Scalable MDC-Based Volumetric Video Delivery for Real-Time One-to-Many WebRTC Conferencing

Anonymous
Anonymous@Anonymous
Anonymous
Anonymous, Anonymous

ABSTRACT

The production and consumption of video content has become a staple in the current day and age. With the rise of virtual reality (VR), users are now looking for immersive, interactive experiences which combine the classic video applications, such as conferencing or digital concerts, with newer technologies. By going beyond 2D video into a 360 degree experience the first step was made. However, a 360 degree video offers only rotational movement, making interaction with the environment difficult. Fully immersive 3D content formats, such as light fields and volumetric video, aspire to go further by enabling six degrees-of-freedom (6DoF), allowing both rotational and positional freedom. Nevertheless, the adoption of immersive video capturing and rendering methods has been hindered by their substantial bandwidth and computational requirements, rendering them in most cases impractical for low latency applications. Several efforts have been made to alleviate these problems by introducing specialized compression algorithms and by utilizing existing 2D adaptation methods to adapt the quality based on the user's available bandwidth. However, even though these methods improve the quality of experience (QoE) and bandwidth limitations, they still suffer from high latency which makes real-time interaction unfeasible. To address this issue, we present a novel, open source [3], one-to-many streaming architecture using point cloud-based volumetric video. To reduce the bandwidth requirements, we utilize the Draco codec to compress the point clouds before they are transmitted using WebRTC which ensures low latency, enabling the streaming of real-time 6DoF interactive volumetric video. Content is adapted by employing a multiple description coding (MDC) strategy which combines sampled point cloud descriptions based on the estimated bandwidth returned by the Google congestion control (GCC) algorithm. MDC encoding scales more easily to a larger number of users compared to performing individual encoding. Our proposed solution achieves similar real-time latency for both three and nine clients (163 ms and 166 ms), which is 9% and 19% lower compared to individual encoding. The MDC-based approach, using three workers, achieves similar visual quality compared to a per client encoding solution, using five worker threads, and increased quality when the number of clients is greater than 20.

MMSys'24, April 2024, Bari, Italy

ACM ISBN 978-x-xxxx-xxxx-x/YY/MM…$15.00
https://doi.org/XXXXXXX.XXXXXXX

CCS CONCEPTS

• **Information systems** → **Multimedia streaming**; • **Human-centered computing** → *Virtual reality*.

KEYWORDS

Volumetric Video, Adaptive Streaming, WebRTC, Virtual Conferencing, Multiple Description Coding, Virtual Reality

ACM Reference Format:
Anonymous. 2024. Scalable MDC-Based Volumetric Video Delivery for Real-Time One-to-Many WebRTC Conferencing. In *Proceedings of ACM MMSys (MMSys'24).* ACM, Bari, Italy, 11 pages. https://doi.org/XXXXXXX.XXXXXXX

