS (Section 1.1.2) and Social XR (Section 1.1.3). In addition, this chapter outlines the problems of the state-of-the-art (Section 1.2) and the scope of the dissertation (Section 1.3).

1.1.1 Remote Communication and Collaboration

In order to better understand immersive and holographic communication, it is good to first look at digital remote communication at large. Apostolopoulos et al. [21] defined a general communication and collaboration framework, see Figure 1.1. The framework clusters different technical solutions according to two axes, the level of information shared, and the level of naturalness of the conversation. The main aim of the framework is to depict the wide spectrum of systems designed for remote communication and their respective methods. With respect to this clustering, we need to consider two important aspects, a) how well can a system convey information sent over it (media richness theory), and b) how suitable is each system for a specific task (task-medium fit).

The theory of media richness was first introduced by Daft and Lengel [22, 23] as a means of describing how well a system can reproduce information. Furthermore, we can group the media richness into four dimentions [24]: feedback immediacy, multiplicity of cues, language variety, and personal focus. The theory of media richness was extended with the task-media fit hypothesis by McGrath and Hollingshead [25]. The task-media fit hypothesis identifies different suitabilities in the effectiveness of different communication mediums and different tasks. Although some follow-up research could not prove the

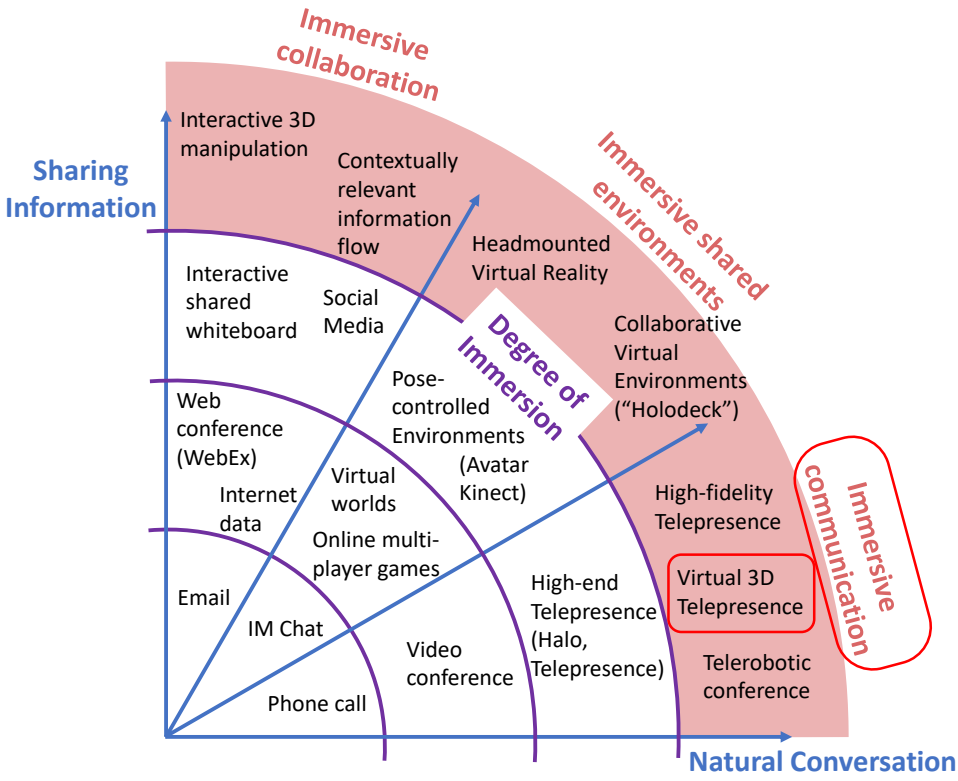


Figure 1.1: Communication and Collaboration framework [21]

task-media fit hypothesis completely accurate [26], it is still a powerful concept for remote communication classification. What is clear from previous research is that not every technical medium is best suited for every task. Thus, the focus of this dissertation is not on a replacement technology (i.e. to replace video conferencing), but rather on a new tool for remote communication to enable better, more natural communication for specific use cases and tasks in photorealistic XR experiences.

1.1.2 Immersive Communication

Table 1.1 gives an overview of the entire landscape of remote communication technology, with examples of key enabling technologies. Starting from general tools for asynchonous and synchonous communication and expanding two levels over Immersive Communication Systems (ICS) into Social XR. Generally, we can note that existing tools, such as video conferencing, already give a sense of remote presence (and social presence), which is believed to increase further in VR [43]. The increase in presence can be further attributed to a sense of non-mediation in VR; simply the "technical" medium disappears [44]. This illusion of "non-mediation" has not been reported for video conferencing [44]. This indicates a potential advantage, or at least a clear difference, when communicating in VR, or in

Asynchronous			Synchronous				
Fax	Email	Message Boards	Phone	SMS	Chat	Video Conferencing	ICS

Immersive communication systems (ICS) [27]					
Life-Like [28] Telepresence	CAVE [29]	Gaming [30]	Social VR (Avatars) [31]	Digital Twin [32]	Holographic (Social XR)

Holographic – Social XR Solutions						
VRComm* (RGBD) [33]	VR2Gather & VRTogether (PC) [34, 35]	TVM [36–38]	Holo-portation [39]	3DTI (3D multi-stream) [40, 41]	V-Cube [42]	…

Table 1.1: General Overview of Remote Communication Solutions

immersive communication systems (ICS) in general.

Immersive Communication Systems (ICS)
There is a wide range of technology that covers communication in immersive 3D environments. Pérez et al. [27] gives a good overview of ICS including its multiple archtypes. These archtypes cover the entire spectrum of immersive technology from "artificial" avatars in shared VR, digital 3D representations of real-world objects (digital twin), and finally photorealistic representations in Social XR and AR. Furthermore, Barzon et al. [45] gives a good overview of ICS and some benefits that ICS can offer. In general, based on previous research, ICS can improve the overall communication [20] due to an increased effectiveness of meetings [2], increased quality of personal creativity [19], and increased perception of immersion [46]. To give a better idea of some ICS technology examples, Table 1.1 shows a non-exhaustive list of the main ICS solutions:

- **Life-Like Telepresence** [28], usually comprises high-class video conferencing equipment that depicts people on big screens in their respective "real" size, ideally matching gaze, and thus can create a high-quality remote communication.
- **CAVE** [29], stands for Cave Automatic Virtual Environment and usually consists of multiple projectors facing white walls, thus forming a virtual environment at room scale.
- **Gaming** [30], besides virtual environment mainly built for social purposes (such as second life), the users avatar representation and how users interact in multi-player scenarios become increasingly important.
- **Social VR (Avatars)** [31], in recent years there has been a steep increase in the ever growing number of solutions that allow people to communicate in virtual

environments with the help of computer-generated artificial avatars. To name some of the most important ones (each offering different features of interaction and slightly different look and feel[1]): Meta Horizon Worlds[2], Microsoft Mesh[3], Odyssey[4], Break Room[5], Frame VR[6], Alakazam[7], VR Chat[8], Rec Room[9], Engage[10], Glue[11], and Spatial[12].

- **Digital Twin** [32], covers a range of technologies with the main aim of reproducing or virtualizing real-world objects as 1-to-1 digital replicas.

- **Holographic (Social XR)**, covers immersive communication based on photorealistic 3D user representation and is described in the following.

1.1.3 Social XR / Holographic Communication

When we look at Social XR [27] (also known as holographic communication [47]), it is all about real-time photorealistic communication in 3D. Users are captured via different video or volumetric capture solutions and transmitted via different protocols with the ultimate goal of a realistic and natural representation of the user in 3D and 3D environments (such as computer-generated graphics or photogrammetry in VR scenes, or blened into the real world in AR). In some definitions, holographic communication covers all 5 types of human sensory in formations (such as hearing, sight, touch, smell, and taste; mulsemendia) [48]. In the scope of this dissertation the focus of the work on Social XR is on visual and auditory information only. Table 1.1 gives a non-exclusive list of representative Social XR systems:

- **VRComm** [33], is representative for RGBD-based 3D user representations utilizing color and depth image information (RGBD: Red, Green, Blue, Depth) and presented in this dissertation.

- **VR2Gather & VRTogether** [34, 35], is representative for point cloud (PC) 3D user representation. In this approach, users are captured with multiple RGB+Depth sensors, reconstructed into a unstructured point cloud raw format, encoded via a unique format, and transmitted via either TCP, Socket.IO or DASH (Dynamic Adaptive Streaming over HTTP). The approach offers acceptable performance in terms of both client resource use (CPU) and end-to-end latency in a complex capture setup with limited point resolution and specific encoding.

[1] https://web.archive.org/web/20240324201920/https://ryanschultz.com/list-of-social-vr-virtual-worlds/
[2] https://web.archive.org/web/20240409150909/https://www.meta.com/horizon-worlds/
[3] https://web.archive.org/web/20240212003720/https://www.microsoft.com/en-us/microsoft-teams/microsoft-mesh
[4] https://web.archive.org/web/20240110204241/https://www.odyssey.stream/
[5] https://web.archive.org/web/20240414220825/https://www.breakroom.net/
[6] https://web.archive.org/web/20240204060700/https://framevr.io/
[7] https://web.archive.org/web/20230303194904/http://alakazam.io/
[8] https://web.archive.org/web/20240502224223/https://hello.vrchat.com/
[9] https://web.archive.org/web/20240204193031/https://recroom.com/
[10] https://web.archive.org/web/20240320093229/http://www.engagexr.co/
[11] https://web.archive.org/web/20231230220328/https://www.glue.work/
[12] https://web.archive.org/web/20240502223440/https://www.spatial.io/

- **TVM** [36–38], is representative for 3D user representation of time-varying meshes (TVM). In this approach, users are captured with four RGB+Depth sensors, reconstructed into time-varying meshes (TVM), the TVM is encoded via existing mesh encoders (such as OpenCTM[13], Draco[14], Corto[15] and O3DGC[16]), the texture is encoded via JPEG compression and transmitted via RabbitMQ. Not all performance details of this approach are known. Mesh reconstruction and compression, however, is computationally complex and both demanding in latency and resources.

- **Holoportation** [39], uses it own unique wire mesh 3D representations format. In this solution, eight rgb + depth capture sensors (connected to 4 PC) are used for the capture of a complete room including multiple users. The capture is then reconstructed into a mesh in a GPU-based computationally expense process. Furthermore, the resulting wire mesh is compressed with LZ4 compression. Additionally, the color image is compressed with LZ4 as well and both is transmitted via unicast TCP and in an approx. bandwidth of 1 Gbit/s. An alternative approach to the "Holoportation" prototype transmits the raw RGB + depth camera signals compressed with "Run length encoding and Variable Length encoding" (RVL) [49], and the Room Alive Toolkit [50]. Reducing the processing load but with similar or increased bandwidth to the wire mesh approach.

- **3DTI** [40, 41], 3D multi-stream. In this solution, a 3D camera array captures "a local scene from various angles" and combines individual streams into stream bundles[51]. This allows clients to receive only the necessary streams to reconstruct a 3D view. This approach, however, can result in high bandwidth demands and in general poses a complex system and setup.

- **V-Cube** [42], VirtualCube combines the concept of a CAVE like render cube with RGBD capture devices and user view position tracking to allow sceen-based virtual photorealistic communication between users. Although V-Cube is based on commonly available hardware, it requires a very specific room setup (for each user). In the given solution [42] it further requires high powered GPU compute units for remote rendering, and utilises a suboptimal image transmission in JPEG. This might hinder the general deployability and scalability of such a system without further research.

In summary, all Social XR systems have different advantages and disadvantages, but all suffer from specific limitations as discussed in the next section (1.2).

1.2 Problem

Looking at the current state of the art in the research, prototypes, and solutions of Social XR (holographic communication), they all have different limitations. That said, there is fragmentation in the technical directions of the solutions, and the technical requirements

[13] https://web.archive.org/web/20240503005210/https://openctm.sourceforge.net/
[14] https://web.archive.org/web/20240108074840/https://google.github.io/draco/
[15] https://web.archive.org/web/20230812175843/https://github.com/cnr-isti-vclab/corto
[16] https://web.archive.org/web/20240322040114/https://github.com/rbsheth/Open3DGC

for Social XR applications are not always clear or clearly defined. Therefore, we need to understand the basic requirements and implications of the different technical building blocks of Social XR. This means creating a better understanding of the impact of current technical limitations and gaps, as well as creating reference architectures, implementations, and performance measures to make different solutions comparable. If we summarize some of the gaps and problems in current holographic communication research, most solutions face one or multiple of the following:

- Complex capture setup
- High bandwidth needs
- High processing needs
- Largely incompatible with existing technology
- Limited in the number of holographic user representations

In this way, most of the latest Social XR solutions are difficult to deploy, configure, and complex to apply in user testing. In addition, most solutions are not fully tested in terms of technical suitability, impact on user experience, and communication behavior. One of the key evaluation methods for remote communication and digital experiences is the measure of Quality of Experience (QoE) [52]. Previous research [1] states that "fully understanding *Telemeeting QoE* requires a holistic approach that takes into account all three types of *Quality Influence Factors (QIF)*, namely concerning the telemeeting technology and system, the experiencing person, and the context(s) in which the telemeeting takes place." Thus, complete QoE research can be complex and costly, and it always requires a stable technical system based on Quality of Service (QoS) constraints [53, 54]. Furthermore, the mapping of QoS to QoE is mostly difficult and complex [55]. For example, before being able to analyze these complex QoE relationships for Social XR we first need to understand the basic building blocks on a technical level.

The focus in this dissertation is on the basic building blocks of Social XR, with the help of technical evaluations and with the help of Quality of Service parameters (i.e., delay, image quality, and resource usage). This is both on an individual component level and on a holistic overall system approach. The goal of this work is to create a new reference for building more complex Social XR systems and executing more user tests in the future.

1.3 Scope

One of the main aims of this work is to extend 2D video conferencing towards photorealistic immersive 3D communication. This paradigm change is motivated by the identified benefits of ICS and Social XR: improved overall communication [20], increased effectiveness of meetings [2], improved quality of personal creativity [19], and increased perception of immersion [46]. Ultimately, our goal is to create new immersive communication systems that allow for similar many-to-many communication possibilities as current 2D video conferencing allows. In the scope of this dissertation, this means creating a generic basis for technical comparisons and to allow further validations of actual benefits for the users and their communication. One of the key differences between a 2D and 3D

photorealistic communication system is in the representation of users in 3D. Furthermore, user representation is also one of the key aspects to convey essential communication cues [18, 56]. Thus, we first need a generic way to create 3D user representations in a simple way, with existing off-the-shelf hardware and in a widely compatible transmission format (**RQ1**). Furthermore, to assess the limitations of the different state-of-the-art building blocks and to create a reference framework, we propose a novel Social XR system. One of the central points of this Social XR system is to allow measurements of various technical characteristics and QoS properties, both on an individual component level and on a system and network level (**RQ2**).

Ultimately, the goal of Social XR is to replicate similar communication sessions as we are used to from the existing 2D video conferencing solution in 3D and enable new forms of remote interaction in multi-user XR. In addition to the technical and performance characteristics of the system, network, and end devices, an essential part is the number of users that can be supported in one single communication session. Current Social XR systems are very restrictive in the number of simultaneous users in one communication session. This means that we need to better understand the impact of different technical restrictions on the number of simultaneous users, as well as how to potentially increase the number of simultaneous users (**RQ3**), to allow a wide range of conversational use cases in Social XR in the future.

To summarize, the main scope of this work is to create a holistic view on the complete end-to-end chain of holographic 3D communication, its minimal (low complexity) building blocks, and technical (performance) limitations. To achieve this, the work was guided by multiple research questions that are explained in the following table.

1

Main Research Question (RQ): What are the technical challenges and limitations in extending existing video conferencing solutions to support photorealistic 3D immersive communication with RGBD (color and depth) video streams?		
RQ 1. 3D User Capture & Representation What are the performance implications of different RGBD capture modalities for 3D user representations?	**RQ 2. Network and System Performance** What is the performance and bandwidth overhead for immersive communication based on RGBD?	**RQ 3. Transmission Optimization** What is the maximum number of simultaneous users in Social XR and can we optimize the server and client resource needs to increase that number?
RQ 1.1. What are the advantages and disadvantages of different RGBD capture modalities, individual processing steps, and their resource needs?	**RQ 2.1.** Is RGBD 2D video transmission suitable for XR communication with volumetric user representations?	**RQ 3.1.** Is tile-based composition possible in real-time with a central composition unit and VP9?
RQ 1.2. How can RGBD data be utilized for 3D photorealistic user representations, and what is its rendering resource overhead?	**RQ 2.2.** Which components need to be extended or added to common video conferencing architectures to allow Social XR?	**RQ 3.2.** What is the impact on resource usage for clients and the central server, when using a tile-enabled VP9 encoding and composition (T-MCU)?
RQ 1.3. Do RGBD-based user representations result in a high social presence for remote communication?	**RQ 2.3.** What performance and network utilization are expected for Social XR?	**RQ 3.3.** How many simultaneous 3D photorealistic users could this approach allow in an immersive communication session?

Table 1.2: Research Questions

1.4 Research Questions

The focus of this work is the development of a generic holographic end-to-end chain and Social XR reference system. That is, we look at the individual building blocks of Social XR, how they form a complete Social XR system, and individual performance implications. As captured in the main research question (Table 1.2). Our main hypothesis is that by extending current video conferencing solutions in architecture and functionality, immersive communication in photorealistic 3D is possible with existing hardware limitations (CPU/GPU/Bandwidth). To address the main research question (Table 1.2), we split the research into three main directions:

a) Investigation of the **3D user representation** (RQ1., Table 1.2) using a new format based on RGBD.

b) Evaluation of the **network and system performance** (RQ2., Table 1.2) using a novel web-based Social XR system.

c) Presentation of a **transmission optimization** (RQ3., Table 1.2) to improve the scalability of the system.

First, the center of any Social XR solution is the **representation of the users in 3D**, as this is the key difference compared to other remote communication solutions. Therefore, when using a new capture format (i.e. RGBD), we first need to understand the performance overhead compared to traditional 2D capture (RQ1., Table 1.2). This means that we need an understanding of the different RGBD modalities, different capture and processing blocks, and their performance implications (RQ1.1., Table 1.2). The capture and processing of the user into a new format also has direct implications on the rendering performance. Thus, we need to study these implications and in particular any resource overhead (including different optimizations that can be applied in the rendering shader, i.e., utilizing GPU resources optimally; RQ1.2., Table 1.2). Finally, when it comes to a photorealistic 3D user representation, we would like to know if it can provide any added benefit. That is, an increase in social presence, compared to traditional 2D video (RQ1.3., Table 1.2).

Second, to fully understand all the components of the end-to-end Social XR system, we cannot look at the user representation in isolation. Therefore, we need a complete system with all the system blocks to evaluate the implications on **network** and various **system performance** (RQ2., Table 1.2). Firstly, from a system perspective, it is important to check on the suitability of the new 3D user format in terms of transmission. Thus, we compare different transmission strategies of RGBD data on its suitability through factors of quality of service (QoS; i.e., bandwidth and PSNR; RQ2.1., Table 1.2). Furthermore, we redesign the 3D video-based communication system into a new Social XR system. Therefore, we need to understand any new components in the architecture (RQ2.2., Table 1.2) and the performance and utilization of the network (RQ2.3., Table 1.2).

Third, for realistic deployments, we need to support a similar number of users as currently available in common video conferencing. In addition, to support new XR devices, such as AR glass-type devices (with limited power and compute resources), the performance footprint on the client running on end devices needs to be low while allowing maximum quality. For this reduction in the resources used, we propose a novel tile-based central composition unit based on VP9 (RQ3., Table 1.2). This requires research to determine

whether such a tiled composition is even possible with VP9 and what extensions must be made to the existing encoding and transmission pipeline (RQ3.1., Table 1.2). Furthermore, we would like to understand the performance implications of this approach for the client and the server and whether it provides the envisioned benefits (RQ3.2., Table 1.2). Ultimately, we would like to understand how many simultaneous users can be supported in a Social XR with a tile-based central composition unit (RQ3.3., Table 1.2).

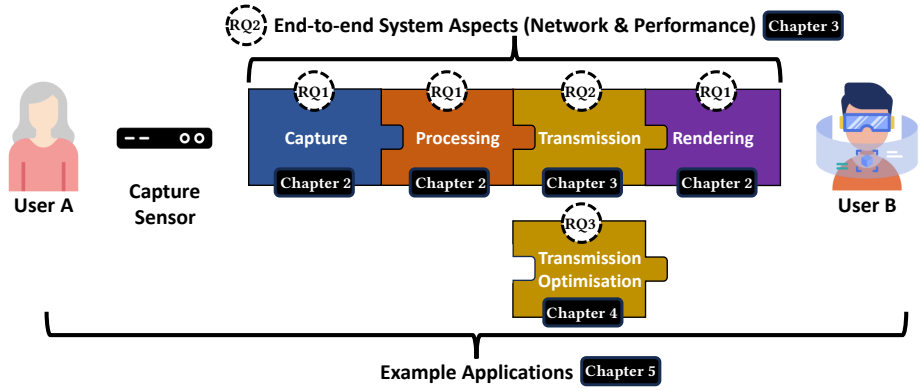


Figure 1.2: Simplified Social XR pipeline in relation to core dissertation chapters and research questions

1.5 Contributions

This work is one of the first to design the full end-to-end minimal building blocks of a Social XR system, breaking its holographic communication pipeline into its minimal and basic components. In addition, we provide an in-depth analysis of the performance and delay impact of each of these single components. Thus, the main contribution of this work is in a Social XR reference framework for modular capture, processing, transmission, and rendering of 3D user representations. The results presented from our evaluation can help to build new and better Social XR systems while making the technical characteristics and limitations easier to compare between different systems. Finally, the main outcome of this research is general guidelines on how to build and measure Social XR systems on a technical level.

Based on the three main research questions, the contributions of this work can also be structured along the three main chapters of the dissertation (see Figure 1.2): 3D User Representation (Chapter 2), Network & Performance (Chapter 3), and Transmission Optimization (Chapter 4). Furthermore, we added another core chapter to outline the applicability of the contributions in Applications (Chapter 5). Figure 1.2, depicts a simplified end-to-end Social XR pipeline, including how the different research questions and dissertation chapters relate to this pipeline. Furthermore, Figure 1.2, shows a generic and simplified pipeline from a sender to a receiver, but not the full complexity of the multiparty (many-to-many) system architecture as addressed in our research. In the following, we present the main contributions aligned with the research questions and the main dissertation chapters.

Capture and 3D reconstruction (RQ1 and Chapter 2)
The basis of this research is a unique RGBD capture and transmission format. In the following, we outline the key contributions related to the capture and rendering of this format.

- **C 1.1.** Design considerations and implementation of a modular real-time capture component for RGBD-based photorealistic 3D user representations (as basis to address **RQ1.1**).
- **C 1.2.** A technical evaluation of the individual modules and processing strategies for RGBD-based capture (**RQ1.1.**) and rendering (**RQ1.2.**).
- **C 1.3.** Two small-scale user evaluations of an AR and a VR communication application that integrate the modular capture approach (**RQ1.3.**).

System and Performance (RQ2 and Chapter 3)
Utilizing the unique RGBD user representation format, the dissertation presents a full end-to-end Social XR system with the following contributions.

- **C 2.1.** A novel transmission scheme for grayscale-based depth information, for 3D user representations. An evaluation of the new transmission format and a comparison with other existing solutions based on QoS characteristics (i.e., bandwidth and PSNR; **RQ2.1**).
- **C 2.3.** The design and implementation of a VR communication system that combines video conferencing technology with Social XR capabilities in a new end-to-end pipeline from user capture, processing, transmission, and rendering photorealisitc 3D users into virtual environments for shared immersive experiences and communication (**RQ2.2**).
- **C 2.3.** An evaluation of the resulting VR communication system with respect to processing delay, CPU and GPU usage (**RQ2.3**).

Transmission Optimization (RQ3 and Chapter 4)
Based on the end-to-end system the dissertation presents further enhancements on the transmission and resource optimization to increase the number of simultanious users with the following contributions.

- **C 3.1.** An extensions to the VP9 reference encoder that allow composition of individual tiles into a single bitstream for parallel decoding (**RQ3.1**).
- **C 3.2.** A new concept of a Tiled-enabled Multipoint Control Unit (T-MCU), and an experimental design study providing in-depth details on its impact on client performance, server performance, and delays (**RQ3.2**).
- **C 3.3.** An example implementation to demonstrate how T-MCU enables support for large user groups in XR with photorealistic volumetric user representations (**RQ3.3**).

1.6 Dissertation Outline

In the following we describe the outline of the dissertation as shown in Figure 1.2.

Chapter 1 Introduction. Introduces the topic of the dissertation, research questions, contributions, and outline of the dissertation.

Chapter 2 3D User Representation. Presents a complete and modular RGBD capture module that includes different capture, processing, and rendering steps to utilize RGBD as a means of photorealistic 3D user representations. In addition, different modules and capture modalities are evaluated both in technical experiments and in two small-scale user tests. Furthermore, Chapter 2 addresses RQ1. and is based on [57]:

- *Simon N.B. Gunkel, Sylvie Dijkstra-Soudarissanane, Hans M. Stokking, and Omar Niamut. From 2d to 3d video conferencing: Modular rgb-d capture and reconstruction for interactive natural user representations in immersive extended reality (xr) communication.* Frontiers in Signal Processing*, 3, 2023*

Chapter 3 Network & Performance. Provides a novel immersive communication framework, which is web-based and utilizes RGBD video. In addition, the chapter provides an evaluation of system and network performance. Furthermore, Chapter 3 addresses RQ2. and is based on [33]:

- *Simon NB Gunkel, Rick Hindriks, Karim M El Assal, Hans M Stokking, Sylvie Dijkstra-Soudarissanane, Frank ter Haar, and Omar Niamut. Vrcomm: an end-to-end web system for real-time photorealistic social vr communication. In* Proceedings of the 12th ACM multimedia systems conference*, pages 65–79, 2021*

Chapter 4 Transmission Optimization. Presents an optimization on the RGBD video transmission to reduce performance load on the system and end device, and thus increase the scalability of simultanious users in one immersive Social XR communication session. The optimization is based on VP9 encoding and a Tile-aware Multipoint Control Unit (T-MCU). In addition, the chapter presents an experimental design study to evaluate the T-MCU under different streaming conditions (in terms of latency, client performance, and server performance). Furthermore, Chapter 4 addresses RQ3. and is based on [58]:

- *Simon N.B. Gunkel, Rick Hindriks, Yonatan Shiferaw, Sylvie Dijkstra-Soudarissanane, and Omar Niamut. Vp9 bitstream-based tiled multipoint control unit: Scaling simultaneous rgbd user streams in an immersive 3d communication system. In* Proceedings of the 15th ACM Multimedia Systems Conference*, MMSys '24, page 23–33, 2024*

Chapter 5 Applications. Outlines three specific application examples based on published research demonstration papers. The demonstrators cover a) a VR shared experience use case [59], b) a screen-based AR social care use case [60], and c) a remote training use case with AR glass-type devices [61]. The given examples show the applicability of the presented research and contributions in the dissertation towards real-world scenarios. Chapter 5 is based on [59–61]:

- *Simon NB Gunkel, Hans M Stokking, Martin J Prins, Nanda van der Stap, Frank B ter Haar, and Omar A Niamut. Virtual reality conferencing: Multi-user immersive vr experiences on the web. In* Proceedings of the 9th ACM Multimedia Systems Conference*, pages 498–501, 2018*

- *Sylvie Dijkstra-Soudarissanane, Simon NB Gunkel, and Véronne Reinders. Virtual visits: life-size immersive communication. In* Proceedings of the 13th ACM Multimedia Systems Conference*, pages 310–314, 2022*
- *Simon NB Gunkel, Sylvie Dijkstra-Soudarissanane, and Omar Niamut. "you ar' right in front of me": Rgbd-based capture and rendering for remote training. In* Proceedings of the 14th Conference on ACM Multimedia Systems*, pages 307–311, 2023*

Chapter 6 Conclusion. The final chapter of the dissertation provides a discussion on how the research questions are addressed, the results of the dissertation, and some directions for future work.

2

3D User Representation

With recent advancements in Virtual Reality (VR) and Augmented Reality (AR) hardware, many new immersive Extended Reality (XR) applications and services arose. One challenge that remains is to solve the social isolation often felt in these XR experiences and to enable a natural multi-user communication with high Social Presence. While a multitude of solutions exist to address this issue with computer-generated "artificial" avatars (based on pre-rendered 3D models), this form of user representation might not be sufficient for conveying a sense of co-presence for many use cases. In particular, for personal communication (for example, with family, doctor, or sales representatives) or for applications requiring photorealistic rendering. One alternative solution is to capture users (and objects) with the help of RGBD sensors to allow real-time photorealistic representations of users. In this chapter, we present a complete and modular RGBD capture application and outline the different steps needed to utilize RGBD as means of photorealistic 3D user representations. We outline different capture modalities, as well as individual functional processing blocks, with its advantages and disadvantages. We evaluate our approach in two ways, a technical evaluation of the operation of the different modules and two small-scale user evaluations within integrated applications. The integrated applications present the use of the modular RGBD capture in both AR and VR communication application use cases, tested in realistic real-world settings. Our examples show that the proposed modular capture and reconstruction pipeline allows for easy evaluation and extension of each step of the processing pipeline. Furthermore, it allows parallel code execution, keeping performance overhead and delay low. Finally, our proposed methods show that an integration of 3D photorealistic user representations into existing video communication transmission systems is feasible and allows for new immersive XR applications.

This chapter is based on Simon N.B. Gunkel, Sylvie Dijkstra-Soudarissanane, Hans M. Stokking, and Omar Niamut. From 2d to 3d video conferencing: Modular rgb-d capture and reconstruction for interactive natural user representations in immersive extended reality (xr) communication. *Frontiers in Signal Processing*, 3, 2023 [57]

Although 2D video conferencing has gained increased popularity for everyday communication, it may not be ideal for all use cases and may lead to exhaustion, as well as other problems related to "zoom fatigue" [8]. One possible solution to improve remote communication systems is to introduce 3D spatial cues in immersive extended reality (XR) applications. Immersive communication applications built on XR technology have the promise of high immersion, presence, and more natural user interactions [62]. Eventually, the goal of immersive communication systems is to maximize social presence between users in remote communication. As described by Lombard and Ditton [44], users ultimately feel the illusion of nonmediation. This is achieved in part by transporting a user virtually to the other user. We firmly believe that at least one other part of it is giving users a feeling of 'the same place', of local togetherness, of physical proximity. Although current virtual reality (VR) communication systems are maturing, they still face significant drawbacks. One main complication is the capture and rendering of users in photorealistic quality. Photorealistic representations can be a contributing factor in allowing the high level of immersion and social presence and might be required in any photorealistic XR environment, simply to blend users into the virtual or augmented world. However, photorealistic capture is complex due to stringent real-time requirements and significant computational demands. Furthermore, the advantages and disadvantages of different processing and analysis strategies are not always clear. This raises the need for a better understanding of the technical steps and their impact on any resource usage in the system and the user's device.

To address this gap, this work outlines the main technical building blocks in a modular capture application to enable immersive communication via RGBD (color & depth) user capture and rendering. Our approach is based on existing real-time streaming components and infrastructure and was integrated into two XR communication systems. Thus, we propose a modular capture application, paired with multiple 3D rendering strategies, and integrated into a commercial communication toolkit. We evaluated our approach with users in their daily work environment. This evaluation assesses the practical usability of the system, our technical approach, and the quality of communication / interaction (i.e., Social Presence). The research in this chapter was based on the following research questions and hypotheses:

RQ 1.1. What are the advantages and disadvantages of different RGBD capture modalities, individual processing steps, and their resource needs?
H1.1.1 RGBD capture and processing is possible within a latency that is acceptable for real-time communication (<500ms glass-to-glass).
H1.1.2 RGBD capture and processing is possible with a performance overhead that is acceptable for any modern PC/Laptop.

RQ 1.2. How can RGBD data be utilized for 3D photorealistic user representations, and what is its rendering resource overhead?
H1.2.1 RGBD data allow 3D rendering of user representations on different devices and with a performance overhead acceptable for low powered stand alone hardware.
H1.2.2 Different GPU shader optimizations allow for higher rendering quality without significant processing overhead.

RQ 1.3. Do RGBD-based user representations result in a high social presence for remote communication?

While previous work outlined the system and transmission aspects of RGBD-based 3D conferencing [33] our works's focus is on the capture and rendering technologies and the evaluation thereof. In order to address our research questions, we followed an experimental design under controlled conditions for RQ 1.1 and RQ 1.2, which offers technical guidance of different RGBD capture and processing approaches and evaluate them in detail with technical measures. Additionally, we conducted two small-scale case studies (based on RQ 1.3) within two realistic deployments of the capture application into complete immersive communication experiences. The work presented in this chapter offers the following contributions:

C 1.1. Design considerations and implementation aspects for real-time modular capture of RGBD-based photorealistic 3D user representation into real-world systems and hardware.

C 1.2. Technical evaluation of individual modules and strategies for RGBD-based capture and rendering.

C 1.3. Small-scale user evaluation of both an AR and a VR communication application that integrates our modular capture approach.

2.1 Related work

Volumetric video is recognized as a crucial technique for creating immersive AR and VR experiences. An ongoing area of research is the capture, encoding and transmission of volumetric video formats such as point clouds and models [63–67]. Volumetric videos, in particular, improve the quality of experience and social presence [68, 69] for remote telepresence and immersive communication [39, 70]. However, most current volumetric video formats, such as video-based point cloud coding (V-PCC)[71] and geometry-based point cloud coding (G-PCC)[65]) require a lot of computing power, making them currently difficult to implement in realistic real-time pipelines. One alternative is to convert the depth information into a 2D image format that can be encoded and transmitted via existing image/video compression techniques. Pece et al. [72] introduced a simple video codec for depth streaming, while [33, 73, 74] considered a more sophisticated conversion of depth information into grayscale images. Furthermore, the use of RGBD images for 3D user rendering is a well-established concept in depth-image-based rendering (DIBR). DIBR was introduced for 3D TV [75] but can also be applied for complex texture synthesis [76]. However, 3D TV was never widely deployed and, to our knowledge, DIBR was not applied in the context of a communication system. Furthermore, AR and VR applications require spatial computing, which is the ability to recognize the environment, the user, and the things surrounding the user (see [77, 78]). In terms of communication, this means that in order to express effective remote interactions, the user's actions must be accurately mirrored in the user's representation. Spatial processing tasks are complex and complicated to test; thus, the modular capture application presented in this chapter is designed for easy extension and reconfiguration to simplify spatial processing prototyping and testing, as well as to form a basic building block for immersive communication applications.

There are some solutions to allow immersive communication through Free Viewpoint Video (FVV) [79, 80]. These approaches run on consumer-grade hardware and existing encoding strategies. The ease of deployment is still hindered by complex calibration and preprocessing steps. There are further solutions based on photorealistic social VR

communication [33, 81, 82]. For example, [83] introduces a web-based framework for communicative and Social VR experiences. Further, [84] outlines multiple use cases of Social VR and the benefit to user experience. These experiences are also considered suitable for real-world situations, such as stand-up meetings[85]. In a different study, Francesca et al. [86] found that face-to-face and photorealistic Social VR show no statistical differences in terms of interaction and social connectivity. Furthermore, immersive communication compared to 2D video conferencing is theorized to differ in its ability to provide social context-related cue transfer, particularly involving body posture, gestures, movement, and eye contact. Furthermore, the way in which presence is typically experienced in VR[43] can also affect communication and interpersonal relationships.

Engaging in remote collaboration benefits organizations, as well as their employees, by offering an advantage over less flexible competitors and supporting employee well-being [87]. However, virtual teams have also been shown to be prone to various problems. Thompson and Coovert [88] propose a categorization of the main issues faced by virtual teams, consisting of: (i) decreased communication quality; (ii) ineffective interpersonal relationships; and (iii) lack of awareness of the work of each other. An important determinant of the decrease in communication quality and the ineffective development of interpersonal relationships appears to be the generally restricted flow of social information between virtual team members. Social information in this context is commonly understood as social context cues and refers to nonverbal cues (e.g. gestures), paraverbal cues (e.g. tone of voice), status and interpersonal cues (e.g. age), and characteristics of physical surroundings (e.g. office size) [89]. In fact, research shows that social context cues help regulate interpersonal interaction and information exchange [89], and the development of trust in teams [90]. In turn, this supports communication and cooperation [91, 92], as well as the development of interpersonal relationships [93]. In addition, social context cues show positive correlations with levels of efficiency [94], perceived communication quality, and satisfaction [95]. As such, it can be argued that the more a medium affords the transfer of social context cues, the more it supports the quality of communication and the development of effective interpersonal relationships.

2.2 Method - RGBD-based 3D Representations

In this section, we outline the technical pipeline and the generic modular capture application (see Figure 2.1). Most immersive communication systems can be simplified into three components: capture, client, and central system components (such as data transmission). It is important to note here that there is a causal relationship between any applied capture method and the reconstruction or rendering of a user. With a focus on RGBD as a format, this section explains the main aspects and modules of the full pipeline to allow capture and reconstruction of users in 3D. Further examples of actual application implementations are presented in Section 2.3.2.

2.2.1 Design Considerations

One of our main requirements is to have a capture and render pipeline that is completely modular to make different individual process blocks easy to measure and to extend. This includes an easy extension to use different capture devices and an open API to access the

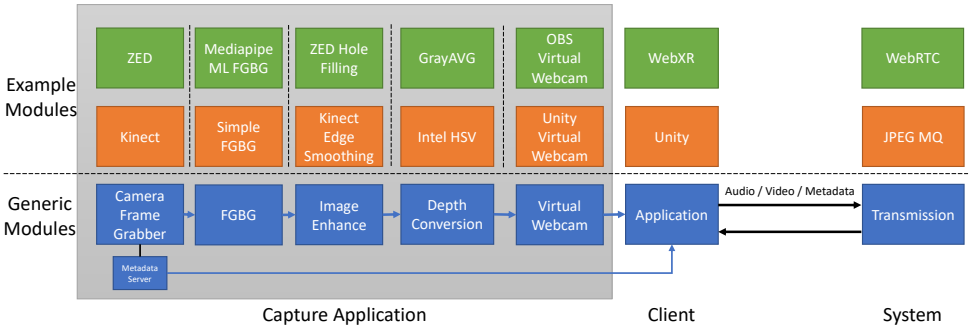


Figure 2.1: Modular Capture Architecture

final capture in any application. All with the main purpose to allow photorealistic 3D user representations in in real-time communication use cases. Therefore, we consider the following design criteria (based on our four hypotheses):

Capture Latency (H1.1.1): One of the most important factors for any real-time communication system is delay. The acceptable end-to-end (capture-to-display) for video conferencing is still debated; however, an acceptable value was identified as less than 500 ms for group discussions [96]. Thus a latency of up to 500ms should be acceptable for most people. Furthermore, latency values between 300-500ms are in line with many existing video conferencing systems [97] and immersive communication systems [33, 98]. To the best of our knowledge, no research has been conducted on the user and communication perception of latency in photorealistic 3D immersive communication. Therefore, in this article, we will focus on technical values and expect more research on user-side aspects in the future.

Capture Performance (H1.1.2): Essentially, the capture system has to be accessible and easy to deploy. Therefore, it should operate under performance constraints (CPU, GPU, and memory utilization) that are acceptable for any modern laptop / PC.

Render Interoperability (H1.2.1): The RGBD-based approach presented in this chapter presents challenges not only in the capture of such data but also in the reconstruction and rendering. Any rendering approach presented has to be adoptable for different render engines (e.g., WebGL and Unity) and under performance constraints that allow low-powered devices (i.e., AR glasses) to reconstruct and render the 3D user representations.

Render Optimizations (H1.2.2): When it comes to the quality of the user representation, the capture quality is a very important factor. However, due to expected lossy compression (otherwise, the data rates would not be suitable for any internet-based system), many more artifacts and distortions may be injected into the representation. Thus creating a need to optimize and clean the image not only at the capturing stage but also during receiving or rendering. Furthermore, any performance overhead resulting from such optimizations should have minimal impact on the rendering device.

2.2.2 Modular Capture and Processing

In this chapter, we introduce a modular capture and processing application that was fully developed in Python. Python was chosen because it offers easy code development and offers a rich set of modules for image processing and machine learning. Furthermore, most RGBD sensors SDKs offer simple wrappers in Python (like the Kinect Azure and ZED used as exampled in this chapter), which offer similar performance as the native C solutions. Also, utilizing OpenCV[1] and Numba[2], many image processing tasks are implemented with hardware acceleration and underlying C bindings, thus keeping the performance overhead low. Although in a production environment it would be more desirable to have a complete native solution, our approach offers an excellent entry point for prototyping, experimentation, and research. This module can be easily replaced for new research needs or extended with new processing functions such as complex spatial user and environment analysis. Finally, all modules are fully interchangeable with the ZED 2i, K4A, and any future sensor capture input, but may lead to undesirable or suboptimal image quality.

The capture application is divided into five generic modules (see Figure 2.1). For optimal performance and parallel processing, each module runs in its own thread loop. Information between modules is exchanged via simple queues that ensure thread safety and image ordering. All modules are glued together via a central governing thread that also is the main tread to interface with OpenCV image rendering (displaying any required final or debug output in a window). In addition to the five modules, a Metadata Server may offer additional metadata information from the capture module to applications on the same computer system. The five generic capture modules, as shown in Figure 2.1, are explained in more detail (including example implementations of these modules) below.

