A Pipeline for Multiparty Volumetric Video Conferencing: Transmission of Point Clouds over Low Latency DASH

Jack Jansen
Shishir Subramanyam
Jack.Jansen@cwi.nl
S.Subramanyam@cwi.nl
CWI
Amsterdam, the Netherlands

Romain Bouqueau
romain.bouqueau@motionspell.com
Motion Spell
Paris, France

Gianluca Cernigliaro
Marc Martos Cabré
gianluca.cernigliaro@i2cat.net
marc.martos@i2cat.net
i2Cat Foundation
Barcelona, Spain

Fernando Pérez
fernando.perez@themoderncultural.com
The Modern Cultural
Spain

Pablo Cesar
P.S.Cesar@cwi.nl
CWI
Amsterdam, the Netherlands
TU Delft
Delft, the Netherlands

ABSTRACT

The advent of affordable 3D capture and display hardware is making volumetric videoconferencing feasible. This technology increases the immersion of the participants, breaking the flat restriction of 2D screens, by allowing them to collaborate and interact in shared virtual reality spaces. In this paper we introduce the design and development of an architecture intended for volumetric videoconferencing that provides a highly realistic 3D representation of the participants, based on pointclouds. A pointcloud representation is suitable for real-time applications like video conferencing, due to its low-complexity and because it does not need a time consuming reconstruction process. As transport protocol we selected low latency DASH, due to its popularity and client-based adaptation mechanisms for tiling. This paper presents the architectural design, details the implementation, and provides some referential results. The demo will showcase the system in action, enabling volumetric videoconferencing using pointclouds.

CCS CONCEPTS

• **Information systems** → **Multimedia streaming**; • **Human-centered computing** → **Interaction paradigms**; • **Networks** → **Network protocols**.

KEYWORDS

pointclouds, low latency dash, social vr

ACM Reference Format:
Jack Jansen, Shishir Subramanyam, Romain Bouqueau, Gianluca Cernigliaro, Marc Martos Cabré, Fernando Pérez, and Pablo Cesar. 2020. A Pipeline for Multiparty Volumetric Video Conferencing: Transmission of Point Clouds over Low Latency DASH. In *11th ACM Multimedia Systems Conference (MMSys'20), June 8–11, 2020, Istanbul, Turkey*. ACM, New York, NY, USA, 4 pages. https://doi.org/10.1145/3339825.3393578

MMSys'20, June 8–11, 2020, Istanbul, Turkey

ACM ISBN 978-1-4503-6845-2/20/06.
https://doi.org/10.1145/3339825.3393578

1 INTRODUCTION

Recent advances in depth sensors, affordable head mounted displays and the computational power of commodity hardware have made it possible to create and render photorealistic reconstructions of humans in real time. This allows a more natural representation of participants in teleimmersive conferencing applications. Mekuria et al [6] conducted a subjective study to compare the user experience of photorealistic reconstruction of users and synthetic avatars in a teleimmersive environment. The authors found that reconstructions increase the presence as well as physical, emotional and user state recognition compared to synthetic avatars.

In this paper we present a low-latency dash based volumetric video conferencing system, as depicted in figure 1. The system allows up to four participants to meet in a shared virtual space, depicted at the top. Participants are captured with 1–4 cameras, and can view the experience either on a screen or through a head-mounted display for increased immersion.

The system described here was created within the scope of VR-together, an EU H2020 project to investigate the creation of end-to-end immersive social VR experiences. This version of the system is used in the second of 3 pilots within that project, with each subsequent pilot enabling increased immersiveness. The system has a number of features that fall outside the scope of this article, such as session creation and management and synchronous shared media playback.

We will demonstrate the volumetric and audio conferencing abilities of the system using a single camera and either a screen or a head mounted display.

2 CONTRIBUTIONS

The pipeline we present in this work has two main novel contributions: it uses pointclouds as the volumetric data representation and it uses MPEG-DASH as the transport mechanism.