1 INTRODUCTION

The increasing demand for remote communication tools such as Microsoft Teams and Zoom has garnered increased attention in recent times [23]. Nonetheless, these platforms provide a restricted interactive experience, as they only allow users to view other participants through two-dimensional imagery. Proposals have been made for immersive six degrees-of-freedom (6DoF) video implementations which allows users to interact with both rotational and positional freedom [1]. In the context of these applications, a distinction should be made based on the interaction requirements of the users. In the scenario of one-on-one interactions, such as video calls, a major requirement is having the lowest possible latency to enable real-time interaction and prevent discomfort while talking to the other user [37]. For virtual conferences, having many senders and receivers, the additional demand of being able to adapt the video content for a large number of users while still maintaining low latency arises [26]. One-to-many applications, such as a virtual classroom in the context of education or a virtual concert in the context of entertainment, represent a hybrid category, wherein, instead of multiple participants generating content, a single peer serves as the content source, thereby simplifying the system design required to ensure low latency. Choosing the correct form of communication is important to ensure that the interactivity requirements are satisfied in addition to having an acceptable bitrate and latency. Several formats exist to facilitate the use of 6DoF content for immersive applications, which are commonly rendered and displayed on a head-mounted display (HMD). These formats may be categorized in two classes of immersive 6DoF video, image-based or volumetric video [47]. Image-based video techniques, such as light fields, leverage a large quantity of images to generate a new rendered image based on the viewing angle of the user. Capturing is done either by a single camera with a multitude of lenses, known as a lenslet camera, or by utilizing an advanced camera setup using tens or hundreds of cameras [11]. The choice of camera setup depends on the positional requirements of the application The necessity of requiring a large quantity of cameras arises from the need to generate any possible combination of position and tilt while watching the video [38]. In contrast, volumetric video relies on the utilization of three-dimensional representations, such as meshes and point clouds, to depict objects within a 3D environment. Meshes are characterized by vertices interconnected by edges in order to form the polygons that are used to render the object. Opposed to point clouds which exclusively consist of vertices. Furthermore, instead of using texture mapping, a point cloud uses a color attribute for each point [9]. These representations are subsequently rendered in a virtual scene, allowing them to be viewed from different angles and positions. Similar to image-based video, a specialized camera setup is required to capture the 3D representation of the object. In the context of point clouds, several cameras are precisely positioned around the object to capture it from different angles. This setup produces multiple distinct point clouds which are subsequently stitched together to remove the duplicate points and retain a singular object as the final result [2]. Compared to light field video, this approach substantially reduces the necessary bandwidth for transmission, as it is no longer required to maintain a large collection of images for each frame; instead, it suffices to transfer a single 3D object. Furthermore, the computational requirements associated with rendering volumetric video are significantly lower compared to image-based solutions, which is a necessity for enabling real-time rendering and interaction at the client side.

Despite the substantial reduction in required bandwidth, the resulting bitrates are often still too high for contemporary network infrastructures. The 8i point cloud dataset [16] consists of multiple high resolution point cloud videos, each of which demands a considerable data rate of up to 5.2 Gb/s which can only be used in very specialized environments with dedicated ultra-high bandwidth connections [47]. Additional mechanisms, such as compression or adaptation, are imperative to allow the use of such immersive content by a variety of users with differing network conditions and bandwidth restrictions. HMDs exhibit the ability to track both the position and field-of-view (FoV) through either internal sensors contained within the headset, which is called inside-out tracking, or by utilizing external devices strategically positioned within the user's environment, which is called outside-in tracking [21]. Leveraging FoV and positional data, it becomes feasible to adapt video content dynamically, rendering only the important details of the video resulting in significantly lower bandwidth usage.

A major component in ensuring real-time communication is the choice of transport protocol. TCP-based solutions such as low-latency HLS (LL-HLS) and low-latency DASH (LL-DASH) do not offer the real-time latency required for one-to-many and many-to-many virtual conferencing, because of the overhead introduced by the forced reliable delivery mechanisms such as acknowledgments, flow control, retransmissions and packet reordering. UDP-based solutions, such as WebRTC, aim for real-time performance by trading TCP features such as guaranteed delivery for a significantly lower latency [7].

This paper proposes a novel end-to-end pipeline designed for one-to-many applications, containing the following contributions:

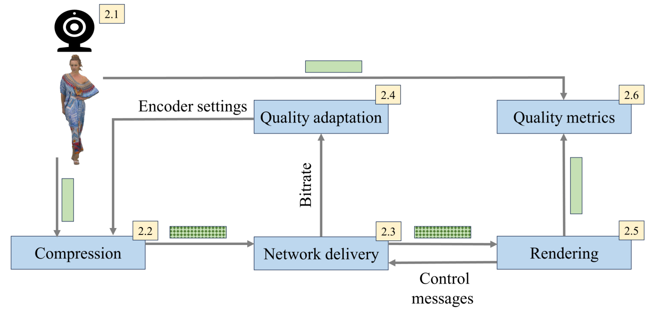


Figure 1: A volumetric video streaming architecture.