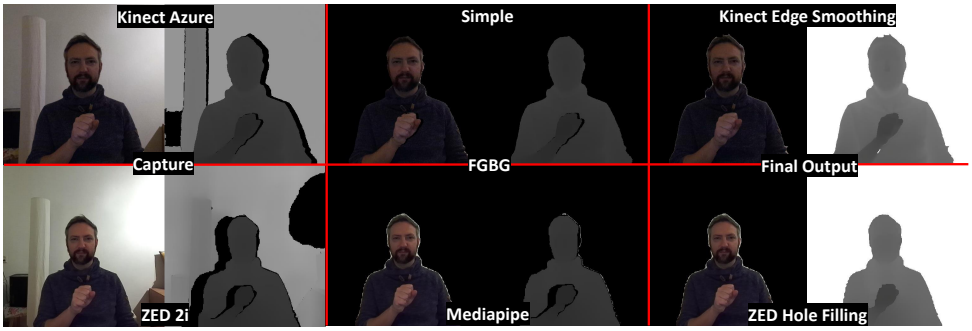


Figure 2.2: Capture Example - Capture & FGBG & Final Images (Top row = Kinect; Bottom row ZED)

Camera Frame Grabber

The first module in the capture chain is a frame grabber to allow simple access to different capture sensors. The main function of this module is to extrapolate any dedicated capture API and to supply a steady real-time flow of synchronized RGB color and depth image frames. Currently, we implemented two examples of this module to support the Kinect

[1] https://web.archive.org/web/20240214001251/https://opencv.org/
[2] https://web.archive.org/web/20240502055447/https://numba.pydata.org/

Azure sensor (based on [3]) and ZED 2i (based on [4]). In addition to images, this module also offers generic metadata of the camera intrinsic (i.e., resolution and focal length), which are important for the correct rendering of the image into the 3D plane. The implementation of the two sensor modules currently offers a high- and low-capture mode with an output resolution of 1024x1024 pixels (high) and 512x512 pixels (low). The image capture is internally cropped, and all metadata is adjusted accordingly. This is, however, not a limitation of our approach but a deliberate decision on order to optimize different parts of the end-to-end chain, as an image with a resolution power of 2 can be handled more performantly in most image/video compression and rendering engines. This is specifically the case for the image mapping in WebGL rendering (as internally any image will be scaled to a power of two for texture-to-shader mapping). Furthermore, the capture rate can be fully adjusted (based on sensor capabilities). For the remainder of the paper, we refer to the capture rate of 15fps. In the following, we explain the main differences between the two sensors.

Kinect Azure (K4A)[5]: The K4A sensor is a Time-of-Flight (ToF) sensor. In addition to a hi-resolution color sensor, it utilizes two near-infrared (NIR) laser diodes enabling near and wide field-of-view (FoV) depth by measuring the time the (modulated) infrared laser beam takes from sending to sensor reading. This gives accurate depth approximations with a limited resolution (640 × 576 px in the near field of view, NFOV, mode). This resolution is still suitable for human capture (in a range of 3 meters). It is important to note that the depth image is also significant scaled with a complex mapping to the color pixels, depending on the color capture resolution. More details on the accuracy of the Kinect Azure are presented in [99]. Although the depth capture quality is generally high, the raw capture may still include different types of distortion. An example of a raw human capture can be seen in Figure 2.2 (image on top left). For example, misalignment between the infrared (IR) sensor (to measure depth) and the color image can create a halo effect, resulting in wrong or no depth values in certain sections of the image. This problem is particularly undesirable around the hand of the user (i.e., in the raw image part of the body, even though color information is present, would be ignored in the 3D rendering). One potential disadvantage of the K4A is that it may cause IR interference with any other devices that may rely on IR device tracking (like the HoloLens 2 AR glasses or the SteamVR Lighthouse tracking system).

ZED 2i[6]: The ZED 2i sensor is a stereo camera with two high resolution color cameras in a predefined fixed setup. With the disparity of the 2 synchronized video streams and (CUDA, hardware accelerated) stereo matching, the sensor provides both color and depth information. As the full image is used for the depth estimation step, the depth image completely resembles the resolutions of the color image. This can also be accounted for in case of high resulting capture (e.g. 2k or 4k) even though some depth estimation steps might be internally scaled by the SDK (to allow real-time capture with low performance overhead). More details on the accuracy of the ZED 2i sensor (and similar stereo matching depth sensor devices) are presented in [100]. Furthermore, the ZED SDK offers two

[3] https://web.archive.org/web/20230618070726/https://github.com/etiennedub/pyk4a/
[4] https://web.archive.org/web/20240415025624/https://www.stereolabs.com/docs/app-development/python/install/
[5] https://web.archive.org/web/20240212204916/https://azure.microsoft.com/en-us/products/kinect-dk/
[6] https://web.archive.org/web/20240314220950/https://www.stereolabs.com/zed-2i/

other important parameters related to depth capture: depth_mode and sensing_mode. In our current implementation, we use depth_mode = DEPTH_MODE.QUALITY. This was determined to be the best option. The alternative mode DEPTH_MODE.ULTRA seems quite pixelated and has a less consistent depth granularity. As sensing_mode we use SENSING_MODE.STANDARD. Compared to the alternative SENSING_MODE.FILL, standard sensing produces a more consistent depth image. The SENSING_MODE.FILL option's main drawback is that it generates a lot of overlap and causes objects and body parts to blend into the backdrop, leading to inconsistent depth sensing findings. Both depth_mode and sensing_mode are fully configurable as parameters in the ZED capture module. An example of a raw human capture can be seen in Figure 2.2 (bottom left image). Compared to the K4A the ZED offers a more uniform depth image but may result into multiple distortions based on the limitations of the stereo matching approach. First of all, the ZED depth image quality often appears more noisy (compared to K4A). Furthermore, uniformly colored surfaces cannot be correctly matched and cannot be represented as depth (see the black blob on the top right of the raw ZED capture, Figure 2.2). This is not necessarily an issue for humans (as body parts or cloth is rather uniformly colored, particularly in common lighting conditions). Finally, we can observe similar image misalignment distortions (see users hand, Figure 2.2) in the ZED capture as in the K4A capture. In addition, these misalignment distortions often lead to distortions in the outline of the person as well. However, one clear benefit of the ZED sensor is that it does not produce any IR interference with any other devices that may rely on IR device tracking (such as the HoloLens 2 AR glasses or the SteamVR Lighthouse tracking system). Furthermore, please note that currently due to CUDA processing requirements the ZED 2i sensor is only supported on a PC with an NVIDIA graphics card.

Foreground-Background extraction (FGBG)

This module's main goal is to enhance the quality of the image while doing real-time foreground-background extraction (FGBG). This is a crucial step, since, when capturing users, we are only concerned with the user's representation and not their background. Additionally, doing so enables us to visually render the users' representation into the XR environment (i.e. virtual environment for VR or augmented into the real environment for AR).

For the K4A image capture we implemented a simple threshold based FGBG method, removing any pixels out of the threshold. This is done by simply overwriting any values out of the threshold with 0 both for the depth and color. This is we currently allow pixels with a depth value in the range of 500 to 2000mm away from the camera. This is based on the following assumptions:

- user will be at least 50 cm away from the camera
- main body is around 1.5 meter away from the sensor
- we only encode a depth range of (approx.) 1.5 meter

As we do not consider the depth images captured by the ZED 2i sensor as reliable as from the K4A, a simple FGBG threshold might not be good enough to provide a high-quality reconstruction. Therefore, we implemented an enhanced FGBG module using a machine

learning approach for human body segmentation (based on Mediapipe [7]). This offers a fine grain body segmentation map (with reasonable performance overhead due to Mediapipe's hardware acceleration, running on the GPU). Example images of the FGBG modules can be seen in Figure 2.2, simple FGBG based on K4A capture (top middle) and enhanced Mediapipe FGBG based on ZED 2i capture (bottom middle).

Other Image enhancements

After the FGBG processing, further enhancements to the image are desirable for a high-quality user representation. That is, we like to clean the image from any distortions and holes. As these holes and image imperfections are rather different for the raw images of the two examples sensor implementations presented in this chapter (K4A and ZED) we implemented two very distinctive enhancement modules. Enhancements and further spatial processing blocks could easily be introduced in the future.

ZED Hole Filling: This module has the main aim to improve the depth image; that is: A. fill small holes and remove imperfections and B. fill any big holes that might be present due do capture misalignment. These image enhancements rely on the (Mediapipe) image segmentation of the previous FGBG module and encompass the following steps:

- Step 0: As a basis for this image enhancement the Mediapipe segmentation from the FGBG module already removed unnecessary pixels from the color and depth image.
- Step 1: clean the edges with cv2.dialate followed by cv2.erode (both with kernel 5 and 2 iterations)
- Step 2: build a mask for all holes (empty depth values that in the segmentation are detected as person)
- Step 3: build a temporary "filler depth image" by copying the depth image and applying cv2.erode (kernel size 8 and 4 iterations) followed by cv2.dilate (kernel size 8 and 8 iterations), this will fill all big holes but also fill across the segmentation map
- Step 4: finally we combine the depth image with the "filler depth image" (from the previous step) by only replacing missing values in the depth image

The resulting depth image fills the complete segmentation and thus offers a value for each color pixel. An example output can be seen in Figure 2.2 (bottom right). Note that the Mediapipe segmentation may result into a halo around the cutout (part of the background is detected as person). This might result into less sharp edges than the final image of the K4A sensor but often result into better rendering of hair that might be cutout in the K4A image. Further, note that the edge of the image will be further optimized in the rendering of the image.

Kinect Edge Smoothing: This module has the main aim to improve the misalignment of the K4A image on the edges and of any foreground objects (including the users hand) in particular. This step is based on the simple FGBG and thus only relies on OpenCV[1] functions. The different enhancement steps are explained in the following:

[7] https://web.archive.org/web/20240406130828/https://google.github.io/mediapipe/

- Step 1: first we need to resize the image in case of high resolution (this means the optimization will always we done in 512 pixel to ensure real-time low performance execution)
- Step 2: create image outline with cv2.Canny
- Step 3: match edges from cv2.Canny algorithm with depth values
- Step 4: fill edges and create contours with cv2.findContours
- Step 5: find biggest contour (this is our person / including holes)
- Step 6: identify a depth background threshold averaging the depth image - 10cm (we asume that in K4A the foreground object, like a hand, will result into holes in the back of the image, so we like to will the background with background depth values)
- Step 7: create a temporary background depth image by coping the depth image and removing all depth values smaller than the background threshold
- Step 8: now we loop 5 times, creating new depth values with a cv2.dilate (kernel 5) and only replace the new values into the temporary background depth image (Note: this works significantly different than cv2.dilate iterations would work, as it would also alter the depth information of pixels that we consider as “correct")
- Step 9: finally we replace any missing values in our original depth image with values from the temporary background depth image (based on the detected biggest contour)

The final image successfully closes any gaps in the depth image edges and offers a sharp and coherent matching of the depth values to its color counter pixel. An example output is shown in Figure 2.2 (top right).

Depth Conversion

In order to transmit depth over existing video / image compression formats, the 16-bit depth data is converted into an 8-bit RGB color format, with a window of 1530 values. Currently we offer two flavors of this conversion GrayAVG, presented in [33], and Intel HSV[8], both implemented in Numba[2] to offer maximum hardware acceleration with Python.

GrayAVG: We use the GrayAVG depth conversion algorithm as presented in [33]. GrayAVG maps depth values with a range of 0 to 1.5 meters (i.e., 1530 values) into gray-color values on the entire RGB color space. By adding a minimum distance (distance the user is minimally away from the sensor) as metadata, this offers a flexible distance. We usually assume a minimal distance of 50cm, thus transmitting a depth range of 500 to 2030mm. To stream it as a single RGB-D video stream, the RGB image and the grayscale depth image are concatenated.

Intel HSV: We use our own implementation of [8], in Python and Numba[2]. This algorithm also converts the 16 bit depth data into 8-bit RGB color by utilizing the Hue color. The hue-color space offers 1529 discrete levels, or approximately 10.5 bits, and 6 gradations

[8] https://web.archive.org/web/20230924160801/https://dev.intelrealsense.com/docs/depth-image-compression-by-colorization-for-intel-realsense-depth-cameras

in the R, G, and B up and down directions. Furthermore, the image never gets too dark because one of the rgb colors is always 255. This has the advantage of making sure that some lossy image/video compression does not lose details.

Output

Besides rendering the output on the screen, the output can be written into different virtual webcam drivers based on pyvirtualwebcam[9]. Currently, this supports two different virtual webcam drivers: OBS[10] and Unity Capture[11]. The main difference between these two drivers is that the OBS virtual driver operates in RGB format, while the Unity Capture virtual driver operates in RGBA format. Thus, choosing the right driver is important depending on the application accessing the data. For example, Unity-based applications generally expect RGBA format from webcam device, while for a browser the alpha channel is usually ignored. The output module will simple copy the final frame buffer (with or without adding an alpha) to the virtual device.

2.2.3 Transmission

The concept of our approach is based on converting any depth information into a 2D RGB image format (see Section 2.2.2) that can be transmitted via existing 2D encoding and distribution technology. This concept was previously reported in [33, 59]. Compared to newer volumetric streaming formats (such as V-PCC [71] and G-PCC [65]) this allows the use of reliable and established real-time streaming components (including full hardware accelerating), allowing high quality stream decoding even on low-powered end devices (like the Microsoft HoloLens 2 [12] and Oculus Quest 2[13]). The actual transmission is encoding agnostic as long as the system supports any type of 2D RGB video format. Examples of applications transmitting the RGBD data are presented in Section 2.3.2.

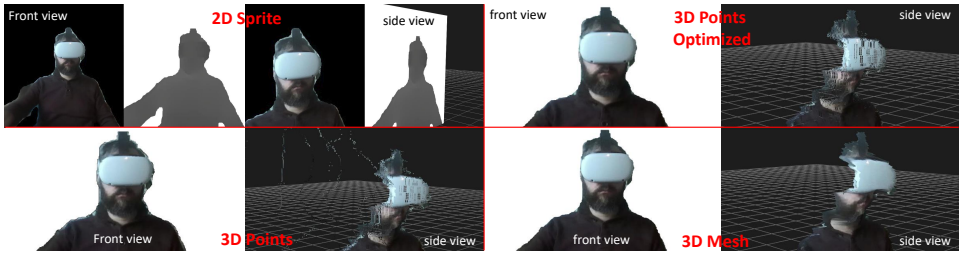


Figure 2.3: Render Example (RGBD Capture with ZED sensor)

2.2.4 RGBD 3D Rendering

In this section we outline our approach on how to render the 2D RGB stream in 3D and what optimizations we can apply in the rendering shader to increase the visual quality. One

[9] https://web.archive.org/web/20240314043542/https://github.com/letmaik/pyvirtualcam
[10] https://web.archive.org/web/20240215044704/https://obsproject.com/
[11] https://web.archive.org/web/20231215032005/https://github.com/schellingb/UnityCapture
[12] https://web.archive.org/web/20240213114005/https://www.microsoft.com/en-us/HoloLens/
[13] https://web.archive.org/web/20240419023753/https://www.meta.com/nl/en/quest/products/quest-2/

important aspect of our approach is that the reconstruction of the RGBD image into 3D space does not require any additional processing steps but is directly executed in the shader code on the GPU and thus increases reliability and reduces resource overhead. Further, this allows us to apply pixel optimizations within the shader and different rendering techniques (i.e. as points or meshes) as explained in the following (in the example of WebGL).

Depth-based rendering

Rendering the RGBD values back into 3D space is a simple geometric function based on the depth value and the intrinsic value of the camera (center pixel and focal length). Thus in order to render points correctly first the depth value needs to be reconstructed from the 2D image (HSV or grayscale) to a z-value in meter followed by the back projection.

For example, for GrayAVG (depth grayscale), the depth-to-z conversion can be done by combining the grayscale value and multiplying it by 1.53 (so that it represents a range of $0-1.53$ meters) and adding the fixed minimal user distance. In a WebGL shader this looks as follows:

$$float\ z = ((color.r + color.g + color.b)/3. * 1.53) + minDepth; \tag{2.1}$$

The back projection will look as follows in a WebGL shader. Note that the camera_center and camera_focalLength is the metadata coming from the capture frame grabber module of the user to be rendered (e.g. the sending application).

$$vec4\ position = projectionMatrix * modelViewMatrix * userTransformationMatrix * vec4(\tag{2.2}$$

$$((pixel\_position.x) - camera\_center.x) * z/camera\_focal\_length.x, \tag{2.3}$$

$$((pixel\_position.y) - camera\_center.y) * z/camera\_focal\_length.y, \tag{2.4}$$

$$z, 1.0); \tag{2.5}$$

The above examples are in WebGL, to render points in space (effectively a point cloud), the shader will write the values in a preallocated array of geometry point vertices[14](an example is shown in Figure 2.3, 3D Points). Changing this preallocated array into a PlaneBufferGeometry[15] also allows rendering the user with all points connected to each other (thus effectively as a Mesh; an example is shown in Figure 2.3, 3D Mesh). The actual rendering is application dependent and can easily adopted, for example, into the different Unity render pipelines.

Edge Smoothing

Simply rendering the RGBD image in 3D may lead to different distortions and imperfections. This can particularity affect the distortions in the edges of the image, based on errors in the capture, FGBG or image compression (i.e. YUV conversion can also inject errors in the corners of the image as encoding always combines information of 4 adjacent pixels). An example of this problem can be seen in Figure 2.3 (3D Points, bottom left), a halo around the user is stretched into space. To counter this effect and increase the rendering

[14]https://web.archive.org/web/20240210024518/https://threejs.org/docs/#api/en/core/BufferGeometry
[15]https://web.archive.org/web/20240210024518/https://threejs.org/docs/#api/en/geometries/PlaneGeometry

quality, we apply a simple edge smoothing. The edge smoothing we apply is based on the assumption that every adjunct pixels are connected to each other in space (different parts of a users body are connected). This means that any pixel that is adjunct but distant in space exceeding a specific threshold can be ignored. Thus, for each pixel, we calculate its distance to each neighboring pixel (x and y +/- one) and apply two thresholds smoothing_threshold and cutoff_threshold. For the cutoff_threshold if any neighboring pixel is over that distance, we will not render this pixel. For the smoothing_threshold if any neighboring pixel is under that distance we average z value with that pixel to create a more smooth surface. Our best practice showed an ideal smoothing_threshold of 3 cm and a cutoff_threshold of 10 cm. An example of the edge smoothing as WebGL shader code is given in the following (please note that in the actual implementation the thresholds are not hard-coded but fully and dynamically configurable):

$$float\ depthDiff = distance(z,\ pixel\_neighbor.z); \tag{2.6}$$
$$if(depthDiff >= 0.1)\{ \tag{2.7}$$
$$\quad gl\_PointSize = 0.0; \tag{2.8}$$
$$\quad gl\_Position = vec4(0.0,\ 0.0,\ -1.0,\ 0.0); \tag{2.9}$$
$$\quad return;\} \tag{2.10}$$
$$if(depthDiff <= 0.03)\{ \tag{2.11}$$
$$\quad z = (z + pixel\_neighbor.z)/2.;\} \tag{2.12}$$

Point Size

Ultimately, when rendering a user representation as a point cloud, each point will be rendered as pixels on the display. The final optimization step of the 3D rendering is to adjust the pixel size of each point based on the distance in space. The reason for this is that by default a capture sensor will capture more dense points if they are closer to the capture device rather than points that are far away. This results in that points further away (even though they are physically close) may be rendered with space in between them, and thus diminish the visual representation (note that this is only relevant when rendering the 3D representation as points, in a Mesh representation points are already connected by default). Thus, ideally, we would like to render points in the same size as the minimal distance to its direct neighbors (neighbors at the same distance in space) according to the focal properties of the capture device. This results in the following formula as WebGL shader code:

$$gl\_PointSize = z\ /\ camera\_focal\_length.x * 1000.; \tag{2.13}$$

Note that in WebGL we cannot create non-uniform pixel sizes on the x and y axes, and thus create the pixel size only based on the x axis. In Unity (e.g. VFX graph) more complex rendering techniques are possible. Further, in WebGL the z value is in meters, while the camera focal length is in mm, thus requiring a multiplication by 1000.

The combination of the “Point Size" and “Edge Smoothing" (section 2.2.4) are shown as example in Figure 2.3, 3D Points Optimized)

2.3 Results & Evaluation

We evaluate our modular capture application in two ways, a technical evaluation of the operation of the different modules (Section 2.3.1) and an integrated application evaluation (Section 2.3.2) of two use cases with a small-scale user study each. The technical evaluation measures the processing delay, CPU, GPU and memory resource usage of each individual module of the capture application, as well as the CPU, GPU and memory resource of the different rendering methods proposed. The application evaluation presents two XR communication solutions integrated with a dedicated capture instantiations.

2.3.1 Technical Evaluation

In this section we present different performance measures for the capture and rendering as proposed in Section 2.2. For the different (processing and rendering) performance measures, we use a customized version of the Resources Consumption Metrics (RCM) [101][16]. The RCM utility is a native Windows application that enables 1-second interval system statistics capture for the CPU, GPU, and memory utilization per process. In all measurements, our capture application was deployed on a XMG NEO 15 [E21] laptop PC (with AMD Ryzen 9 5900HX, GeForce® RTX 3080 and 32GB RAM). With a capture framerate of 15fps.

For (capture-to-display) processing delay measurements, we use VideoLat [102][17]. VideoLat is a tool for dedicated one-way delay measurements. A PC (Mac) with a webcam serves as a measuring device, constantly depicting QR-codes and measuring the time from display to webcam capture. VideoLat is calibrated in an initial step by pointing its on camera to the own screen, it measures the default (hardware) delay of the system (which will be subtracted from subsequent measures). For each measure condition, our capture application records the images of VideoLat and displayed them locally (while deploying different module configurations). In our setup VideoLat was used on a MacBook Retina (15-inch, early 2013 model with 2.4 GHz Intel Core i7, 8GB RAM and MacOS 10.13.6) with a Logitech Brio[18] as capture camera.

Capture Performance

We measure the delay of each capture module by measuring the displayed output of each module against the delay from frame grabbing with VideoLat. Different module configurations were deployed and tested as shown in Figure 2.1, with an additional step of a video conferencing application displaying the results of the virtual webcam. This is to show the delay in a realistic deployment, we used Microsoft Teams (MS Teams)[19] to show a self-view of the virtual webcam output.

For all ZED sensor conditions this means the following module deployment:

- Capture = ZED Frame Grabber
- FGBG = Mediapipe ML FGBG

[16] https://web.archive.org/web/20230524052810/https://github.com/ETSE-UV/RCM-UV
[17] https://web.archive.org/web/20240113164432/https://videolat.org/
[18] https://web.archive.org/web/20240306140702/https://www.logitech.com/nl-nl/products/webcams/brio-4k-hdr-webcam.html
[19] https://web.archive.org/web/20240216070617/https://www.microsoft.com/en-us/microsoft-teams/group-chat-software

- Enhance = ZED Hole Filling
- Depth = GrayAVG
- Output = OBS Virtual Webcam
- Application = Microsoft Teams (camera self view)

For all Kinect Azure sensor (K4A) conditions this means the following module deployment:

- Capture = Kinect Frame Grabber
- FGBG = Simple FGBG
- Enhance = Kinect Edge Smoothing
- Depth = Intel HSV
- Output = Unity Virtual Webcam
- Application = Microsoft Teams (camera self view)

Table 2.1: Capture Module Processing Delays (VideoLan; 500+ samples each; mean values in ms; maximum confidence interval {alpha = 0.05} of 5ms; standard deviation as ±)

Sensor	Res.	Capture	FGBG	Enhance	Depth	Output	Application
K4A	512	156 (± 32)	159 (± 37)	194 (± 42)	241 (± 41)	319 (± 40)	297 (± 40)
K4A	1024	189 (± 33)	225 (± 45)	333 (± 46)	480 (± 57)	572 (± 58)	573 (± 59)
ZED	512	209 (± 19)	240 (± 36)	252 (± 34)	287 (± 30)	285 (± 18)	390 (± 37)
ZED	1024	245 (± 38)	324 (± 39)	380 (± 42)	429 (± 52)	416 (± 45)	475 (± 48)

Table 2.1 shows the different delays measured after each module. Overall, to read the RGBD information in an application (i.e., MS Teams), the delay is approximately doubled to the input (fame grabber capture) delay. Further, the capture delay of the Kinect Azure can be seen as comparable to a “normal" webcam (e.g. the Logitech Brio calibration delay in VideoLat was measured with approx. 190ms) while the ZED capture delay is slightly higher, which is expected as to the expense of stereo matching depth image creation. Otherwise, all processing delays were in a somewhat expected range. However, it is also important to recognize that higher resolution capture significantly increases processing delay. With the highest delay being over 500ms (K4A sensor, high resolution in MS Teams), this configuration might not be suitable for every XR communication use case.

To measure the resource utilization of each individual module, we added a “pass-through" debug option into each module. In pass-through mode, each module does not execute any processing but simply forwards pre-defined example data (i.e., data that are representative for the normal operation of this module). With this feature, we tested each individual module by setting all modules, besides the one to test, into pass-through and measure the CPU, GPU and memory usage via RCM. The module deployment is otherwise identical to the delay measures, with two additional conditions:

- None = No module active, only pass through
- All = All modules active, “normal" operation of the capture application

Table 2.2 shows the measurements of the different individual modules. As mentioned in all conditions (besides “All" and “None"), only one module is fully active in each condition. Overall, the results show a stable and acceptable performance overhead, even for the higher resolution RGBD streams. This might offer some possibility to utilize further resources in order to add more spatial image recognition and improvement tasks (like HMD replacement) or to further split processing into more parallel tasks to further decrease the processing delay.

Rendering Performance

In order to evaluate the different rendering options (see Figure 2.3) we deployed a dummy capture module and used the VRComm system [33] with different devices and browsers as clients. The dummy capture module facilitates a playback of representative pre-recorded RGBD content based on the following modules: Azure Kinect Frame Grabber (with a resolution of 1024 and 512 pixel), Simple FGBG, Kinect Edge Smoothing, GrayAVG, and OBS Virtual Webcam. Besides the three 3D rendering conditions (3D Points, 3D Points Optimized and 3D Mesh), as presented in Figure 2.3 (see Section 2.2.4), we added two baseline conditions: (1) “None" not rendering any user and (2) “2D" rendering the user as a 2D sprite. This is the case where for each condition (besides “None") an upload client is uploading a RGBD video stream into VRComm (WebRTC/peer-to-peer) and the receiving client (marked as “Device / Browser" in the table) renders the RGBD image texture according to the condition. The performance of the “PC" condition was measured with RCM and the performance of the “HoloLens 2" condition was measured with the HoloLens Windows Device Portal[20]. For the HoloLens 2 we measured the complete device performance (only running the browser client). Under all conditions, the memory usage of the HoloLens 2 device was measured below 3GB.

Table 2.3 shows the rendering performance under the different device / browser and rendering conditions. Overall, the different browsers all perform very similar. Generally, all three browsers are very capable for WebXR processing and rendering. However, it is important to note that even though Chrome&Edge indicate a significantly higher GPU usage than Firefox, this is mainly due to internal rendering fame rate management. While Chrome&Edge targeted a framerate of approximately 90 fps, Firefox fluctuated around approximately 60 fps. On the Hololens 2 however we could observe a drop in framerate due to reaching the maximum resource utilization. While most conditions (None and 2D) run with the expected 60fps, including the 512 3D Mesh condition, the higher resolution and point cloud conditions decreased in framerate (512 3D and 3D Optimised had a framerate of ~30fps, 1024 3D and 3D Optimised had a framerate of ~15fps, and 1024 3D Mesh had a framerate of ~30fps). We can observe that the different configurations are technically suitable for photorealistic XR communication as previously observed in [[33, 59, 103]]. Adding multiple users with high resolution might still be challenging according to the high GPU usage in our measures (this is also because for the high resolution more than 1M pixels

[20] https://web.archive.org/web/20231229012417/https://learn.microsoft.com/en-us/windows/mixed-reality/develop/advanced-concepts/using-the-windows-device-portal

point needs to be rendered and constantly updated). However, the different 3D render options (and whether optimizations were applied) had little influence on the resource usage, and thus can be chosen according to the use case and users needs. This is for the Hololens 2 the 3D Mesh rendering seems to result into a better technical performance.

Table 2.2: Capture Performance of different modules (RCM; 5min samples each; average and SD values; maximum confidence interval {alpha = 0.05} is CPU = 1.98%, GPU = 4.3%, and Memory = 48.86MB)

Sensor	Res.	Measure	None	Capture	FGBG	Enhance	Depth	Output	All
K4A	512	CPU in %	1.22	1.62	0.94	3.2	1.5	1.04	4.13
		SD in %	0.85	1.16	0.54	2.46	0.89	0.71	1.9
K4A	512	GPU in %	7.65	10.02	6.73	6.35	6.72	6.56	10.62
		SD in %	4.39	2.18	1.6	1.85	1.68	1.68	2.27
K4A	512	Memory in MBs	253.63	219.87	254.12	268.49	294.51	262.33	277.74
		SD in %	40.67	33.49	38.77	2.32	2.82	1.92	6.09
K4A	1024	CPU in %	3.25	5.01	3.3	7.06	7.33	2.88	15.58
		SD in %	3.13	2.63	4.06	5.04	2.76	2.38	6.1
K4A	1024	GPU in %	6.49	13.52	6.63	6.68	6.71	6.74	13.36
		SD in %	1.88	2.62	1.68	1.6	1.63	1.64	2.55
K4A	1024	Memory in MBs	388.71	321.06	387.27	406.81	420.41	406.21	414.34
		SD in %	60.31	3.05	60.05	4.13	66.81	3.79	68.59
ZED	512	CPU in %	0.27	2.28	0.0	0.45	0.04	0.37	6.71
		SD in %	0.38	1.02	0.0	0.56	0.1	0.55	1.65
ZED	512	GPU in %	0.0	27.45	0.0	0.0	0.0	0.0	15.34
		SD in %	0.0	12.88	0.0	0.0	0.0	0.0	13.09
ZED	512	Memory in MBs	406.19	393.5	231.35	421.38	406.63	417.79	480.18
		SD in %	62.13	60.24	34.81	0.69	0.18	0.35	20.47
ZED	1024	CPU in %	1.17	4.81	6.8	2.77	0.08	1.06	16.03
		SD in %	1.23	2.36	4.21	1.3	0.14	0.68	4.2
ZED	1024	GPU in %	0.0	22.78	0.0	0.0	0.0	7.38	16.08
		SD in %	0.0	4.0	0.0	0.0	0.0	3.79	9.08
ZED	1024	Memory in MBs	587.61	538.67	780.41	609.11	621.25	255.36	758.06
		SD in %	93.53	85.77	71.5	4.52	1.01	31.97	144.41

Table 2.3: Render Performance of different Browsers (RCM; 5min samples each; average and SD values; maximum confidence interval {alpha = 0.05} is CPU = 1.35%, GPU = 1.17%, and Memory = 8.81MB)

Device / Browser	Res.	Measure	None	2D	3D	3D Optimized	3D Mesh
PC / Chrome	512	CPU in %	7.04 (±1.07)	8.7 (±1.25)	8.57 (±1.35)	8.69 (±1.18)	8.84 (±1.22)
PC / Chrome	512	GPU in %	34.09 (±6.81)	39.17 (±3.74)	49.12 (±4.1)	48.83 (±4.89)	49.43 (±6.52)
PC / Chrome	512	Memory in MBs	601 (±50)	501 (±31)	531 (±18)	541 (±32)	514 (±49)
PC / Chrome	1024	CPU in %	7.71 (±1.34)	9.09 (±1.19)	9.81 (±1.64)	9.66 (±1.53)	9.76 (±1.42)
PC / Chrome	1024	GPU in %	20.26 (±6.52)	31.69 (±2.63)	68.09 (±6.62)	69.71 (±8.38)	62.21 (±8.31)
PC / Chrome	1024	Memory in MBs	446 (±26)	522 (±18)	603 (±35)	604 (±39)	637 (±35)
PC / PC / Edge	512	CPU in %	4.85 (±0.99)	37.24 (±2.04)	18.06 (±2.24)	12.79 (±2.36)	17.75 (±2.35)
PC / PC / Edge	512	GPU in %	23.79 (±7.49)	58.64 (±7.59)	67.21 (±3.54)	65.35 (±3.23)	63.1 (±4.47)
PC / PC / Edge	512	Memory in MBs	451 (±22)	540 (±78)	614 (±81)	566 (±84)	626 (±73)
PC / PC / Edge	1024	CPU in %	3.58 (±1.1)	34.19 (±1.96)	13.63 (±2.54)	13.86 (±2.7)	8.05 (±1.85)
PC / PC / Edge	1024	GPU in %	21.74 (±5.0)	29.55 (±4.55)	48.53 (±6.68)	48.82 (±6.38)	76.47 (±2.33)
PC / PC / Edge	1024	Memory in MBs	388 (±23)	585 (±17)	675 (±34)	689 (±53)	751 (±15)
PC / Firefox	512	CPU in %	2.69 (±0.98)	8.9 (±1.37)	8.79 (±1.4)	8.63 (±1.42)	8.54 (±1.54)
PC / Firefox	512	GPU in %	19.78 (±2.96)	49.63 (±6.08)	55.65 (±6.58)	55.33 (±6.04)	56.26 (±6.16)
PC / Firefox	512	Memory in MBs	399 (±11)	501 (±52)	499 (±53)	526 (±15)	534 (±53)
PC / Firefox	1024	CPU in %	3.35 (±1.17)	9.76 (±1.34)	9.97 (±1.36)	9.95 (±1.38)	9.96 (±1.41)
PC / Firefox	1024	GPU in %	32.34 (±8.77)	30.1 (±2.62)	54.02 (±6.1)	55.29 (±5.58)	49.1 (±5.26)
PC / Firefox	1024	Memory in MBs	447 (±13)	559 (±9)	691 (±48)	700 (±26)	701 (±13)
HoloLens 2 / Edge	512	CPU in %	30.15 (±3.79)	35.54 (±2.6)	32.36 (±3.41)	30.02 (±3.84)	35.66 (±2.12)
HoloLens 2 / Edge	512	GPU in %	33.0 (±1.96)	40.44 (±3.93)	80.88 (±11.28)	78.64 (±13.44)	85.26 (±2.83)
HoloLens 2 / Edge	1024	CPU in %	30.37 (±7.8)	33.41 (±12.03)	29.06 (±3.89)	27.5 (±4.52)	28.63 (±4.23)
HoloLens 2 / Edge	1024	GPU in %	32.75 (±3.67)	38.62 (±10.42)	88.53 (±4.58)	91.7 (±4.9)	80.77 (±4.76)

Figure 2.4: Side view of Example User inside the immersive VR communication application (Connec2) [104]

2.3.2 XR Application and Use Cases Evaluation

Virtual Reality for Business Meetings

Face-to-face (f2f) meetings are widely acknowledged to be the most productive and interesting way to conduct business meetings [105]. One of the causes is that meetings using existing 2D videocon ferencing solutions often lack engagement and effectiveness. Participants in meetings often struggle with background noise, lack of social presence, and not recognizing who is speaking. Most of the time, these problems arise in large group meetings [106]. Thus, we assume that business meetings are a good testing use case for immersive communication in VR. To test our modular capture system in a business meeting use case, we developed an immersive VR communication tool that integrates 3D photorealistic capture and rendering (RGB + depth) into the Connec2[21] commercial VR application. Our approach enables movement in a 3D environment with auditory and visual spatial awareness, in the ideal view frustum. This is due to deploying only a single RGBD camera solution, individuals are advised to stay in a small area in order to prevent 3D point cloud distortions (i.e., simply because sections that are not captured cannot be displayed). However, staying in a small, confined space can be considered normal for most business meetings.

VR User Study Setup To evaluate the technical feasibility of our modular capture application, we created a unique communication tool that integrates the presented capture application (for 3D photorealistic user representations) into the commercial VR application and communication platform Connec2. The capture application consisted of the following modules: ZED Frame Grabber (with a resolution of 512x512 pixel), Mediapipe ML FGBG, ZED Hole Filling, GrayAVG, and OBS Virtual Webcam. Deployed with a single RGBD (ZED) sensor and an Oculus Quest for business (same hardware as Oculus Quest 2). Figure 2.4

[21] https://web.archive.org/web/20240201143219/https://connec2.nl/

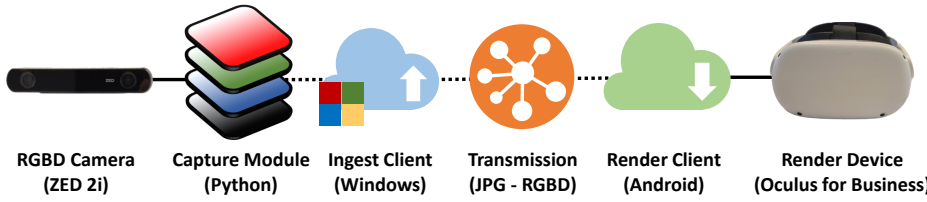


Figure 2.5: Application Pipeline VR [107]

shows an example of the rendering of a remote user within the Connec2 VR communication application.

The complete application pipeline can be seen in Figure 2.5, and includes the following components:

- Modular capture as presented in Section 2.2
- Ingest client receiving RGBD images from the OBS virtual webcam driver and uploading them into the Connec2 system
- Connec2 utilizes a motion JPEG compression sent via structured networked message queues
- The render client runs on an Oculus Quest device sending/receiving audio and rendering the audio as well as the 3D point cloud (via VFX-graph)

Part of this study was previously presented in [107]. In total, 12 participants (6 meeting pairs) used the Social VR system to have 1 or 2 meetings with each other from various offices in The Netherlands. There are 11 male subjects and 1 female subject, and the participants' ages range from 20 to 60+. All pairs already knew each other through their work on previous or ongoing projects. Subsequent encounters were held with each other using the VR system over a 12 week period. All participants were required to complete a questionnaire after each meeting that contained questions based on several established surveys on social presence, immersion, usability, embodiment, quality of experience (QoE), quality of interaction (QoI), and quality of communication (QoC) [10, 108–114].

Table 2.4: Overview of variables of VR communication, scores are presented as mean average with standard deviation (SD) and confidence interval (CI; alpha = 0.05)

Variable	Embodiment	QoS	Usability	QoE	QoI	Immersion	Meeting Engagement	Social Presence	QoC
Mean	-1.44	-0.81	-0.30	-0.25	-0.10	0.61	0.68	0.75	0.90
SD	1.24	1.26	1.17	0.78	0.87	1.04	0.79	0.65	0.96
CI	0.52	0.54	0.49	0.33	0.37	0.43	0.33	0.27	0.40

Evaluation To analyze the scores of the different measured aspects of the communication and the system, the answers to the questions corresponding to either embodiment, Quality of Service (QoS), usability, Quality of Interaction (QoI), immersion, social presence, and Quality of Communication (QoC) have been combined for their respective elements. The average rating for each of the 22 meetings for each of the elements for which the data was gathered is shown in Table 2.4. The average score for the embodiment was the lowest, coming in at -1.44 on a scale from -3 to 3, with 0 being neutrality. Positive numbers show a satisfactory or excellent response to the element, whilst negative values show an inadequate or bad response from the participants. With a score of 0.9, QoC received the best rating of all elements. Thus while technically users were able to operate and use the system with an acceptable level of social presence and quality of communication, there is still room to improve many of the technical aspects. Some of the limitations in this study are the low image resolution of 512 pixels, the VFX graph used in Unity did not apply edge smoothing, and the users face being occluded by the VR HMD. Currently, (whiteout applying any HMD replacement strategy) using HMDs is counterintuitive for photorealistic communication as it obfuscates the face and prevents eye gaze.

Figure 2.6: Example User experiencing immersive communication in AR (with RGBD capture shown on the screen)

Augmented Reality for Personal Communication

A demographic that is often neglected in daily personal communication is the elderly. There is a severe absence of contact between elderly residents of nursing facilities and their families and loved ones. Elderly people's health, cognitive decline, and quality of life are negatively impacted by social isolation and loneliness. Furthermore, the Covid-19 measures applied in care institutions in the Netherlands further "increased loneliness and restricted decision-making for" care home residents [115]. To address this problem, an AR tool with 3D user capture and rendering on an iPad and a large TV screen was previously presented in [60, 116] and enabled high-quality mediated social communication (effectively

rendering remote users is life size). We identified this use case to integrate and test our capture application in AR communication (i.e., with AR glasses).