Figure 1: Virtual space, studio setup, portable setup and participant viewpoints

2.1 Pointclouds

A pointcloud represents a 3D object as an unstructured collection of points with X, Y and Z coordinates plus additional attributes. In our case, the additional attributes are the RGB color values of the surface. Pointclouds are a simple representation that do not require much preprocessing, unlike other representations such as polygon meshes. This makes pointclouds well suited for realtime applications. Pointclouds do have a drawback: they tend to require a lot of storage and therefore transmission bandwidth.

We use the codec proposed by Mekuria et al [5] in an all intra configuration. This codec uses octree occupancy to represent geometry and projects the colors onto a 2D grid to use JPEG image compression. This configuration allows for low delay encoding and decoding making it suitable for real time applications. Point cloud compression has received significant research interest in recent years including an MPEG standardisation activity [10] we expect a new standard to be announced later this year, but the current state of the VPCC encoder does not seem suitable for real-time applications. We expect future releases to allow low delay encoding and decoding especially if hardware acceleration can be used to code the point cloud video streams.

2.2 MPEG-DASH

Most video conferencing solutions rely on RTP-based transport stacks such as WebRTC, but for this application we faced several challenges which have led us to using MPEG-DASH as transport mechanism. Based on [11] we wanted to apply chunked HTTP transfers and fragmented ISOBMFF/MP4 as a transport stack for live streaming.

MPEG-DASH is an adaptive bitrate streaming standard that usually leverages HTTP as a network protocol to deliver small chunks of data, typically a few seconds long. In [1] it is shown that in such case the inner latency of MPEG-DASH is the ISOBMFF fragment duration and that lowering the fragment duration does not lead to significant bitrate overhead, which makes performance comparable to WebRTC.

MPEG-DASH allows each consumer to select only the tiles it is interested in, at an appropriate resolution (and therefore bitrate). It also allows to easily scale the number of viewers using CDNs (Content Delivery Network i.e. HTTP replication servers) in the future.

MPEG-DASH makes agnostic transport easier than RTP, allowing for faster experiments, and metadata handling. The fact that MPEG-DASH is based on HTTP allows us to conduct remote experiments without dealing with NAT issues.

3 RELATED WORK

Volumetric video conferencing systems have been proposed in previous research. Kauff et al [4] describe a shared virtual table environment for conferencing among a small team using multiple video streams from a multi camera setup that is then MPEG-4 encoded and transmitted to other participants using RTP via IP. Mekuria et al [8][7] propose a 3-D immersive telepresence system using UDP/TCP multi-streaming systems to transmit mesh reconstructions in real time. Collet et al. [2] presents a system to create high quality streamable free viewpoint video by tracking meshes into subsequences that can then be streamed on demand. Qian et al. [9] propose Nebula, a DASH based streaming system to deliver volumetric video on demand to smartphones. Their system uses an edge server to transcode volumetric objects to 2D video that can then be efficiently transmitted to and decoded by smartphones. Wu et al [12] present a psychophysical approach to Color-plus-Depth Level of Detail for polygon mesh based 3D tele immersive video. Doumanoglou et al [3] propose a skeleton based approach to compress human time varying meshes by modifying the MPEG4 TFAN codec. In this work we use real time photo realistic point cloud reconstructions of users along with low latency DASH. To our knowledge this is the first work in point cloud volumetric video conferencing to use low latency DASH. Using a standards compliant DASH transport mechanism will help with system adoption. This will also allow us to make further optimizations in future with user and network adaptive transmission using independently decodable point cloud tiles.