- A scalable multiple description coding (MDC) quality adaptation solution, allowing for higher quality streaming for a large number of clients ($n > 20$) when compared to a per client encoding solution
- A capture-to-render pipeline that employs WebRTC, which was adapted to support point cloud streaming, supporting real-time communication with an average capture-to-display latency of 163 ms
- An adaptive bitrate allocation algorithm, ensuring that each client receives a quality representation tailored to their network conditions and position within the virtual environment
- A comparative analysis between our proposed MDC-based approach and an encoding solution optimized per client, evaluating their respective performance in terms of quality and scalability

The remainder of the paper is organized as follows. Section 2 provides an overview of the current state of the art concerning volumetric video streaming. Section 3 introduces our proposed approach to facilitate the adaptive streaming of volumetric video. Section 4 presents our evaluation methodology followed by the results and discussion of our experiments. Finally, Section 5 concludes the paper with a brief overview of the most important results and potential future work.

2 RELATED WORK

This section presents an overview of the literature related to the components required to establish a real-time one-to-many volumetric video streaming architecture, as illustrated in Figure 1. First, we describe the possible volumetric video representations and capturing methods, followed by an overview of the state-of-the-art point cloud codecs. For each of these codecs we discuss the advantages and disadvantages concerning their application in an end-to-end pipeline. We then explain low-latency delivery mechanisms, quality adaptation and quality metrics in the context of both traditional video and volumetric video.

2.1 Volumetric Video and Capturing Methods

Both meshes and point clouds are used as a source of immersive video content, each having their own advantages and disadvantages. Meshes leverage certain graphics pipeline features and optimizations, such as anisotropic filtering, to increase their visual quality and performance [49]. However, point clouds employ better, optimized culling and tiling algorithms due to relying purely on vertices [12]. Additionally, the capturing process is more simplistic when utilizing point clouds, since the captured points and their color attribute can be used directly. In contract, meshes require the captured points to be converted into a geometric topology before rendering, this process is time consuming and negatively impacts the overall latency of the system [15]. For the remainder of this paper, the focus will be on the usage of point clouds as a source of volumetric video.

LiDAR cameras use reflected lasers to attain a highly accurate 3D representation of a large area at the cost of increased power consumption [18]. In contrast, depth-based systems employ a stereoscopic sensor to estimate the depth for each pixel of a depth image [29]. For volumetric video it is common to use depth cameras due to their lower cost and power consumption.

2.2 Point Cloud Compression and Codecs

Point cloud compression significantly reduces the required bandwidth, two categories of compression codecs can be distinguished: projection-based and geometric-based codecs. Projection-based codecs, such as V-PCC, project the point cloud on a 2D plane, allowing existing 2D video codecs to be used on the projection plane. Contrary, geometric codecs, such as Draco or G-PCC, are designed to encode the point cloud as a 3D structure [22].

V-PCC employs 2D projections of the point cloud to create an image containing the different projected views. Subsequently, this image is encoded using existing video codecs [22]. This method achieves a high compression ratio at the cost of non-real-time coding speeds [6].

In contrast to V-PCC, geometric encoders utilize 3D data structures, such as octrees and kd-tree, to encode the point cloud. Of the geometric encoders only Draco achieves real-time encoding when using large point clouds of more than 750,000 points [6].

2.3 Low-Latency Delivery

Given that volumetric video is fundamentally an extension of conventional video, it is feasible to adapt existing 2D video streaming solutions to incorporate features required to support volumetric video. Typically, HTTP-based implementations are considered for their reliable transmissions, which are a prerequisite for most applications [17]. HTTP adaptive streaming (HAS) represents a widely adopted approach for achieving adaptive 2D video playback. HAS-based streaming formats, such as DASH and HLS, divide video content into segments which are encoded at multiple quality representations at varying target bitrates [40]. This process introduces a latency between 5 and 18 seconds [48], making it unsuitable for real-time streaming. However, even the low-latency variants, such as LL-DASH and LL-HLS, are only able to achieve a latency between 1 and 5 seconds [48], which is still inadequate [10]. The aforementioned streaming formats utilize TCP, which further increases latency due to overhead created by TCP mechanisms, such as the TCP handshake or forced reliability [13]. Web real-time communication (WebRTC) is a UDP-based solution, designed specifically to enable real-time interactive communication [28]. Compared to other streaming formats, WebRTC achieves a sub one second delay [48]. Although WebRTC was designed as a peer-based solution, which presents challenges in the context of scalability, several server-client architectures have been designed to allow for large scale low-latency transmissions. Architectures such as multipoint control unit (MCU) and selective forwarding unit (SFU) employ a central server responsible for adapting the quality for each user based on their bandwidth. For these reasons, we will use WebRTC in the context of a server-client architecture to enable low-latency adaptive streaming in a multi-client environment.