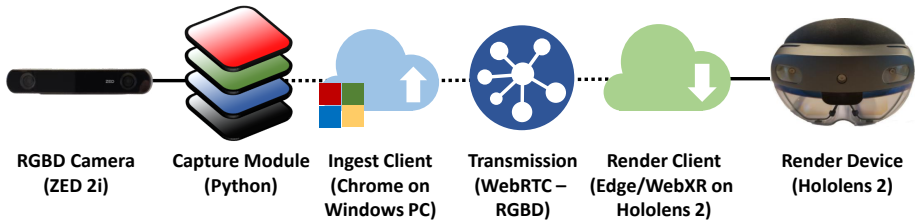


Figure 2.7: Application Pipeline AR

AR User Study Setup To test our capture application within an AR communication use case, we integrated our capture application into a web-based immersive communication platform called VRComm [33]. VRComm is a web-enabled (WebXR) XR system that allows both VR and AR rendering on OpenXR devices. In order to enable our AR usecase in VRComm we made minor improvements to the system, like a new room configuration and the placement of users in AR. Users are placed opposite each other in AR, thus a remote user appearing in your own environment, with the same geometric and size properties as captured. For this integration, we deployed the following capture modules: ZED Frame Grabber (with a resolution of 1024x1024 pixel), Mediapipe ML FGBG, ZED Hole Filling, GrayAVG, and OBS Virtual Webcam. The final user setup consists of a single RGBD (ZED) sensor, a capture ingest PC and a HoloLens 2 running a Social XR web client. An example of a user in an immersive communication, while wearing a Hololens 2 AR device, is shown in Figure 2.6. Note that the Kinect Azure could not be used, as it resulted in interference with the HoloLens 2.

The complete application pipeline can be seen in Figure 2.7, and includes the following components:

- Modular capture as presented in Section 2.2
- Ingest web client receiving RGBD images from the OBS virtual webcam driver and uploading them into the VRComm system
- VRComm utilizes per-to-peer WebRTC transmission for the video (and audio)
- The render web client runs on a HoloLens2 device sending/receiving audio and rendering the audio, as well as the 3D point cloud

We replicated the same testing environment as described in [117], to measure social presence in a personal communication environment of AR. We applied two different conditions than [117] for our small-scale user study: (1) a condition using MS teams on a 46 inch (life-size) display and (2) using VRComm and the HoloLens 2 as described above. To make both conditions comparable, users had to wear the HoloLens 2 in both conditions

(i.e. the HoloLens 2 was turned off in the MS Teams condition). We conducted our study in a setup between subjects with 18 pairs (N=36; 9 pairs per condition) of people who know each other (friends, couples, family, and close colleagues). This means in 7 male and 11 female (with an average age of 29.5) in the MS Teams condition, and 9 male and 9 female (with an average age of 34.39) in the HoloLens 2 condition. In each condition, two participants have a conversation in two different rooms followed by questionnaires. We included the following questionnaires in our study:

- Holistic Framework for Quality Assessment of Mediated Social Communication (H-MSC-Q) [118]: to assess both the spatial and social presence.
- Networked Minds questionnaire (NMQ) [108]: assessing co-presence, psychological involvement, and behavioral engagement.
- Inclusion of Other in the Self (IOS) [119]: to assess interpersonal closeness via "a single-item, pictorial measure of closeness".
- User Experience Questionnaire (UEQ) [120]: evaluating the user experience on six scales attractiveness, perspicuity, efficiency, dependability, stimulation, and novelty

We also asked users to rate the overall perceived video quality and performed post-experiment interviews for qualitative feedback. We ensured good quality audio, given its importance in video conferencing [121].

Table 2.5: Overview of presence and quality scores for AR communication (HoloLens 2), scores are presented as mean average with standard deviation (SD) and confidence interval (CI; alpha = 0.05)

Condition	IOS	H-MSC-Q	NMQ	Overall quality	Image quality
MS Teams 46" & HL2	3.78	6.1	5.27	4.11	4.16
	(SD 0.82)	(SD 0.86)	(SD 0.77)	(SD 0.66)	(SD 0.5)
	(CI 0.66)	(CI 0.40)	(CI 0.36)	(CI 0.30)	(CI 0.23)
HoloLens 2	3.44	5.73	5.6	3.78	2.72
	(SD 1.42)	(SD 0.81)	(SD 0.58)	(SD 0.71)	(SD 0.80)
	(CI 0.66)	(CI 0.39)	(SD 0.27)	(CI 0.33)	(CI 0.37)

Evaluation The results of our user study can be seen in Table 2.5, the perceived quality of our 3D video pipeline was lower compared to the conditions of the MS Teams. Interestingly, the feeling of social presence did not decrease significantly under these conditions, despite the lower perceived video quality. We performed a Principal Component Analysis of the various conditions to gain further insight into all the factors that influence the results. This analysis basically shows two main factors. The first is the novelty effect of the use of the HoloLens 2, for which the scores on factors such as attractiveness, novelty, and stimulation had a great influence on the social presence scores. However, perceived quality has a large positive influence on social presence scores.

This explorative study strengthens our belief that Augmented Reality can be used to increase the social presence when communicating at a distance. By further improving 3D video quality in the future, immersive communication will ultimately outperform current

video conferencing in delivering social presence. This is further supported by remarks during the post-experiment interviews. Users indicated that the HoloLens 2 brought a feeling of closeness, of experience of the other person as being actually there with them in the room. Multiple users specifically mentioned the lack of a screen as a big plus. Even with these remarks, we still feel that AR headsets are not good enough yet for AR conferencing. Multiple users also remarked that they had to sit still, because with (head) movement, the other user would disappear, pointing to the limited field of view of the HoloLens 2. Additionally, with our limited single capture solution, the eyes were not visible, which was a serious detriment. In the Teams + HoloLens condition, the eyes were somewhat visible, but ideally more transparent glasses are available for eye contact. Given current announcements on new AR headsets, we feel they may be suitable in 2 to 4 years from now, both offering a bigger field of view and more transparent glasses.

Although the user study confirms the technical feasibility of integrating the capture modules into an immersive communication application that allows people to communicate in a way comparable to Microsoft Teams, it still raises multiple questions about the technical readiness of multiple components (such as the AR glasses itself) as well as multiple questions relating to the user experience.

2.4 Discussion

To discuss our modular capture application approach and evaluation results, we follow the structure of our research questions:

RQ 1.1. What are the advantages and disadvantages of different RGBD capture modalities, individual processing steps, and their resource needs?
In this chapter, we present an extensive performance analysis of the different capture and processing modalities. Each processing step was included in an individual module and independently measured, both in terms of processing delay and resource usage. Our results show low performance overhead with a good balance between CPU and GPU utilization (and a maximum of 16% for GPU and CPU for high resolution streams). This means that the performance requirements are significantly higher than for traditional video conferencing capture but still acceptable for any modern PC (H1.1.1). Furthermore, we have an acceptable delay overhead for low-resolution streams. This is because with a complete delay of ~300 ms (K4A) and ~400 ms (ZED) we can assume a complete capture-to-glass delay for a communication application of less than 500 ms (depending on the encoding and network conditions), which is acceptable for most communication scenarios (H1.1.2). This may not be true for the high quality stream setting with a complete delay of <500 ms (ZED) and <600 ms (K4A). One of the main reasons for the 100 ms higher delay in the K4A condition is that the Intel_HSV depth conversion seems to not perform optimal for the higher-resolution streams (needing ~150 ms execution time vs. ~50 ms execution time of GrayAVG), as well as the Unity Virtual Webcam driver (adding a delay of ~100 ms vs. ~50 ms for the OBS virtual webcam driver). We assume in its current state our proposed capture application, reconstruction and rendering might result into a delay double to a 2D webcam video approach. In an operational deployment, however, this delay could further be optimized, i.e. by processing within the integrated circuit (IC) of the capture device itself.

RQ 1.2. How can RGBD data be utilized for 3D photorealistic user representations and what is its resource overhead?

In our rendering performance measures, all rendering conditions performed in an acceptable range of CPU/GPU usage within the laptop / browser combination (see Table 2.3). Furthermore, all browsers performed similar, which is to be expected since we choose main browsers that are in long-term development and well equipped for 3D WebGL processing. One surprising aspect is that the Chromium-based Edge browser seemed to perform slightly less than Chrome, even though both rely on the same rendering engine. However, we can conclude that the RGBD-based photorealistic user rendering is easily possible on a modern PC with any of the tested browsers (H1.2.1). With the internal structure of the WebGL rendering engine in mind, we expect that any native application (e.g., Unity or Unreal XR application) would achieve similar or better performance results.

Concerning the different 3D rendering techniques, we were unable to observe any significant differences between them. This means that our introduced shader optimizations are able to increase the render quality while not introducing any further processing overhead (H1.2.2). Furthermore, we could not observe much difference between 3D Mesh and 3D point-based rendering, allowing us to choose the optimal rendering technique based on the user case and user aesthetic preferences. When comparing the 3D rendering with the 2D baseline, we do see a significant increase in GPU and memory usage. This is expected as, for example, in the high-resolution case, the GPU needs to handle 1M individual pixel data points and its geometric properties. Furthermore, the stable CPU utilization between the 2D and 3D conditions indicates that the rendering pipeline works optimally and reliably.

Our tests also show that rendering on the HoloLens 2, even though possible, is still challenging, as it practically maxes out the CPU/GPU usage of the device (H1.2.1). This effectively only allows to add one high resolution user. Adding more users would only be possible in a lower resolution, and thus accepting a low-quality image for the user representations. However, with the next generation of hardware and further optimizations such as remote rendering, this could be mitigated in the future.

RQ 1.3. Do RGBD-based user representations result in a high social presence for remote communication?

We integrated our modular capture application into two XR communication applications, covering both AR and VR modalities, including one commercial application. Our examples show that a modular integration is feasible and can add 3D photorealistic user representations to existing video communication transmission systems. Even though motion JPEG compression (as in the case of Connec2) can be considered suboptimal to transmit RGBD data (and thus is currently restricted to 512x512 pixel resolution for the color and depth stream), it further shows the usability and technical feasibility of our approach. However, the question of an increased level of social presence can only be partially answered due to remaining technical limitations and the complexity of user studies itself. Although the AR personal communication use case study showed an indicative result of its benefits, technically the AR glasses do not appear to be fully ready for our use case. Multiple technical improvements are required in the pipeline to reach the same sophisticated state of a commercial application like Microsoft Teams. Further, more user research is needed to investigate XR communication use cases in more detail. This includes a need for a

better understanding of technical and functional limits as well as the benefits of immersive photorealisitc communication. We believe that our current modular capture approach offers the ideal testbed for dedicated user studies under different constraints.

Finally, we like to stress that the modular capture application and rendering, presented in this chapter, can be reused in any established existing 2D video conferencing system (incl. its 2D video transmission). However, building such immersive communication systems in a reliable, scalable way with ever-growing higher resolution of streams and simultaneous users remains a challenge.

2.5 Conclusion

In this chapter, we present a modular capture application as well as two examples of integration with two XR communication applications, one of which is a commercial product. We show the advantages and disadvantages of the different modules with dedicated in-depth evaluations of each module. This includes two types of foreground background extraction and two RGBD image quality enhancement strategies. In addition, we propose and evaluate different techniques to render the 3D user representation. Our small-scale user evaluation of the two XR communication applications shows the technical suitability of our approach and shows that an easy integration of 3D photorealistic user representations can be achieved to allow immersive communication. Although this chapter focuses on different capture and rendering aspects, it does not cover many important aspects relating to key building blocks and the performance of the system and transmission. This is discussed in the following chapter.

3

Network & Performance

Tools and platforms that enable remote communication and collaboration provide a strong contribution to societal challenges. Virtual meetings and conferencing, in particular, can help to reduce commutes and lower our ecological footprint and can alleviate physical distancing measures in the case of global pandemics. In this chapter, we outline how to bridge the gap between common video conferencing systems and emerging social VR platforms to allow immersive communication in Virtual Reality (VR). In this chapter, we present a novel VR communication framework that enables remote communication in virtual environments with real-time photorealistic user representation based on color-and-depth (RGBD) cameras and web browser clients, deployed on common off-the-shelf hardware devices. The chapter's main contribution is threefold: (a) a new VR communication framework, (b) a novel approach for real-time depth data transmitting as a 2D grayscale for 3D user representation, including a central MCU-based approach for this new format, and (c) a technical evaluation of the system with respect to processing delay, CPU and GPU usage.

This chapter is based on Simon NB Gunkel, Rick Hindriks, Karim M El Assal, Hans M Stokking, Sylvie Dijkstra-Soudarissanane, Frank ter Haar, and Omar Niamut. Vrcomm: an end-to-end web system for real-time photorealistic social vr communication. In *Proceedings of the 12th ACM multimedia systems conference*, pages 65–79, 2021 [33]

Communication and collaboration are an important part of everyday life, both in professional and private settings. Having tools to help one communicate over distance, such as the video conferencing applications Skype, Google Hangouts, or Zoom, has rapidly become a normality in a modern and globalized world. Nowadays, these applications can be run both as standalone, dedicated apps and programs, as well as web applications (apps) in browsers that implement rich media tooling such as WebRTC and support advanced video decoding. However, looking at these tools today, common video conferencing applications have clear limitations regarding enabling a sense of (co-)presence and immersion. That is, "the video calling stuff breaks down beyond a few people, because you have this big grid of tiny faces." Simply by the absence of "the full range of social cues, from posture to eye gaze to facial expressions, things like head nodding and hand gesturing', which can convey crucial non-verbal information (Blair MacIntyre[1]). And even when video conferencing systems manage to convey information communicated by facial expressions and body gestures, they do so at the expense of a sense of shared space.

Not only recent events (such as the global covid-19 pandemic) raise the need for better solutions that increase the feeling of togetherness while communicating remotely. One of the first steps in adding space sharing in 2D video conferencing can be seen in Microsoft Teams together mode[2] that blends the 2D webcam video of users in a simply lecture hall inspired background. Furthermore, from other works [122–124] we know that spatial awareness, presence, and immersion can be provided by communication in Virtual Environments and Virtual Reality (VR) experiences. However, in the initial VR frameworks, the lack of social interaction [86, 125] prevented users from experiencing co-presence. With recent social VR applications and platforms for VR meetings and conferencing, social interactions have also started to make their way into the virtual realm. These platforms typically make use of model-based avatars, i.e. graphical representations of participants whose movements are steered by input from the VR headset and/or controllers. In order to allow users to consume existing media streams as well (for example, shared consumption of live and on-demand video streams like sport), the integration of video into these platforms is currently taking shape. An example of a widely accessible social VR system is Mozilla Hubs[3]. It solved many interesting aspects of social VR from a technical perspective, for example, objects and states synchronization across multiple clients. However, Moxilla Hubs currently deals with video only in a limited way (i.e., streams for presentations and static 2D webcam views).

In this chapter, we seek to bridge the gap between common video conferencing systems and emerging social VR platforms. That is, we aim to reuse proven technologies and frameworks from the domain of video conferencing and build a platform for VR communication experiences that incorporates photorealistic user representations. We mainly consider the end-to-end video processing chain and the use of a multipoint control unit (MCU) to bridge multiple video conferencing connections . Our main hypothesis is that by reusing these components from common video conferencing systems, we can support VR conferenc-

[1] https://web.archive.org/web/20230130172421/https://spectrum.ieee.org/tech-talk/consumer-electronics/audiovideo/forget-video-conferencinghost-your-next-meeting-in-vr

[2] https://web.archive.org/web/20240308153844/https://news.microsoft.com/innovation-stories/microsoft-teams-together-mode/

[3] https://web.archive.org/web/20240216000442/https://hubs.mozilla.com/

ing under network requirements similar to those for traditional video conferencing. The research in this chapter was based on the following research questions:

RQ 2.1. Is RGBD 2D video transmission suitable for XR communication with volumetric user representations?
H2.1. By introducing a new depth-to-grayscale transmission scheme existing encoders can be reused for high-quality RGBD video transmission.

RQ 2.2. Which components need to be extended or added to common video conferencing architectures to allow Social XR?
H2.2. The higher resolution and data rates of RGBD videos increase the need for a better (central) stream management in the system.

RQ 2.3. What performance and network utilization are expected for Social XR?
H2.3. With some modification, the reuse of existing video conferencing technologies allows RGBD-based volumetric XR communication with reasonable processing overhead.

The motivation for this work lies in the relevance of end-to-end video processing technology to provide real-time performance in web-based social VR applications. While the importance of recent volumetric video formats to provide true 6-degrees-of-freedom VR experiences is becoming apparent [68, 69], the end-to-end workflow to process such data from capture to rendering is far from real-time. The use of video-based methods allows us to benefit from existing deployed infrastructures and interfaces(such as hardware acceleration, robust coding, and streaming) and ultimately extended support in many browser platforms. The contributions are threefold:

i.) we describe a new VR communication system that combines video conferencing technology with social VR capabilities in a new end to end pipeline from user capture, processing, transmission and rendering users into virtual environments for shared immersive experiences and communication

ii.) we report on a novel transmission scheme for grayscale based depth information, for 3D user representations, including a central MCU-based approach for the transmission of this video format

iii.) we perform an evaluation of the resulting VR communication system with respect to processing delay, CPU and GPU usage

3.1 Related work

3.1.1 Video Conferencing

The first video conferencing systems were based on one-on-one connections between two sites. Scalability for multi-person video conferences could either be achieved by a full-mesh exchange of media streams between all participants, or by using potentially available multicast mechanisms. As IP multicast technology is not widely deployed across the public internet, centralized mixing facilities called Multipoint Control Units (MCU) were developed in the 1990s. These MCUs multiplex in some form the various media streams, so only a single stream needs to be sent to each participant. With many participants, such a centralized scheme allows for improved scaling of bandwidth requirements and can scale

the video conference to a large set of simultaneous users, in particular by designing hybrid centralized forwarding architectures [126].

Various developments have been made more recently to further achieve scalability without sacrificing quality. For MCUs, recent cloud developments give the opportunity to use processing on-demand, thereby allowing conferencing sessions to scale up to many hundreds or even thousands of participants [127]. For efficient stream multiplexing without any media processing, Selective Forwarding Units (SFU) are developed; see [128] and [129]. These SFUs forward streams from one participant to all other participants, thereby alleviating the need for one participant to send out separate streams. To support bandwidth adaptation, each participant can send its stream in a few different bitrates so that an SFU can select and forward individualized streams depending on the bandwidth availability for each participant.

More recently, MCU architectures have also improved through the use of tiling mechanisms [130]. Tiling allows for stitching together video parts in the encoded domain, creating an architecture that sits somewhere between the MCU and the SFU: all incoming streams are mixed together so that each participant receives only a single media stream to be decoded, while at the same time no decoding-mixing-encoding is required.

3.1.2 Social VR

In the 1990s, much work was put into creating high-end shared virtual environments. Various universities have set up cave augmentation virtual environments (CAVE) and CAVE-like systems which could be used to communicate remotely, of which [131] and [132] provide good examples, using what they call "video avatars". These environments typically used back projection and large calibrated camera rigs to produce a coherent virtual environment. Other examples such as [133] used large screens, again together with calibrated camera rigs, to also offer a sense of togetherness.

Other work from this era consists of using graphical avatars to create large shared virtual environments, of which [29, 134, 135] give some overview. In these days, the impact of avatar realism on participants' perception was also studied [16]. Virtual reality saw renewed interest with the rise of high quality but affordable AR and VR HMDs, most notably the Oculus Development Kit, which carried the promise of bringing high-quality VR to the masses. This development has also led to new initiatives in shared and social VR experiences. Today, virtual reality is mostly associated with graphical avatars in a graphical environment. Main examples are Facebook Horizon, AltSpaceVR, BigScreen, Glue, High Fidelity, vTime, Hubs by Mozilla, VR chat, SteamVR and Spatial.

While considering the social VR services mentioned, all represent users as graphical avatars displayed in a shared VR environment and allow users to play games, share screens, share web browsing, watch videos, explore spaces, or share other experiences together. However, very little studies exist that compare those new services with existing communication tools or compare real-time photorealistic representations with artificial avatars. One study [125] suggests that graphical avatars (Facebook Horizon, formerly Facebook Spaces[4]) have limitations in terms of (co-)presence as "the social cues that you would normally have about someone … weren't there". Another study [86] presents the results that real-time photorealistic representations show no statistical differences in terms of interaction and social connectedness compared to a face-to-face meeting, while the avatar-based system

(Facebook Horizon[4]) did. In the context of collaboration, VRComm and "traditional" video conferencing are theorized to differ in their affordability of social context cue transfer, specifically in terms of body posture, gestures, and eye contact. Additionally, the experience of presence, which is usually found in VR (e.g. [43]), may affect communication and interpersonal relationships as well.

3.1.3 Spatial Computing & HMD replacement

Spatial Computing, which is the ability to understand the environment, the user, and objects surrounding the user, is an essential part of AR and VR applications (good examples can be found in [77, 78]). In terms of communication, this means that user's actions should be correctly reflected into the users' representation to convey good remote interactions. One example of complex and processing intensive Spatial Computing tasks is the replacement of the HMD that by default occludes the participants face when using a VR-HMD as a display device. As facial expressions and eye gaze are important factors in communication, the HMD replacement process becomes essential for a qualitative experience. The so-called facial reconstruction is relevant to improve the user experience in (video-based) social VR. When capturing the user with a video and/or depth sensor, the captured footage will include the HMD, and thus occlude parts of the face, including the eyes. Our previous experiments have shown that it still allows for natural interaction and communication with an increased feeling of co-presence [84]. Takemura [136] was one of the first researchers to describe a method that performs HMD replacement by detecting the location of the HMD in the video and replacing pixels with a 3D facial model captured at an earlier process. Li [137] followed a similar approach by using strain gauges inside the HMD to measure facial expressions. Burgos-Artizzu [138] used various facial models to represent various expressions and detect expressions based on the part of the face still visible in the video recording. More recently, Thies [139] and Google [140] elaborated on this approach by combining these methods with an eye-tracking camera inside the HMD to reproduce the "correct" eye direction. Furthermore, recent work aims to make HMD replacement more robust and flexible by using RGB-D image inpainting techniques [141].

3.1.4 Volumetric video capture and transmission

Volumetric video is regarded worldwide as a key technology in the context of immersive AR and VR experiences. Capture, encoding, and transmission of volumetric video formats such as point clouds and meshes is an active field of research [63–67] as well as industry standardization[71]. In particular, for remote telepresence and immersive communication[39, 70], volumetric videos provide a higher quality of experience and social presence [68, 69]. The most recent volumetric video formats (i.e., video-based (V-PCC)[71] and geometry-based (G-PCC)[65]) point cloud coding) require significant processing resources for capturing, coding, transmitting, and rendering[65, 71]. Although V-PCC is currently not suitable for real-time communication (due to its encoding latency), recent work has begun to optimize G-PCC for telepresence scenarios [142, 143] by data reduction and fusion. Still, there are many open challenges in terms of volumetric media delivery [144]. This includes high data rates, high processing load, high encoding delays, low

[4] https://web.archive.org/web/20240117003810/https://www.oculus.com/facebook-horizon/

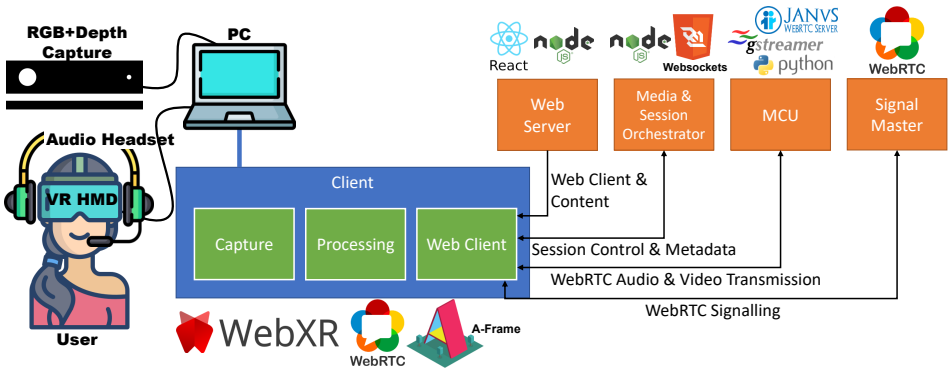


Figure 3.1: VRComm System components and user setup

resolution, or low frame rate (or a combination of those). To address the current gaps of V-PCC and G-PCC our work mainly considers a video-based (RGB plus depth) transmission approach as an initial step towards full volumetric representation of users. The adaptation of standard video codecs for depth streaming was studied in [72], while more advanced conversion of depth information to grayscale images was considered in [74] and [73].

3.1.5 Tele-immersion and telepresence systems

The research direction closest to this work can be seen in tele-immersion and telepresence. While telepresence systems offer a professional video conferencing experience with dedicated hardware setups to showcase people more realistic (looking more natural; examples are Lifelike and Cisco), tele-immersion is making the step to offer communication systems in virtual environments. Multiple research efforts have been conducted in the last decades to address tele-immersion and telepresence [145–149]. One of the first examples of a tele-immersion system is TEEVE [150] which allows 3D capture and rendering of users in a complex setup with a frame rate of 4 5 frames per second. Another example of a tele-immersion system is the Roomalive toolkit[5][50] from Microsoft. One research using Roomalive is Room2Room [151], limited to one-to-one interactions, it allows projection-based AR telepresence with the help of a depth sensor and projector. We can summarize the efforts of tele-immersion and telepresence systems in a complex setup that require dedicated hardware, have high performance and network requirements, or support only a very limited number of users. In our work, we bridge the gap from low-cost simple video conferencing solutions towards volumetric user representations in VR, with a focus on off-the-shelf hardware and web-based client software for a low entry burden. Furthermore, we aim to support capture framerate, capture resolution, and network utilization similar to existing video conferencing solutions.

[5] https://web.archive.org/web/20240415071242/https://www.microsoft.com/en-us/research/project/roomalive-toolkit/)

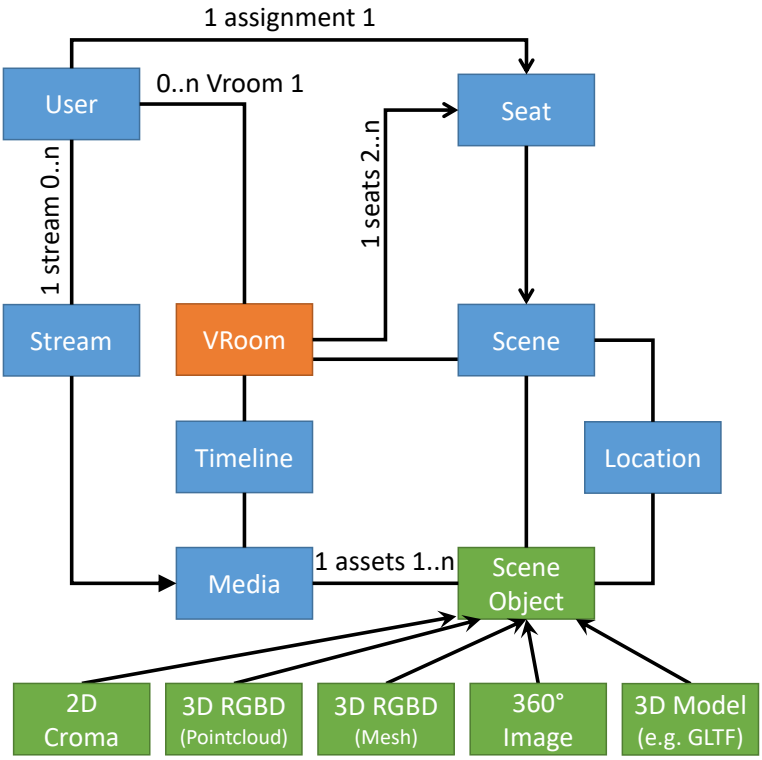


Figure 3.2: Virtual Scene Description Metadata Schema

3.2 VRComm Framework

Moving from traditional video conferencing to virtual reality (VR) conferencing brings about new requirements on the system and device setup [84]. One of the main differences between video conferencing and VR communication is that in VR, we work on a geometry-based 3D environment rather than a window-like 2D arrangement for video conferencing. VR conferencing involves multiple aspects of supporting new media formats, interaction paradigms, and ways to orchestrate and synchronize media in the virtual environment. In particular, new ways to capture and blend users into geometric spaces are crucial to increase immersion and presence for natural communication [84], which includes 3D user representation, enabling self-representation and maintaining eye gaze with others (i.e., not being restricted by wearing an HMD). In addition to supporting novel user representation formats, VR conferencing systems should scale to support numbers of users that are similar to those of traditional 2D video conferencing applications. Figure 3.1, shows the VRComm system with the different components and technology aspects that are explained in the following subsections.

3.2.1 Architecture and media orchestration

VRComm is a web-based framework for building and consuming shared and social VR experiences. Our main motivation to utilize Web-based technology is to cater for an easy and widespread deployment and low entry burden for end users and developers. In this way, we currently only use off-the-shelf hardware and currently available web technologies.

Figure 3.1, shows the overall framework architecture of VRComm. To initiate a web client instance, first the client JavaScript code and any multimedia files are downloaded from a web server, second the web client will register at the Media Orchestrator by selecting and exchanging session metadata, and finally will negotiate WebRTC stream connections with the help of the Signal Master (either as peer-to-peer or MCU transmission scheme). This results in the following video transmission modules and processing steps:

1. Capture of raw sensory data (see Section 3.2.3)
2. Capture Processing (see Section 3.2.3):
   (a) Video Background Removal
   (b) (optional) Camera calibration
   (c) (optional) HMD Replacement
3. RGBD grayscale conversion (see Section 3.2.4)
4. Web client ingest (see Section 3.2.2)
5. (opt.) local rendering of self-representation (see Section 3.2.2)
6. WebRTC Transmission (see Section 3.2.4), either
   (a) Peer-to-peer (P2P), or
   (b) (optional) central server / MCU-based (see Section 3.2.5)
7. rendering of remote user(s) (see Section 3.2.2)

One central component of VRComm is a media orchestration server, which manages communication sessions that users can join to communicate with each other. For example, this also includes the calibration data of the user capture to allow 3D reconstruction of the user representation. Furthermore, the orchestration server maintains all metadata to synchronize and modify the virtual environments of each client at run-time. The orchestration server powers monitoring and modifying all client properties relevant to the VR experience in real-time and thus facilitates complex interactive VR multimedia experiences. In particular, the content that is displayed can be modified, e.g., a game or movie, and the placement of objects and users.

To facilitate the calibration and design of the VR rooms, we have developed a metadata format to position users and immersive media objects into the virtual space. The schema of the rooms metadata is shown in Figure 3.2. In the center is the VRoom which can cover different objects related to users, the virtual scene (or environment), and different media objects (like video panes or interaction elements). We can currently support many different

media objects like 2D video (including DASH), images, 360-degree content, 3D models (e.g., in the OBJ or glTF format), and our own 2D/3D RGBD format (see Section 3.2.2]). Furthermore, the metadata-based virtual scene creation and media object allocation also serves as a global coordinate system and simplifies the synchronization, interaction, and remote configuration of different user-client states in the system. The media orchestration provides an admin console that grants fine-grained control of all metadata properties in real-time. This remote control is designed with the main aim of supporting any user experience research.

3.2.2 Web Client

The entry point for the VRComm web client is offered by a web server back-end. WebXR-enabled web browsers can connect to this server back-end to obtain the client application, which is based on open source software JavaScript frameworks Node.js, React, SimpleWebRTC and A-Frame[6]. This allows any modern browser to display the VR content on a screen or any Open VR hardware enabled VR-HMD (e.g., Oculus Rift or Windows MR). Furthermore, the client can access the image produced from the RGBD capture module (see Section 3.2.3) to be displayed as self-representation or sent via WebRTC to one or multiple other clients. To support such a multi-user connection, the client is connected to a signalling server that handles streaming orchestration.

The rendering of users in the VR environment is done via custom WebGL shaders that alpha-blend user representations into the virtual environment for a natural visual representation. We first record users with a RGB-plus-depth sensor (e.g., in VRComm we currently support Kinect v2, RealSense™ and Azure Kinect) and then have two options for transmission:

i we replace the users background with a fixed "chroma" color before transmission (over WebRTC), and after reception apply alpha-blending to remove the background, resulting in a transparent image showing just the user without his/her physical background.

ii we convert the depth values into grayscale and transmit them along the image to render the user in 3D (see section 3.2.4).

For capture and transmission, we currently use a resolution of 540x800 pixels for RGB (with chroma background) or 1080x800 pixels for RGBD images. This resolution is matching the depth resolution of most depth sensors. However, our system is fully adaptable to any resolution. Audio is also captured, transmitted, and made spatially audible with the help of the Google Resonance API[7].

When utilizing a 360-degree VR environment, we will transmit and render users in 2D only (RGB + chroma). While for 3D geometry-based environments, we render users in such a way that they can observe themselves (as self-representation) via 3D point cloud. The point cloud is created by first converting the grayscale depth data to a depth value and then recalculating the 3D position of each point based on the calibration data of each RGB-D sensor. This is done in a dedicated WebGL (GLSL) shader and thus runs efficiently

[6] https://web.archive.org/web/20240214000513/https://aframe.io/
[7] https://web.archive.org/web/20240304195927/https://resonance-audio.github.io/resonance-audio/

on the GPU. The following 3D mapping is used (per pixel):

$$z = depth \tag{3.1}$$
$$x = (position.x - outputCenter.x - 0.5) * z/focalLength \tag{3.2}$$
$$y = (position.y - outputCenter.y - 0.5) * z/focalLength \tag{3.3}$$

Where depth is the distance of the pixel in meters, position is the pixel coordinate on the RGB video, outputCenter is the centre pixel coordinate of the sensor metadata, and focalLength is the focal length of the sensor.

Similar to the self-representation, we display remote users based on the video from a WebRTC connection. To render remote users, we developed 3 types of components for 3 types of rendering:

1. Rendering in 2D (RGB + chroma background) via a shader that alpha blends the user into the virtual scene
2. Rendering in 3D (RGB + Depth) as point cloud (mapping the 3D points on a THREE.js Geometry[8])
3. Rendering in 3D (RGB + Depth) as mesh (mapping the 3D points on a PlaneBufferGeometry[9])

A VRComm client experience consists of VR environments referred to as 'VRooms' (see Figure 3.2). For the rendering of visual and audio information (e.g., the VR Environment, objects and users), the client will utilize the A-Frame framework, which combines the WebXR API with Three.js to provide a simple scripting framework for the design and development of web-based XR experiences. This allows to easily create 360-degree and 3D volumetric VR applications while supporting many 3D scenes and models (including glTF™[10]). A-Frame has integrated support for most common consumer hardware (including any SteamVR and OpenXR enabled device), among which the Oculus Quest, Windows Mixed Reality (WMR) headsets and HTC Vive. Further A-Frame is supported by major PC and laptop browsers like Mozilla Firefox, Google Chrome and Microsoft Edge, several mobile browsers including Chrome and Firefox. As a result, easy access to VR technology has become available on the web. Internally, the client follows a completely modularized structure via a combination of React Modules and A-Frame components. This allows an easy and dynamic creation of individual VR applications with different features.

3.2.3 Capture

One of the main challenges in any immersive communication system is on how to capture the users. For natural interaction and true social presence, it is important that a visual representation accurately reflects the appearance and actions of each user. Therefore we focus in this work on a camera-based capture solution to capture a photorealistic representation of users in real-time. Many factors have to be considered for this capture, as an example, some of the main factors include lighting, color, edge accuracy, and other

[8] https://web.archive.org/web/20240210024518/https://threejs.org/docs/#api/en/core/BufferGeometry
[9] https://web.archive.org/web/20240210024518/https://threejs.org/docs/#api/en/geometries/PlaneGeometry
[10] https://web.archive.org/web/20240209132249/https://www.khronos.org/gltf/

capture artefacts. The capture in VRComm is developed in a modular architecture and currently allows the use of one or two RGBD sensors and different processing modules. In the following, we outline the capture modules:

Raw Sensor Capture The first capture module does real-time scene and user capture by reading the raw RGBD data from the capture sensor and mapping the color and depth images to a shared memory location on a client PC. Currently, we have modules to support three types of RGBD sensors: Kinect v2[11], RealSense™ (Intel®RealSense™ SDK 2.0[12]) and Azure Kinect[13]. However, different RGBD sensors could easily be added without affecting any of the subsequent modules.

Foreground-background removal (FGBG) The raw RGBD sensor needs some further processing in a second capture module. The main aim of this module is to improve the image quality and perform real-time foreground-background (FGBG) extraction using color and depth images from shared memory. This is an important step, as for the user capture, we are exclusively interested in the user itself rather than his or her background. Furthermore, this allows us to transmit only the captured user and blend the visual (rendered) representation of users into the virtual environment.

Multi-cam capture and calibration Currently, we support the capture with one or two depth sensors. When two cameras are used, we calibrate and align both cameras. The calibration phase concerns the alignment of the two RGB-D sensors used to capture the participant. The registration and alignment of the two sensors is done with the help of a large ArUco[14] marker ($30x30cm$) and pose matching. This results in a near 180° 3D representation of the user (front view), from the RGB-D frame pairs [152]. The calibration parameters from the rigid body transformation are sent as metadata together with the RGB-D visual data.

HMD Replacement The HMD Replacement module consists of an open source available ArUco marker detection (implemented using OpenCV[15] in Python[16]) applied to the RGB-D image. When the marker is attached to the HMD of the subject, the HMD can be detected in real-time without assumptions on the position of the subject or the capturing device. With additional markers on all sides of the HMD, the detection also works when the user looks left or right, up or down. The 2D detection in the RGB stream is combined with the depth to acquire an accurate 3D position and orientation of the HMD. There are multiple applications possible for this 3D position and orientation:

1. 3D head removal for self-representation, such that the view of the subject is not occluded by the scan of his/her face

[11] https://web.archive.org/web/20240330120932/https://developer.microsoft.com/en-us/windows/kinect/
[12] https://web.archive.org/web/20240420024225/https://www.intelrealsense.com/sdk-2/
[13] https://web.archive.org/web/20221211184548/https://azure.microsoft.com/en-in/services/kinect-dk/
[14] https://web.archive.org/web/20210820131926/http://www.uco.es/investiga/grupos/ava/node/26
[15] https://web.archive.org/web/20240214001251/https://opencv.org/
[16] https://web.archive.org/web/20240427111212/https://www.python.org/

2. Auto-calibration of multiple sensors, such that when two sensors detect the same marker they can auto-calibrate

3. Integration with FGBG removal, to only have one foreground subject in VR

4. 3D HMD replacement and 3D face repair in RGB-D or VR

3.2.4 RGBD grayscale transmission

The VRComm streaming approach relies on a web framework with a peer-to-peer (P2P) nature for delivering video-based social VR experiences to each of the participants. This web streaming framework employs WebRTC for browser-based real-time communication. All 2D video streams and users' audio are transmitted via WebRTC. Any associated metadata are transmitted via a central media orchestration server and Socket.IO . Despite the newer volumetric streaming formats (like V-PCC [71] and G-PCC [65]), this allows us to reuse many existing real-time streaming components (including to benefit from full hardware acceleration).

We use the SimpleWebRTC library to support direct WebRTC-based peer-to-peer communication between users for audio and video. At this moment, voice communication is monaural and spatially positioned in the receiver client (utilizing the Google Resonance Audio SDK for Web[17]). The integration of WebRTC with a VRComm client is managed through an (Node.js) orchestration server.

Grayscale conversion For a depth transmission in VRComm we target a simple and reliable approach that works in real-time and is applicable to be used in any modern browser. This already implies a couple of design choices, e.g., transferring of RGB and depth should be done in one frame as a separate transmission and frame accurate synchronisation is very difficult in the browser (as the underlying media APIs needed for frame accurate synchronization are not exposed in JavaScript in every browser). Similarly, many underlying WebRTC transmission and decoding APIs are also not exposed by a browser raising the need to make any color conversion and depth mapping directly in the WebGL (GLSL) shader itself.

For transmitting the RGB-D frame data over WebRTC to VR over the Internet, we convert the depth data into a grayscale image to comply with current video encoders. For this conversion, we use an improved version of [59]. While [59] does not utilise the full RGB range, we convert depth values corresponding to a real-world distance of $0-1.5m$ into gray-color values that are mapped to the full RGB color space (contrary to other approaches that directly modifies the YUV values, which will not be possible to convert back to depth in a browser WebGL shader). The grayscale depth image is concatenated to the RGB image to stream it as a single RGB-D video stream. In the VR environment, the depth image is converted back into the 3D positions of individual pixels. In the following, we will refer to our algorithm as "GrayAVG". To ensure that GrayAVG works within a depth range of 1.5 m we first subtract a fixed value (this is the minimum distance a person is away from the camera to allow full body capture and is transmitted as metadata and added in the reconstruction) and remove all values outside of the 1.5 m range. The following shows our

[17] https://web.archive.org/web/20240304195927/https://resonance-audio.github.io/resonance-audio/

Table 3.1: Capture Performance According to sensor (1000+ samples each; mean values)

RGBD Sensor	Capture		Processing	RDA Delay (in ms)	Processing Delay (in ms)	Browser Delay (in ms)
	(CPU in %)	(GPU in %)	(CPU in %)			
One Kinect v2	10.53% (SD 1.13)	6.63% (SD 2.73)	19.08% (SD 2.70)	78.72 (SD 15.87)	138.19 (SD 18.71)	342.01 (SD 39.31)
One RealSense™ D415	3.94% (SD 1.21)	n/a	21.81% (SD 4.10)	79.54 (SD 15.79)	152.18 (SD 18.89)	190.16 (SD 19.56)
One Azure Kinect	4.43% (SD 2.00)	4.36% (SD 1.12)	9.64% (SD 1.90)	112.29 (SD 15.51)	155.99 (SD 16.90)	261.74 (SD 24.92)
Two RealSense™ D415	10.79% (SD 1.32)	n/a	10.36% (SD 2.32)	82.84 (SD 15.08)	147.06 (SD 16.35)	234.89 (SD 24.63)
Two Azure Kinect	9.11% (SD 2.49)	9.22% (SD 0.45)	9.52% (SD 1.62)	162.28 (SD 13.38)	306.28 (SD 29.35)	382.99 (SD 35.43)

algorithm in Python/NumPy[18]:

$$depth = depth - min\_distance \tag{3.4}$$

$$depth[depth > 1500] = 0 \tag{3.5}$$

$$r = depth/3 \tag{3.6}$$

$$g = (depth - r)/2 \tag{3.7}$$

$$b = depth - r - g \tag{3.8}$$

3.2.5 RGB(D) Multipoint control Unit

One drawback of a P2P based WebRTC approach for transmission is scalability [128], as multiple users can quickly elevate the (CPU/GPU) resource usage. To mitigate this, we can (optionally) deploy a Multipoint Control Unit (MCU), to aggregate streams centrally and reduce the processing burden on individual clients.