4 ARCHITECTURE

The architecture of the pipeline is depicted in figure 2. The architecture has been designed and implemented in a way to allow easy replacement of modules, to allow us to experiment with different

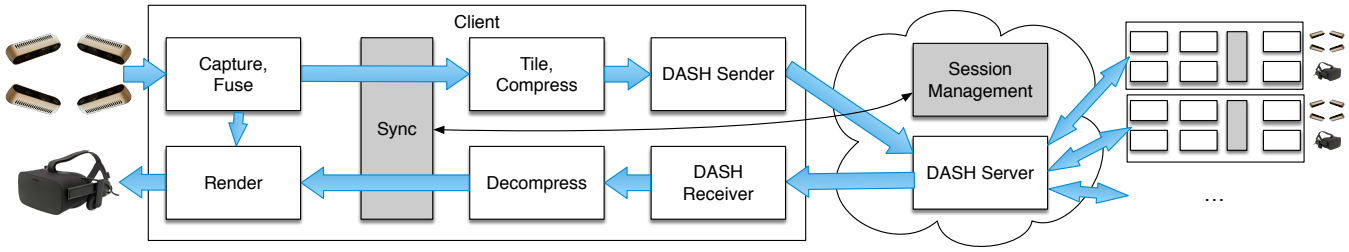

Figure 2: Architecture and data flow

capturers, encoders and transport pipelines and measure the effect of that on quality, delay and bandwidth consumption.

The architecture also allows for adaptation to different use cases and budgets (both money budgets and bandwidth budgets): 3D video conferencing will generally be applied in an immersive activity like a game or telepresence situation. These different use cases make it more important to allow adaptation for volumetric videoconferencing when compared to normal videoconferencing, which is generally used as a standalone application.

The architectural description in this section only pertains to the pointcloud pipeline: the actual implementation also contains an audio pipeline with a very similar overall structure and modules for session setup and management.

In each instance of our architecture there is a single transmission section (tile, compress, sender), and an independent reception section per other participant (receive, decompress).

The capture module interfaces to the cameras and captures RGB and Depth images, which are transformed using the intrinsic and extrinsic matrices and converted to pointcloud representation. There is an intrinsic 4x4 matrix for each camera, and it governs the transformation between RGB and D images. It is used to create a pointcloud per camera. The extrinsic matrices convert the per-camera pointclouds to world coordinates and therefore allow the fusing of the pointclouds. These matrices are determined in a calibration procedure once, after the cameras have been set up.

Pointclouds are fed to the tiling and compression module. Tiling splits a single pointcloud into a small number (1-8) of non overlapping pointclouds that together comprise the whole original pointcloud. Through tiling we enable viewers to save bandwidth by only downloading the relevant parts of a pointcloud, for example omitting invisible tiles. Subsequently each tile is downsampled into a small number (1-3) of resolutions. This downsampling is generally done through voxelization: overlaying a 3D cubic grid over the pointcloud space and replacing all points within each cube by a single voxel with the average position and color. The resulting pointclouds are fed into a lossy compressor which creates a linearized compressed data block for each tile at each resolution.

The DASH sender is instantiated with a description of the number of tiles and resolutions and creates a manifest file based on this information, which is uploaded to the DASH server. Subsequently, as compressed data blocks become available they are uploaded as DASH segments and the manifest file is periodically updated.

The DASH server stores the manifest file and the segments, and serves these to the receivers over HTTP or HTTPS. We have opted for a simple DASH server, not an MCU with mixing and fusing capabilities because this will give us a baseline against which to compare if we later want to investigate cloud-based composition of pointclouds.

The DASH receiver downloads the manifest file and makes the information on available tiles and compression levels available to the renderer. Based on view point, gaze direction and distance between viewer and subject the renderer selects the tiles it wants to receive and the quality level for each of those, and the DASH receiver starts requesting the segments for these tiles in parallel. This is done in an eager fashion: the DASH receiver will attempt to request segments as soon as they are expected to be available.

The per-tile compressed data blocks are fed to the decompressor which converts them back to pointclouds.

The synchronization module is responsible for synchronizing the tile streams and the audio, but it has not been implemented yet because in practice it has not been needed yet: in our current experimental setup the network delays turn out to be fairly stable so the synchronization between individual tiles is good enough. For audio adding a simple fixed delay is applied. We are aware of the fact that this is not generally true, so our architecture does cater for a synchronization module.

The renderer receives the self-view pointcloud from the capturer and a number of tiles per other participant and renders these in 3D space based on viewpoint and position of the participants. The mechanism for determining these positions as well as what other 3D content that may be rendered in the scene is out of scope for this paper.