2.4 Quality Adaptation

The concept of quality adaptation is a well-established practice in conventional streaming to ensure smooth video playback without packet loss [36]. Most encoders generate several quality representations by supplying varying target bitrates [8]. In the context of a one-to-one scenario, the provided bitrate can be an estimation of the available bandwidth for the user. However, for a larger number of users, it becomes unfeasible to have individual encoding, rather, several default bitrates are carefully chosen to ensure a low latency and acceptable quality video stream for each user. Alternatively, rather than employing separate quality levels, a MDC approach can be adopted. With MDC-based adaptation, the video content is encoded in individual descriptions that are combined to achieve varying levels of quality. While these aforementioned adaptation methods have been designed for 2D video, certain aspects can adapted and integrated into immersive video scenarios.

In volumetric video, a user is able to alter his viewpoint, causing objects to fall out of the current viewpoint and making it inefficient to continue transmitting these objects. Van der Hooft et al. [46] encode each point cloud object in the scene into several quality representations. A quality adaptation algorithm is proposed that uses the FoV and position of the user to determine what quality representation for each object has to be transmitted to achieve optimal quality for the available bandwidth. However, a DASH-based approach is proposed, where the encoded point clouds are concatenated into segments, making it unsuitable for real-time communication. An MDC-based approach alleviates this problem, as it allows for the creation of more quality levels, with less processing time. However, currently no MDC-based volumetric video streaming solution utilizing WebRTC exists.

To adapt the quality an accurate estimation of the bandwidth is required. Compared to TCP, UDP inherently lacks flow control and a bandwidth estimation mechanism. Commonly, bandwidth estimation is implemented at the application layer. Congestion controllers, such as Google congestion control (GCC) [14] and Self-Clocked Rate Adaptation for Multimedia (SCReAM) [27], are implemented in the multimedia application and require an exchange of control messages between the sender and receiver in order to estimate the available bandwidth. For this reason, we will utilize GCC to implement bandwidth estimation in WebRTC.

2.5 Rendering of Volumetric Video

Most rendering solutions are optimized to efficiently render meshes. Meshes use the triangle primitives of graphics application programming interfaces (APIs) to render their objects efficiently. Schütz et al. [39] use point primitives to render the point cloud. Additionally, the size of the points is changed to reduce the number of gaps, increasing the user-perceived quality.

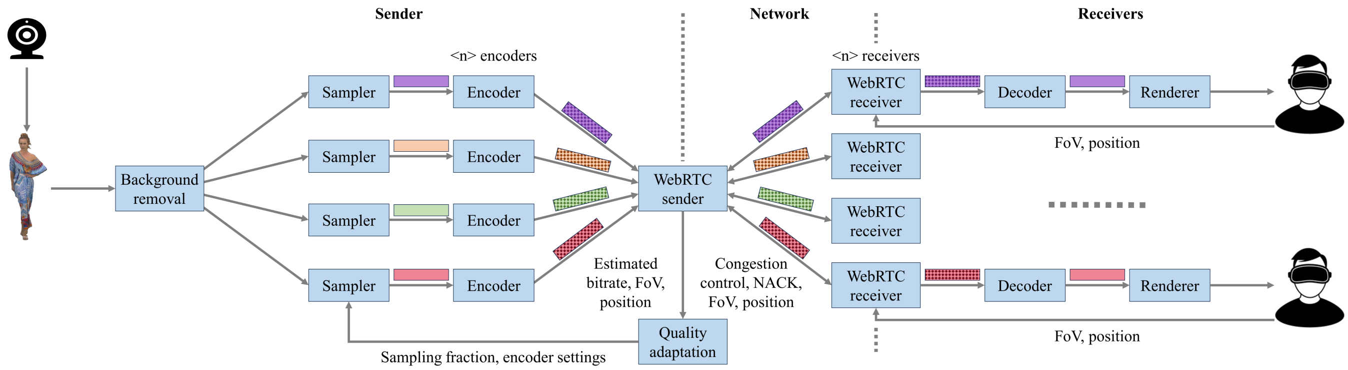


Figure 2: The system architecture for the individually encoded one-to-many volumetric video streaming solution.