Figure 3.3 depicts the architecture of our MCU. It reuses existing open source components, such as the Janus Video Bridge [153], which is a general purpose, central WebRTC server. While Janus takes care of all WebRTC stream handling (i.e., SDP negotiation and stream forwarding, as well as any audio transmission), the MCU composes all uploaded video streams it receives from clients into a single output stream which is then published via WebRTC to all clients. This output stream resembles a video mosaic combining all user streams, as depicted in Figure 3.4. As a result, clients are able to retrieve all relevant streams together instead of separately. This optimizes the network bandwidth due to more efficient routing (each client only sends its video stream to the MCU, and no longer to all other clients), as well as the decoding resources (clients typically have a limited amount of hardware decoders, which can result in higher CPU usage with many receiving streams).

Incoming WebRTC streams to the server are first remuxed by the Janus WebRTC Gateway into RTP streams that are sent to the MCU (based on GStreamer[19] and Python). The GStreamer media pipeline will then decode each individual user video stream into frames and convert them into NumPy[18] arrays. All NumPy images are then mapped into one complete output mosaic (see Figure 3.4) with efficient in-memory functions. This mosaic

[18]https://web.archive.org/web/20240503065753/https://numpy.org/
[19]https://web.archive.org/web/20240212010719/https://gstreamer.freedesktop.org/

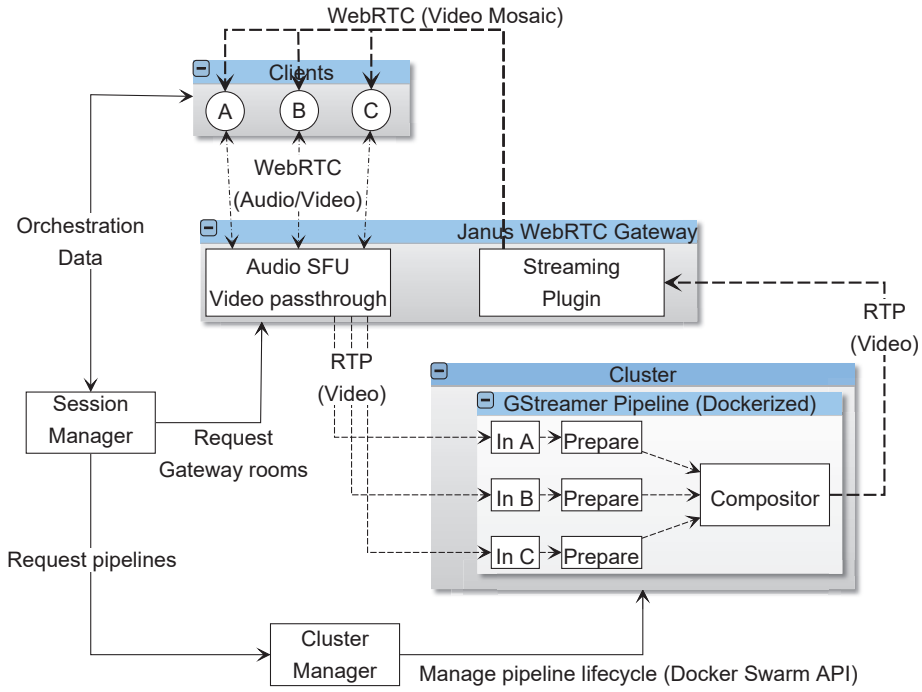


Figure 3.3: MCU Architecture [103]

image is encoded into video frames and sent as a single RTP stream to Janus for distribution to an all client broadcast. The MCU system has been designed as a containerized service such that given enough hardware, it is horizontally scalable over multiple parallel sessions. These services can be managed using Docker Swarm[20] and the Media Orchestrator.

The user experience in VR Conferencing is greatly enhanced by using spatial audio. This requires to uniquely address each individual audio stream, to render it at the appropriate spatial location. Therefore, we use an SFU architecture (within the MCU) for the audio, such that clients can selectively request and retrieve individual audio streams from the MCU. The SFU part of the MCU is implemented purely using Janus (using the Videoroom plugin, see [153]), and requires no further processing.

[20]https://web.archive.org/web/20240427090044/https://www.docker.com/

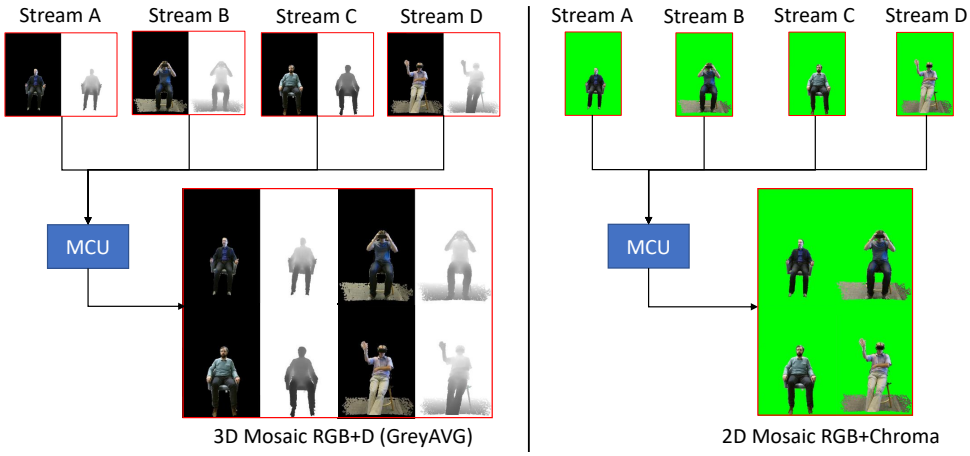


Figure 3.4: MCU Mosaic composition of RGBD (GrayAVG, left) and RGB+Chroma (right) input streams

3.3 System Evaluation

In this section, we focus on the technical evaluation of our system only. A summary of user evaluation of our system can be found in [84]. The evaluation of the core components of the system are structured into three parts, capture (Section 3.3.1), grayscale based depth transmission (Section 3.3.2) and web client evaluation based on using P2P vs. central MCU-based transmission (Section 3.3.3).

3.3.1 Capture Evaluation

We evaluate the capture performance with different sensor set-ups and at different points in our system. Figure 3.5 shows the different components and measure points for delay and CPU/GPU usage:

1. Read sensor data via the sensor SDK into RDA. RDA (Remote Data Access) is a flexible infrastructure for real-time distributed data access and data acquisition. It allows easy exchange of video frame data between different software modules and allow a high flexibility for development and different hardware set-ups.

2. Display data from RDA on the screen.

3. Processing (i.e. FGBG and grayscale mapping)

4. Screen capture the processed image and display it in the browser as self-representation, or transmit via WebRTC. We are currently following this procedure as no current browser implements the Media Capture Depth Stream Extensions [21] thus making it impossible to capture depth data directly in the browser.

All measurements were done on a MSI GS65-Stealth-Thin-8RF (with Intel®Core™ i7-8750H, GeForce®GTX 1070 Max-Q and 32GB RAM). The capture-to-display delays

[21] https://web.archive.org/web/20230929081931/https://w3c.github.io/mediacapture-depth/

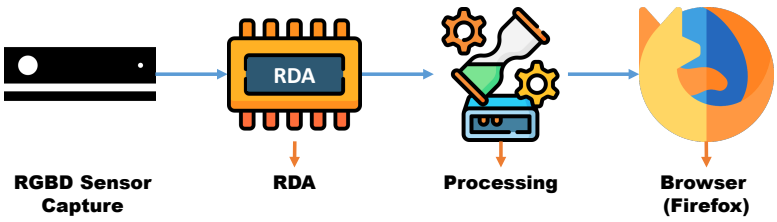


Figure 3.5: Measurements points of capture system

were measured with VideoLat[102][22], with at least 1000 samples each. CPU and GPU performance was measured with a modified version of the Resources Consumption Metrics (RCM) measurement tool[23] [101]. The RCM tool is a native Windows application that allows to capture CPU, GPU, memory usage per process and network statistics of the system in a 1-second interval. Each performance measure was done with a representative sample size of at least 30 minutes. Table 3.1 shows the results of the different measurements.

Overall, the different capture requirements and delays are all in the expected range. However, the Azure Kinect shows an overall higher delay. We also observed high CPU and GPU usage under certain conditions (i.e., all other GPU processes being idle). As the Azure Kinect is still a relatively new sensor, we expect that the performance of the sensor might still improve in the future with further updates to the SDK (we did our tests with firmware version 1.6.108079014 and SDK 1.4.0). Furthermore, the Kinect v2 and RealSense^{TM}D415 included FGBG and thus show 10% more processing loads compared to the other capture methods. In conclusion, our current approach, including RDA, proofs beneficial for testing and rapid prototyping. However, in an operational environment, we expect to decrease the delay by at least 1-2 frames (30-60ms).

3.3.2 Depth Transmission Evaluation

We compared our depth-based conversion (GrayAVG) under different encoding and bandwidth conditions with two other algorithms (naive/simple, HSV). The "simple" algorithm is the most simple conversion based on direct depth to RGB mapping (and thus only serves as a minimal baseline). The HSV conversion is a reimplementation of the HoloTuber Kit[24]. Thus, the different mapping functions in the following:

$$Depth(simple) = (r+g+b)/3 * 4 + min\_distance \tag{3.9}$$

$$Depth(HSV) = h * 4 + min\_distance \tag{3.10}$$

$$Depth(GrayAVG) = (r+g+b) * 2 + min\_distance \tag{3.11}$$

In this test, we focus on encoding formats that are widely available in modern browsers and compatible with WebRTC [25]: VP8, VP9 and H.264. For the test, we used 3 clips of 10

[22] https://web.archive.org/web/20240113164432/https://videolat.org/
[23] https://web.archive.org/web/20230524052810/https://github.com/ETSE-UV/RCM-UV
[24] https://web.archive.org/web/20240415081046/https://github.com/TakashiYoshinaga/HoloTuberKit-for-AzureKinect
[25] https://web.archive.org/web/20231212075318/https://developer.mozilla.org/en-US/docs/Web/Media/Formats/WebRTC_codecs

seconds length with 30 fps (resulting into 300 frames in total for each video). Furthermore, we analysed the videos using the mean absolute difference (MD) to indicate the motion in the video stream:

low: user sitting, no movement, MD 44,92 (SD 13,51)

med: user sitting, some movement, MD51,34 (SD 27,27)

hi: user standing, high movement, MD61,52 (SD 33,34)

For each test condition, we made a full reference analysis measured with the peak signal-to-noise ratio (PSNR) of each frame and with 30 different encoding bitrates (0.1 Mbit to 3 Mbit). We encoded the video sequences with FFmpeg[26] (libvpx, libvpx-vp9 and libx264) with real-time encoding flags (deadline="realtime" for VP8/9 and tune="zerolatency" for H.264), GOP size of 6 and yuv420 pixel format. Results of the 3 depth conversion methods are shown in Figure 3.6. While the HSV conversion is more robust on lower bandwidth constrains (up to 1,7Mbit), our approach (GrayAVG) outperforms the HSV on higher bit-rates that are common for real-time video conferencing (2-3Mbit). However, the main benefit of our approach is that it does not require any HSV or color mapping but works with a simple RGB conversion function that can be directly implemented in a 3D rendering shader (i.e. OpenGL GLSL). This is of particular importance for a web/browser implementation.

While Figures 3.6(a,b,c) include the overall average of all 3 video sequences, we also compared our conversion to the 3 sequences in more detail in Figure 3.6 (d). For this comparison, we excluded VP8 as it significantly underperformed on all methods before (see Figure 3.6 a,b,c). As to be expected, the videos with a lower motion achieve a higher PSNR under the same bandwidth and VP9 achieves a slightly higher PSNR as H.264 (overall average of 44,47 VP9 vs. 42,72 H.264). Thus, similar to other analysis of VP9. However, what is not shown in this graph is that still VP9 is not on pair with H.264 in regards to real-time encoding (delay and required CPU/GPU performance). In this regard, VP9 only offers a marginal PSNR increase from H.264.

3.3.3 P2P vs MCU for SocialVR (Simulation)

To evaluate our system and client performance, we compare MCU vs. peer-to-peer (P@P) transmission with different numbers of simulated users and two types of streams RGB with green chroma background (rendered as flat 2D sprite) and RGB + Depth (rendered as 3D point cloud). Users were simulated with the same stream used in the grayscale evaluation (med - user siting with some movement, MD 51,34 - SD: 27,27) and pre-encoded in H.264 with 2Mbit for the RGB stream and 4 MBit for the RGBD stream. We utilize H.264 under these bit-rates as it provides the best balance of performance and stream quality (based on our results from Section 3.3.2). When using VP9 we observed higher CPU usage in the client and lower frame rate throughput in the MCU. Furthermore, 2/4MBit aligns with the values that are negotiated by Chrome in the P@P case.

We run the server and MCU on a Microsoft®Azure cloud instance (Standard F8s_v2) with 8 vcpus (Intel®Xeon®Platinum 8168 @ 2.70GHz) and 16GB memory. The client runs in a Chrome browser on a VR laptop, MSI GS65-Stealth-Thin-8RF (Intel®Core™ i7-8750H, GeForce®GTX 1070 Max-Q and 32GB RAM).

[26]https://web.archive.org/web/20240213090355/https://ffmpeg.org/

Table 3.2: User Devices for performance evaluation (CPU as Intel Core i7; GPU as NVIDIA GeForce)

Name	CPU	GPU	Memory	Sensor	Location
NL1	i7-8750H CPU @ 2.20GHz	GTX 1070 Max-Q	32 GB	Azure Kinect	NL, Amsterdam
NL2	i7-8700 CPU @ 3.20GHz	RTX 2080	32 GB	Azure Kinect	NL, Katwijk
NL3	i7-6700K CPU @ 4.00Ghz	GTX 980 Ti	16 GB	Kinect V2	NL, The Hague
NL4	i7-7820HK CPU @ 2.9GHz	GTX 1070	24 GB	Azure Kinect	NL, Enschede
FR1	i7-8750H CPU @ 2.20GHz	GTX 1070	16 GB	Kinect V2	FR, Rennes
FR2	i7-47770K CPU @ 3.50GHz	GTX 1050 Ti	16 GB	Kinect V2	FR, Paris
DE1	i7-8750H CPU @ 2.20GHz	GTX 1070 Max-Q	32 GB	Azure Kinect	DE, Berlin

The measurement results of our tests can be seen in Figure 3.7a (for 2D RGB+Chroma) and Figure 3.7b, for 3D RGBD). Further, Figure 3.7c shows the performance of the MCU under the conditions tested. The P@P transmission shows a much steeper curve in terms of CPU resource usage than using an MCU, and both RGB and RGBD behave very similarly. This is as the overhead from multiple encoding and decoding streams is significant against an one stream upload and download in all MCU conditions. This said, P@P has clear advantages on a lower number of users, this is also to be expected as the MCU stream also transmits the uploaded stream back to each client and thus creates overhead on a lower number of users. Under all tests, the Chrome memory usage was kept in a reasonable boundary ranging from 473 MB (2ppl RGB) up to 1368 MB (16ppl RGB) on average. Thus, given our results, it is beneficial from 4 users on to follow an MCU methodology and the system becomes unstable and unusable (when adding capture modules and adding further processing needs when using an VR HMD) in P2P from 6 RGB and 5 RGBD clients. In our current implementation, however, the MCU can support a maximum of 16 RGB and 8 RGBD clients (see MCU performance in Figure 3.7,c). This said, as to be expected from a central transcoding entity, the improved performance on end clients comes at the price of added delay, the full end-to-end (capture to display) delays of the MCU vs P2P in the following (measured with VideoLat and >1000 samples):

RGB delay: P2P 396ms (SD 41) / MCU 564ms (SD 69)

RGBD delay: P2P 384ms (SD 44) / MCU 622ms (SD 68)

Overall, this shows that the MCU might add an significant overhead in terms of delay (but still in a considerable range of real-time communication). Further enhancements in the MCU like GPU accelerated encoding or tiled based compositions (that do not need transcoding) can further increase the number of maximum users and decrease the delay in the future.

3.3.4 Evaluation in realistic user setting

To further evaluate the simulation results, we conducted a set of user sessions in a realistic setting connecting 4 and 6 users (from Netherlands, France, and Germany). The details of the user end points used can be found in Table 3.2. We conducted 6 sessions with a duration of at least 25 minutes. An example of the user test is shown in Figure 3.8. Four user sessions were conducted between the nodes NL1, NL2, FR1 and FR2 in four conditions: P2P with 2D and 3D presentation, MCU with 2D and 3D representation. As the performance is not stable enough for a 6 user P2P condition, we only tested 2 conditions: MCU with 2D and

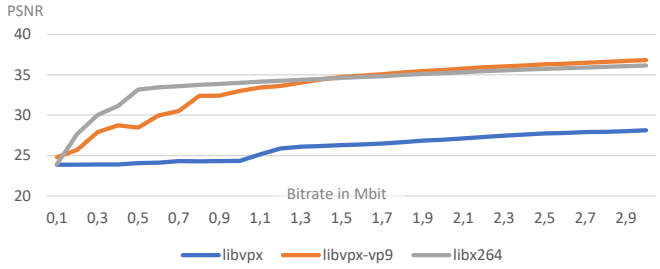
3D representation. The 6 user sessions were conducted between the nodes NL2, NL3, NL4, FR1, FR2 and DE1. The results of these tests are presented in the following. CPU, GPU and network performance was measured with the same Resources Consumption Metrics (RCM) measurement tool [101] as in the simulation evaluation (section 3.3.3), the frame rate was measured via the Aframe stats and the WebRTC delay was measured via the Chrome WebRTC stats.

Figure 3.9, shows the overall performance average per condition (of all user end points) of the Chrome instance running the Web client in terms of CPU, GPU, and rendering frame rate. The values are slightly lower in terms of GPU/CPU usage than in the simulation due to more powerful end points with the same trend in MCU vs P2P resource usage: The MCU condition allows to reduce the CPU load at the cost of GPU usage. As CPU resources are more sparse and necessary for many more processes, this is particularly beneficial to support more simultaneous users and constant high-quality rendering. Important to note is that the frame rates are only indicative and not realistic for the rendering performance in an VR HMD. This is, we conducted the evaluation without a VR HMD to simplify the measurements and user interactions. Furthermore, the browser executes various optimisation strategies to balance the performance load with visual rendering quality. None of the users perceive stuttering or visual impact due to performance and the CPU/GPU load was low enough to allow higher frame rates in VR.

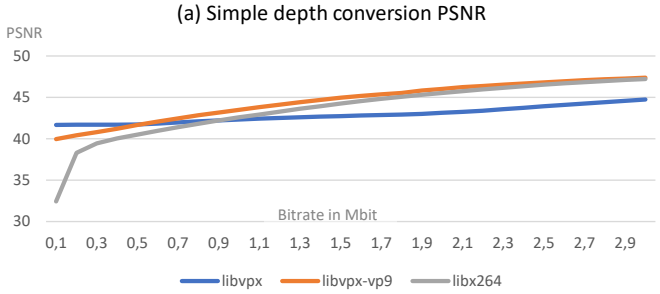
Figure 3.10, shows the overall average per condition (of all user end points) of the video upload round trip delay and jitter. These delays are additional to the overall delay values reported in the simulation evaluation (section 3.3.3). We can observe an overall higher delay and jitter for P2P transmission compared to central MCU transmission. However, one condition "4 users MCU with 2D representation" shows a high standard derivation as one client (FR1) observed higher delay values. This is to be expected in such a test (with a realistic and varying internet connection) and is in the normal boundaries of delay to expect for WebRTC transmission (and in the delay range acceptable for remote communication).

Figure 3.11, shows the overall average per condition (of all user end points) in network traffic. Our results reflect the main benefits of an central WebRTC approach (MCU) as the upload traffic is significantly decreased. This is as in the MCU condition the representation of a user is only uploaded once, while in the P2P condition the user representation has to be uploaded to each other end point. Overall, the MCU is capable to use network resources much more efficiently (based on the cost of central computation, see 3.3.3).

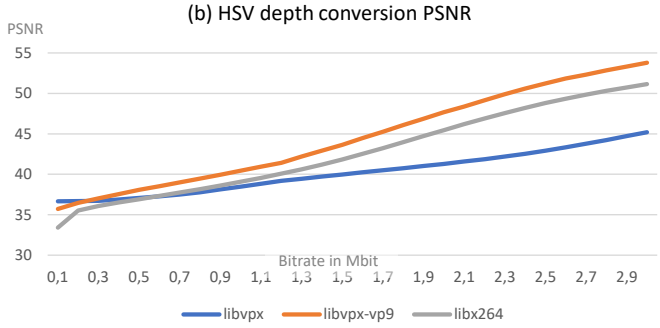
In general, the user evaluation confirms the performance measures of the simulation in more realistic settings. The results show that we can achieve VR communication with similar network transfer rates and slightly more CPU/GPU resource usage compared to video conferencing solutions while being able to render users in 2D and volumetric 3D.



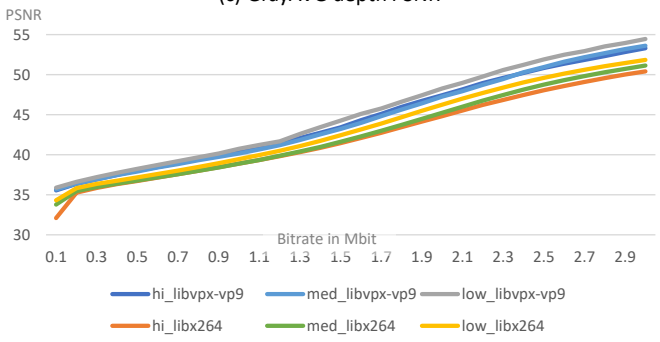
(a) Simple depth conversion PSNR



(b) HSV depth conversion PSNR

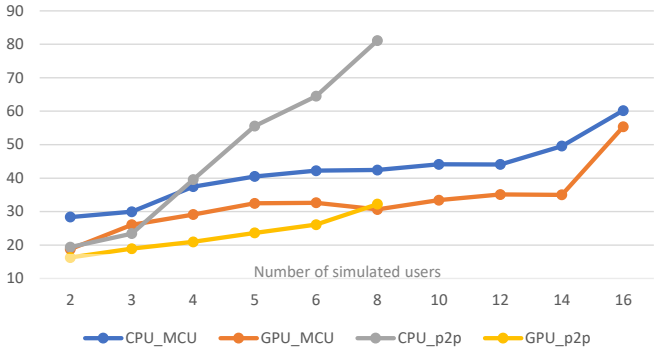


(c) GrayAVG depth PSNR

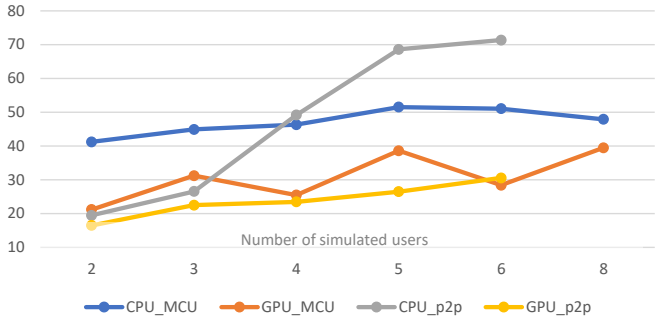


(d) GrayAVG PSNR of 3 video conditions

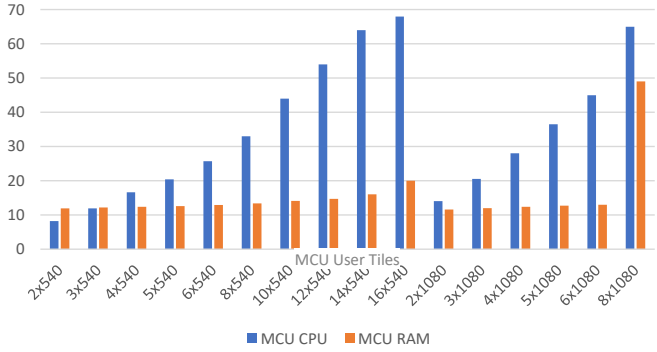
Figure 3.6: Different Depth conversion with VP8, VP9 and H.264 encoding and different bitrate (in MBit)



(a) Chrome performance (CPU/GPU) in % for MCU and p2p transmission of RGB user streams (540x800)



(b) Chrome performance (CPU/GPU) in % for MCU and p2p transmission of RGBD user streams (1080x800)



(c) MCU Performance in %

Figure 3.7: Web client (Chrome) and MCU performance

Figure 3.8: 6 users in 3D performance test

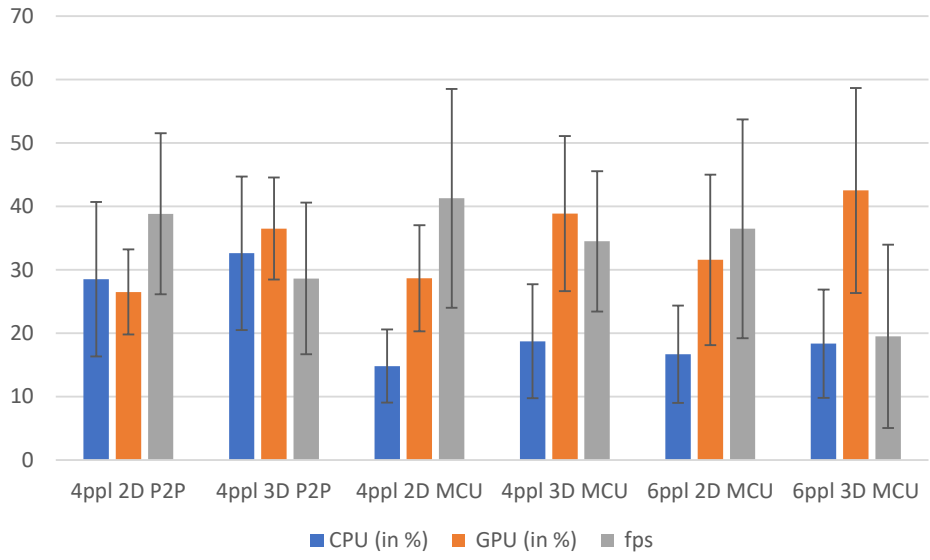


Figure 3.9: Chrome Browser Performance of user test

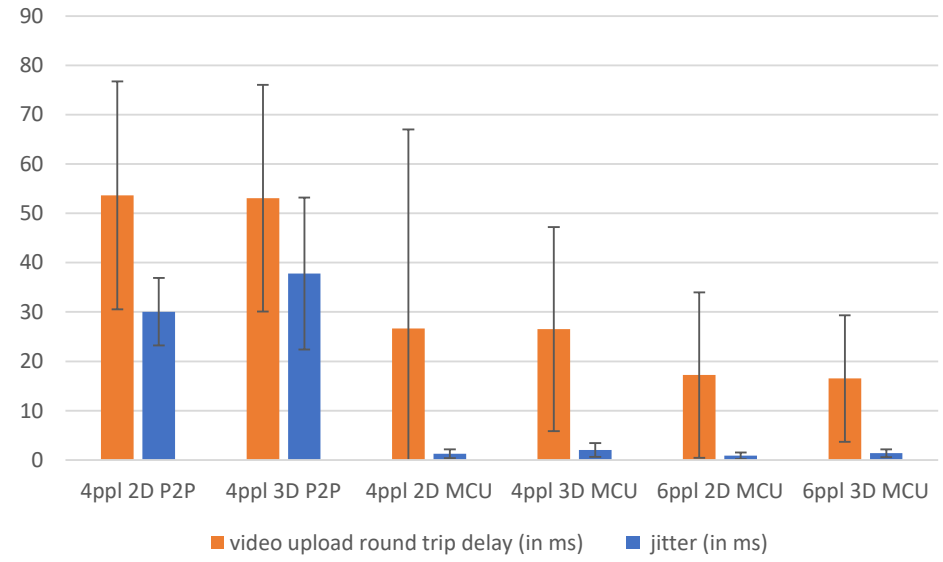


Figure 3.10: WebRTC video upload delay and jitter

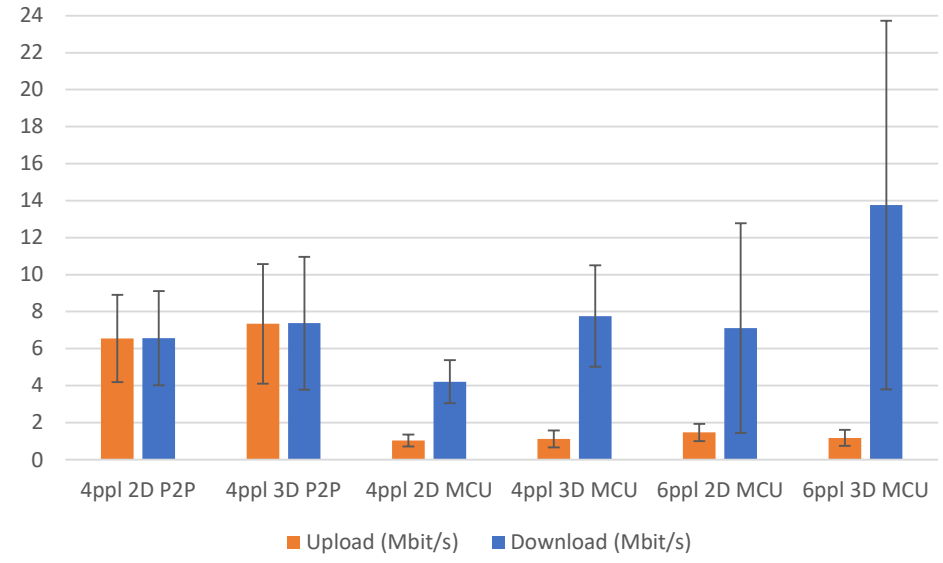


Figure 3.11: Video Upload & Download Traffic

3.4 Discussion

Our current system design and example VR experiences show that multi-user photorealistic immersive media applications are possible in real-time on the web. Allowing to use such applications without downloading and installing large software packages. However, current browser implementations still have some drawbacks with regard to WebXR and other immersive media functions. One problem is that performance can vary significantly between different browsers and different browser versions. This can make it difficult to widely support your application with constant high quality. For the test presented in this chapter, we only used Chrome as a browser client. We also tested a working solution of VRComm with other browsers like Firefox and Edge (e.g., using VP8 or VP9 encoding for WebRTC transmission). Currently, however, daily updates and changes in APIs can still break different aspects of the application and might make a widespread deployment cumbersome. In general, video codec support, frame-accurate synchronization, underlying WebRTC functionality, and the connection of WebXR with different headsets (or the Steam[27] OpenXR[28] Api) still need to mature across browsers to offer a constant and high-quality user experience. To discuss our web-based Social XR system and evaluation results, we follow the structure of our research questions.

RQ 2.1 Is RGBD 2D video transmission suitable for XR communication with volumetric user representations?

Our evaluation (section 3.3) shows that our proposed capture and depth-to-grayscale conversion for RGBD video data is suitable for real-time video transmission under bandwidth considerations typical for current video conferencing systems (H2.1.). However, for other bitrates (i.e., below 1.5 Mbit and above 3 Mbit), as well as pre-encoded content, other solutions might result into a better visual quality. The real strength of our method is that it works on the RGB color space and thus does not require direct access to encoding APIs (e.g., as YUV mapping would require), which makes it a suitable solution for web applications (due to its limitations in not revealing many underlying native media APIs do not allow many other RGB+Depth transmission techniques).

RQ 2.2. Which components need to be extended or added to common video conferencing architectures to allow Social XR?

With VRComm, we extend common video conferencing optimisation solutions, i.e., an MCU to support RGBD video which is important to address the dedicated performance requirements of VR applications. In our current setup, we can support 16 2D (RGB + chroma) users and 8 3D (RGB + Depth) users, while the use of an MCU proves to be efficient from 4 users onwards. It is important to note here that our additions to the MCU in terms of handling our RGBD video format are fully compatible to any other MCU optimisation technique. For example, in more complex use cases one MCU might not be enough to cater for many geographical distributed users. Then a multi-MCU solution (or extended with multiple SFUs) could be deployed. Together with other optimisations like not rendering all users at the same time, this can increase scalability, visual quality and reduce delay.

[27] https://web.archive.org/web/20240404221551/https://store.steampowered.com/steamvr
[28] https://web.archive.org/web/20240420190815/https://www.khronos.org/openxr/

RQ 2.3. What performance and network utilization are expected for Social XR?
Our evaluation of different capture configurations (section 3.3.1) shows reasonable CPU / GPU usage in all conditions. Furthermore, it offers a modular design to add different processing and image improvements like foreground background removal, HMD replacement, and image alignment calibration. One of the main bottle necks of our current approach is the connection to the browser, as this is currently done via a screen rendering and screen capture approach. In the future, this can be mitigated by direct access of the browser to the APIs of the depth sensor (i.e., via the W3C Media Capture Depth Stream Extensions) and a combination of processing within the browser client and in the network.

3.5 Conclusion

In this chapter, we present a web-enabled video-based social VR framework that allows to rapidly develop, test, and evaluate photorealistic VR communication experiences. By combining video conferencing technology with social VR capabilities, we offer a new end-to-end pipeline (capture, processing, transmission, and rendering) to allow real-time shared immersive experiences and volumetric communication. Our novel transmission scheme for grayscale-based depth information via 2D video proofs particularity usefully for web applications (that do not allow other depth conversion due to browser API limitations). Finally, the evaluation of our system in a simulation and realistic user setting shows that our solution utilizes processing (CPU / GPU) acceptable for modern (VR-ready) PCs and under network bandwidth constraints similar to existing video conferencing solutions. Still, more research is necessary to get all aspects of the technology ready (i.e., spatial computing, HMD replacement, enhanced quality transmission of real-time 2D/3D video data, and system scalability for large sets of simultaneous users).

Currently, VRComm can support up to 16 single-camera (low-resolution) users in VR communication experiences. To support more complex and large groups of users and higher-resolution streams, the transmission approach needs to be improved. A novel transmission optimization for RGBD immersive user representation is presented in the next chapter.

4

Transmission Optimization

Video conference applications that allow group communication through video and audio over distance have become commonplace and mainstream. To scale the number of participants in video conferencing systems, the usual practice is to deploy centralized streaming components such as Multipoint Control Units (MCU) or Selective Forwarding Units (SFU). Using these components can become problematic due to significant server or client resource overheads.

In this chapter, we propose a Tile-aware Multipoint Control Unit (T-MCU) that is capable of combining and forwarding streams on a bitstream level. In our solution, multiple VP9 user video streams are combined into a single bitstream, requiring only a single video transmission (single receiving socket, single video decoder, and single rendering texture). Multiple changes (relating to dynamic header and motion vectors) had to be made to the reference VP9 encoder to allow combining bitstream tiles. These changes are available as open source and are described in this chapter, including their impact on encoding performance. Furthermore, all changes are fully compatible with existing VP9 decoders. The chapter also presents an experimental design study to evaluate our new T-MCU under different streaming conditions (including the impact on the receiving client) within an immersive 3D communication system utilizing RGBD video data. Our approach significantly reduces the performance requirements of the client (10-20%) and server (~80%) at the cost of increased bandwidth (~16%). Ultimately, the T-MCU allows immersive 3D communication applications to support at least 16+ simultaneous users.

This chapter is based on Simon N.B. Gunkel, Rick Hindriks, Yonatan Shiferaw, Sylvie Dijkstra-Soudarissanane, and Omar Niamut. Vp9 bitstream-based tiled multipoint control unit: Scaling simultaneous rgbd user streams in an immersive 3d communication system. In *Proceedings of the 15th ACM Multimedia Systems Conference*, MMSys '24, page 23–33, 2024 [58]

The rise of video conferencing as a basis for communication has transformed the way we interact with each other. This is specifically to allow remote working and to allow communication over distance [154]. However, most video conferencing applications still have significant limitations to allow complex communication. One solution to allow for better and more natural interactions is immersive technology (that is, Extended Reality, XR) [21, 155, 156]. However, moving from existing remote video conferencing towards immersive technology creates a pressing need for immersive volumetric communication systems that can support large groups of users. Designing such immersive systems presents a significant challenge, as it requires meeting demanding and rather stringent performance requirements for handling many simultaneous users in a communication session in real-time.

Currently, most multiparty communication systems function according to two fundamental paradigms [157], (i) peer-to-peer (P2P), or through (ii) an intermediary centralized entity (i.e., Selective Forwarding Unit, SFU, or Multipoint Control Unit, MCU).

In the P2P paradigm, streams are sent and received between all clients, effectively creating a full mesh streaming topology. Therefore, for scalability, the P2P approach can be considered the worst-case scenario, since each end device needs to encode and upload its own streams to all other users, creating an quadratic increase in resource demands.

This problem is mitigated by an SFU where each client only encodes and uploads its stream once, and a central entity forwards the stream to all other clients (as needed). The use of an SFU results in many stream connections, decoders, and rendering entities that all need to be correctly managed in the receiving client, which can still lead to high resource demands when dealing with many users in a simultaneous communication session.

Using an MCU can shift resource demands from end devices to the central server entity by not simply forwarding the streams (SFU), but decoding, composing, and encoding all user streams into a single stream. This single stream reduces the client needs to a single receiving transmission and a single decoder instance, thus reducing performance needs, but can create high resource demand on the central processing unit.

Therefore, the current architecture and implementation of traditional SFU & MCU systems may not be sufficient to support demanding performance requirements, such as high quality, low package losses, and real-time handling of many simultaneous users in complex communication sessions with many high-resolution user streams. Such performance requirements increase further when dealing with photorealistic immersive 3D communication solutions. That is, the visual information and transmitted streams increase in size and complexity, e.g., by adding a depth layer (D) in addition to the regular (RGB) color information, and thus require more bandwidth and computational resources in the system and clients.

To address this challenge, we propose a solution that supports modern video encoding techniques to stream RGBD content. Our tile-aware MCU can combine streams on the bitstream level without the need for decoding/encoding the video streams. Thus, enabling the benefits of a single-stream MCU but significantly reducing the performance needs of the server.

Although previous research described bitstream-based tiling for real-time communication as an initial concept (tested within a limited nonreal-time study) [130]. This chapter is the first to present real-time VP9-tiling (as most restrictive encoder) in a full end-to-

end communication system and an experimental design study on the system delay, client processing, and server processing. We hypothesize that a T-MCU system can increase scalability and support large user groups in simultaneous communication sessions, allowing high-quality real-time visual communication between users.

The research in this chapter and the evaluation based on an experimental design approach of our system and its components were based on the following research questions and hypotheses:

RQ3.1. Is tile-based composition possible in real-time with a central composition unit and VP9?
H3.1. With modern video encoders, we can create a tile-enabled MCU (T-MCU) that can continuously combine video streams in real-time on the bitstream without the need for decoding and encoding.

RQ3.2. What is the impact on resource usage for clients and the central server, when using a tile-enabled VP9 encoding and composition (T-MCU)?
H3.2. By managing streams on a bitstream level in a central component, we expect a significant performance improvement on both the system and the client, but at the cost of increased bandwidth.

RQ3.3. How many simultaneous 3D photorealistic users could this approach allow in an immersive communication session?
H3.3. We expect that the performance improvements of a T-MCU will allow one to transmit and render at least 16 3D users in parallel.

By addressing these three research questions, we show that our system can support large groups of users in virtual communication while maintaining real-time performance and high-quality visual representation. The results of this research will provide valuable insight for designing more scalable immersive and volumetric communication systems and increasing their applicability.

In this chapter, we present several contributions that address the scalability challenge. First, we propose extensions to the VP9 encoder that allow for parallel encoding, with decodable tiles. Second, we introduce the concept of Tiled-enabled Multipoint Control Unit (T-MCU), a new technical concept for volumetric user communication that significantly reduces system complexity and overhead compared to traditional Multipoint Control Units (MCU). Third, we demonstrate how T-MCU enables support for large user groups in XR with photorealistic volumetric user representations.

Finally, our experimental design study provides in-depth details on client and server performance, as well as the delay impact of the T-MCU approach. Our contributions provide a valuable framework for designing scalable volumetric representation of users in immersive communication systems, which will alter the way we interact with virtual spaces.

4.1 Related Work

Although 2D video conferencing is commonplace in our current lives, recent approaches try to elevate remote communication with immersive communication systems (ICS) in 3D [34, 158–160]. One way to enable photorealistic user representation in ICS is through RGBD (color and depth) capture devices [161]. There are multiple examples that use RGBD sensors for remote telepresence and immersive communication [33, 35, 39, 70]. An advantage of

RGBD-based approaches for immersive communication is that they can utilize standard 2D video compression, which was studied in [72]. More recent examples show the conversion of depth information to grayscale [57, 73, 74]. Using standard 2D video compression for the depth part also allows one to reuse existing 2D video transmission pipelines. The suitability of using an existing 2D video encoder for the transmission of user images in RGBD for immersive communication was studied in [33]. Alternative 3D representation and transmission formats are point clouds (PC) [34, 35], time-varying mesh (TVM) [36–38], or 3D teleimersion (3DTI) [40, 41, 162]. However, these alternatives face one or more of the following problems: complex capture setup, high bandwidth, high processing needs, and largely incompatible with existing transmission technology. The focus of this chapter is on RGBD video over existing 2D video conferencing transmission.