5 IMPLEMENTATION

The client-side modules (with the exception of the renderer) have all been implemented in a portable language-agnostic way, so they are usable from C, C++, C#, Python and probably others. They use a memory model with clear ownership of memory areas and explicit freeing via the same module that allocated the memory. The renderer module is in reality not so modular as depicted in figure 2 because it is an integral part of the Unity application that implements the whole VR experience.

The client side modules are:

- Point Cloud Capture (cwipc_realsense2) implements the capturer. It uses the open source Intel librealsense2 to capture RGBD data from one or more cameras and reconstructs a point cloud frame.
- Codec (cwipc_codec) implements the encoder and decoder. It is based on the codec proposed by Mekuria et al. [5]

- Utilities (cwipc_util) is an auxiliary module to handle opaque pointcloud objects, so they can be passed efficiently between modules without copying or knowing the internal representation
- Transmission (bin2dash) handles the DASH transmission and is based on the GPAC toolset
- Reception (Signals Unity Bridge) handles DASH reception and is based on the GPAC toolset

The first three will be made available as open source. A distribution scheme for the latter two is still under consideration.

6 INITIAL MEASUREMENTS

We have done some initial measurements of the system using the setup depicted in figure 1. The studio setup consisted of an 8-core 4.2GHz i7, 32GB memory, with a NVIDIA GTX 1080Ti graphics card running Windows 10, 4 Realsense D415 cameras and an Oculus Rift. The portable setup consisted of a 12-core 2.6GHz i7, 16GB memory, with an NVIDIA GeForce RTX 2070 graphics card running Windows 10 and one Realsense D435. The two systems were co-located in Amsterdam, but the central services such as the DASH server were off-site in Paris, France. The network connection had a round-trip time of approximately 25ms and a bandwidth well over 100Mbps.

The studio setup captured 8–10 fused pointclouds per second of approximately 50Kpoints. These were compressed to 50–70Kbyte packets per pointcloud. Latency from studio system to the portable system (not including capture and display) was between 330ms and 450ms.

The portable setup captured 13–15 pointclouds per second of approximately 18Kpoints. These were compressed to 20–25Kbyte packets per pointcloud. Latency from portable system to the studio system (not including capture and display) was between 130ms and 160ms.

DASH segment used duration was 4 seconds, but as we outlined in section 2.2 the segment duration had very little impact on the average latency.

While not a rigid measurement these numbers suggest that the number of points per pointcloud is an important factor in determining the latency and frame rate of the system. Based on this, we expect to make further performance improvements.

7 FUTURE WORK

In the current implementation we have not used tiling, nor encoding in multiple quality levels. The infrastructure is in place, the client application will be adapted in the near future. This will allow us to experiment with different tiling strategies. In addition this will allow us to evaluate the quality of experience offered by various network and user adaptation strategies, as well as providing a more rigorous performance analysis.

We plan to implement other capturers, with higher quality and for different cameras and depth sensors such as the Azure Kinect.

We will also experiment with other codecs such as the upcoming MPEG standard for pointcloud compression [10].

ACKNOWLEDGMENTS

This work is partially funded by the European Commission H2020 program, under the grant agreement 762111, VRTogether, http://vrtogether.eu/. The authors would like to thank Francesca De Simone, Fons Kuijk, Irene Viola and the partners in the VRTogether project for their contributions.