2.6 Quality Metrics

The quality of a volumetric video is contingent on a multitude of factors. In real-time scenarios, the frame rate might be limited by bandwidth constraints or due to the latency introduced by the codec, which negatively impacts the user's experience [51]. The assessment of objective quality can be conducted through peak signal-to-noise ratio (PSNR) metrics, which are categorized into two types. Tian et al. [41] introduce multiple geometric metrics which assess the deformation of the mesh or point cloud relative to the original object, independent of potential viewports. In contrast, Torlig et al. propose a projection-based approach. This method uses multiple angles to project the point cloud onto a 2D image, enabling the use of existing 2D PSNR metrics. Both of the mentioned PSNR types suffer from limitation of not being able to prioritize sections of the object which draw more attention compared to others, such as the hands [50], thereby posing challenges in objectively approximating user-perceived quality.

Video Multi-Method Assessment Fusion (VMAF) is a video metric that combines the scores from multiple video assessment metrics in order to produce a single metric, ranging from 0 to 100, that better correlates to the user-perceived quality [35]. For this reason, VMAF will be used in this paper to assess the impact of the proposed approach on the visual quality of the volumetric video.

3 PROPOSED APPROACH

This section provides an overview of the proposed one-to-many MDC-based volumetric video streaming architecture. Our proposed architecture is an extension of a simple individually encoded architecture, an example of which is depicted on Figure 2. The goal of our architecture is to solve the problems that occur when employing the architecture illustrated on Figure 2 in terms of scalability. In this section, we describe the architectural components, their design choices and innovative methods in detail.

3.1 Depth Camera Capturing

For the acquisition of point cloud data, a single capturing source is assumed to capture a frontal perspective of an individual. Utilizing the depth frame, the positions of each point in the cloud are determined. The color frame is used to map an RGB value to the corresponding points

3.2 Preprocessing

Our primary focus is on the individual subject, and not on the (static) surrounding environment. Thus, a rudimentary distance filter will be applied to retain points that are too far away from the camera. This filter significantly reduces the number of points, allowing the available bandwidth to be used more efficiently.

In specific scenarios, a lower quality or sampled point cloud might be sufficient compared to the version with the full resolution. In particular, this occurs when viewing distant objects, where it is more difficult to distinguish the finer details [31]. To streamline this process, we employ sampling techniques to generate varying levels of point cloud quality.

In order to prevent inter-frame dependencies, the frames are exclusively encoded through intra-coding. This ensures that each frame is able to be decoded independently, leading to an increased quality when frames need to be dropped in the case of packet loss.

3.3 Quality Adaptation

The initial step of our quality adaptation algorithm is the creation of diverse quality representations. To facilitate this process, we propose two distinct methods that are able to create these representations. The objective of these methods is to produce a quality representation for each user, while simultaneously ensuring that the preprocessing time remains below the inter-frame time for real-timeliness. Both methods employ the principle of sampling, which reduces the number of points used in the encoder. We propose to use a uniform random sampler, as this ensures a balance between the sampling time and resulting quality [6].

The first method generates a unique quality representation for each connected client. This is accomplished by estimating the maximum achievable sampling rate, using the available bandwidth together with the frame size to calculate the required compression factor. This compression factor is used together with the fitted polynomial in Figure 4 to estimate the sampling factor. This polynomial was created by using a random subset (n=600) of our experimentation point cloud sequence, for which each frame of this subset was repeatably sampled across the range of 10% to 100% of the original point clouds before encoding. The used polynomial depends on the resolution of the encoded point clouds, as a more dense point cloud impacts the resulting compression ratios. Figure 2 illustrates the integration of this method into a WebRTC streaming pipeline. A