In general, 2D video conferencing solutions are very established and widely tested [97, 163, 164]. Usually 2D video conferencing systems follow one of the following paradigms [157] : peer-to-peer (P2P), centralized Selective Forwarding Unit (SFU) or centralized Multipoint Control Unit (MCU). Recent examples also introduced central composition units for immersive communication [33, 165]. One of the main challenges for all these immersive communication system solutions [33, 35, 39, 70, 158–160] is the scalability of users, since either the user rendering client or the server / system experiences performance bottlenecks, thus limiting the maximum number of simultaneous users in one communication session. Therefore, in this chapter, we investigate whether a central composition method based on combining bitstream tiles can solve these bottlenecks.

Currently, to the best of our knowledge, there are only four commonly available codec formats that allow tiling: VP9 [166], H.265 [167], H.266 [168] and AV1 [169]. As many communication applications run on the browser (over WebRTC), it is important to note that H.265, H.266 and AV1 are currently not widely supported in all browsers, compared to VP9. For example, there is initial support for AV1 in Google Chrome that can also be used for WebRTC (tested[1] with v113.0.5672.64) but does not work in Microsoft Edge (tested[1] with v112.0.1722.68) and many other configurations. Any test using H.265[2] was not successful. Thus, even though we expect better tiling support in H.265, H.266 and AV1 (as they are more modern encoders), in this chapter, we focus on VP9 as encoding entity to address a generic solution with wider applicability.

The concept of tiling [170] and how to utilize tiling in mixing compressed video stream [171, 172] is well known for H.265/HEVC encoding. Furthermore, bitstream-based tiling for real-time communication was previously studied for H.265 in [130]. This is, [130] offers an initial study by pre-recorded sequences only, but not testing a real-time end-to-end communication system. The work presented in this chapter is the first of its kind to present real-time VP9-tiling (assuming that it will be the most restrictive encoder) and a complete end-to-end evaluation of tiling in a communication system (including system delay, client processing, and server processing).

[1] https://web.archive.org/web/20240503144309/https://janus.conf.meetecho.com/echotest.html?vcodec=av1
[2] https://web.archive.org/web/20240503144425/https://janus.conf.meetecho.com/echotest.html?vcodec=h265

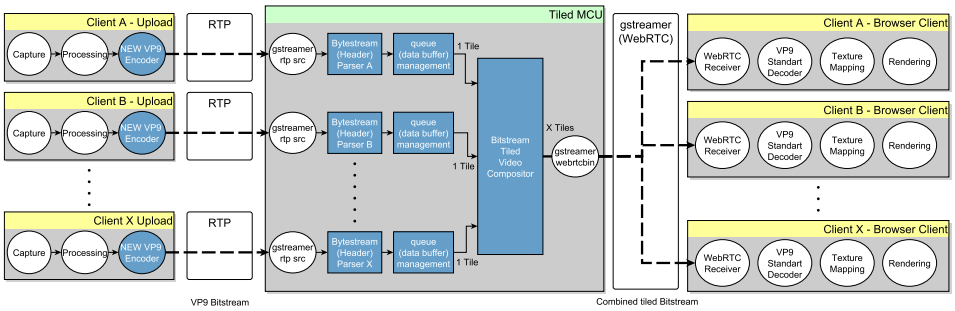


Figure 4.1: Tiled MCU schematics

4.2 VP9 Tiled-Encoding & Composition

Our work is based on the libvpx[3] WebM VP8/VP9 Codec SDK. It is important to note that the VP9 specification is still in the draft stage [173] at the time of writing this document. However, VP9 is widely used and its frame format is well documented [173][4]. The strength of VP9 lies in its inter-frame prediction using spatial and temporal redundancy and advanced entropy coding methods [166]. This allows VP9 to offer high encoding gains [174] with acceptable performance.

Regarding tiling and VP9 there are some limitations to consider that we will outline in this section and how we addressed them in order to allow combining multiple bitstreams that were produced by different encoder instances (and different source content) into one standard complied (mosaic) bitstream.

For simplicity, our approach is fixed to profile 0, and we do not address golden, alt-ref and superframes. The profile defines the core feature set of the encoding relating to color depth and chroma subsampling. Pofile 0 is the most restrictive with a color depth of 8 bit per channel and a YUV420 chroma subsampling. With these settings, our preliminary tests indicated that any stream resolution greater than 16,384 pixels was not rendered (in the Google Chrome browser and FFplay) compared to the theoretical maximum of VP9 of up to 65,536 pixels. Further, it is important to note that in VP9 the number of tiles needs to be a power of two (in the work presented in this chapter, we tested tile sizes of 2, 4, 8, 16 and 32 tiles).

To understand the tiling in VP9 it is important to first understand the frame structure of VP9 ([173], page 25). A frame always consists of (a) an uncompressed header, (b) a compressed header, and (c) the frame (tile) data. Another requirement is that the video format and encoder settings are the same for all streams, including that all tiles must have the same fixed width, height, framerate, and group of pictures (GOP) structure. Based on these settings, we work with the following assumptions to combine streams of different sources (actions per frame need):

[3] https://web.archive.org/web/20240503144440/https://chromium.googlesource.com/webm/libvpx/+/8874873bef4089164a697c62fbfdeeaa1cc678ac

[4] More technical details about VP9 can be found at https://web.archive.org/web/20240502000048/https://forum.doom9.org/showthread.php?t=168947

(a). change header to reflect combined resolution and # of tiles

(b). align all compressed and uncompressed header bits

(c). eliminate all conflicting header settings in the encoder

In order to address (c) and produce frames with compatible headers, we made the following changes to the VPX/VP9 encoder[5]: It is important to note that our changes affect only the encoding gain, but do not make changes on the encoding itself or the compatibility with existing VP9 decoders.

Ignore coef_probs: Coefficient probabilities are part of the VP9 entropy encoding (related to motion vectors). These are usually created for the entire frame and part of the compressed header. Modifying or combining these probabilities in the compressed domain (in the context of the current VP9 design) is probably not possible or would add significant complexity to the central forwarding unit. In our current approach, we avoid the use of coefficient probabilities and some motion vectors. That is, effectively, we completely deactivate the compressed header, with the expectation of a significant increase in bandwidth compared to the reference encoder.

Fix interpolation: The method of interpolation in VP9 can change per frame and is stated in the uncompressed header. As this is a per-frame property, misalignment between frames can cause distortions in the rendering. Therefore, we fix the interpolation method for all frames to BILINEAR.

Skip adaptive_pred_interp_filter: Aligned with the fix of the interpolation method to BILINEAR we also skip any adaptive interpolation prediction.

Fix TX_MODE: The transform mode is related to the block decomposition and allows the encoder to bundle similar values to save bandwidth. This is a value that is selected and can vary per frame. Thus, any misalignment in transform mode between tiles can have unpredictable consequences (i.e., completely corrupted frames). We fix the transform mode to ONLY_4X4, which is the smallest level and thus results in the highest quality with a potential increase in bandwidth compared to the reference encoder. Increasing this fixed transform mode might result in bandwidth savings with a decrease in quality.

Fix base_q_idx: The base_q_idx indicates the base frame quantization index. This is used for Y AC coefficients and as the base value for the other quantizers ([173], page 67). Although the resulting quantizer levels are specified in the segmentation map, they will be based on the base_q_idx that is set in the uncompressed header. Thus, having a misalignment between frames can result in unpredictable behavior, i.e., distortions and completely corrupted frames. Therefore, the encoder has to adhere to a fixed quality encoding, which is not ideal for video conferencing, where a stable bandwidth is usually preferred. Furthermore, it means that the quality parameter is shared between all users, making adjustments for individual user streams more challenging.

The effect of these changes is to limit any dynamic encoding behavior that can have an impact on combining bitstream tiles of different initiating sources. This effectively will deactivate the motion vectors and result in a constant bitrate (CBR) encoding scheme on the basis of a configurable fixed base_q_idx. Further, all changes are limiting the encoding

[5] *Our tile-enabled VP9 encoder modifications are published as open source patches to the WebM VP9 encodering SDK:* https://web.archive.org/web/20240503144954/https://github.com/TNO/vp9-bitstream-tiling

Table 4.1: Encoding performance under different conditions

sequence	motion SI	motion TI	VPX bitrate	VPX psnr	VPT bitrate	VPT psnr	delta bitrate /psnr	delta %
User W-1	32.63	2.74	1447k	50.18	1583k	47.85	4.24	+14.71%
	±0.19	±0.41	±885.75	±3.15	±691.36	±3.55		
User W-2	32.60	3.57	1671k	50.08	1831k	47.83	4.92	+14.74%
	±0.77	±1.07	±975.16	±3.17	±766.46	±3.52		
User W-3	33.81	4.61	1963k	49.70	2157k	47.56	5.87	+14.86%
	±0.78	±1.62	±1125.17	±3.29	±886.68	±3.51		
User M-1	37.53	2.08	1322k	49.66	1468k	47.18	4.50	+16.89%
	±0.11	±0.63	±879.21	±3.25	±664.56	±3.42		
User M-2	37.90	3.62	1717k	49.48	1926k	47.30	6.01	+17.33%
	±0.81	±1.95	±1048.89	±3.32	±806.95	±3.57		
User M-3	39.30	5.21	2021k	49.32	2227k	47.17	6.23	+15.21%
	±1.15	±2.92	±1136.41	±3.33	±882.42	±3.57		
Total	-	-	1690k	49.74	1865k	47.48	5.30	+15.61%
			±1044.49	±3.27	±834.83	±3.54		

but do not change anything outside of the VP9 specification, thus all streams are decoded via existing VP9 decoders.

Note: Throughout the chapter, we use the name "VPX" as a reference to the original encoder (libvpx VP9 encoder) and "VPT" for our modification to allow tile-based composition on the bitstream. Furthermore, both VPT and VPX encoders are compiled via MINGW[6] (gcc version 12.2.0).

[6]https://web.archive.org/web/20240503133109/https://www.mingw-w64.org/

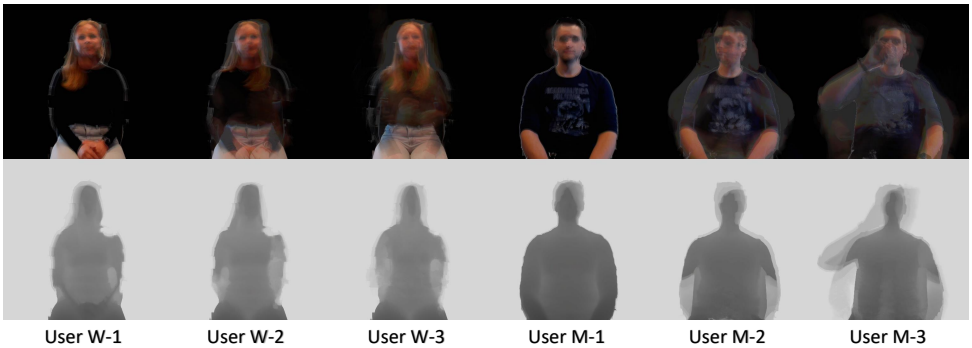


Figure 4.2: User mosaic of testing sequences with different motion (motion shown as motion blur)

4.2.1 Communication System & T-MCU

As a basis for implementation and comparison of the T-MCU we use the Web-based immersive communication system as presented in [33]. Figure 4.1 shows the revised schematics of the T-MCU and the communication between the T-MCU and the clients. The client upload uses a Python-based capture component, presented in [57]. An Azure Kinect was used as a RGBD capture device. The output of the capture module consists of an RGB + grayscale depth image with a resolution of 512x1024 pixels (top half color and bottom half depth) with 30fps. An example representation of this stream is shown in Figure 4.2. The captured images are then encoded with the VPT (VP9 modified libvpx) encoder and uploaded via RTP streams to the T-MCU. Important to note is that the new T-MCU has the same API as the MCU presented in [33]. Thus, both components are interchangeable for easier testing. One of the main differences to [33] is that the output of the (T-)MCU component is available via a direct GStreamer WebRTC link (versus a Janus Gateway server, i.e., as presented in [33]). This has the advantage of a more controlled and easier session setup (as well as producing identical constraints to test an SFU condition).

Internally, the T-MCU is a complete new implementation. For each incoming user stream, a GStreamer component will format and repackage each frame and forward it to a bitstream frame (header) parser. Each parsed frame is queued in preparation for the Bitstream Tiled Video Compositor (BTVC). The BTVC loops in a defined framerate and GOP structure over all frames (we use 30fps with a GOP size of 5). On each "tick", the BTVCompositor will collect all input frames, update the header to the correct (complete) image size and number of tiles, and append the tile data of all input frames. Finally, the output image is forwarded through the GStreamer WebRTC proxy in a broadcast to all receiving clients. After receiving, each client will decode the video with a standard VP9 decoder, map the output to one texture, and render it in 3D.

Note that we added the function to have a single render texture and render shader for the (T-)MCU image plane in the client. This offers a significant performance improvement vs. a texture mapping and render shader per user, but requires a more complex geometric transformation in the render shader to position remote user at their "correct" location in the 3D environment.

Additionally, we implemented a debug mode that allows one to disable the 3D rendering and in this way emulate a more traditional 2D video conferencing condition. We implemented a viewing mode in the receiving client that allows it to connect to many individual upload streams; in this case, each upload client sends its stream via an individual GStreamer WebRTC proxy. This allows us to emulate a condition that is comparable with a SFU, where the client receives all user tiles as an individual stream and that required an individual encoder per stream. Finally, we implemented a dummy client uploader that does not upload real-time video streams from the Azure Kinect but precaptured and preencoded video frames to allow system performance tests under identical conditions.

4.3 Evaluation and Results

In this section, we present our evaluation of the encoder, the central composition server, and its impact on the rendering client. We tested our encoder with six test sequences (explained in the following) and our system under two conditions (i) real-time capture through an Azure Kinect RGBD capture device and a Python-based capture application [57] for delay

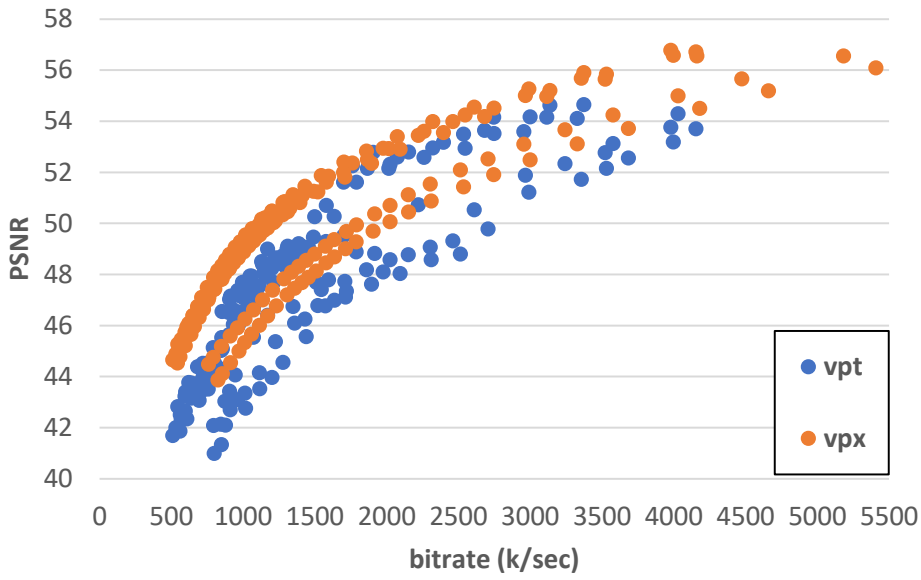


Figure 4.3: Encoder Bitrate vs. PSNR

measures and (ii) a dummy upload client that uploads one of the pre-encoded video streams of the test sequence (test sequence = M-2) for performance measures. All measurements were executed on a single PC (Acer Predator PH315-55 PC, with Intel i9-12900H, GeForce® RTX 3080 and 32GB RAM), running all server and client components, and stream settings per tile of 512x1024 pixel resolution and 30fps. The resolution of 512x1024 was chosen, as a resolution power of 2 is usually beneficial in the encoding, decoding and rendering of the image [57].

4.3.1 VP9 encoding performance

To test the encoding performance and compare our new VPT encoder with the existing VPX encoder, we created 6 test sequences with 2 different users with varying motion (both users performed motion with low, medium, and high motion, simulating remote communication). The test sequences are shown in Figure 4.2 (motion is visualized by combining the entire video into a long-exposure image and thus showing the user's movement as motion blur). Furthermore, we measured motion as spatial information (SI) and temporal information (TI) with VQEG SI / TI calculation tools[7], as defined in ITU-T P.910 [175] (see Table 4.1). Although the user itself did not change much (consistent complexity of SI), the different amount of movement can be observed in an increasing TI. Each sequence consists of 30 seconds with 30fps and 512x1024 pixel resolution. Both encoders were tested with the 6 sequences under 30 conditions each and constant bitrate encoding settings (VPT $base\_q = 28 - 128$ in steps of 4; VPX $q = 6 - 32$). For VPT, each base_q was established by rebuilding the encoder, and for VPX, the q was s in the command-line argument, the

[7] https://web.archive.org/web/20240324134728/https://github.com/VQEG/siti-tools

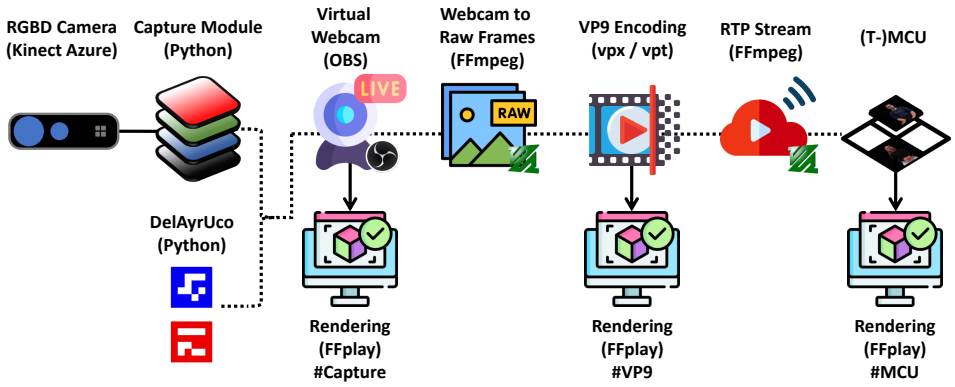


Figure 4.4: Pipeline for Latency measures

encoders were used with the following commands:

VPT: ./vpxenc –passes=1 –width=512 –height=1024 –fps=30/1
–profile=0 –i420 –kf-max-dist=5 –kf-min-dist=5
–error-resilient=1 –lag-in-frames=0 –tile-rows=0
–tile-columns=0 –cpu-used=9 –rt - –ivf -o -

VPX: ./vpxenc –passes=1 –width=512 –height=1024 –fps=30/1 –profile=0 –i420 –kf-max-dist=5 –kf-min-dist=5
–lag-in-frames=0 –tile-rows=0 –tile-columns=0 –cq-level=***q*** –end-usage=***q*** –min-q=***q***
–cpu-used=9 - –ivf -o -

After encoding each condition, a peak signal-to-noise ratio (PSNR) value per frame was calculated by decoding the encoded video and comparing it to the uncompressed original input frames. Table 4.1 shows the results of the sequence tests. As expected, with higher motion, the bitrate of the encoding increased (considering the same PSNR quality). Furthermore, Figure 4.3 shows the relation between all sequence bitrates and their resulting PSNR values. We can observe that the consistent quality mode that we use for our encoders results in a very consistent and stable PSNR. However, it is important to note that the PSNR values cannot be identical between the two encoders, since the q setting for VPX works completely differently from the base_q value in VPT. To compensate for the difference in average PSNR, we calculate the delta of both by dividing the bitrate by the PSNR value. Our results show an overall encoding overhead of VPT of approximately 16% (with a similar PSNR). This is to compress image frames with the same PSNR VPT requires a 16% higher bitrate.

Table 4.2: (T)MCU performance and latency measured with DelAyrUco (values as mean, SD as ±)

Condition	2-Tiles	4-Tiles	8-Tiles	16-Tiles	32-Tiles
MCU CQ - latency (in ms)	367.62	371.37	376.55	419.80	37178.73
	(± 23.33)	(± 21.82)	(± 20.69)	(± 23.48)	(± 41600.44)
MCU CBR - latency (in ms)	351.55	389.64	373.14	409.98	69618.20
	(± 17.12)	(± 23.56)	(± 24.87)	(± 25.67)	(± 54483.56)
T-MCU - latency (in ms)	282.64	304.01	319.65	379.26	423.59
	(± 17.51)	(± 28.00)	(± 48.03)	(± 53.17)	(± 55.91)
MCU CQ - loss (in fps)	5.51 (18.4 %)	3.84 (12.8 %)	4.04 (13.5 %)	2.97 (9.9 %)	29.52 (98.4 %)
MCU CBR - loss (in fps)	2.33 (7.8 %)	3.49 (11.6 %)	4.29 (14.3 %)	5.28 (17.6 %)	29.60 (98.7 %)
T-MCU - loss (in fps)	0.66 (2.2 %)	0.47 (1.6 %)	0.24 (0.8 %)	0.55 (1.8 %)	1.18 (3.9 %)
t-test MCU CQ Δ MCU CBR	s=69.7	s=-102.8	s=26.7	s=100.3	
	p=0.00	p=0.00	p=0.00	p=0.00	
t-test MCU CQ Δ T-MCU	s=369.6	s=354.0	s=292.3	s=265.0	
	p=0.00	p=0.00	p=0.00	p=0.00	
t-test MCU CBR Δ T-MCU	s=373.5	s=437.1	s=264.4	s=195.3	
	p=0.00	p=0.00	p=0.00	p=0.00	

4.3.2 Latency

To test the delay of the VPT encoder and the T-MCU, we measured three types of delay (capture, encoding and full end-to-end). The complete delay measurement chain is shown in Figure 4.4. To execute the delay measurements, we utilized two delay measurement tools, VideoLat[8] [102] and DelAyrUco[9]. Compared to Figure 4.1 in Figure 4.4, the upload client can be exchanged with DelAyrUco, and the "Browser Client" is replaced with FFplay.

VideoLat is a one-way glass-to-glass delay measurement tool. A PC (Mac) with webcam acts as a measuring device, constantly displays QR-codes, and measures the time from display to webcam recording. VideoLat is calibrated in an initial step by pointing its own camera to the own screen; it measures the default (hardware) delay of the system (which will be subtracted from subsequent measures). In our setup VideoLat was used on a MacBook Retina (15-inch, early 2013 model with 2.4 GHz Intel Core i7, 8GB RAM and MacOS 10.13.6) with a Dell WB7022 as capture camera.

DelAyrUco, similar to VideoLat, measures the one-way delay but then on a system level, excluding the delay of the webcam and the rendering (screen) device. Utilizing a virtual webcam driver, it consistently creates visual (ArUco) markers that are recorded via screen capture (at 100fps) and analyzed post-measurement. Compared to VideoLat, DelAyrUco significantly reduces the error and standard deviation and provides measurements based on 30fps streams, including loss estimates.

Capture & encoding delay To measure delays with VideoLat, we use an Azure Kinect as a RGBD capture sensor and a Python-based capture module [57], which outputs RGBD images with 512x1024 pixel resolution and 30 fps to a virtual webcam driver (provided via Open Broadcaster Software, OBS Studio[10]). In the DelAyrUco measuremets, images with 512x1024 pixels resolution and 30 fps are directly sent by DelAyrUco to the same virtual webcam driver, instead of the Azure Kinect capture.

To measure the baseline capture delay, we use the webcam image rendered on the screen using FFPlay[11] ("#Capture" in Figure 4.4). Alternatively, we get the raw frames from the virtual webcam driver (python capture module) via FFmpeg[12] and send them to the encoder (either VPT or VPX); the encoder will pipe the encoded frames to FFplay. FFPlay will decode and render the frames on screen ("#VP9" in Figure 4.4). Both encoders use constant quality encoding mode with the same command as presented in Section 4.3.1 (VPT $base\_q = 76$; VPX $q = 24$). All measurements (for both tools) were executed with at least 5 min measure time (resulting in a minimum of 500 samples for VideoLat and 9000 for DelAyrUco).

Table 4.3, shows the delay measurements of the encoder ("#Capture" and "#VP9" in Figure 4.4). Compared to the baseline capture delay, we can observe that the encoder adds a delay of approximately 2 frames ($1s/30frames = 33ms * 2$). Furthermore, the latency of both encoders is similar (with a difference of approximately 2.4 % higher delay of VPT). This holds true for both types of measurement, system delay (DelAyrUco) and glass-to-glass

[8] https://web.archive.org/web/20240113164432/https://videolat.org/
[9] https://web.archive.org/web/20240503144828/https://github.com/TNO/DelAyrUco
[10] https://web.archive.org/web/20240215044704/https://obsproject.com/
[11] https://web.archive.org/web/20240211164442/https://ffmpeg.org/ffplay.html
[12] https://web.archive.org/web/20240213090355/https://ffmpeg.org/

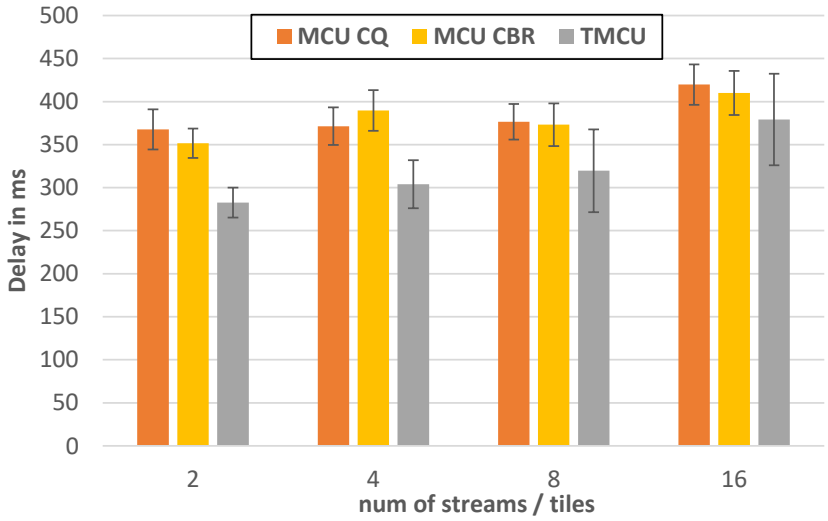


Figure 4.5: (T-)MCU latency comparision

delay (VideoLat). The DelAyrUco also show a stable stream with no significant loss. Finally, we can conclude that with <300ms (capture-to-render) delay, the encoders will be suitable for real-time communication.

Table 4.3: Encoding and decoding performance latency (e2e)

latency condition	Capture	VPX	VPT MOD
VideoLat (ms)	216 (±27)	283 (±29)	290 (±34)
DelAyrUco (ms)	9 (±18)	72 (±19)	74 (±18)
DelAyrUco loss (fps)	0.41 (1.37%)	0.45 (1.48%)	0.45 (1.51%)

Full end-to-end delay We tested the latency of the T-MCU in a complete end-to-end setup ("#MCU" in Figure 4.4), and compared it under two conditions MCU (as presented in [33])) and T-MCU (as presented in this chapter). Only the VPT encoder and the DelAyrUco tool are used for this test. This is in relation to Figure 4.1, the upload client is exchanged with DelAyrUco, and the "Browser Cleint" is exchanged with FFplay. The VPT encoder is set to constant bitrate encoding ($base\_q = 76$; see Section 4.3.1). We tested the delay for the MCU under two output encoding modes, i) constent quality (CBR, as in previous tests of this article with GStreamer settings of end-usage=cq and max-quantizer=24) and ii) constent bitrate (CBR, similar to the settings presented in [33] with GStreamer settings of end-usage=cbr and target-bitrate={3000 * 1000 * #tiles}). Each user stream has its own VPT encoder (grabbing frames from the same virtual webcam device) and pipe the encoded frame as data to FFmpeg, which converts the raw bitstream to RTP and sends it to either an MCU or T-MCU. The (T-)MCU will send the composed mosaic video via RTP to FFplay,

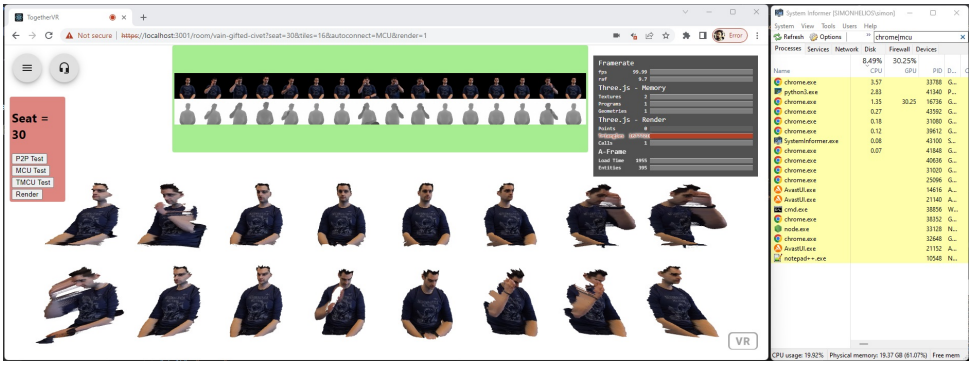


Figure 4.6: WebXR Browser Client rendering 16 tiles (MCU)

which will display it on the screen, via FFPlay ("#MCU" in Figure 4.4). In addition, we used two configurations for the MCU, with constant quality (CQ, max-quantizer = 24) and constant bitrate (CBR, bitrate = 2M x #tiles).

Table 4.2 and Figure 4.5, show the delay measurements of the end-to-end (T-)MCU test. Adding the approximate 200 ms capture and rendering delay (see Table 4.3), the MCU delays are similar to those previously reported in [33]. Please note that the MCU presented in [33] used H.264 to receive and send streams, while in this chapter we use VP9. However, the MCU measures show some inconsistencies in the frame rate with 10-18% loss and fully break with more than 16 tiles (98% loss for 32 tiles). In comparison, the T-MCU shows a consistent frame rate under all conditions and an improved delay of 8-30%. we can also observe in the T-MCU conditions a slight increase in delay and standard deviation with an increase in stream tiles. To further analyze the results, we performed a Welch's t test (assuming unequal sample sizes and unequal variances, the results of the t test are reported in Table 4.2). The t-test confirms that all measures are significantly different ($p = 0$) and there is a slight decrease in difference with increasing tiles. Please note that we did not include the results of the t-test for 32 tiles as the MCU values are not consistent under that condition.

It is important to note that the numbers reported in Table 4.2 and Figure 4.5 are the total average (and standard deviation) across all tiles. The main reason for this increase is the scheduling of the different streams, based on I-frame alignment (within the GOP distance). For the work in this chapter, improving the delays (and stream management) further was out of scope; however, we believe that all delay values can be greatly improved by a) improved stream scheduling, b) I-frame alignment based on I-Frames on request (align streams via the sending encoder), and c) fine tuning of all buffers (e.g., RTP jitter buffers in the sender, MCU, and playout).

Table 4.4: Chrome Client CPU & GPU (in mean %, Decoding & rendering) performance (SD as ± in %) and t-test (p-value as p)

Render.	Condition	00 Tiles CPU	00 Tiles GPU	02 Tiles CPU	02 Tiles GPU	04 Tiles CPU	04 Tiles GPU	08 Tiles CPU	08 Tiles GPU	16 Tiles CPU	16 Tiles GPU
2D	SFU	1.66	8.1	2.26	12.12	2.61	15.5	3.23	22.58	4.62	33.63
		(±0.19)	(±4.49)	(±0.19)	(±4.73)	(±0.2)	(±5.26)	(±0.21)	(±6.08)	(±0.27)	(±6.84)
2D	MCU	1.73	7.93	2.52	6.12	2.6	8.49	2.8	12.85	3.22	21.26
		(±0.2)	(±4.05)	(±0.2)	(±1.1)	(±0.21)	(±1.39)	(±0.19)	(±1.73)	(±0.21)	(±2.36)
2D	T-MCU	1.71	7.95	2.45	6.85	2.52	9.72	2.64	15.55	2.81	26.81
		(±0.17)	(±3.86)	(±0.17)	(±1.15)	(±0.18)	(±1.43)	(±0.19)	(±2.04)	(±0.2)	(±3.2)
3D	SFU	1.67	8.21	2.45	15.25	2.94	25.03	3.78	37.3	5.22	50.37
		(±0.17)	(±4.11)	(±0.18)	(±2.99)	(±0.21)	(±1.07)	(±0.2)	(±1.26)	(±0.29)	(±1.73)
3D	MCU	1.73	8.08	2.6	14.75	2.7	20.87	2.99	30.02	3.41	39.22
		(±0.17)	(±4.02)	(±0.18)	(±0.76)	(±0.2)	(±0.88)	(±0.28)	(±0.95)	(±0.25)	(±1.43)
3D	T-MCU	1.69	8.17	2.54	15.36	2.62	22.08	2.72	32.32	2.94	42.63
		(±0.19)	(±3.87)	(±0.19)	(±0.78)	(±0.25)	(±0.94)	(±0.21)	(±1.1)	(±0.22)	(±1.64)
	T-Test										
2D	SFUΔMCU	p=0.23	p=0.23	p=0.00	p=0.00	p=0.00	p=0.00	p=0.00	p=0.00	p=0.00	p=0.00
2D	SFUΔT-MCU	p=0.28	p=0.28	p=0.00	p=0.00	p=0.00	p=0.00	p=0.00	p=0.00	p=0.00	p=0.00
2D	MCUΔT-MCU	p=0.89	p=0.89	p=0.00	p=0.00	p=0.00	p=0.00	p=0.00	p=0.00	p=0.00	p=0.00
3D	SFUΔMCU	p=0.36	p=0.36	p=0.00	p=0.00	p=0.00	p=0.00	p=0.00	p=0.00	p=0.00	p=0.00
3D	SFUΔT-MCU	p=0.78	p=0.78	p=0.14	p=0.14	p=0.00	p=0.00	p=0.00	p=0.00	p=0.00	p=0.00
3D	MCUΔT-MCU	p=0.51	p=0.51	p=0.00	p=0.00	p=0.00	p=0.00	p=0.00	p=0.00	p=0.00	p=0.00

4.3.3 T-MCU Performance

To test the performance of the server and client, we deployed a complete communication system and the conditions explained in Section 4.2.1. The capture upload client is replaced for performance measures with a dummy uploader. The dummy uploader loop uploads the pre-VPT-encoded video of sequence M-2. This upload consists of up to 16 individual user streams in the modified tile-able VP9 encoding format (that is, all conditions use identical streams). Our deployment uses a WebXR client [33] via Google Chrome (version 113.0.5672.64), with a render framerate of 100fps for all conditions. The performance measure was taken using the System Informer[13] (version 3.0.6043) system resources monitoring tool in one second intervals. Measurements were taken under 2 render conditions, 3 transmission conditions, and 5 user conditions, totaling 30 conditions. Each condition was measured in 30 minute sessions (with a sample interval of 1/second = 1800 samples per condition). Additionally, a 30-second wait time was added after each client startup condition (to exclude startup performance peaks).

2D: Displaying the users only as a 2D video image, thus representing a condition similar to regular 2D video conferencing, without the lower half).

3D: Rendering of remote users as 3D meshes in the virtual environment.

The transmission conditions used are as follows:

SFU: The SFU condition simply connects to X number of WebRTC download entities to receive X number of streams, where X is the number of users according to the condition.

MCU: Using the MCU [33] as a central composition unit.

T-MCU: The T-MCU introduced in this chapter is used as a central composition unit.

The 5 user conditions represent the use of a different number of user stream, while 00-Tiles serves as baseline, not receiving any streams, but only starting the client and rendering an empty environment (thus showing the generic overhead of the client).

Table 4.4 shows the results of the Web browser performance measurements. The result shows a steady increase in CPU and GPU usage with the increase in the number of users. Overall in all conditions (besides the 00-Tiles), the CPU/GPU load is reduced in the (T-)MCU condition vs. SFU. Table 4.4, also includes the results of the t-tests to check that the values are significantly different (assuming unequal sample sizes and similar variances). The t-tests indicate that there is no significant difference in the baseline condition (00-Tiles) as expected and indicate that all other values are significantly differnt ($p < alpha=0.05$), with one expection of SFU comapred to MCU in 2 tiles 3D. Furthermore, our measurements show that a central transmission strategy via (T-)MCU can significantly reduce the performance load on the client on both the CPU and GPU (10%∼20%) compared to an SFU. The improved performance of the (T)-MCU is based on the difference between one and multiple decoder instances (observable in the 2D conditions) and one vs. multiple 3D render textures and shaders (observable in the 3D conditions). The difference between SFU and (T-)MCU

[13] https://web.archive.org/web/20240416071123/https://systeminformer.sourceforge.io/nightly.php

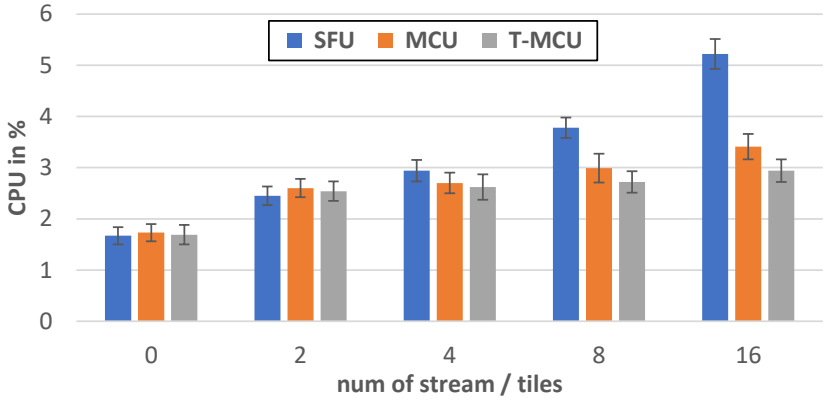


Figure 4.7: Chrome Client CPU performance (3D)

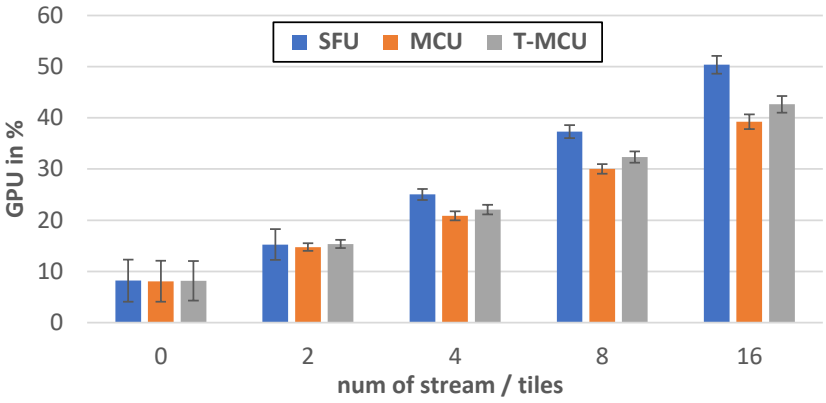


Figure 4.8: Chrome Client GPU performance (3D)

in client performance can be particularly observed under the 3D conditions (Figure 4.7 and Figure 4.8). We also observe slightly higher GPU usage in the client for the T-MCU condition versus the MCU condition (Figure 4.8), which can be explained by the slight frame loss of the MCU as observed in the latency tests (Table 4.2).

Finally, we would like to mention that we excluded any test beyond 16 tiles (that is, 32 tiles) as it resulted in client instabilities in the SFU condition and instabilities in the MCU (as shown in Table 4.2). However, for a 32-tile T-MCU stream, the client runs with a stable 100fps, utilizing 16.67% CPU (±1.94) and 68.18% GPU (±2.13).

Table 4.5 shows the measurement results of the (T-)MCU Python server. Currently both the T-MCU and MCU do not use any GPU, thus results are only reported as CPU utilization. The MCU shows significantly higher CPU usage as it utilizes one decoder per user stream, a demanding composition function, and a (high-resolution) stream encoding. That is, our tests show an improvement with the T-MCU of 85-90% reduction in server

Table 4.5: Server CPU Performance (in mean %; SD as ± in %)

Server	02-Tiles	04-Tiles	08-Tiles	16-Tiles	32-Tiles
MCU	2.25	4.62	9.64	21.81	*
	(±0.14)	(±0.31)	(±0.74)	(±1.57)	*
T-MCU	0.57	0.88	1.56	2.59	5.94
	(±0.02)	(±0.05)	(±0.12)	(±0.22)	(±0.44)

CPU usage compared to the MCU. Note that the MCU CPU load is not reported for 32-Tiles as the output was not stable (see Table 4.2).

4.4 Discussion

In the following we discuss the work presented in this chapter based on our research questions:

RQ3.1. Is tile-based composition possible in real-time with a central composition unit and VP9?