REFERENCES

[1] Nassima Bouzakaria, Cyril Concolato, and Jean Le Feuvre. 2014. Overhead and performance of low latency live streaming using MPEG-DASH. In *IISA 2014, The 5th International Conference on Information, Intelligence, Systems and Applications*. IEEE, New York, New York, USA, 92–97. https://doi.org/10.1109/IISA.2014.6878732

[2] Alvaro Collet, Ming Chuang, Pat Sweeney, Don Gillett, Dennis Evseev, David Calabrese, Hugues Hoppe, Adam Kirk, and Steve Sullivan. 2015. High-Quality Streamable Free-Viewpoint Video. *ACM Trans. Graph.* 34, 4, Article 69 (July 2015), 13 pages. https://doi.org/10.1145/2766945

[3] Alexandros Doumanoglou, Dimitrios Alexiadis, Dimitrios Zarpalas, and Petros Daras. 2014. Towards Real-Time and Efficient Compression of Human Time-Varying Meshes. *IEEE Transactions on Circuits and Systems for Video Technology* 24 (12 2014), 2099–2116. https://doi.org/10.1109/TCSVT.2014.2319631

[4] Peter Kauff and Oliver Schreer. 2002. An Immersive 3D Video-Conferencing System Using Shared Virtual Team User Environments. In *Proceedings of the 4th International Conference on Collaborative Virtual Environments* (Bonn, Germany) *(CVE '02)*. Association for Computing Machinery, New York, NY, USA, 105–112. https://doi.org/10.1145/571878.571895

[5] Rufael Mekuria, Kees Blom, and Pablo Cesar. 2017. Design, Implementation, and Evaluation of a Point Cloud Codec for Tele-Immersive Video. *IEEE Transactions on Circuits and Systems for Video Technology* 27, 4 (apr 2017), 828–842. https://doi.org/10.1109/TCSVT.2016.2543039

[6] Rufael Mekuria, Pablo Cesar, Ioannis Doumanis, and Antonella Frisiello. 2015. Objective and subjective quality assessment of geometry compression of reconstructed 3D humans in a 3D virtual room. 95991M. https://doi.org/10.1117/12.2203312

[7] Rufael Mekuria, Antonella Frisiello, Marco Pasin, and Pablo Cesar. 2015. Network Support for Social 3-D Immersive Tele-Presence with Highly Realistic Natural and Synthetic Avatar Users. In *Proceedings of the 7th ACM International Workshop on Massively Multiuser Virtual Environments* (Portland, Oregon) *(MMVE '15)*. Association for Computing Machinery, New York, NY, USA, 19–24. https://doi.org/10.1145/2723695.2728605

[8] Rufael Mekuria, Michele Sanna, Stefano Asioli, Ebroul Izquierdo, Dick C. A. Bulterman, and Pablo Cesar. 2013. A 3D Tele-Immersion System Based on Live Captured Mesh Geometry. In *Proceedings of the 4th ACM Multimedia Systems Conference* (Oslo, Norway) *(MMSys '13)*. Association for Computing Machinery, New York, NY, USA, 24–35. https://doi.org/10.1145/2483977.2483980

[9] Feng Qian, Bo Han, Jarrell Pair, and Vijay Gopalakrishnan. 2019. Toward Practical Volumetric Video Streaming on Commodity Smartphones. In *Proceedings of the 20th International Workshop on Mobile Computing Systems and Applications* (Santa Cruz, CA, USA) *(HotMobile '19)*. Association for Computing Machinery, New York, NY, USA, 135–140. https://doi.org/10.1145/3301293.3302358

[10] 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. 2019. Emerging MPEG Standards for Point Cloud Compression. *IEEE Journal on Emerging and Selected Topics in Circuits and Systems* 9, 1 (March 2019), 133–148. https://doi.org/10.1109/JETCAS.2018.2885981

[11] Viswanathan Swaminathan and Sheng Wei. 2011. Low latency live video streaming using HTTP chunked encoding. In *2011 IEEE 13th International Workshop on Multimedia Signal Processing*. IEEE, New York, New York, USA, 1–6. https://doi.org/10.1109/MMSP.2011.6093825

[12] Wanmin Wu, Ahsan Arefin, Gregorij Kurillo, Pooja Agarwal, Klara Nahrstedt, and Ruzena Bajcsy. 2011. Color-plus-depth level-of-detail in 3D tele-immersive video: A psychophysical approach. *MM'11 - Proceedings of the 2011 ACM Multimedia Conference and Co-Located Workshops*, 13–22. https://doi.org/10.1145/2072298.2072302