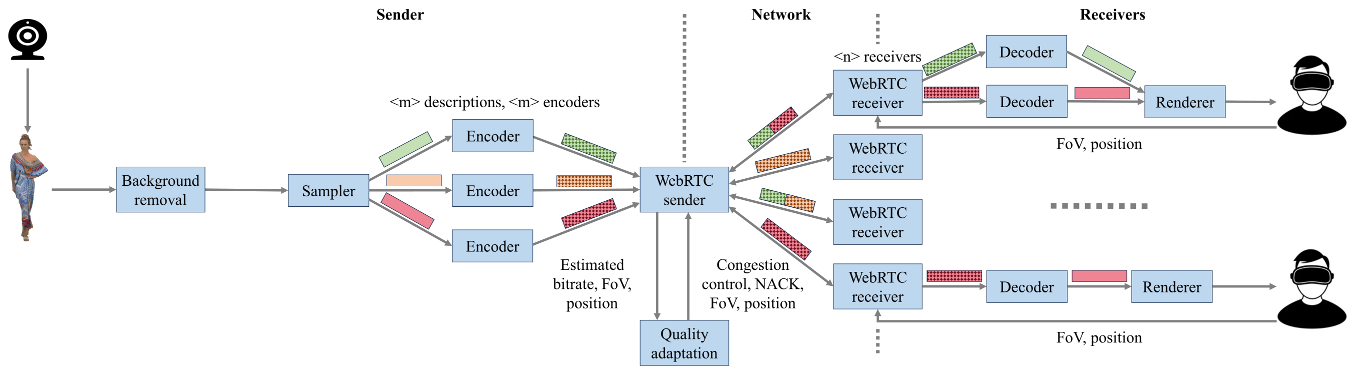


Figure 3: The system architecture for the proposed MDC-based one-to-many volumetric video streaming solution, requiring a fixed number of encoders based on the amount of descriptions, the number of local decoders for each client scales with the number of received descriptions.

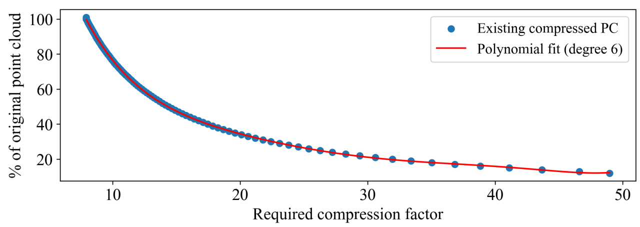


Figure 4: The usage of a fitted polynomial together with the required compression factor allows to determine the required sampling rate.

substantial disadvantage of this method is the number of required encoders being equal to the number of clients. If we want each client to have a similar preprocessing latency we are required to scale the number of workers, which is limited by the available computational resources. Additionally, updating the architecture to support a many-to-many scenario would require additional computational resources, further restricting the maximum number of clients.

The second method, which is shown in Figure 3, segments the point cloud in n descriptions, each containing distinct points. The combination of these descriptions permits access to additional quality representations. Consequently, this approach facilitates the streaming of $2^n - 1$ possible combinations of quality representations. To generate the available descriptions, three approaches are suggested:

- **Percentage:** Use n fixed percentages to sample the point cloud, creating n descriptions that each contain a fraction p_i of the points with i in $1, ..., n$ which corresponds with the respective percentage
- **Fixed Size:** Limit the number of points to x, with x being an expected lower limit, by randomly sampling x points. Afterwards use the first approach to create n descriptions
- **Fixed Bitrate:** Leverage the polynomial, illustrated in Figure 4, to estimate n sampling percentages necessary for creating n descriptions with varying bitrate targets

Using Algorithm 1 we first calculate the rotation matrix and viewing angles by utilizing the yaw, pitch and roll of the camera. Following this we calculate the distance of the user to the streamed object. Subsequently we use the viewing angle, FoV and distance

Algorithm 1 Bitrate allocation using FoV and position to calculate the transmitted descriptions.