The VP9 standard in its current draft form does not sufficiently support tile composition at the bitstream level, when utilizing all common encoding features, which is clear from the significant changes that we had to make in the libvpx encoder. However, by the modifications we made, we were able to demonstrate that it is still possible to perform bitstream level composition within a standard-compliant bitstream. In our current solution, this comes at the cost of an increase in bandwidth of approximately 16%. Even though this increase is significant, our current VPT approach still offers multiple points for improvement, for example, to optimize the transform mode and base_q settings. In addition, other encodings (such as AV1, H.265, and H.266) could improve this increase in bandwidth but need wider support in end devices and browsers.

Another disadvantage of the presented solution is a fixed quality encoding; as for remote communication, the best practice is a fixed bitrate approach (to maintain a stable network connection). On a general note, for complex high-resolution video and immersive communication this best practice might also need to be challenged, as rendering quality demand and general availability of internet bandwidth increases. The more problematic limitation of our current approach is that the bandwidth and quality cannot be altered for individual streams (Note: the bandwidth will fluctuate but is fully defined by the base_q and the complexity, such as motion, of the user image). Effectively, each stream is prioritized externally in terms of high quality, while this can be a "fair" strategy for many communication situations. Natural human communication behavior dictates that attention is most often focused on individuals, e.g. the person who is speaking. One potential way to mitigate this is the addition of Scalable Video Coding (SVC) [176]. SVC is already widely supported in browsers [14] and thus could offer a powerful combination if possible combined with an T-MCU.

One clear advantage the bitstream-based composition allows (that is currently not explored in this chapter) is the possibility to activate and deactivate individual streams,

[14] https://web.archive.org/web/20240114080912/https://www.w3.org/TR/webrtc-svc/

without any decoder reset, stream renegotiation, or any other signaling. This can also mitigate the increased bandwith by allowing complex real-time stream management by only transmitting streams that are in sight of the user (in an immersive environment) or only the most important users in high resolution.

RQ3.2. What is the impact on resource usage for clients and the central server, when using a tile-enabled VP9 encoding and composition (T-MCU)?

The MCU and T-MCU single stream transmission show a similar performance impact to the (web) rendering client resource. This is, the T-MCU offers a significant performance advantage compared to a P2P/SFU. The main reason is that with a single stream, the client needs to utilize only one decoder, one texture mapping, and one rendering shader. Texture mapping is an expensive step in the rendering pipeline, particularly when considering high-resolution streams, as required for immersive communication.

One key advantage of the T-MCU is that it does not require any decoding and encoding steps in the server. Therefore, resource usage decreases significantly (85- 90%) compared to MCU. In addition to transcoding, the other bottleneck in the MCU is the composition itself, that is, building the output mosaic on the RGB image plane. Images are not simply concatenated but must be memory-copied, possibly combining an image per image row. The MCU (as presented in [33]) uses internally the GStreamer compositor element [15], which can create high CPU load and latency. But this also might make the MCU not suitable for many streams (beyond 16) or high-resolution streams. In our test, we can observe that the MCU is not capable of supporting 32 user streams, while the T-MCU still offers a stable stream output. The need for decoding, encoding and composition in the MCU also results in a significantly higher latency (10-20%) compared to the T-MCU.

In our tests, the T-MCU adds some delay (Table 4.2) to the raw encoding delay (Table 4.3), which is based on header parsing and frame composition scheduling. Currently, we follow a best-effort approach. This is to grab frames as they arrive and align them on the GOP structure. With a GOP structure of 5 the interval of an inter frame (I-frame) is 165 ms ($1sec/30frames = 33ms/frame * 5frames = 165ms$), thus the maximum theoretical de-sync of streams can be up to 165 ms (this is to say that ultimately the GOPs will align via I-frames). This de-sync is also the most likely explanation for the increase in delay with an increase in tiles. We expect that with further improvement on the parsing and I-frame scheduling (e.g., adding an I-frame on demand requests), the overall delay of the T-MCU can be further reduced. However, the T-MCU already offers a significant delay reduction compared to the MCU.

Finally, we assume that the VP9 encoding has the most limitations in terms of tiling (compared to H.265 and AV1). This is with our approach, we show that resource usage can already be significantly improved for remote real-time video and immersive communication systems (with a current drawback of increased bandwidth). We expect that with a wider deployment of AV1 and H.265 further improvements will be possible to a bitstream-based central composition approach.

RQ3.3. How many simultaneous 3D photorealistic users could this approach allow in an immersive communication session With the result presented in this chapter we show that both our system and (web) render client can easily handle 16 user streams (and theoretically up to 32 user streams). Furthermore, with a CPU load of ~3%

[15] https://web.archive.org/web/20230322143355/https://GStreamer.freedesktop.org/documentation/compositor/

and GPU load of ~43% in case of a 16-tile T-MCU (running on a modern powerful PC such as the Acer Predator Helios 300), it offers enough room for complex virtual environments, spatial capture, and other overhead to utilize complex XR rendering devices. This resource footprint might, however, not hold true for mobile end devices like VR glasses with limited CPU, GPU, and battery resources.

The use case that motivated the work presented in this chapter, to facilitate RGBD streams in immersive environments for 3D user representation, can currently be seen as high demand, upper limit requirements. In a real deployment, it is not clear whether 16 user streams with ~2Mbit, resulting in ~32Mbit down traffic, are a realistic real-world scenario for the forseable time. However, the T-MCU (and its underling components like the Bitstream Tiled Video Compositor and VP9 parser) can be seen as basic building blocks, allowing many new complex central composition units. Firstly, a deployment with less and less quality streams can already be beneficial for many video conferring scenarios, and thus lowering performance needs for traditional conferencing. Second, the current T-MCU could be combined with other streaming approaches (like SVC as mentioned, or) and could only forward a subset of the user streams in a communication session. Important to note is that any switch in the bitstream composition in the T-MCU will not require any renegotiation or decoder reset, which can be a powerful advantage for many use cases.

4.5 Conclusion

In this chapter, we introduce a novel Tiled-enabled Multipoint Control Unit (T-MCU) that allows real-time stream composition on the bitstream without the need to decode or encode the stream in the central composition step. The component is built on a modified version of the VP9 encoder (libvpx). Our modifications to the encoder are fully standard-compliant and thus can be decoded by any VP9 decoder, but result in a ~16% bandwidth overhead compared to the standard libvpx VP9 encoder.

Our new T-MCU offers similar resource advantages in the receiving rendering client as a traditional MCU [33] with a performance gain of approximately 20% compared to a P2P/SFU transmission approach. Most importantly, the T-MCU improves the transmission delay by ~20% and the performance gain on the server by 85-90% compared to an MCU [33]. The T-MCU can offer a solution for the scalability of simultaneous users in demanding immersive communication applications and can support up to 32 simultaneous user streams.

Finally, the concept of the T-MCU can also be applied to other use cases such as traditional video conferencing or multiview streaming, as recently introduced by YouTube[16], significantly reducing server processing overhead.

[16] https://web.archive.org/web/20240218015334/https://blog.youtube/news-and-events/multiview-on-youtube-tv/

5

Applications

In this chapter, we show the applicability of the Social XR research presented in this dissertation in three use cases. The use cases cover communication scenarios in VR and AR as well as remote education in AR. We choose the use cases based on an initial use case study and develop them in collaboration with both industry stakeholders and end users. By presenting the prototypes, we show that the individual concepts of the dissertation can easily be adopted to real-world applications and are technically ready to allow immersive remote communication with 3D photorealistic user representations.

This chapter is based on:

- Simon NB Gunkel, Hans M Stokking, Martin J Prins, Nanda van der Stap, Frank B ter Haar, and Omar A Niamut. Virtual reality conferencing: Multi-user immersive vr experiences on the web. In *Proceedings of the 9th ACM Multimedia Systems Conference*, pages 498–501, 2018 [59]
- Sylvie Dijkstra-Soudarissanane, Simon NB Gunkel, and Véronne Reinders. Virtual visits: life-size immersive communication. In *Proceedings of the 13th ACM Multimedia Systems Conference*, pages 310–314, 2022 [60]
- Simon NB Gunkel, Sylvie Dijkstra-Soudarissanane, and Omar Niamut. "you ar' right in front of me": Rgbd-based capture and rendering for remote training. In *Proceedings of the 14th Conference on ACM Multimedia Systems*, pages 307–311, 2023 [61]

To better understand the general applicability of ICS and Social XR, we conducted an initial use case study involving 91 participants at a virtual reality industry event [177]. This study identified "Conferencing" and "Education" as the two most relevant use cases for immersive communication in VR. Based on these two main areas, we developed multiple use cases in collaboration with key stakeholders in the industry and end users. In this chapter, we present three use cases and how we integrated our system (see chapter 3) into prototype applications to address these use cases. Through these examples, we show how our research findings can address immersive communication in different real-world applications in both VR and AR.

Table 5.1: Prototype application components overview

#	Capture Device	Capture Processing	Transmission	Rendering	Rendering Device	XR Modality
5.1	Kinect v2	Minimal	WebRTC peer-to-peer	WebGL: 2D sprite, 3D point cloud	Oculus Rift CV1	360 VR and 3D VR
5.2	Zed 2i	Edge cleaning and hole filling	WebRTC peer-to-peer	Unity: 3D point cloud	TV	Static AR
5.3	Azure Kinect	Fully modular (as in section 3.2.3)	WebRTC peer-to-peer	WebGL: 2D sprite, 3D point cloud, 3D mesh	Hololens 2	AR

Table 5.1 shows how the different example prototypes relate to different concepts presented in the dissertation. While all prototypes are based on the transmission of users representations in our own RGBD format over 2D WebRTC video peer-to-peer connections, they utilize different capture sensors, different render devices, different user rendering modalities, and are deployed into different XR modalities. Finally, the applications presented in this chapter are technical examples on how the research can be applied to specific XR use cases. The three use cases are explained in the following:

Communication in 360-degree and 3D VR (5.1): This use case was developed as part of the VRTogether EU project[1]. The focus of VRTogether was on improving user experiences by innovating on different media formats (i.e., audio, video capture, graphics, transmission, and 3D user rendering) to increase the feeling of being together. Our prototype presents a simple communication use case that allows users to communicate in virtual reality environments of either 360-degree (with billboard cut-out representations of user) or fully 3D volumetric.

Life-size communication in AR (5.2): This use case was developed in collaboration between TNO, MeanderGroup, connec2, vodafone, Ziggo Holding B.V., Ericsson and Mastrich University (covering a wide range of industry and end-user perspective to create the

[1] https://web.archive.org/web/20240503122720/https://vrtogether.eu/)

use case). The focus of this collaboration was on developing a high-quality XR communication system for residents (elderly) in care homes to mitigate social isolation (which was particularly increased by the COVID-19 pandemic). Our prototype takes into account the needs of both end users (the elderly in the nursing home and the family at home). The end result allows the resident (elderly) in the care home to experience 'immersiveness', where there is the feeling that the family visitor is present in their environment.

Remote Education in AR (5.3): This use case was developed in collaboration between TNO, Fraunhofer EMFT, Fraunhofer IGCV and the Fraunhofer Center for Secure Intelligent Systems to address teaching and expertise at distance. The interest in online training and expertise over distance has increased significantly due to distance restrictions and accessibility of remote locations. Our prototype provides a practical approach to transfer the advantages of classroom training to online training via a Mobile Learning Hub (MLH) and AR. Combining the mobile workstation (MLH) with practice-oriented AR applications and immersive communication (including holographic user representations) is intended to fully simulate high-quality classroom training remotely.

5.1 Communication in 360-degree and 3D VR

We developed a VR framework that extends current video conferencing capabilities with new VR functionalities [46, 83, 178]. Our framework is modular and based on web technologies, and allows to both easily create VR experiences that are social and to consume them with off-the-shelf hardware. With our framework, we aim to allow users to interact or collaborate while being immersed in interactive VR content. In this demo, we focus on a communication use case, where multiple people (up to three) can sit around a table to communicate within VR. We evaluated this experience with 54 people, in unstructured testing sessions with a duration of 3-10 minutes. Each testing session was followed by a questionnaire and informal discussions. These tests were done in a static 360-degree VR environment. Furthermore, in this demo we like to compare such static 360-degree environments with new media formats and full 3D environments. We present an approach to enhance our system with color+depth video data, allowing to show users as 3D point cloud in a 3D VR environment and evaluate this approach on a technical level. Although we believe that such representation will add value, a QoE comparison with 2D is not yet performed and is out of the scope of the work presented in this section.

Figure 5.1: Three-person experience: Virtual Reality design (left), VR User View (middle) and Setup diagram (right).

Figure 5.2: Different Video Capture modes A (top left) full color, B (bottom left) green-screen cut-out and C (right) color cut-out (top) with depth (bottom).

5.1.1 Multi-user VR Communication

In this section we present a three-people experience based on our TogetherVR framework [46, 83, 178]. TogetherVR is a completely web-based framework to build and consume shared and social VR experiences. Our main motivation to utilize web technology is to allow an easy and widespread deployment and low entry burden for end users and developers. TogetherVR is utilizing many established web frameworks (like WebVR, AFrame, three.js, WebRTC, WebAudioAPI, WebGL, soket.io, Angular, DASH) to create virtual spaces and to enable users to communicate in such spaces. To allow communication via audio and video via our system, we alpha-blend people into the environment based on WebGL shaders. In our system users are recorded with a Kinect 2 RGB-plus-depth camera, and the background of the users is replaced with a uniform green color before transmission (see Figure 5.2, bottom left). After receiving the video, the background is removed via alpha-blending in the receiving browser, thus leaving a transparent image showing just the user without his/her physical background. For capture and transmission we use a resolution of 960x540 pixels with 25fps.

In this three-people experience, three people sit around a round table in VR. The view of each user is the same; that is, each user sees the other two users on the opposite side of the table and a video playing on the top of the table (see Figure 5.1). In our setup (Figure 5.1, right), we have three specific and similar user places, each with a laptop (MSI GT62VR), Oculus Rift HMD (CV1), Kinect camera, headset (Pioneer SE-M631TV) and unidirectional microphone (Power Dynamics PDT3). Users are recorded from the front so that they are able to see and interact with two people at the same time. We held a 1-day experiment trial in an informal and uncontrolled setting at our lab facilities. In this experiment, we collected feedback through a short questionnaire from 54 participants (avg. age of 32,5 and 42% Female), who communicated and watched a video with our system.

We evaluated the quality of the (visual) experience and the sense of (co-)presence, asking participants to rank their experience on a 5-point Likert-type scale. We performed a Cronbach's Alpha test to check the consistency of our questionnaire items (i.e. whether

Table 5.2: Performance of different video encoding, decoding and rendering.

Performance matrix	Encoding	Decoding	Bandwidth	10 sec P8 file
A: RGB Full Color (540x960)	34%CPU 15%GPU	16%CPU 30%GPU	2.7Mbps	356KB
B: Greenscreen cut-out (540x960)	40%CPU 16%GPU	17%CPU 30%GPU	2.7Mbps	449KB
C: Cut-out + depth (1080x960)	38%CPU 24%GPU	24%CPU 30%GPU	5.2Mbps	466KB

the scale used for our questionnaire would yield reliable measurements). The test reveals an overall good consistency, all questions show a high alpha value (avg. of >0.7). People appreciated the overall quality of the experience (98% of the participants scored 4 or higher, avg. 4.3, SD 0.51). They also felt involved in the virtual environment (experience, avg. 4.1, SD 0.76). Further, they reported a sense of being in the VR scene (presence, avg. 4.2, SD 0.80) and the sense of being with the other users (co-presence, avg. 4.1, SD 0.86).

Within the experience we observed more interaction between participants compared to our previous co-watching experience [46, 178], with 2 users sitting beside each other, watching a movie. This is not only because more people were involved; while seeing people from the front as well as seeing the video playback and the other people at the same time, people expressed a more natural conversation setting. Further, no participant reported on any experience of motion sickness. However, participants reported not seeing themselves (self-view) as well as not being able to stand up or get close to the camera as diminishing factor for the experience.

5.1.2 3D User Representation

In this section, we present our ongoing effort to include new media formats. This is, using RGB+depth (RGBD) information, to display users as a 3D point cloud. To do so, we mainly changed 2 components from the three-person experience (see Section 3): (A) the Kinect v2 camera capture to record both color and depth, and (B) the WebGL shader to display the user video. Regarding the camera capture (A), we combined the color image with the depth image (see Figure 5.2, right). However, as the browser and current WebRTC implementation does not support depth encoding we need to use an intermediate step. We map the 16 bit depth value from the Kinect into the RGB color space using the green and red color only. We use green and red because these colors are prioritized in an YUV color mapping. An example of such a mapping can be seen in Figure 5.2 (bottom right). To display this image as a point cloud (B) we changed the WebGL shader, in order to not display a flat image, but each pixel with coordinates in the 3D space, based on the depth image. Our shader is based on the work by George MacKerron[2].

5.1.3 Performance

We run performance tests, to evaluate our depth-based point cloud approach on a technical level. This is to compare the resource usage under different video transmission to better understand the strict performance requirements and limitations of our web-based solution.

[2]https://web.archive.org/web/20230530022141/http://blog.mackerron.com/2012/02/03/depthcam-webkinect/)

In this analysis, we compared 3 conditions (see Figure 5.2 and Table 5.2), transmitting the full color image (top left), the image with a replaced green background (bottom left) and the image with color + depth information (right). We ran 2 computers (measuring CPU, GPU and bandwidth usage), one with a Kinect sensor to capture and send the video over WebRTC and one to receive and display the video. Additionally, we encoded 10 sec clips in all three conditions with FFmpeg (VP8) to show the differences in encoding sizes of the three approaches. Table 5.2 shows the measurements of our tests. The differences in CPU/GPU usage are minimal in all 3 cases, showing only little overhead for the color+depth video. Further, the bandwidth used is the same for all conditions (in relation to the resolution), which is based on the automatic WebRTC settings. Looking at the encoding size of the 3 clips, the differences are more complicated. Particularly, that condition B (cut-out) is larger than condition A (full color) seems counter intuitive, as a large part of B consist of a uniform color. In this regard, it is good to mention that most encoders (i.e. H.264 and VP8) are not optimized for such video files, but for color scenes. Thus, this result is not surprising.

5.2 Life-size Communication in AR

Virtual meetings became essential for our new way of daily communication with relatives. This is especially true for family members who are categorized as "at risk", such as the elderly in care homes. Digital communications such as 2D video conferencing systems (e.g., Microsoft Teams, Zoom, Messenger, or Whatsapp) provide an omnipresence capability, where people have the freedom to call whenever and wherever they feel. In most 2D systems, although easy and practical to use, they do not provide a satisfying feeling as a "real life" meeting. 2D video conferencing systems simply do not provide a natural experience in which essential social and spatial cues play a crucial role [179, 180].

Furthermore, elderly people in care homes often lack knowledge and motivation to use current communication technology [181, 182]. A caregiver is often required to explain how the system works on their phone [183, 184]. Older patients often suffer from an eye sight problem that prevents them from properly seeing their relatives on small mobile screens.

We propose an easy-to-deploy end-to-end communication system that facilitates high-quality natural communication through 3D capture and rendering, catering for a real-time immersive experience on life-size displays. Our system is easy to use, and a potential user only needs limited assistance from a caregiver. Just sitting in front of a massive screen is enough to allow life-size communication with physically distant family and friends [28].

We have previously shown our early research findings in the use case called 'Elderly in Care Homes' in the domain of Social XR [116]. The system we developed is an AR-based communication tool that enables the elderly to communicate with their relatives while feeling like being together. The system architecture proposed in this section is defined based on an extensive user study that is reflected in the definition of each module of the transmission pipeline.

In this section, we present enhancements to this first set-up and extend our AR communication framework to enable remote communication taking place in the environment of elderly nursing homes, with real-time environment capture combined with real-time 3D photorealistic representation of family members in a full end-to-end demonstrator.

5.2.1 Life-size AR

On the client side, the screen setup influences the realness of the other person and the sense of being together. In our initial tests, we observed a preference for vertically oriented screens for one-on-one video calls. This requirement is also a well-known concept in photography, where a vertical photo frame is chosen to capture a person's personality and emotions [185–188].

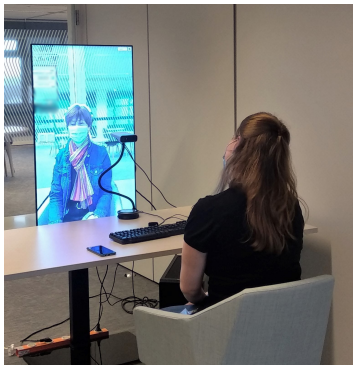


Figure 5.3: Static AR system without perspective projection. A 3D sensor captures the user in photorealistic representation.

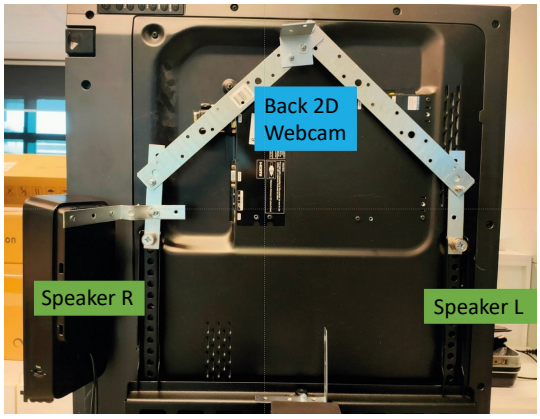


Figure 5.4: Structure at the back of the life-size screen. Two speakers are attached at the sides to enhance the audio quality. A 2D webcam is attached to capture the background of the person to reflect the AR environment.

The size of the screen also influences the sense of being together [189]. Previous research [190] showed a higher sense of being together for a human-sized screen versus a small screen. This study confirms that the screen size of 24 and 28.2 inches is perceived as too small and is, indeed, not big enough to project a subject in a life-size manner. 55 inch screens are perceived as too big, as this screen size takes up too much of the user's visual field to see both the real world and the virtual world seamlessly. Therefore, the preferred screen size is defined to be at around 40-46 inches. Our final choice for life-size AR consists of a 46 inch vertically oriented screen placed on a chair on the other side of the table, opposite the observer, as shown in Fig. 5.3.

5.2.2 System architecture

Our system design is loosely based on our previous findings presented in [116]. We refactored and enhanced the different modular building blocks to allow remote communication for elderly people in care homes with remote family members. Our new solution presented in this section introduces a new completely independent capture module based on the ZED 2i camera and introduces a new rendering client that supports life-size displays while allowing AR-type environment rendering. The result is a rendered image that blends the remote user into your real space, as shown in Fig. 5.5.

Fig. 5.6, shows a simplified view of the different architectural building blocks of our system. The texture (RGB) image and depth (D) information are captured using a ZED 2i sensor through a Python capture module (see Section 5.2.3). This capture results in a

Figure 5.5: Life-size Immersive AR video call.

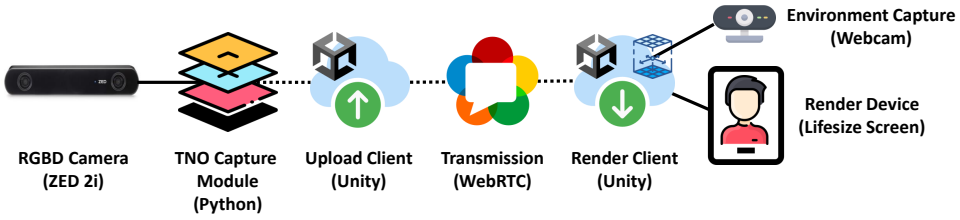


Figure 5.6: System building blocks.

processed RGB+Depth (RGBD) image in RGB color format accessible via a virtual webcam (note: we utilize the same method as presented in (chapter 3) [33] for the conversion of the 16-bit depth into an 8-bit RGB grayscale format). An Unity-based upload client can simply grab the RGBD frames from the virtual webcam device and send them through a WebRTC [191] transmission pipeline to the remote rendering client. The "upload client" and "render client" are part of the same Unity Engine program and could, in principle, also be two complete separate software instances.

The combined upload/render client receives two webcam input images, one for the user representation (in front of the client as shown in Fig. 5.3) and one for the environment behind the life-size render display as shown in Fig. 5.4. In our Unity rendering pipeline, we place the environment on a simple 2D texture and render the remote user in front of this texture as a 3D representation. For this 3D rendering, a VFX graph converts each pixel into its respective 3D coordinates to effectively render a point cloud. In the future, this approach can easily be adopted for more rendering modalities such as AR glass-type displays (e.g. Hololens[3] or NReal[4]) and holographic displays (such as Looking Glass[5])

[3]https://web.archive.org/web/20240213114005/https://www.microsoft.com/en-us/hololens/)
[4]https://web.archive.org/web/20240325081732/https://www.nreal.ai/
[5]https://web.archive.org/web/20240503105026/https://lookingglassfactory.com/)

Figure 5.7: Capture In/Out and processing steps.

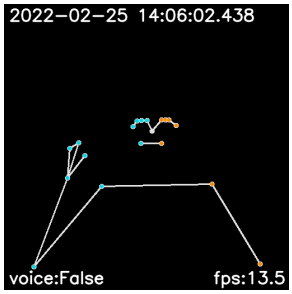


Figure 5.8: Body tracking recording.

5.2.3 Capture Module & Processing Steps

One promising solution to capture a person and create a 3D human reconstruction is through a color + depth (RGBD) sensing capture device. While for such a 3D reconstruction the raw depth accuracy is most crucial, for a "good" reconstruction, the capture devices face multiple limitations. RGBD sensors have already been considered in the past for human 3D capture, such as Azure Kinect, RealSense, and Kinect v2 (e.g., [33, 192, 193]). In this section we explore a different sensor: the ZED 2i[6]. The ZED 2i camera is an RGBD capture device based on stereo matching (via two color cameras) and thus particularly faces difficulties with the depth reconstruction of uniform surfaces. Fig. 5.7 (top left) shows the raw capture of a user with the ZED 2i, with many spatial gaps and pixel matching inconsistencies. For our use case, it is most important to achieve a clean, sharp, and complete picture of the user that can be rendered into the AR scene. In the following section, we will outline the details of our ZED-based capture and the different processing steps involved to enhance the overall quality of the picture. Our capture module is fully implemented in Python and thus easy to extend, modify, and deploy. It uses different Python native bindings to ensure a high performance and low resource footprint:

- OpenCV[7], Numpy[8] and Numba[9] for image processing.
- ZED python SDK[10] to access the ZED images
- Mediapipe[11] for machine learning (ML), body segmentation and cutting out the user from his/her background
- pyvirtualcam[12] to expose the final image via a virtual webcam (OBS[13]).

[6] https://web.archive.org/web/20240314220950/https://www.stereolabs.com/zed-2i/
[7] https://web.archive.org/web/20240214001251/https://opencv.org/
[8] https://web.archive.org/web/20240503065753/https://numpy.org/
[9] https://web.archive.org/web/20240502055447/https://numba.pydata.org/
[10] https://web.archive.org/web/20240415025624/https://www.stereolabs.com/docs/app-development/python/install/
[11] https://web.archive.org/web/20240406130828/https://google.github.io/mediapipe/
[12] https://web.archive.org/web/20240314043542/https://github.com/letmaik/pyvirtualcam
[13] https://web.archive.org/web/20240215044704/https://obsproject.com/

To capture the color and depth images from the ZED 2i camera, we utilize the following settings from the ZED SDK:

- **camera_resolution = HD720**: 720p resolution is enough as we directly crop the input image to a resolution of 512x512, this is on the one hand to increase the processing and encoding performance and on the other to get the best (centered) crop of the user (based on the distance of the user to the camera).
- **camera_fps = 15**: we identified 15 as the ideal framerate, however, utilizing Mediapipe ML processes can easily easily achieve frame-rates of >20fps
- **depth_mode = DEPTH_MODE.QUALITY**: we identified this as the best solution. The alternative
  DEPTH_MODE.ULTRA contrary to expectation, looks very pixelated and less uniform on the depth granularity
- **sensing_mode = SENSING_MODE.STANDARD**: Standard sensing results in a more consistent depth image compared to the alternative SENSING_MODE.FILL. The main issue with FILL is that it creates a lot of overlap and objects and body parts merge with the background creating inconsistent depth sensing results.

However, based on these capture parameters, the image is still not of sufficient quality. To improve the image quality (that is, on the depth plane), we follow these processing steps (the outcome of each step is shown in Fig. 5.7):

- **Step 1.**: First, we want to cut out the user from its background (in order to mitigate any blending of the user and his/ her background). To do so, we create a segmentation mask with the Mediapipe machine learning framework (see selfie segmentation[14]).
- **Step 2.**: With the help of the segmentation mask, we clean the raw depth data.
  A. Remove any values outside of the segmentation mask.
  B. Remove small holes
     (cv2.MORPH_OPEN followed by cv2.MORPH_CLOSE with kernel size 4).
- **Step 3.**: Based on any missing values, we then build a hole mask (missing depth values & segmentation mask).
- **Step 4.**: To fill any missing values, we create a dilated version of the depth image from step 2 (cv2.dilate with kernel size 4 and 8 iterations).
- **Final Step. (output)**: Finally, we replace the missing (hole mask, step 3) with the values from step 4. and send the final image to the virtual webcam (pyvirtualcam).

The resulting output image is shown in Fig. 5.7. Note that the conversion of depth to grayscale image follows the same method as described in (chapter 3) [33]. Even though the final image is not perfect on the depth plane (e.g. the parts between your fingers

[14] https://web.archive.org/web/20240406130828/https://google.github.io/mediapipe/solutions/selfie_segmentation.html

become part of the hand even though they are part of the background), the image delivers a consistent and fully filled representation of the user. This allows us to render the user and ultimately blend the user into his/her AR environment.

Furthermore, the capture module incorporates two features to better support any qualitative user study with our system at a later state. (I) On the basis of the Mediapipe image analysis, we receive and store body tracking information (see the example recording image in Fig. 5.8). The body tracking information promises to identify movement patterns and possible engagement of users in a non-intrusive way (no audio or video is recorded of users). (II) We added voice activity detection (based on the WebRTC Voice Activity Detector[15] to allow (together with body pose data) further reasoning about the communication behavior of users (without the need to record any audio).

The performance of the capture module was measured at 221.67 ms (SD 17.72) capture-to-display delay, a CPU usage of 11.32% (SD 2.41) and a GPU usage of 6.98% (SD 2.73). This includes all processing and handling of body tracking and voice activity. Measurements were made on a XMG NEO 15 e21 laptop[16] (XNE15AE21, AMD Ryzen 9 5900HX, NVIDIA GeForce RTX 3080, 32 GB RAM).

5.3 Remote Expertise & Training (glass-type AR)

It is compelling to enable expertise or skills to be provided at a distance through existing 2D communication tools such as Teams, Whatsapp calls, Zoom. In industry use cases, where training, co-working, inspection, and maintenance are required, facilitating remote education and training, and supporting inclusion of citizens with accessibility barriers became a necessity to address the current labor market gap for skilled persons. However, the learning outcome and the willingness to learn on current 2D platforms decreases due to the lack of interaction between the trainer and training participants.

One approach that has gained recent popularity in industry is to use simulations in Extended Reality (XR) applications to facilitate (pre-recorded) specialized training's. While [194] identified some clear benefits of this approach, they also identified a clear need for more interactions. Additionally, while novel 3D immersive platforms may enable new forms of remote education and training, not all training procedures can be simulated with the required high level of accuracy and sensory immersion. For example, high-end ESA-certified soldering training requires students to feel the heat and assess soldering fumes during training, whereas the teacher must assess the quality of the soldering result.

In particular, for providing expertise at a distance, holographic telepresence may significantly enhance the sense of immersion and (co-)presence, and we estimate that immersive learning platforms can contribute to the effectiveness of the task at hand; for remote training and skill transfer, in particular, leveraging XR technologies may enhance the effectiveness and shorten the duration of training and education.

The prototype presented in this section is guided by an AR use case studied together with Fraunhofer EMFT. We have investigated the specific use case of teaching soldering to remote students on a mobile soldering station. Over the past 2 years of collaboration, we have endeavored to design and develop a system that makes use of off-the-shelf equipment that

[15] https://web.archive.org/web/20240329141349/https://github.com/wiseman/py-webrtcvad/
[16] https://web.archive.org/web/20231129124612/https://www.xmg.gg/en/xmg-neo-15-amd-e21/

Figure 5.9: Soldering station with student (left) interacting with a remote teacher (right) via a AR glasses (center).

is easily deployable and for a reasonable price by overlaying a real-time 3D representation of a trainer into the field of view of the training participants. The interaction between the trainer and the trainee is significantly improved by conveying a sense of co-presence. Figure 5.9 (middle) shows an example view through the AR glasses of a training participant. The participant is sitting in front of a mobile soldering station (Figure 5.9, left) and is able to interact with the remote trainer (Figure 5.9, right). Additional information was placed in the participant's field of view, such as remote video sequences, to enhance the theoretical training.

5.3.1 Modular RGBD capture and rendering

One of the main differences between an immersive communication system and a 2D video conferencing solution is the aspect of space. In 2D solutions, the concept of shared space is relatively absent, and users usually communicate from their own environment, displayed as flat representations on screens. In the case of AR, the spatial relationship (including the capture) is important as the user should be rendered in 3D into the real environment of another user. Such capture and reconstruction can be done with the help of RGBD sensors [39, 158] and depth image-based rendering [75]. However, most RGBD sensors are still prone to acquisition errors, resulting in distractions and artifacts in the 3D reconstruction. Selecting the right capture configuration, as well as the correct image enhancement optimization technique, remains a challenging process.

Our proposed prototype integrates a modular application for capture and processing that was entirely written in Python and can easily be configured in real-time for different capture, render, and processing options. Python was chosen because it allows simple code creation and has a wide range of existing image processing and machine learning components. Additionally, the majority of RGBD sensor SDKs include Python wrappers that operate similarly to native C solutions (such as the Kinect Azure and ZED 2i used in this section). Moreover, many image processing tasks using OpenCV and Numba are implemented using hardware acceleration and underlying C bindings, reducing the performance overhead. Although a complete native solution would be preferable in a production setting, our approach provides a great starting point for prototyping, testing, and research. This flexible Python module also allows for easy extension with new sensors or different image enhancement and image processing needs.

Five generic modules make up the capture application (see Figure 5.10). First, the camera frame grabber module captures RGBD image frames in a resolution and framerate defined by the settings. The raw RGBD image is further processed by the foreground

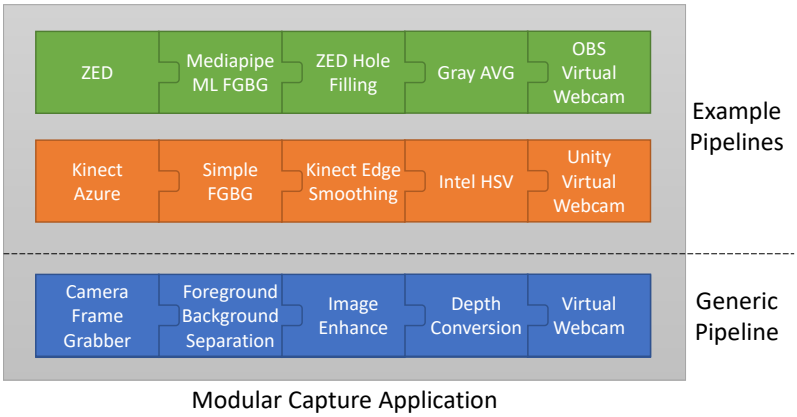


Figure 5.10: Capture Application and Processing Modules

background (FGBG) separation module (in order to delete any unnecessary information, thus offering some initial cleaning of the data). The resulting image is processed by the image enhancement module, which improves the image quality based on various edge and hole-filling intelligent imaging tasks. In order to allow the image to be transmitted in the communication system with existing (hardware-accelerated) 2D video encoders, the depth information is converted into a 2D RGB format and concatenated with the color image (depth conversion module). Finally, the complete image is sent to the virtual webcam driver to be used in the final client application. Each module is executed in a separate thread loop for optimal speed and simultaneous processing. Simple queues are used to convey information between modules, ensuring thread safety and image ordering. The primary thread for interacting with OpenCV image rendering serves as the glue that holds all modules together (displaying in a window any necessary final or debug output). A Metadata Server may provide more metadata data from the capture module to programs on the same computer system in addition to the five modules.

Furthermore, to add to the modularity aspect of the capture application, we added convenient deployment features and a large set of configurable debug options. The full set of features in overview:

- Easy Deployment support with Auto updater
- Support for different Capture Sensors:
  - ZED 2i [17]
  - Kinect Azure [18]
- Mediapipe Machine Learning Body Segmentation to perform foreground background extraction (cut-out of the user)

[17] https://web.archive.org/web/20240415025624/https://www.stereolabs.com/docs/app-development/python/install/

[18] https://web.archive.org/web/20230618070726/https://github.com/etiennedub/pyk4a/

- Different Image Quality Improvements (cleaning and hole filling) based on the raw RGBD image of the ZED and Kinect
- Different depth to color conversion for transmission
  - GrayAVG (chapter 3) [33]
  - Intel HSV[19]
- Different Output Options
  - Render on Screen (with and without frame-rate counter)
  - Virtual Webcam support (OBS and Unity Driver) accessible from any application
  - Write final image to files
- Extensive Debug Options for Testing
  - Render after each module
  - Skip any module
  - Capture playback via pre-recorded files

More information on the details of the individual capture modules can be found in (chapter 2) [57].

5.3.2 Prototype

For a complete immersive (AR) communication prototype, we integrated the modular capture application into a web-based immersive communication platform called VRComm (chapter 3) [33]. VRComm is a web-enabled (WebXR) XR system that allows VR and AR rendering on OpenXR devices. A few adjustments had to be made to VRComm to fully allow a HoloLens 2 device to use the platform:

- AR environment according to remote teaching use case;
- performance fixes for HoloLens 2;
- Functionality to let the HoloLens 2 join with and without streaming the rendered view of what is shown within the glasses itself;
- allowing to disable individual upload streams (the webcam image of the student should not be send to the students HoloLens 2).

Figure 5.11 shows the setup of the demo prototype. The final setup for the student consists of a PC running a Social XR web client to ingest the webcam feed from the soldering station (we utilize the MAC mini PC embedded in the soldering station without any modifications) and a HoloLens 2 also running a Social XR web client to ingest the view from the HoloLens 2 into the system and to render the teacher's audio and 3D user representation.

[19] https://web.archive.org/web/20230924160801/https://dev.intelrealsense.com/docs/depth-image-compression-by-colorization-for-intel-realsense-depth-cameras

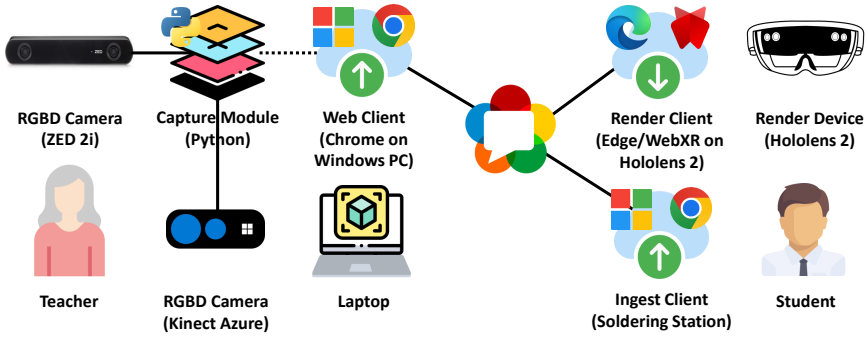


Figure 5.11: System architechture of the AR training pipeline, connecting a remote teacher to a student.

The final user setup for the teacher consists of a single RGBD sensor, and a capture ingest PC running a Social XR web client. For the demo, we deploy both a Kinect Azure and a ZED 2i as RGBD sensor solutions. Although only one of the sensors is active at a time, we allow a real-time switch between cameras to offer a direct visual comparison between both sensors. The teacher PC is the main interface for the teacher to interact with the student and show both the view from within the HoloLens 2 and the student webcam (or other video feeds selected by the student, such as the soldering station microscope). Finally, we added an example augmentation interaction in the form of text messages. The teacher can input text messages (additionally to the already present audio link) that will be rendered into the augmented view of the student. This system capability allows us to experiment and determine optimal font size, color, and text position for the AR headset.

The combination of modular capture and communication systems offers multiple settings in real-time. These settings enable us to test different capture parameters, image optimization techniques, and stream qualities to determine the right balance of hardware and software configurations for the presented use case (and future similar use cases). The following capture, rendering, and processing settings can be modified in real-time:

- depth thresholds;
- depth to color code - HSV vs. GrayAVG;
- output virtual webcam;
- render options:
  - point size;
  - render point size by distance;
  - perform shader based edge cleaning;
  - render geometry option (point cloud or mesh).

The different rendering options are shown in Figure 5.12. These options offer a real-time comparison of different rendering and image enhancement options while at the same time also providing the ease of measuring the performance footprint of those options.

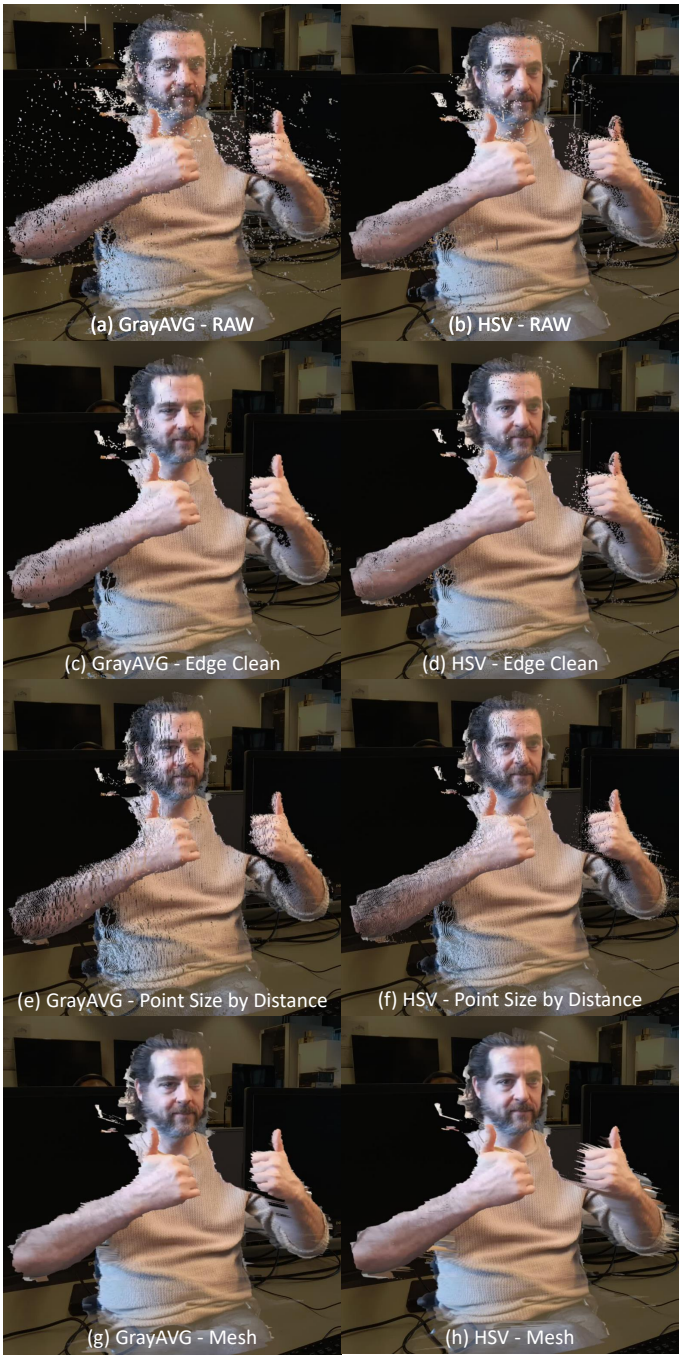


Figure 5.12: Different render options displayed in HoloLens 2

5.4 Discussion and Conclusion

In this chapter, we presented three prototypes showing the technical feasibility of the dissertation research within "real-world" use cases. The use cases were developed together with stakeholders from industry and end users and cover multiple XR modalities in AR and VR. In addition, the prototypes cover a range of technical concepts presented in the dissertation, including different capture sensors, different levels of processing on the RGBD image, and different render modalities in Unity and WebGL. Ultimately, all three prototypes allow different communication interactions with 3D photorealistic user representations. One limitation of these examples is the scalability of the users that facilitates a minimal number of simultaneous users via peer-to-peer network connections. The main reason for this limitation is to reduce the complexity of the use case to have clear real-life applicability. This is due to the novelty of Social XR communication and XR devices. Thus simplified use cases allow for easier testing of the applicability and benefits of these concepts towards end users. Finally, the presented prototypes show technical compatibility and technical readiness of the use cases, but the actual user tests are out of scope of this work. Therefore, in the future, complex detailed user and QoE testing is required to better understand the value and benefits of Social XR for end users in everyday life.

6

Conclusion

This chapter concludes the dissertation with final observations and an outlook on future research directions. In our work, we extended 2D video conferencing towards photorealistic immersive 3D communication along three main research questions. First, the work presents a modular RGBD capture component that allows for detailed technical tests on different capture, processing, and transmission formats. Second, we present a web-enabled Social XR system that allows full end-to-end evaluations. Third, we present a novel central transmission optimization that allows for the scaling of simultaneous holographic participants to allow numbers similar to current video conferencing solutions. In addition to addressing the research questions, the dissertation presents multiple example applications to show how immersive 3D communication system concepts can be applied in real-world use cases. In general, the result of this work is a generic holographic communication pipeline that can be used as a reference for technical comparisons and future QoE research.

In the current state-of-the-art, a comprehensive and generic view of the individual building blocks for creating Social XR systems and evaluating and comparing Social XR systems is missing. The research described in this dissertation addresses this gap, by developing a comprehensive understanding of the entire process of Social XR, encompassing all stages of the holographic 3D communication pipeline and its technical (performance) limitations. In this chapter, we examine how the dissertation addressed the research questions and present the key findings and contributions of our work. Finally, we outline ongoing and future work related to the results of this dissertation.

6.1 Revisiting the Research Questions

In this section, we discuss how our research addressed the different research questions that motivated our work. We reassess the main research question and RQ1-3, indicating the main advantages, disadvantages, and conclusions of our results.

RQ 1. (3D User Capture & Representation): What are the performance implications of different RGBD capture modalities for 3D user representations?

To address this research question, we created a modular capture pipeline and implemented representative capture, processing, and rendering components within example applications. Our evaluation of the example applications and pipeline components shows the performance implications at the individual component level. In general, for the complete modular capture pipeline, we measured a maximum CPU and GPU utilization of 16% for high-resolution streams and 5-15% for low-resolution streams. Furthermore, our experiments show a capture and processing delay of 300-400 ms. Although this may be significantly higher than for existing video conferencing capture solutions, it should result in an acceptable delay for most use cases of below 500 ms (previous research [52] indicates that delays beyond 500 ms significantly impact remote communication). The overhead in the rendering strongly depends on the device and browser capabilities (for example, hardware decoding). Our experiments show, however, that modern PCs and mobile devices, like the HoloLens 2, can support rendering to a limited number of users.

RQ 2. (Network and System Performance): What is the performance and bandwidth overhead for immersive communication based on RGBD?

To address this research question, we developed a web-enabled video-based Social XR framework that allows us to rapidly develop, test, and evaluate photorealistic XR communication experiences. By combining video conferencing technology with Social XR capabilities, we offer a new end-to-end pipeline (capture, processing, transmission, and rendering) to allow shared immersive experiences in real-time and volumetric communication. Our novel transmission scheme for grayscale-based depth information via 2D video is particularity beneficial for web applications (that do not allow other depth conversion due to browser API limitations). Finally, the evaluation of our system in a simulation and realistic user setting shows that our solution utilizes processing (CPU / GPU) acceptable for modern (VR-ready) PCs and under network bandwidth constraints similar to existing video conferencing solutions. Still, more research is necessary to get all aspects of the

technology ready (i.e. spatial computing, HMD replacement, enhanced quality transmission of real-time 2D/3D video data and system scalability for large sets of simultaneous users).

RQ3 (Transmission Optimization): What is the maximum number of simultaneous users in Social XR and can we optimize the server and client resource needs to increase that number?
To address this research question, we implemented both a state-of-the-art transmission technique for the RGBD pipeline as well as a novel tiled-based encoding extension to evaluate the suitability of Social XR under different numbers of simultaneous users. Our naive approach (Capter 3) shows that we can easily support around 4 users in a full mesh network architecture (which is similar to most state of the art systems); this can significantly be increased with an SFU topology (to a number of 16, with some performance limitations). Combining the advantages of an SFU and an MCU, we created a novel central composition unit (T-MCU) that reduces both the resource usage on the rendering client and the central composition server. The T-MCU is built on a modified version of the VP9 encoder (libvpx). Our modifications to the encoder are fully standard-compliant and thus can be decoded by any VP9 decoder, but result in a ~16% bandwidth overhead compared to the standard libvpx VP9 encoder. Our new T-MCU offers similar resource advantages to the receiving rendering client as a traditional MCU chapter 3 with a performance gain of approximately 20% compared to a P2P/SFU transmission approach. Most importantly, the T-MCU improves transmission delay by ~ 20% and server performance gain by 85-90% compared to an MCU chapter 3. Finally, the T-MCU can offer a solution for the scalability of simultaneous users in demanding immersive communication applications and can support up to 32 simultaneous user streams.

Main RQ: What are the technical challenges and limitations in extending existing video conferencing solutions to support photorealistic 3D immersive communication with RGBD (color and depth) video streams?
To better understand our findings relating to technical challenges and limitations of the state-of-the-art, it is good to structure the discussion according to the simplified Social XR pipeline (Figure 1.2): capture, processing, transmission and rendering.

Regarding capture and processing, we have shown a new modality of formatting 3D RGBD data and different processing steps for this new format. Ultimately, RGBD-based capture creates higher resource demands than traditional video conferencing (our results show a performance overhead of 5-15% CPU & GPU load). In addition, we introduced a novel RGBD transmission scheme. The main advantage is that this allows reuse of existing encoding and decoding technologies and hardware acceleration. In this way, the overhead is low, as shown in various evaluation experiments. This forms a good basis for reference, but further enhancements and new transmission formations for immersive 3D user representation will be needed in the future to increase visual quality and overall user experience.

Finally, the rendering client needs significant changes to traditional 2D multimedia clients. Defining the environment, positioning of users and audio in 3D, rendering users in 'correct size' in virtual reality or blended into the real world. This required specific metadata, as presented in chapter 3. Furthermore, the 3D rendering client opposes increasing resource needs depending on the number of simultaneous users to display. To reduce the resources

on the end device and increase the number of simultaneous users, we proposed a novel tile-enabled multipoint composition unit (T-MCU). The extension of the T-MCU allows 32 users in simultaneous user communication spaces, while not significantly increasing the resource needs in the central composition server.

6.1.1 Outcome

Ultimately, the research presented in this dissertation provides a reference system and pipeline for many-to-many communication in Social XR, including in detail technical building blocks. We report multiple performance indicators for individual components of the system and the end client that can be directly compared with existing video conferencing solutions. We conclude that our work outlines the applicability and technical functioning of Social XR under different device, processing, and networking constraints. It can be seen as a new basic layer for Social XR research and development and opens up many new possibilities for further research.

6.2 Lessons learned

While 2D video conferencing is an established tool both from the technology and its usability, immersive media and XR are still novel concepts. This created multiple challenge in the development and research of the Social XR as presented in this dissertation. In the following we outline some of our most important challenges and personal lessons as an outcome of our work.

6.2.1 Novelty of XR

One of the key challenges of our work was that we dealt with novel technology on both the software and hardware side. The XR devices used throughout the research have gone through multiple stages of technical advancement. Devices have significantly increased their technical capabilities in the past years, including processing power and rendering quality (i.e. display resolution and pixel quality). Thus, creating generalized results was often a concern. Our work tried to mitigate this as much as possible by creating a frame of reference and results that can easily be put into new contexts and adopted to new technology. Furthermore, we can generally note that the increase in performance goes hand in hand with more demanding applications and resource needs for higher quality rendering.

We can also note that VR and AR developments are still predominantly driven by industry rather than user demands. This means that there is still a lack of many user aspects, such as clear design guidelines for user interfaces, user interaction, and user experiences. This makes it difficult to test technical concepts with users. Furthermore, the development of XR devices goes very fast in term of complexity and technical feature also often resulting into changes in corresponding APIs. Therefore there is significant need to align developments and standardize interoperable solutions.

6.2.2 Standardization and interoperability

Despite great efforts in the different standardization bodies (such as MPEG, 3GPP, W3C, ITU, and many more), many technical aspects related to XR are not well aligned. One positive

example is the OpenXR[1] standard of the Khronos Group, which specifies and aligns many low-level XR APIs. However, when it comes to immersive media formats, some metadata, applications, and end-to-end XR services, there are still many gaps that are currently addressed by the respective groups. Many of the concepts are simply not fully ready (technically and interoperable) and hinder the mass adaptation and further productization of XR services and devices. We do not expect that these developments will fundamentally change the results presented in this dissertation, but would have been of great help in our research. Furthermore, it is important to note that new developments in terms of XR technology and its standards often lead to the need to investigate and redevelop the basic assumptions of existing multimedia research. Simply because immersive technology is fundamentally different in user perception from existing 2D display methods.

6.2.3 Immersive media formats & representation quality

In general we can observe that at present media standards and the industry at large is lacking behind to fully embrace new immersive 3D media formats. This is while lots of new formats and techniques are in developments and discussed in the different standardization bodies (like MPEG, 3GPP, Khronos, and others) it will still take a significant time to have fully (hardware optimized) interoperable support in different devices. This means throughout the research of this dissertation, many concept were experimental and the amount and quality of reference implementations and datasets was low. This resulted into some additional challenges as better reference software and comparable datasets would have significantly helped to focus on core system aspects alone. Ultimately this means we had to keep a detailed focus on all aspects of the pipeline and develop our own transmission and representation format. Within this dissertation we were able to show the technical readiness and good results in technical quality of these new Social XR concepts, however the mapping to the users quality of experience is still unclear and out of scope of the presented work.

6.2.4 On QoS and QoE

The lack of well-defined technical requirements for Social XR was one of the biggest concerns in our research. This is no problem exclusive to Social XR as most technical requirements are also not well defined video conferencing systems (e.g. ideal delay, spatial resolution, performance demands, and more). We know, however, that in video conferencing requirements can vary strongly, according to the task and the many personal and impersonal influencing factors [1]. Furthermore, it is very difficult to map individual technical factors and QoS improvements to the active impact on the user benefits. Many different technical factors (besides the non-technical factors) interact and always influence each other with an impact on the user experience an task performance [1]. This means that all aspects of the system always need to work optimally and all aspects need to be of sufficient quality. In the context of the dissertation, even though we could show technical readiness and initial applicability in different use cases, how ready the presented immersive communication concepts are towards creating an improvement in user tasks remains to be discovered. We also expect that more open accessible immersive media software and data

[1] https://web.archive.org/web/20240420190815/https://www.khronos.org/openxr/

sets (such as [192]) can significantly simplify future research.

6.3 Future Work

With the scientific and technical contributions of the work presented in this dissertation, there are multiple directions for further research. In general, the work in this dissertation is related to a larger portfolio of work and ongoing activities in both industry and academic research. Some of these research directions are summarized below.

6.3.1 QoE

Although significant advancements have been made for holographic QoE, its research is still in its infancy. Some initial work established metrics and testing for 3D user point cloud representations [195, 196], a generic Social XR questionnaire [197] and some first complex evaluation of photorealistic 3D multi-user experiences in VR [198, 199]. To get conclusive conclusions on QoE in Social XR, more tests need to be done. This is also to better understand the different advantages and disadvantages of Social XR related to the user experience under different use cases and communication tasks. Furthermore, existing multimedia techniques for mapping QoS to QoE need to be revisited. This is because immersive experiences in Social XR are significantly different from traditional multimedia experiences. Creating the need for more research, particularly related to understanding the impact of user representations in Social XR.

6.3.2 3D user encoding

Uncompressed 3D user representations produce a very high data load and, as such, cannot be transmitted over regular network connections. Thus, high-quality compression is essential for holographic user data. In our research, we used RGBD video data for user representations due to its compatibility with existing transmission channels and hardware encoding/decoding. However, to increase effectiveness, efficiency, and visual quality, new transmission/encoding strategies are needed and are currently under development and research [71, 200]. One main direction is to encode users as point clouds, either video-based (V-PCC) or geometry-based (G-PCC). V-PCC supports the reuse of (hardware enabled) encoders and decoder, but is stalled in real-time execution due to complex point-to-video patch conversions. This is a current area of research. G-PCC is a very promising direction for the 3D user encoding and constantly advancing, but will still need a significant time to reach maturity and wide-scale (hardware) device support. Finally, another direction of research is to use AI and ML methods such as (Pixel) Codec Avatars [201, 202] and Nvidea Maxime [203], encoding user representations via facial expressions and keypoint extraction [204]. Beyond the representation quality, these methods, however, require a better understanding of the context, behavior, and expressions of the user.

6.3.3 Spatial Computing and Metadata

In order to allow complex interaction between users and (virtual) objects in Social XR, detecting their context, behavior, and movement is essential. This relates to many research topics in the area of sensing and intelligent image analysis, often referred to as spatial computing [205]. In addition to the sensing and semantic understanding that spatial

computing may create, it is also important how spatial (meta)data is stored, transmitted, and syncronized. One of the promising works that defines spatial metadata is the royalty-free specification of glTF™[2] by the khronos®group. Although glTF™ covers many aspects of the presentation of the environment, objects, and users, it does not fully cover the aspects of dynamic object interactions. We see a need to better define spatial computing use cases with complex dynamic interaction between a mix of real and virtual people, objects, and environments to better define new requirements and enable more research on those topics. In recent years, there has already been much research and development on spatial computing, and it is expected that with the recent introduction of the Apple Vision Pro (AVP) spatial computing solutions will rapidly increase in capabilities [206]. This new wave of devices like the AVP allow many new sensing and processing capabilities, including optimized on-device AI/ML, to allow many new spatial computing solutions.

6.3.4 Network-based (split) Rendering

With this new wave of spatial computing, sensing, and hardware capabilities of XR devices, we also expect many more similar devices in the future. Specifically, we expect new light-weight glass-type AR devices that can be easily applied in many new scenarios of everyday life. These glass-type devices are already a focus point in the industry; for example, in 3GPP SA4 [207, 208]. However, their form factor, limited computing power, and limited battery bring new challenges that need to be investigated. For example, one requirement is to reduce the performance needs for immersive experience by moving (parts of) processing from the device into new processing components within the network (in the cloud or edge compute nodes). One well-established concept for moving processing is remote rendering. Remote rendering is an established concept in gaming [209, 210], but is still challenging in XR scenarios [211]. Furthermore, from the processing components presented in this dissertation, we believe that many more processing functionality could benefit from network assistance. Finally, the operation of these new AR-type devices in ubiquitous everyday situations also creates new requirements on mobile networks.

6.3.5 Beyond 5G & QoS

Communication became intrinsically mobile and in situ. Therefore, it is important for any communication technology how it is applied and integrated into mobile networks. When it comes to mobile networks, recent advances in 5G technology allow many new benefits beyond higher bandwidth and lower delays. For example, 5G introduces new networking concepts, such as network slicing (dedicated virtual networks), edge computing (processing very close to the end device on the network edge), and QoS guarantees. We expect that these benefits and the complexity of network functions will increase further when moving beyond 5G and towards 6G functionalities. In addition, these new network topologies and new envisioned services, such as immersive multimedia and Social XR, also need new dedicated QoS metrics that indicate the health of the network. Therefore, there is currently a need to research and define many of the 5G-related functionality in relation to immersive communication, as well as to define requirements for the next generation of mobile networks (6G). In addition, with the increasing complexity of the network and

[2] https://web.archive.org/web/20240209132249/https://www.khronos.org/gltf/

the demand for novel immersive services, concerns about energy efficiency and resource needs are also growing.

6.3.6 Sustainability & Energy efficiency

Sustainability and energy efficiency in the multimedia domain have not received much attention in recent decades in technical developments, research, or standardization. Most optimizations in resources and related research focused on device or network limitations, rather than overall energy consumption. This has changed a lot in recent years, strongly motivated by issues related to global warming. It is good to observe that many conferences expect the papers to publish detailed resource and energy footprints of their solutions, and many new standardization activities emerge with the focus on sustainability. Although this dissertation is largely motivated by improving remote communication (also in the scope of reducing but not replacing travels), it does not address sustainability research sufficiently. The optimizations presented in chapter 4 can be a good starting point for further research in creating better and more sustainable video transmission technologies with reduced research usage in server and client components.

6.3.7 Privacy, Identity & Ethical considerations

In addition to sustainability, one other common concern with immersive communication systems is privacy and identity. How can we guarantee that user data including images are handled in a secure and privacy aware way, and how can we guarantee that depicted users are indeed the same person as in reality? In the dissertation, we addressed this issue by designing a transparent end-to-end system that has no central data storage or manipulation. In this way, we currently always capture and render the user as is (at a particular moment). However, with moving processing to system components (for example, into 5G/6G mobile edge), or introducing more AI/ML image manipulation techniques (e.g. replacing part of a user captured with a VR HMD with a realistic representation of his/her face), new problems arise. Intelligent imagining techniques are becoming hyper-realistic, and AI generated content is becoming indistinguishable from content captured in the real world. This creates the need for many discussions on how to apply and restrict such technology and possibly requires clear industry guidelines and government policies.

6.4 Final Remarks

RGBD video has proven as a key enabler for Social XR. Even if parts of the transmission might be replaced by new techniques and encoding formats, it will still be used within many immersive technology solutions in the future. This is many 3D sensing technologies, as well as many AR technologies (i.e. Android, iPhone, Microsoft HoloLens) built on top of RGBD data as basis technology.

Immersive communication and multi-user immersive experiences can be a great way to break the boundaries of space and move towards a sustainable future. The virtual, augmented, and real worlds can blend into each other and could have a strong impact on the way we communicate in and perceive the world around us. Imagine being able to visit any real or virtual part of the world from any time in history, wherever you are, and with whomever you like. However, it is important to note that we do not see Social XR as a

replacement technology for existing communication channels or meetings in real life, but as a new and more natural way of interacting over distance. ICS and Social XR are here to stay and in the future might be used on a daily basis as video conferencing is now.

Bibliography

References

[1] Janto Skowronek, Alexander Raake, Gunilla H Berndtsson, Olli S Rummukainen, Paolino Usai, Simon NB Gunkel, Mathias Johanson, Emanuël AP Habets, Ludovic Malfait, David Lindero, et al. Quality of experience in telemeetings and videoconferencing: a comprehensive survey. *IEEE Access*, 10:63885–63931, 2022.

[2] Willem Standaert, Steve Muylle, and Amit Basu. Business meetings in a postpandemic world: When and how to meet virtually. *Business Horizons*, 65(3):267–275, 2022.

[3] OECD. Teleworking in the covid-19 pandemic: trends and prospects, 2021.

[4] Anne-Laure Fayard, John Weeks, Mahwesh Khan, et al. Designing the hybrid office. *Harvard Business Review*, 99(2):114–123, 2021.

[5] Lynda Gratton. How to do hybrid right. *Harvard Business Review*, 99(3):66–74, 2021.

[6] Melanie S Brucks and Jonathan Levav. Virtual communication curbs creative idea generation. *Nature*, 605(7908):108–112, 2022.

[7] Anastasia Kuzminykh and Sean Rintel. Low engagement as a deliberate practice of remote participants in video meetings. In *Extended Abstracts of the 2020 CHI Conference on Human Factors in Computing Systems*, pages 1–9, 2020.

[8] Hadar Nesher Shoshan and Wilken Wehrt. Understanding "zoom fatigue": A mixed-method approach. *Applied Psychology*, 71(3):827–852, 2022.

[9] Geraldine Fauville, Mufan Luo, Anna CM Queiroz, Jeremy N Bailenson, and Jeff Hancock. Zoom exhaustion & fatigue scale. *Computers in Human Behavior Reports*, 4:100119, 2021.

[10] Alexander Toet, Tina Mioch, Simon NB Gunkel, Omar Niamut, and Jan BF van Erp. Holistic framework for quality assessment of mediated social communication. 2021.

[11] Philipp A Rauschnabel, Reto Felix, Chris Hinsch, Hamza Shahab, and Florian Alt. What is xr? towards a framework for augmented and virtual reality. *Computers in human behavior*, 133:107289, 2022.

[12] Paul Milgram, Haruo Takemura, Akira Utsumi, and Fumio Kishino. Augmented reality: A class of displays on the reality-virtuality continuum. In *Telemanipulator and telepresence technologies*, volume 2351, pages 282–292. Spie, 1995.

[13] Richard Skarbez, Missie Smith, and Mary C Whitton. Revisiting milgram and kishino's reality-virtuality continuum. *Frontiers in Virtual Reality*, 2:647997, 2021.

[14] Elliot Aronson and Joshua Aronson. *The Social Animal*. W.H. Freeman, New York, NY, 12 edition, June 2018.

[15] Michael Haller. Photorealism or/and non-photorealism in augmented reality. In *Proceedings of the 2004 ACM SIGGRAPH International Conference on Virtual Reality Continuum and Its Applications in Industry*, VRCAI '04, page 189–196, New York, NY, USA, 2004. Association for Computing Machinery.

[16] Maia Garau, Mel Slater, Vinoba Vinayagamoorthy, Andrea Brogni, Anthony Steed, and M Angela Sasse. The impact of avatar realism and eye gaze control on perceived quality of communication in a shared immersive virtual environment. In *Proceedings of the SIGCHI conference on Human factors in computing systems*, pages 529–536, 2003.

[17] Marc Erich Latoschik, Daniel Roth, Dominik Gall, Jascha Achenbach, Thomas Waltemate, and Mario Botsch. The effect of avatar realism in immersive social virtual realities. In *Proceedings of the 23rd ACM symposium on virtual reality software and technology*, pages 1–10, 2017.

[18] Daniel Roth, Jean-Luc Lugrin, Dmitri Galakhov, Arvid Hofmann, Gary Bente, Marc Erich Latoschik, and Arnulph Fuhrmann. Avatar realism and social interaction quality in virtual reality. In *2016 IEEE virtual reality (VR)*, pages 277–278. IEEE, 2016.

[19] Xiaozhe Yang, Lin Lin, Pei-Yu Cheng, Xue Yang, Youqun Ren, and Yueh-Min Huang. Examining creativity through a virtual reality support system. *Educational Technology Research and Development*, 66:1231–1254, 2018.

[20] Divine Maloney, Guo Freeman, and Donghee Yvette Wohn. " talking without a voice" understanding non-verbal communication in social virtual reality. *Proceedings of the ACM on Human-Computer Interaction*, 4(CSCW2):1–25, 2020.

[21] John G. Apostolopoulos, Philip A. Chou, Bruce Culbertson, Ton Kalker, Mitchell D. Trott, and Susie Wee. The road to immersive communication. *Proceedings of the IEEE*, 100(4):974–990, 2012.

[22] Richard L Daft and Robert H Lengel. Information richness. a new approach to managerial behavior and organization design. Technical report, Texas A and M Univ College Station Coll of Business Administration, 1983.

[23] Richard L Daft and Robert H Lengel. Organizational information requirements, media richness and structural design. *Management science*, 32(5):554–571, 1986.

[24] Vivian C. Sheer. *Media Richness Theory*, pages 1–14. John Wiley & Sons, Ltd, 2020.

[25] Joseph E McGrath and A. B. Hollingshead. Putting the" group" back in group support systens: Some thepretical issues about dynamic processes in groups with technological enhancements. *Group Support systems: New perpectives*, pages 78–96, 1993.

[26] Brian E Mennecke, Joseph S Valacich, and Bradley C Wheeler. The effects of media and task on user performance: A test of the task-media fit hypothesis. *Group Decision and Negotiation*, 9:507–529, 2000.

[27] Pablo Pérez, Ester Gonzalez-Sosa, Jesús Gutiérrez, and Narciso García. Emerging immersive communication systems: Overview, taxonomy, and good practices for qoe assessment. *Frontiers in Signal Processing*, 2, 2022.

[28] Jason Lawrence, Dan B Goldman, Supreeth Achar, Gregory Major Blascovich, Joseph G. Desloge, Tommy Fortes, Eric M. Gomez, Sascha Häberling, Hugues Hoppe, Andy Huibers, Claude Knaus, Brian Kuschak, Ricardo Martin-Brualla, Harris Nover, Andrew Ian Russell, Steven M. Seitz, and Kevin Tong. Project starline: A high-fidelity telepresence system. *ACM Transactions on Graphics (Proc. SIGGRAPH Asia)*, 40(6), 2021.

[29] Jason Leigh, Andrew E Johnson, Thomas A DeFanti, Maxine Brown, M Dastagir Ali, Stuart Bailey, Andy Banerjee, P Benerjee, Jim Chen, Kevin Curry, et al. A review of tele-immersive applications in the cave research network. In *Proceedings IEEE Virtual Reality (Cat. No. 99CB36316)*, pages 180–187. IEEE, 1999.

[30] John Wiecha, Robin Heyden, Elliot Sternthal, and Mario Merialdi. Learning in a virtual world: experience with using second life for medical education. *Journal of medical Internet research*, 12(1):e1, 2010.

[31] Joshua McVeigh-Schultz, Anya Kolesnichenko, and Katherine Isbister. Shaping pro-social interaction in vr: An emerging design framework. In *Proceedings of the 2019 CHI Conference on Human Factors in Computing Systems*, pages 1–12, 2019.

[32] David Jones, Chris Snider, Aydin Nassehi, Jason Yon, and Ben Hicks. Characterising the digital twin: A systematic literature review. *CIRP Journal of Manufacturing Science and Technology*, 29:36–52, 2020.

[33] Simon NB Gunkel, Rick Hindriks, Karim M El Assal, Hans M Stokking, Sylvie Dijkstra-Soudarissanane, Frank ter Haar, and Omar Niamut. Vrcomm: an end-to-end web system for real-time photorealistic social vr communication. In *Proceedings of the 12th ACM multimedia systems conference*, pages 65–79, 2021.

[34] Irene Viola, Jack Jansen, Shishir Subramanyam, Ignacio Reimat, and Pablo Cesar. Vr2gather: A collaborative social vr system for adaptive multi-party real-time communication. *IEEE MultiMedia*, 2023.

[35] Sergi Fernández Langa, Mario Montagud, Gianluca Cernigliaro, and David Rincón Rivera. Multiparty holomeetings: Toward a new era of low-cost volumetric holographic meetings in virtual reality. *Ieee Access*, 10:81856–81876, 2022.

[36] Dimitrios S Alexiadis, Anargyros Chatzitofis, Nikolaos Zioulis, Olga Zoidi, Georgios Louizis, Dimitrios Zarpalas, and Petros Daras. An integrated platform for live 3d human reconstruction and motion capturing. *IEEE Transactions on Circuits and Systems for Video Technology*, 27(4):798–813, 2016.

[37] Kyriaki Christaki, Konstantinos C Apostolakis, Alexandros Doumanoglou, Nikolaos Zioulis, Dimitrios Zarpalas, and Petros Daras. Space wars: An augmentedvr game. In *MultiMedia Modeling: 25th International Conference, MMM 2019, Thessaloniki, Greece, January 8–11, 2019, Proceedings, Part II 25*, pages 566–570. Springer, 2019.

[38] Kyriaki Christaki, Emmanouil Christakis, Petros Drakoulis, Alexandros Doumanoglou, Nikolaos Zioulis, Dimitrios Zarpalas, and Petros Daras. Subjective visual quality assessment of immersive 3d media compressed by open-source static 3d mesh codecs. In *MultiMedia Modeling: 25th International Conference, MMM 2019, Thessaloniki, Greece, January 8–11, 2019, Proceedings, Part I 25*, pages 80–91. Springer, 2019.

[39] Sergio Orts-Escolano, Christoph Rhemann, Sean Fanello, Wayne Chang, Adarsh Kowdle, Yury Degtyarev, David Kim, Philip L Davidson, Sameh Khamis, Mingsong Dou, et al. Holoportation: Virtual 3d teleportation in real-time. In *Proceedings of the 29th Annual Symposium on User Interface Software and Technology*, pages 741–754, 2016.

[40] Ahsan Arefin, Zixia Huang, Klara Nahrstedt, and Pooja Agarwal. 4d telecast: Towards large scale multi-site and multi-view dissemination of 3dti contents. In *2012 IEEE 32nd International Conference on Distributed Computing Systems*, pages 82–91, 2012.

[41] Shannon Chen, Zhenhuan Gao, Klara Nahrstedt, and Indranil Gupta. 3dti amphitheater: Towards 3dti broadcasting. *ACM Transactions on Multimedia Computing, Communications, and Applications (TOMM)*, 11(2s):1–22, 2015.

[42] Yizhong Zhang, Jiaolong Yang, Zhen Liu, Ruicheng Wang, Guojun Chen, Xin Tong, and Baining Guo. Virtualcube: An immersive 3d video communication system. *IEEE Transactions on Visualization and Computer Graphics*, 28(5):2146–2156, 2022.

[43] Carlos Coelho, JG Tichon, Trevor J Hine, GM Wallis, and Giuseppe Riva. Media presence and inner presence: the sense of presence in virtual reality technologies. In *From communication to presence: Cognition, emotions and culture towards the ultimate communicative experience*, pages 25–45. IOS Press, Amsterdam, 2006.

[44] Matthew Lombard and Theresa Ditton. At the heart of it all: The concept of presence. *Journal of computer-mediated communication*, 3(2):JCMC321, 1997.

[45] Paolo Barzon, Simon Gunkel, Evangelos Alexiou, and Sylvie Dijkstra-Soudarissanane. The value of immersive communication systems in online meetings: A problem statement and literature review. In *Proceedings of the 2023 IEEE International Conference on Metrology for eXtended Reality, Artificial Intelligence, and Neural Engineering (IEEE MetroXRAINE 2023)*, 2023.

[46] Simon NB Gunkel, Martin Prins, Hans Stokking, and Omar Niamut. Social vr platform: Building 360-degree shared vr spaces. In *Adjunct publication of the 2017 ACM international conference on interactive experiences for tv and online video*, pages 83–84, 2017.

[47] Alexander Clemm, Maria Torres Vega, Hemanth Kumar Ravuri, Tim Wauters, and Filip De Turck. Toward truly immersive holographic-type communication: Challenges and solutions. *IEEE Communications Magazine*, 58(1):93–99, 2020.

[48] Ian F Akyildiz and Hongzhi Guo. Holographic-type communication: A new challenge for the next decade. *ITU Journal on Future and Evolving Technologies*, 2022.

[49] Andrew D Wilson. Fast lossless depth image compression. In *Proceedings of the 2017 ACM International Conference on Interactive Surfaces and Spaces*, pages 100–105, 2017.

[50] Andrew D. Wilson and Hrvoje Benko. Projected augmented reality with the roomalive toolkit. In *Proceedings of the 2016 ACM International Conference on Interactive Surfaces and Spaces*, ISS '16, page 517–520, New York, NY, USA, 2016. Association for Computing Machinery.

[51] Pooja Agarwal, Raoul Rivas Toledano, Wanmin Wu, Klara Nahrstedt, and Ahsan Arefin. Bundle of streams: Concept and evaluation in distributed interactive multimedia environments. In *2010 IEEE International Symposium on Multimedia*, pages 25–32. IEEE, 2010.

[52] M.R. Schmitt. *Personal Quality of Experience: Accurately modelling Quality of Experience for multi-party desktop video-conferencing based on systems, context and user factors*. Phd dissertation, Vrije Universiteit Amsterdam, 2019.

[53] M Angela Sasse, Mark J Handley, and Nermeen M Ismail. Coping with complexity and interference: Design issues in multimedia conferencing systems. In *Design Issues in CSCW*, pages 179–195. Springer, 1994.

[54] Oliver Frick. Multimedia conferencing systems as building blocks for complex cooperative applications. In *Proceedings International Workshop on Multimedia Software Development*, pages 61–68. IEEE, 1996.

[55] Yanjiao Chen, Kaishun Wu, and Qian Zhang. From qos to qoe: A tutorial on video quality assessment. *IEEE Communications Surveys & Tutorials*, 17(2):1126–1165, 2014.

[56] Joël Tapie, Patrice Terrier, Laurence Perron, and J-M Cellier. Should remote collaborators be represented by avatars? a matter of common ground for collective medical decision-making. *AI & SOCIETY*, 20:331–350, 2006.

[57] Simon N.B. Gunkel, Sylvie Dijkstra-Soudarissanane, Hans M. Stokking, and Omar Niamut. From 2d to 3d video conferencing: Modular rgb-d capture and reconstruction for interactive natural user representations in immersive extended reality (xr) communication. *Frontiers in Signal Processing*, 3, 2023.

[58] Simon N.B. Gunkel, Rick Hindriks, Yonatan Shiferaw, Sylvie Dijkstra-Soudarissanane, and Omar Niamut. Vp9 bitstream-based tiled multipoint control unit: Scaling simultaneous rgbd user streams in an immersive 3d communication system. In *Proceedings of the 15th ACM Multimedia Systems Conference*, MMSys '24, page 23–33, 2024.

[59] Simon NB Gunkel, Hans M Stokking, Martin J Prins, Nanda van der Stap, Frank B ter Haar, and Omar A Niamut. Virtual reality conferencing: Multi-user immersive vr experiences on the web. In *Proceedings of the 9th ACM Multimedia Systems Conference*, pages 498–501, 2018.

[60] Sylvie Dijkstra-Soudarissanane, Simon NB Gunkel, and Véronne Reinders. Virtual visits: life-size immersive communication. In *Proceedings of the 13th ACM Multimedia Systems Conference*, pages 310–314, 2022.

[61] Simon NB Gunkel, Sylvie Dijkstra-Soudarissanane, and Omar Niamut. "you ar' right in front of me": Rgbd-based capture and rendering for remote training. In *Proceedings of the 14th Conference on ACM Multimedia Systems*, pages 307–311, 2023.

[62] Kay M Stanney, Hannah Nye, Sam Haddad, Kelly S Hale, Christina K Padron, and Joseph V Cohn. extended reality (xr) environments. *Handbook of human factors and ergonomics*, pages 782–815, 2021.

[63] Dimitrios S Alexiadis, Anargyros Chatzitofis, Nikolaos Zioulis, Olga Zoidi, Georgios Louizis, Dimitrios Zarpalas, and Petros Daras. An integrated platform for live 3d human reconstruction and motion capturing. *IEEE Transactions on Circuits and Systems for Video Technology*, 27(4):798–813, 2017.

[64] Alvaro Collet, Ming Chuang, Pat Sweeney, Don Gillett, Dennis Evseev, David Calabrese, Hugues Hoppe, Adam Kirk, and Steve Sullivan. High-quality streamable free-viewpoint video. *ACM Trans. Graph.*, 34(4), July 2015.

[65] Rufael Mekuria, Kees Blom, and Pablo Cesar. Design, implementation, and evaluation of a point cloud codec for tele-immersive video. *IEEE Transactions on Circuits and Systems for Video Technology*, 27(4):828–842, 2017.

[66] Jounsup Park, Philip A Chou, and Jenq-Neng Hwang. Rate-utility optimized streaming of volumetric media for augmented reality. *IEEE Journal on Emerging and Selected Topics in Circuits and Systems*, 9(1):149–162, 2019.

[67] Oliver Schreer, Ingo Feldmann, Sylvain Renault, Marcus Zepp, Markus Worchel, Peter Eisert, and Peter Kauff. Capture and 3d video processing of volumetric video. In *2019 IEEE International conference on image processing (ICIP)*, pages 4310–4314. IEEE, 2019.

[68] SungIk Cho, Seung-wook Kim, JongMin Lee, JeongHyeon Ahn, and JungHyun Han. Effects of volumetric capture avatars on social presence in immersive virtual environments. In *2020 IEEE conference on virtual reality and 3D user interfaces (VR)*, pages 26–34. IEEE, 2020.

[69] Shishir Subramanyam, Jie Li, Irene Viola, and Pablo Cesar. Comparing the quality of highly realistic digital humans in 3dof and 6dof: A volumetric video case study. In *2020 IEEE Conference on Virtual Reality and 3D User Interfaces (VR)*, pages 127–136. IEEE, 2020.

[70] Nikolaos Zioulis, Dimitrios Alexiadis, Alexandros Doumanoglou, Georgios Louizis, Konstantinos Apostolakis, Dimitrios Zarpalas, and Petros Daras. 3d tele-immersion platform for interactive immersive experiences between remote users. In *2016 IEEE International Conference on Image Processing (ICIP)*, pages 365–369, 2016.

[71] S. Schwarz, M. Preda, V. Baroncini, M. Budagavi, P. Cesar, P. A. Chou, R. A. Cohen, M. Krivokuća, S. Lasserre, Z. Li, J. Llach, K. Mammou, R. Mekuria, O. Nakagami, E. Siahaan, A. Tabatabai, A. M. Tourapis, and V. Zakharchenko. Emerging mpeg standards for point cloud compression. *IEEE Journal on Emerging and Selected Topics in Circuits and Systems*, 9(1):133–148, 2019.

[72] Fabrizio Pece, Jan Kautz, and Tim Weyrich. Adapting standard video codecs for depth streaming. In *EGVE/EuroVR*, pages 59–66, 2011.

[73] Sam Ekong, Christoph W. Borst, Jason Woodworth, and Terrence L. Chambers. Teacher-student vr telepresence with networked depth camera mesh and heterogeneous displays. In George Bebis, Richard Boyle, Bahram Parvin, Darko Koracin, Fatih Porikli, Sandra Skaff, Alireza Entezari, Jianyuan Min, Daisuke Iwai, Amela Sadagic, Carlos Scheidegger, and Tobias Isenberg, editors, *Advances in Visual Computing*, pages 246–258, Cham, 2016. Springer International Publishing.

[74] Yunpeng Liu, Stephan Beck, Renfang Wang, Jin Li, Huixia Xu, Shijie Yao, Xiaopeng Tong, and Bernd Froehlich. Hybrid lossless-lossy compression for real-time depth-sensor streams in 3d telepresence applications. In *Advances in Multimedia Information Processing–PCM 2015: 16th Pacific-Rim Conference on Multimedia, Gwangju, South Korea, September 16-18, 2015, Proceedings, Part I 16*, pages 442–452. Springer, 09 2015.

[75] Christoph Fehn. Depth-image-based rendering (dibr), compression, and transmission for a new approach on 3d-tv. In *Stereoscopic displays and virtual reality systems XI*, volume 5291, pages 93–104. SPIE, 2004.