$R \leftarrow calculateRotationMatrix(yaw, pitch, roll)$
$viewAngles \leftarrow calculateAngles(R)$
$distance \leftarrow getDistanceToUser(userPos, clientPos)$
$category \leftarrow getClientCategory(fov, viewAngles, distance)$
$allowedDescriptions \leftarrow getAllowedDescriptions(category)$
for each $description \in allowedDescriptions$ **do**
  if $bitrate >= getDescriptionSize(description)$ **then**
    $bitrate \leftarrow bitrate - getDescriptionSize(description)$
    $addDescriptionToFrame(description)$
  end if
end for

Figure 5: Quality category is assigned based on distance and field of view.

to impose restrictions on the maximum achievable quality that can be transmitted to the client. Figure 5 illustrates the working of this mechanism. The high-quality representations will only be available for transmission to the user if the object is close and directly in the FoV of the user. Furthermore, if the object is outside of the FoV, or too far away from the user, no frame will be transmitted. Consequently, we are able to calculate the allowed descriptions, of which an example implementation is shown in Table 1. For individual encoding, Table 1 shows the maximum allowed sampling rate and, for the MDC approaches it indicates the allowed combinations of descriptions. Finally, we iterate over the allowed descriptions, starting from the highest quality description, and validate if there is enough remaining bandwidth to add the description.

3.4 WebRTC-Based Delivery

As we strive for the lowest possible latency, we utilize WebRTC to ensure real-time communication between capturing and rendering

Table 1: Example implementation of the possible quality levels of MDC encoding through the combination of description (higher description ID = greater quality).

Quality Category	Description IDs Used	% of Points in Cloud
Low Quality	1	~15%
Medium Quality	2	~25%
	1, 2	~40%
High Quality	3	~60%
	1, 3	~70%
	1, 2	~85%
	1, 2, 3	~100%

components. Our pipeline is focused on a one-to-many scenario, which results in a central server at the capturing side that is connected to each client. The central server is based on MCU and SFU architectures, sharing similar responsibilities such as quality adaptation. Contrary to MCU and SFU, the central server is uni-directional and only transmits frames to the client. The clients connect to the server using a signaling algorithm. WebRTC does not specify how the signaling process should be implemented. However, recent efforts have been made to standardize signaling, to allow for easier integration of different WebRTC-based systems. Prior to streaming the video, an exchange of the session description protocol (SDP) of both the server and the client takes place. These SDP messages alert the other application of the available network routes, as well as the supported codecs. In the event a requested codec is not supported by the the other side of the connection, a back-off occurs and the connection is terminated.

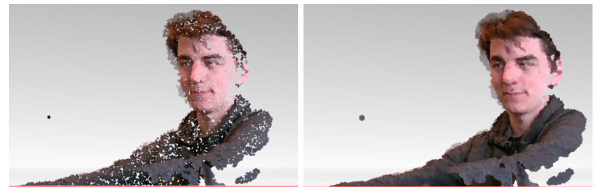
Furthermore, as a measure to address potential packet loss, we opt for the usage of negative acknowledgments (NACKs) to transmit lost packets. The utilization of NACK packets serves the purpose of ensuring that frames remain decodable in the event of packet loss, assuming the packet loss remains within a certain margin. Currently, no partial decode and rendering is supported by the architecture, requiring the full frame to be received. Compared to ACK control messages, NACK packets are only sent for packets that were not received by the client, lowering the overhead of the control messages. Given that WebRTC rearranges packets to achieve sequential ordering, NACK packets are commonly employed to detect packet loss when gaps in sequence numbers occur.

In addition to transmitting the position and FoV, the WebRTC client also transmits feedback messages to the server which are subsequently used by the congestion control algorithm of the server to estimate the bandwidth, allowing for the quality adaptation to take place.

3.5 Virtual Reality Interactivity and Rendering

Due to the sampling that occurs in the quality adaptation algorithm, gaps will occur in the lower quality representations. In order to increase the perceived quality we increase the point size based on the sampling ratio, an example of this can be observed in Figure 6.

To facilitate an immersive experience, a simple rendering solution is not sufficient. To allow for interactivity with the environment, the point cloud needs to be displayed on an HMD. There are two methods of doing this: either by rendering on the HMD itself, or by having a remote system render the point cloud at the correct



(a) 40% sampled point cloud (b) Upscaled point cloud

Figure 6: Increasing the size of the rendered points enables a simple and fast upscaling method. Point size is illustrated left of the person as a dot.