[76] Patrick Ndjiki-Nya, Martin Koppel, Dimitar Doshkov, Haricharan Lakshman, Philipp Merkle, Karsten Muller, and Thomas Wiegand. Depth image-based rendering with advanced texture synthesis for 3-d video. *IEEE Transactions on Multimedia*, 13(3):453–465, 2011.

[77] Carmine Elvezio, Mengu Sukan, Ohan Oda, Steven Feiner, and Barbara Tversky. Remote collaboration in ar and vr using virtual replicas. In *ACM SIGGRAPH 2017 VR Village*, SIGGRAPH '17, New York, NY, USA, 2017. Association for Computing Machinery.

[78] RA Grier, H Thiruvengada, SR Ellis, P Havig, KS Hale, and JG Hollands. Augmented reality–implications toward virtual reality, human perception and performance. In *Proceedings of the Human Factors and Ergonomics Society Annual Meeting*, volume 56, pages 1351–1355. SAGE Publications Sage CA: Los Angeles, CA, 2012.

[79] Pablo Carballeira, Carlos Carmona, César Díaz, Daniel Berjon, Daniel Corregidor, Julián Cabrera, Francisco Morán, Carmen Doblado, Sergio Arnaldo, María del

Mar Martín, et al. Fvv live: A real-time free-viewpoint video system with consumer electronics hardware. *IEEE Transactions on Multimedia*, 24:2378–2391, 2021.

[80] Sverker Rasmuson, Erik Sintorn, and Ulf Assarsson. A low-cost, practical acquisition and rendering pipeline for real-time free-viewpoint video communication. *The Visual Computer*, 37:553–565, 2021.

[81] Sergi Fernández Langa, Mario Montagud Climent, Gianluca Cernigliaro, and David Rincón Rivera. Toward hyper-realistic and interactive social vr experiences in live tv scenarios. *IEEE Transactions on Broadcasting*, 68(1):13–32, 2021.

[82] Ignacio Reimat, Yanni Mei, Evangelos Alexiou, Jack Jansen, Jie Li, Shishir Subramanyam, Irene Viola, Johan Oomen, and Pablo Cesar. Mediascape xr: A cultural heritage experience in social vr. In *Proceedings of the 30th ACM International Conference on Multimedia*, pages 6955–6957, 2022.

[83] Martin J Prins, Simon NB Gunkel, Hans M Stokking, and Omar A Niamut. Togethervr: A framework for photorealistic shared media experiences in 360-degree vr. *SMPTE Motion Imaging Journal*, 127(7):39–44, 2018.

[84] Simon NB Gunkel, Hans Stokking, Tom De Koninck, and Omar Niamut. Everyday photo-realistic social vr: Communicate and collaborate with an enhanced co-presence and immersion. In *Proceedings of International Broadcast Conference (IBC 2019)*, 2019.

[85] Simon NB Gunkel, Marleen DW Dohmen, Hans Stokking, and Omar Niamut. 360-degree photo-realistic vr conferencing. In *2019 IEEE Conference on Virtual Reality and 3D User Interfaces (VR)*, pages 946–947. IEEE, 2019.

[86] Francesca De Simone, Jie Li, Henrique Galvan Debarba, Abdallah El Ali, Simon NB Gunkel, and Pablo Cesar. Watching videos together in social virtual reality: an experimental study on user's qoe. In *IEEE VR Conference*, 2019.

[87] Alain Pinsonneault and Martin Boisvert. The impacts of telecommuting on organizations and individuals: A review of the literature. In *Telecommuting and virtual offices: Issues and opportunities*, pages 163–185. IGI Global, 2001.

[88] Lori Foster Thompson and Michael D Coovert. Understanding and developing virtual computer-supported cooperative work teams. In *Creating high-tech teams: Practical guidance on work performance and technology*, pages 213—-241. American Psychological Association, 2006.

[89] Susan G Straus. Technology, group process, and group outcomes: Testing the connections in computer-mediated and face-to-face groups. *Human–Computer Interaction*, 12(3):227–266, 1997.

[90] Wayne F Cascio. Managing a virtual workplace. *Academy of Management Perspectives*, 14(3):81–90, 2000.

[91] Benoit A Aubert and Barbara L Kelsey. Further understanding of trust and performance in virtual teams. *Small group research*, 34(5):575–618, 2003.

[92] Heather A Priest, Kevin C Stagl, Cameron Klein, and Eduardo Salas. Virtual teams: Creating context for distributed teamwork. 2006.

[93] Jeanne M Wilson, Susan G Straus, and Bill McEvily. All in due time: The development of trust in computer-mediated and face-to-face teams. *Organizational behavior and human decision processes*, 99(1):16–33, 2006.

[94] Elizabeth A Boyle, Anne H Anderson, and Alison Newlands. The effects of visibility on dialogue and performance in a cooperative problem solving task. *Language and speech*, 37(1):1–20, 1994.

[95] Abigail J Sellen. Remote conversations: The effects of mediating talk with technology. *Human-computer interaction*, 10(4):401–444, 1995.

[96] Marwin Schmitt, Simon Gunkel, Pablo Cesar, and Dick Bulterman. Asymmetric delay in video-mediated group discussions. In *2014 sixth international workshop on quality of multimedia experience (QoMEX)*, pages 19–24. IEEE, 2014.

[97] Chenguang Yu, Yang Xu, Bo Liu, and Yong Liu. "can you see me now?" a measurement study of mobile video calls. In *IEEE INFOCOM 2014-IEEE Conference on Computer Communications*, pages 1456–1464. IEEE, 2014.

[98] David Roberts, Toby Duckworth, Carl Moore, Robin Wolff, and John O'Hare. Comparing the end to end latency of an immersive collaborative environment and a video conference. In *2009 13th IEEE/ACM International Symposium on Distributed Simulation and Real Time Applications*, pages 89–94. IEEE, 2009.

[99] Gregorij Kurillo, Evan Hemingway, Mu-Lin Cheng, and Louis Cheng. Evaluating the accuracy of the azure kinect and kinect v2. *Sensors*, 22(7):2469, 2022.

[100] Vladimir Tadic, Attila Toth, Zoltan Vizvari, Mihaly Klincsik, Zoltan Sari, Peter Sarcevic, Jozsef Sarosi, and Istvan Biro. Perspectives of realsense and zed depth sensors for robotic vision applications. *Machines*, 10(3):183, 2022.

[101] Mario Montagud, Juan Antonio De Rus, Rafael Fayos-Jordan, Miguel Garcia-Pineda, and Jaume Segura-Garcia. Open-source software tools for measuring resources consumption and dash metrics. In *Proceedings of the 11th ACM Multimedia Systems Conference*, MMSys '20, page 261–266, New York, NY, USA, 2020. Association for Computing Machinery.

[102] Jack Jansen. Videolat: An extensible tool for multimedia delay measurements. In *Proceedings of the 22nd ACM International Conference on Multimedia*, MM '14, page 683–686, New York, NY, USA, 2014. Association for Computing Machinery.

[103] Sylvie Dijkstra-Soudarissanane, Karim El Assal, Simon Gunkel, Frank ter Haar, Rick Hindriks, Jan Willem Kleinrouweler, and Omar Niamut. Multi-sensor capture and network processing for virtual reality conferencing. In *Proceedings of the 10th ACM Multimedia Systems Conference*, pages 316–319, 2019.

[104] Simardeep Singh. Towards guidelines for facilitating engaging social vr business meetings. Master's thesis, 2022.

[105] Jon Martin Denstadli, Tom Erik Julsrud, and Randi Johanne Hjorthol. Videoconferencing as a mode of communication: A comparative study of the use of videoconferencing and face-to-face meetings. *Journal of Business and Technical Communication*, 26(1):65–91, 2012.

[106] Rieks op den Akker, Dennis Hofs, Hendri Hondorp, Harm op den Akker, Job Zwiers, and Anton Nijholt. Supporting engagement and floor control in hybrid meetings. In *Cross-Modal Analysis of Speech, Gestures, Gaze and Facial Expressions*, pages 276–290. Springer, 2009.

[107] Simardeep Singh, Sylvie Dijkstra-Soudarissanane, and Simon Gunkel. Engagement and quality of experience in remote business meetings: A social vr study. In *Proceedings of the 1st Workshop on Interactive eXtended Reality*, pages 77–82, 2022.

[108] Frank Biocca, Chad Harms, and Jenn Gregg. The networked minds measure of social presence: Pilot test of the factor structure and concurrent validity. In *4th annual international workshop on presence, Philadelphia, PA*, pages 1–9, 2001.

[109] John Brooke et al. Sus-a quick and dirty usability scale. *Usability evaluation in industry*, 189(194):4–7, 1996.

[110] Maia Garau, Mel Slater, Simon Bee, and Martina Angela Sasse. The impact of eye gaze on communication using humanoid avatars. In *Proceedings of the SIGCHI conference on Human factors in computing systems*, pages 309–316, 2001.

[111] Charlene Jennett, Anna L Cox, Paul Cairns, Samira Dhoparee, Andrew Epps, Tim Tijs, and Alison Walton. Measuring and defining the experience of immersion in games. *International journal of human-computer studies*, 66(9):641–661, 2008.

[112] Kristine L Nowak and Frank Biocca. The effect of the agency and anthropomorphism on users' sense of telepresence, copresence, and social presence in virtual environments. *Presence: Teleoperators & Virtual Environments*, 12(5):481–494, 2003.

[113] Eric N Wiebe, Allison Lamb, Megan Hardy, and David Sharek. Measuring engagement in video game-based environments: Investigation of the user engagement scale. *Computers in Human Behavior*, 32:123–132, 2014.

[114] Bob G Witmer and Michael J Singer. Measuring presence in virtual environments: A presence questionnaire. *Presence*, 7(3):225–240, 1998.

[115] Johanna ER Rutten, Ramona Backhaus, Jan Ph Hamers, and Hilde Verbeek. Working in a dutch nursing home during the covid-19 pandemic: experiences and lessons learned. *Nursing open*, 9(6):2710–2719, 2022.

[116] Sylvie Dijkstra-Soudarissanane, Tessa Klunder, Aschwin Brandt, and Omar Niamut. Towards xr communication for visiting elderly at nursing homes. In *ACM International Conference on Interactive Media Experiences*, IMX '21, page 319–321, New York, NY, USA, 2021. Association for Computing Machinery.

[117] Marina Alvarez, Alexander Toet, and Sylvie Dijkstra-Soudarissanane. Virtual visits: Ux evaluation of a photorealistic ar-based video communication tool. In *Proceedings of the 1st Workshop on Interactive eXtended Reality*, pages 69–75, 2022.

[118] Alexander Toet, Tina Mioch, Simon NB Gunkel, Omar Niamut, and Jan BF van Erp. Towards a multiscale qoe assessment of mediated social communication. *Quality and User Experience*, 7(1):4, 2022.

[119] Arthur Aron, Elaine N Aron, and Danny Smollan. Inclusion of other in the self scale and the structure of interpersonal closeness. *Journal of personality and social psychology*, 63(4):596, 1992.

[120] Martin Schrepp. User experience questionnaire handbook. *All you need to know to apply the UEQ successfully in your project*, 2015.

[121] John Beerends and Niels Neumann. Conversational quality assessment of advanced video conferencing systems. In *Presented at the 4th International Conference of the Acoustical Society of Nigeria*, 2020.

[122] Mel Slater. Immersion and the illusion of presence in virtual reality. *British Journal of Psychology*, 109(3):431–433, 2018.

[123] Jeremy N Bailenson and Nick Yee. Digital chameleons: Automatic assimilation of nonverbal gestures in immersive virtual environments. *Psychological science*, 16(10):814–819, 2005.

[124] Jim Blascovich and Jeremy Bailenson. *Infinite reality: Avatars, eternal life, new worlds, and the dawn of the virtual revolution.* William Morrow & Co, 2011.

[125] Jessica Outlaw and Beth Duckles. Why woman don't like social virtual reality, 2017.

[126] Juan C Granda, Pelayo Nuño, Francisco J Suárez, and Daniel F García. Overlay network based on webrtc for interactive multimedia communications. In *2015 International Conference on computer, information and telecommunication systems (CITS)*, pages 1–5. IEEE, 2015.

[127] Pelayo Nuño, Francisco G Bulnes, Juan C Granda, Francisco J Suárez, and Daniel F García. A scalable webrtc platform based on open technologies. In *2018 International Conference on Computer, Information and Telecommunication Systems (CITS)*, pages 1–5. IEEE, 2018.

[128] Stefano Petrangeli, Dries Pauwels, Jeroen van der Hooft, Tim Wauters, Filip De Turck, and Jürgen Slowack. Improving quality and scalability of webrtc video collaboration applications. In *Proceedings of the 9th ACM Multimedia Systems Conference*, pages 533–536, 2018.

[129] M. Westerlund and S. Wenger. Rtp topologies. RFC 7667, RFC Editor, November 2015.

[130] Christian Feldmann, Christopher Bulla, and Bastian Cellarius. Efficient stream-reassembling for video conferencing applications using tiles in hevc. In *Proc. of International Conferences on Advances in Multimedia (MMEDIA)*, pages 130–135, 2013.

[131] Michitaka Hirose, Tetsuro Ogi, and Toshio Yamada. Integrating live video for immersive environments. *IEEE MultiMedia*, 6(3):14–22, 1999.

[132] Tetsuro Ogi, Toshio Yamada, Ken Tamagawa, Makoto Kano, and Michitaka Hirose. Immersive telecommunication using stereo video avatar. In *Proceedings IEEE Virtual Reality 2001*, pages 45–51. IEEE, 2001.

[133] Peter Kauff and Oliver Schreer. An immersive 3d video-conferencing system using shared virtual team user environments. In *Proceedings of the 4th international conference on Collaborative virtual environments*, pages 105–112, 2002.

[134] Ralph Schroeder. *The social life of avatars: Presence and interaction in shared virtual environments*. Springer Science & Business Media, 2012.

[135] Steve Benford, Chris Greenhalgh, Tom Rodden, and James Pycock. Collaborative virtual environments. *Communications of the ACM*, 44(7):79–85, 2001.

[136] Masayuki Takemura and Yuichi Ohta. Generating high-definition facial video for shared mixed reality. In *MVA*, pages 422–425, 2005.

[137] Hao Li, Laura Trutoiu, Kyle Olszewski, Lingyu Wei, Tristan Trutna, Pei-Lun Hsieh, Aaron Nicholls, and Chongyang Ma. Facial performance sensing head-mounted display. *ACM Transactions on Graphics (ToG)*, 34(4):1–9, 2015.

[138] Xavier P Burgos-Artizzu, Julien Fleureau, Olivier Dumas, Thierry Tapie, François LeClerc, and Nicolas Mollet. Real-time expression-sensitive hmd face reconstruction. In *SIGGRAPH Asia 2015 Technical Briefs*, pages 1–4. 2015.

[139] Justus Thies, Michael Zollhöfer, Marc Stamminger, Christian Theobalt, and Matthias Nießner. Demo of facevr: real-time facial reenactment and eye gaze control in virtual reality. In *ACM SIGGRAPH 2017 Emerging Technologies*, pages 1–2. 2017.

[140] Christian Frueh, Avneesh Sud, and Vivek Kwatra. Headset removal for virtual and mixed reality. In *ACM SIGGRAPH 2017 Talks*, pages 1–2. 2017.

[141] Nels Numan, Frank Haar, and Pablo Cesar. Generative rgb-d face completion for head-mounted display removal. In *2021 IEEE Virtual Humans and Crowds for Immersive Environments (VHCIE)*. IEEE, IEEE, 03 2021.

[142] Gianluca Cernigliaro, Marc Martos, Mario Montagud, Amir Ansari, and Sergi Fernandez. Pc-mcu: Point cloud multipoint control unit for multi-user holoconferencing systems. In *Proceedings of the 30th ACM Workshop on Network and Operating Systems Support for Digital Audio and Video*, NOSSDAV '20, page 47–53, New York, NY, USA, 2020. Association for Computing Machinery.

[143] Jack Jansen, Shishir Subramanyam, Romain Bouqueau, Gianluca Cernigliaro, Marc Martos Cabré, Fernando Pérez, and Pablo Cesar. A pipeline for multiparty volumetric video conferencing: transmission of point clouds over low latency dash. In *Proceedings of the 11th ACM Multimedia Systems Conference*, pages 341–344, 2020.

[144] J. v. d. Hooft, M. T. Vega, T. Wauters, C. Timmerer, A. C. Begen, F. D. Turck, and R. Schatz. From capturing to rendering: Volumetric media delivery with six degrees of freedom. *IEEE Communications Magazine*, 58(10):49–55, 2020.

[145] Gregorij Kurillo and Ruzena Bajcsy. 3d teleimmersion for collaboration and interaction of geographically distributed users. *Virtual Reality*, 17(1):29–43, 2013.

[146] Zhenyu Yang, Bin Yu, Klara Nahrstedt, and Ruzena Bajscy. A multi-stream adaptation framework for bandwidth management in 3d tele-immersion. In *Proceedings of the 2006 international workshop on Network and operating systems support for digital audio and video*, pages 1–6, 2006.

[147] Jyh-Ming Lien, Gregorij Kurillo, and Ruzena Bajcsy. Multi-camera tele-immersion system with real-time model driven data compression. *The Visual Computer*, 26(1):3, 2010.

[148] Jason Leigh, Thomas A DeFanti, A Johnson, Maxine Brown, and D Sandin. Global tele-immersion: Better than being there. In *Proceedings of ICAT*, volume 97, pages 3–5, 1997.

[149] Renata M. Sheppard, Mahsa Kamali, Raoul Rivas, Morihiko Tamai, Zhenyu Yang, Wanmin Wu, and Klara Nahrstedt. Advancing interactive collaborative mediums through tele-immersive dance (ted): A symbiotic creativity and design environment for art and computer science. In *Proceedings of the 16th ACM International Conference on Multimedia*, MM '08, page 579–588, New York, NY, USA, 2008. Association for Computing Machinery.

[150] Zhenyu Yang, K. Nahrstedt, Yi Cui, Bin Yu, Jin Liang, Sang-hack Jung, and R. Bajscy. Teeve: the next generation architecture for tele-immersive environments. In *Seventh IEEE International Symposium on Multimedia (ISM'05)*, pages 8 pp.–, 2005.

[151] Tomislav Pejsa, Julian Kantor, Hrvoje Benko, Eyal Ofek, and Andrew Wilson. Room2room: Enabling life-size telepresence in a projected augmented reality environment. In *Proceedings of the 19th ACM conference on computer-supported cooperative work & social computing*, pages 1716–1725, 2016.

[152] Leonor Fermoselle, Simon Gunkel, Frank ter ter Haar, Sylvie Dijkstra-Soudarissanane, Alexander Toet, Omar Niamut, and Nanda van van der Stap. Let's get in touch! adding haptics to social vr. In *ACM International Conference on Interactive Media Experiences*, IMX '20, page 174–179, New York, NY, USA, 2020. Association for Computing Machinery.

[153] Alessandro Amirante, Tobia Castaldi, Lorenzo Miniero, and Simon Pietro Romano. Performance analysis of the janus webrtc gateway. In *Proceedings of the 1st Workshop*

on All-Web Real-Time Systems, AWeS '15, New York, NY, USA, 2015. Association for Computing Machinery.

[154] Demetrios Karis, Daniel Wildman, and Amir Mané. Improving remote collaboration with video conferencing and video portals. *Human–Computer Interaction*, 31(1):1–58, 2016.

[155] Jose Luis Rubio-Tamayo, Manuel Gertrudix Barrio, and Francisco García García. Immersive environments and virtual reality: Systematic review and advances in communication, interaction and simulation. *Multimodal Technologies and Interaction*, 1(4), 2017.

[156] Jie Li and Pablo Cesar. Chapter 22 - social virtual reality (vr) applications and user experiences. In Giuseppe Valenzise, Martin Alain, Emin Zerman, and Cagri Ozcinar, editors, *Immersive Video Technologies*, pages 609–648. Academic Press, 2023.

[157] Stefano Petrangeli, Dries Pauwels, Jeroen Van Der Hooft, Matúš Žiak, Jürgen Slowack, Tim Wauters, and Filip De Turck. A scalable webrtc-based framework for remote video collaboration applications. *Multimedia Tools and Applications*, 78:7419–7452, 2019.

[158] Henry Fuchs, Andrei State, and Jean-Charles Bazin. Immersive 3d telepresence. *Computer*, 47(7):46–52, 2014.

[159] Stephan Beck, Andre Kunert, Alexander Kulik, and Bernd Froehlich. Immersive group-to-group telepresence. *IEEE transactions on visualization and computer graphics*, 19(4):616–625, 2013.

[160] Jie Li. From immersive experiences to the metaverse: How can we engage more users? *Interactions*, 30(3):48–53, may 2023.

[161] Ju Shen, Po-Chang Su, Sen-ching Samson Cheung, and Jian Zhao. Virtual mirror rendering with stationary rgb-d cameras and stored 3-d background. *IEEE Transactions on Image Processing*, 22(9):3433–3448, 2013.

[162] Zhenyu Yang, Wanmin Wu, Klara Nahrstedt, Gregorij Kurillo, and Ruzena Bajcsy. Enabling multi-party 3d tele-immersive environments with viewcast. *ACM Transactions on Multimedia Computing, Communications, and Applications (TOMM)*, 6(2):1–30, 2010.

[163] Yang Xu, Chenguang Yu, Jingjiang Li, and Yong Liu. Video telephony for end-consumers: Measurement study of google+, ichat, and skype. In *Proceedings of the 2012 Internet Measurement Conference*, pages 371–384, 2012.

[164] Kyle MacMillan, Tarun Mangla, James Saxon, and Nick Feamster. Measuring the performance and network utilization of popular video conferencing applications. In *Proceedings of the 21st ACM Internet Measurement Conference*, pages 229–244, 2021.

[165] Gianluca Cernigliaro, Marc Martos, Mario Montagud, Amir Ansari, and Sergi Fernandez. Pc-mcu: Point cloud multipoint control unit for multi-user holoconferencing systems. In *Proceedings of the 30th ACM Workshop on Network and Operating Systems Support for Digital Audio and Video*, NOSSDAV '20, page 47–53, New York, NY, USA, 2020. Association for Computing Machinery.

[166] Debargha Mukherjee, Jingning Han, Jim Bankoski, Ronald Bultje, Adrian Grange, John Koleszar, Paul Wilkins, and Yaowu Xu. A technical overview of vp9—the latest open-source video codec. *SMPTE Motion Imaging Journal*, 124(1):44–54, 2015.

[167] Gary J Sullivan, Jens-Rainer Ohm, Woo-Jin Han, and Thomas Wiegand. Overview of the high efficiency video coding (hevc) standard. *IEEE Transactions on circuits and systems for video technology*, 22(12):1649–1668, 2012.

[168] Benjamin Bross, Ye-Kui Wang, Yan Ye, Shan Liu, Jianle Chen, Gary J Sullivan, and Jens-Rainer Ohm. Overview of the versatile video coding (vvc) standard and its applications. *IEEE Transactions on Circuits and Systems for Video Technology*, 31(10):3736–3764, 2021.

[169] Yue Chen, Debargha Murherjee, Jingning Han, Adrian Grange, Yaowu Xu, Zoe Liu, Sarah Parker, Cheng Chen, Hui Su, Urvang Joshi, et al. An overview of core coding tools in the av1 video codec. In *2018 picture coding symposium (PCS)*, pages 41–45. IEEE, 2018.

[170] Kiran Misra, Andrew Segall, Michael Horowitz, Shilin Xu, Arild Fuldseth, and Minhua Zhou. An overview of tiles in hevc. *IEEE journal of selected topics in signal processing*, 7(6):969–977, 2013.

[171] Peter Amon, Madhurani Sapre, and Andreas Hutter. Compressed domain stitching of hevc streams for video conferencing applications. In *2012 19th International Packet Video Workshop (PV)*, pages 36–40. IEEE, 2012.

[172] Yago Sanchez, Ralf Globisch, Thomas Schierl, and Thomas Wiegand. Low complexity cloud-video-mixing using hevc. In *2014 IEEE 11th Consumer Communications and Networking Conference (CCNC)*, pages 213–218. IEEE, 2014.

[173] Adrian Grange, Peter De Rivaz, and Jonathan Hunt. Vp9 bitstream & decoding process specification. *WebM Project*, 2016.

[174] Zhuoran Li, Zhengfang Duanmu, Wentao Liu, and Zhou Wang. Avc, hevc, vp9, avs2 or av1?—a comparative study of state-of-the-art video encoders on 4k videos. In *Image Analysis and Recognition: 16th International Conference, ICIAR 2019, Waterloo, ON, Canada, August 27–29, 2019, Proceedings, Part I 16*, pages 162–173. Springer, 2019.

[175] P.910 ITU-T RECOMMENDATION. Subjective video quality assessment methods for multimedia applications. 2022.

[176] Gonca Bakar, Riza Arda Kirmizioglu, and A Murat Tekalp. Motion-based rate adaptation in webrtc videoconferencing using scalable video coding. *IEEE Transactions on Multimedia*, 21(2):429–441, 2018.

[177] Simon Gunkel, Hans Stokking, Martin Prins, Omar Niamut, Ernestasia Siahaan, and Pablo Cesar. Experiencing virtual reality together: Social vr use case study. In *Proceedings of the 2018 ACM international conference on interactive experiences for TV and online video*, pages 233–238, 2018.

[178] Simon Gunkel, Martin Prins, Hans Stokking, and Omar Niamut. Webvr meets webrtc: Towards 360-degree social vr experiences. In *2017 IEEE Virtual Reality (VR)*, pages 457–458. IEEE, 2017.

[179] Éric Jamet. An eye-tracking study of cueing effects in multimedia learning. *Computers in Human Behavior*, 32:47–53, 03 2014.

[180] Richard Atkinson and R.M. Shiffrin. Human memory: A proposed system and its control processes. *The psychology of learning and motivation*, 2:89–195, 12 1968.

[181] Meg Morris, Elizabeth Ozanne, Kimberly Miller, Nick Santamaria, Alan Pearce, Catherine Said, and Brooke Adair. *Smart technologies for older people.* The University of Melbourne, 2012.

[182] Meg Morris, Brooke Adair, Kimberly Miller, Elizabeth Ozanne, Ralph Hampson, Alan Pearce, Nick Santamaria, L Viegas, Maureen Long, and Catherine Said. Smart-home technologies to assist older people to live well at home. *Journal of Aging Science*, 1:101, 01 2013.

[183] Blanka Klimova. *Elderly People and Their Use of Smart Technologies: Benefits and Limitations*, pages 405–412. 06 2016.

[184] Sarah Abdi, Luc de Witte, and Mark Hawley. Emerging technologies with potential care and support applications for older people: Review of gray literature. *JMIR Aging*, 3(2):e17286, Aug 2020.

[185] Scott Kelby. *Digital Photography Book, The: Part 1.* Peachpit Press, 2 edition, 2013.

[186] Dunja Vucic, Lea Skorin-Kapov, and Mirko Suznjevic. The impact of bandwidth limitations and video resolution size on qoe for webrtc-based mobile multi-party video conferencing. pages 59–63, 08 2016.

[187] Yang Xu, Chenguang Yu, Jingjiang Li, and Yong J. Liu. Video telephony for end-consumers: Measurement study of google+, ichat, and skype. *IEEE/ACM Transactions on Networking*, 22:826–839, 2014.

[188] Gunilla Berndtsson, Mats Folkesson, and Valentin Kulyk. Subjective quality assessment of video conferences and telemeetings. *2012 19th International Packet Video Workshop (PV)*, pages 25–30, 2012.

[189] Sasa Junuzovic, Kori Inkpen Quinn, Rajesh Hegde, and Zhengyou Zhang. Towards ideal window layouts for multi-party, gaze-aware desktop videoconferencing. In *Graphics Interface*, 2011.

[190] Dohyun Ahn, Youngnam Seo, Minkyung Kim, Joung Huem Kwon, Younbo Jung, Jungsun Ahn, and Doohwang Lee. The effects of actual human size display and stereoscopic presentation on users' sense of being together with and of psychological immersion in a virtual character. *Cyberpsychology, Behavior, and Social Networking*, 17(7):483–487, 2014.

[191] Alan B. Johnston and Daniel C. Burnett. *WebRTC: APIs and RTCWEB Protocols of the HTML5 Real-Time Web.* Digital Codex LLC, 2013.

[192] Ignacio Reimat, Evangelos Alexiou, Jack Jansen, Irene Viola, Shishir Subramanyam, and Pablo Cesar. Cwipc-sxr: Point cloud dynamic human dataset for social xr. In *Proceedings of the 12th ACM Multimedia Systems Conference*, pages 300–306, 2021.

[193] Anargyros Chatzitofis, Leonidas Saroglou, Prodromos Boutis, Petros Drakoulis, Nikolaos Zioulis, Shishir Subramanyam, Bart Kevelham, Caecilia Charbonnier, Pablo Cesar, Dimitrios Zarpalas, et al. Human4d: A human-centric multimodal dataset for motions and immersive media. *IEEE Access*, 8:176241–176262, 2020.

[194] Sanika Doolani, Callen Wessels, Varun Kanal, Christos Sevastopoulos, Ashish Jaiswal, Harish Nambiappan, and Fillia Makedon. A review of extended reality (xr) technologies for manufacturing training. *Technologies*, 8(4):77, 2020.

[195] Ashutosh Singla, Shuang Wang, Steve Göring, Rakesh Rao Ramachandra Rao, Irene Viola, Pablo Cesar, and Alexander Raake. Subjective quality evaluation of point clouds using remote testing. In *Proceedings of the 2nd International Workshop on Interactive EXtended Reality*, IXR '23, page 21–28, New York, NY, USA, 2023. Association for Computing Machinery.

[196] Xuemei Zhou, Evangelos Alexiou, Irene Viola, and Pablo Cesar. Pointpca+: Extending pointpca objective quality assessment metric. In *2023 IEEE International Conference on Image Processing Challenges and Workshops (ICIPCW)*, pages 1–5, 2023.

[197] Alexander Toet, Tina Mioch, Simon NB Gunkel, Camille Sallaberry, Jan BF van Erp, and Omar Niamut. Holistic quality assessment of mediated immersive multisensory social communication. In *International Conference on Virtual Reality and Augmented Reality (EuroVR)*, pages 209–215. Springer International Publishing Cham, 2020.

[198] Mario Montagud, Jie Li, Gianluca Cernigliaro, Abdallah El Ali, Sergi Fernández, and Pablo Cesar. Towards socialvr: evaluating a novel technology for watching videos together. *Virtual Reality*, 26(4):1593–1613, 2022.

[199] Jie Li, Shishir Subramanyam, Jack Jansen, Yanni Mei, Ignacio Reimat, Kinga Ławicka, and Pablo Cesar. Evaluating the user experience of a photorealistic social vr movie. In *2021 IEEE International Symposium on Mixed and Augmented Reality (ISMAR)*, pages 284–293, 2021.

[200] Giuseppe Valenzise, Martin Alain, Emin Zerman, and Cagri Ozcinar. *Immersive Video Technologies.* Academic Press, 2022.

[201] Hang Chu, Shugao Ma, Fernando De la Torre, Sanja Fidler, and Yaser Sheikh. Expressive telepresence via modular codec avatars. In *Computer Vision–ECCV 2020: 16th European Conference, Glasgow, UK, August 23–28, 2020, Proceedings, Part XII 16*, pages 330–345. Springer, 2020.

[202] Shugao Ma, Tomas Simon, Jason Saragih, Dawei Wang, Yuecheng Li, Fernando De La Torre, and Yaser Sheikh. Pixel codec avatars. In *Proceedings of the IEEE/CVF Conference on Computer Vision and Pattern Recognition*, pages 64–73, 2021.

[203] Sid Sharma. Ai can see clearly now: Gans take the jitters out of video calls. *The Official NVIDIA Blog*, 2020.

[204] Roshan Prabhakar, Shubham Chandak, Carina Chiu, Renee Liang, Huong Nguyen, Kedar Tatwawadi, and Tsachy Weissman. Reducing latency and bandwidth for video streaming using keypoint extraction and digital puppetry. *arXiv preprint arXiv:2011.03800*, 2020.

[205] Erin Pangilinan, Steve Lukas, and Vasanth Mohan. *Creating augmented and virtual realities: theory and practice for next-generation spatial computing*. " O'Reilly Media, Inc.", 2019.

[206] Jiangnan Xu, Konstantinos Papangelis, Garreth W Tigwell, Nicolas LaLone, Pengyuan Zhou, Michael Saker, Alan Chamberlain, John Dunham, Sanzida Mojib Luna, and David Schwartz. Spatial computing: Defining the vision for the future. 2024.

[207] Hakju Ryan Lee, Hyun-Koo Yang, Sungryeul Rhyu, Eric Yip, and Jaeyeon Song. Overview of 3gpp standardization for 5g ar/mr experiences using glasses-type devices. In *2022 IEEE International Symposium on Mixed and Augmented Reality Adjunct (ISMAR-Adjunct)*, pages 441–446. IEEE, 2022.

[208] 3GPP. 3rd generation partnership project; lte; 5g; media capabilities for augmented reality (3gpp ts 26.119 version 18.0.0 release 18). Technical Report TS 26.119 version 18.0.0 (2024-04-09), 3GPP, 2024.

[209] Shu Shi and Cheng-Hsin Hsu. A survey of interactive remote rendering systems. *ACM Computing Surveys (CSUR)*, 47(4):1–29, 2015.

[210] Shu Shi, Won J Jeon, Klara Nahrstedt, and Roy H Campbell. Real-time remote rendering of 3d video for mobile devices. In *Proceedings of the 17th ACM international conference on Multimedia*, pages 391–400, 2009.

[211] Ruizhi Cheng, Nan Wu, Songqing Chen, and Bo Han. Will metaverse be nextg internet? vision, hype, and reality. *IEEE Network*, 36(5):197–204, 2022.

Glossary

2D Two dimensional.

3D Three dimensional.

3GPP The 3rd Generation Partnership Project.

5G Fifth-generation technology standard for cellular networks (3GPP).

6G Sixth-generation technology standard for cellular networks (3GPP).

AI Artificial intelligence.

AMD Advanced Micro Devices, Inc..

API Application programming interface.

AR Augmented reality.

AV1 AOMedia Video 1 (AV1).

AVG Average.

BTVC Bitstream Tiled Video Compositor.

CAVE Cave automatic virtual environment.

CBR Constant bitrate.

CPU Central processing unit.

CQ Constant quality.

CUDA Compute Unified Device Architecture.

CV1 Oculus Rift consumer version 1.

CWI Centrum Wiskunde & Informatica.

DASH Dynamic Adaptive Streaming over HTTP.

DE Germany (ISO 3166-1 alpha-2 country code).

DIBR Depth-image-based rendering.

EMFT Fraunhofer EMFT is the research institute for microsystem and sensor technology.

ESA European Space Agency.

FGBG Foreground-Background extraction.

FoV Field of view.

FR France.

FVV Free Viewpoint Video.

G-PCC Geometry based Point Cloud Compression.

GB United Kingdom (ISO 3166-1 code).

GLSL OpenGL Shading Language.

glTF (glTF) Graphics Library Transmission Format or GL Transmission Format and formerly known as WebGL Transmissions Format or WebGL TF.

GOP Group of pictures.

GPU Graphics processing unit.

H-MSC-Q Holistic Mediated Social Communication Questionaire.

H.264 Advanced Video Coding (AVC).

H.265 High Efficiency Video Coding (HEVC).

H.266 Versatile Video Coding (VVC).

H2020 The Framework Programmes for Research and Technological Deve.

HD720 High-definition (HD) in 720p resolution of 1280 x 720 pixels.

HEVC High Efficiency Video Coding.

HL2 Mirosoft HoloLens 2.

HMD Head-mounted display.

HoloLens2 Microsoft HoloLens 2.

HSV HSV color space in Hue - Saturation - Value.

HTC HTC Corporation or High Tech Computer Corporation.

HTTP Hypertext Transfer Protocol.

IC Integrated circuit.

ICS Immersive communication system.

IOS Inclusion of Other in the Self.

IP Internet Protocol.

IR Infrared.

ISBN International Standard Book Number.

ITU International Telecommunication Union.

JPEG Joint Photographic Experts Group.

JPG JPG or JPEG - Joint Photographic Experts Group.

K4A Azure Kinect SDK.

KAIST Korea Advanced Institute of Science and Technology.

LZ4 LZ4 lossless data compression algorithm.

MacOS Apple Macintosh operating system.

MB Megabyte.

Mb Megabit.

MCU Multipoint control unit.

MD Mean absolute difference.

MINGW Minimalist GNU Compiler Collection (GCC) for Windows.

ML Machine learning.

MOD Modification.

MPEG Moving Picture Experts Group.

MR Mixed reality.

MS Teams Microsoft Team.

MSI Micro-Star International.

NFOV Narrow Field-of-View.

NIR Near-infrared.

NL Netherlands (ISO 3166-2 code).

NMQ Networked Minds questionnaire.

NVIDIA Nvidia Corporation.

OBJ (OBJ file format) a simple data-format that represents 3D geometry.

OBS Open Broadcaster Software (OBS Studio).

OpenCTM OpenCTM is a 3D geometry technology for storing triangle-bas.

OpenCV Open Source Computer Vision Library.

OpenGL Open Graphics Library.

OpenXR Open-source and royalty-free standard for access to virtual reality and augmented reality platforms and devices.

P2P Peer-to-peer.

PC Personal computer.

PSNR Peak signal-to-noise ratio.

PU Processing unit.

QIF Quality Influence Factors.

QoC Quality of communication.

QoE Quality of experience.

QoI Quality of interaction.

QoS Quality of service.

QR QR code (Quick Response code).

RabbitMQ Implementation of the Advanced Message Queuing Protocol.

RAM Random-access memory.

RCM Resources Consumption Metrics.

RDA Remote Data Access.

RGB RGB color model - red green blue.

RGBA RGBA color model - red green blue alpha.

RGBD RGBD color model - red green blue depth.

RQ Research question.

RTP Real-time Transport Protocol.

RTX Nvidia GeForce RTX.

RVL Run length encoding and Variable Length encoding.

SD Standard deviation.

SDK Software development kit.

SDP Session Description Protocol.

SFU Selective Forwarding Unit.

SI Spatial Information.

SMS Short Message/Messaging Service.

Socket.IO Socket.IO is an event-driven library for real-time web appli.

SteamVR Steam is a video game digital distribution service and store.

SVC Scalable video coding.

T-MCU Tiled-enabled Multipoint Control Unit.

TCP Transmission Control Protocol.

THREE.js JavaScript 3D Library.

TI Temporal Information.

TNO Netherlands Organisation for Applied Scientific Research.

ToF Time of flight.

TU Berlin Technische Universität Berlin.

TV Television.

TVM Time-varying mesh.

UEQ User Experience Questionnaire.

UX User experience.

V-PCC Video-based Point Cloud Compression.

VFX Visual effects.

VP8 An open and royalty-free video compression format.

VP9 An open and royalty-free video compression format.

VPT Tiled-based modified libvpx vp9 encoder.

VPX Reference libvpx vp9 encoder.

VQEG Video Quality Experts Group.

VR Virtual reality.

W3C World Wide Web Consortium.

WebGL Web Graphics Library.

WebM An audiovisual media file format.

WebRTC Web Real-Time Communication.

WebVR Web application VR device API.

WebXR Web application XR device API.

WMR Windows Mixed Reality.

XR Extended reality.

YUV Color model with a luma and two chroma components.

YUV420 YUV chroma subsampling grouping 4 pixels.

List of Publications

In the following is the list of references that are included in the dissertation.

1. Simon NB Gunkel, Hans M Stokking, Martin J Prins, Nanda van der Stap, Frank B ter Haar, and Omar A Niamut. Virtual reality conferencing: Multi-user immersive vr experiences on the web. In *Proceedings of the 9th ACM Multimedia Systems Conference*, pages 498–501, 2018

2. Simon NB Gunkel, Rick Hindriks, Karim M El Assal, Hans M Stokking, Sylvie Dijkstra-Soudarissanane, Frank ter Haar, and Omar Niamut. Vrcomm: an end-to-end web system for real-time photorealistic social vr communication. In *Proceedings of the 12th ACM multimedia systems conference*, pages 65–79, 2021

3. Sylvie Dijkstra-Soudarissanane, Simon NB Gunkel, and Véronne Reinders. Virtual visits: life-size immersive communication. In *Proceedings of the 13th ACM Multimedia Systems Conference*, pages 310–314, 2022

4. Simon N.B. Gunkel, Sylvie Dijkstra-Soudarissanane, Hans M. Stokking, and Omar Niamut. From 2d to 3d video conferencing: Modular rgb-d capture and reconstruction for interactive natural user representations in immersive extended reality (xr) communication. *Frontiers in Signal Processing*, 3, 2023

5. Simon NB Gunkel, Sylvie Dijkstra-Soudarissanane, and Omar Niamut. "you ar' right in front of me": Rgbd-based capture and rendering for remote training. In *Proceedings of the 14th Conference on ACM Multimedia Systems*, pages 307–311, 2023

6. Simon N.B. Gunkel, Rick Hindriks, Yonatan Shiferaw, Sylvie Dijkstra-Soudarissanane, and Omar Niamut. Vp9 bitstream-based tiled multipoint control unit: Scaling simultaneous rgbd user streams in an immersive 3d communication system. In *Proceedings of the 15th ACM Multimedia Systems Conference*, MMSys '24, page 23–33, 2024

Won best demo paper award.