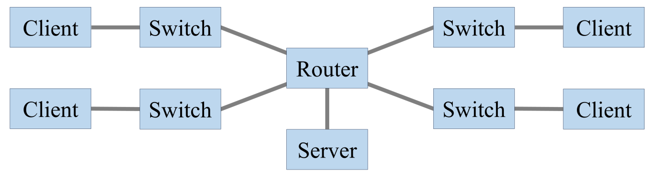


Figure 7: Hardware configuration used in the experimental setup.

position, and transmit a video stream to the HMD with the rendered frames. In Section 4, the latter approach will be considered, for increased computational resources at the decode side.

3.6 VMAF in Volumetric Video

In the context of volumetric videos, VMAF is used by capturing the rendered view at either a fixed viewpoint or by using captured movement traces which allows the viewpoint to be changed during the video [44]. In contrast to PSNR-based metrics, VMAF thus produces a metric that corresponds to the imagery as perceived by the user [19].

4 EVALUATION AND DISCUSSION

This section first presents the experimental setup, providing all hardware and implementation details. Then, we discuss the considered evaluation metrics. Subsequently, we present and discuss the results of several experiments related to the proposed WebRTC-based framework namely: the capturing delay, scalability of the encoder, bandwidth usage, VMAF quality, throughput and latency of WebRTC, and finally an overview of the total end-to-end latency.

4.1 Experimental Setup

We perform the experiments concerning preprocessing latency, scalability and VMAF quality on a machine with the following specifications: *CPU: Intel 12th Gen i7-1265u (4.8 GHz Turbo), RAM: 16GB DDR4 (3200 MHz).* The throughput and WebRTC latency experiments are carried out at the Anonymous [4] (omitted for anonymity), a test bed with a large number of interconnected bare metal nodes. We use several connected bare metal nodes with the following hardware specifications; *CPU: Intel Xeon E5520 (2.27GHz), RAM: 12GB DDR3 (1066 MHz), NIC:* 1 Gb/s. We apply traffic control on the switch nodes in order to limit the available bandwidth, no artificial latency is added. The full setup is illustrated on Figure 7.

To evaluate the suggested approaches with the same data, we use a captured point cloud sequence of 1800 frames [5]. This sequence

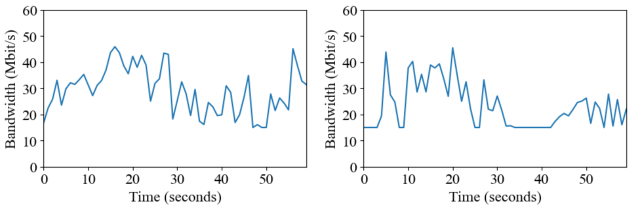


Figure 8: Two example 4G bandwidth traces.

contains point clouds generated from a single camera, with alternating sequences of little to no movement and hand gestures to mimic possible behavior during a conversation. Additionally, to achieve realistic results, we employ a dataset containing 4G traces [45] in a subset of our experiments to simulate fluctuating bandwidth. This dataset is used in the following experiments: scalability, bandwidth usage and VMAF quality. Two example traces are shown in Figure 8.

To capture this sequence, we have used an Intel Realsense D455 depth camera [24] at a frame rate of 30 FPS and a resolution of 848x480 pixels. The laser power of the depth sensor was increased to its maximum value of 360 to increase visual quality and reduce the number of holes in the resulting point cloud.

The preprocessing steps (background removal, sampling and encoding) were collocated with the capturing component, utilizing the same hardware. However, due to the requirement of having a fully implemented congestion controller, we have implemented both the WebRTC server and client in separate applications using Golang [20] together with the Pion package [34]. Both the preprocessing and rendering applications interact with their respective WebRTC counterpart using a UDP socket to transfer the required data. We have opted for GCC as the congestion controller implementation to estimate the bandwidth using the transport wide congestion control (TWCC) messages transmitted by clients.

For the adaptation methods we utilize the polynomial depicted in Figure 4 for estimating the required sampling rate of the individual and bitrate-based MDC encoders. For the MDC encoders we use the implementation shown in Table 1 to divide our point clouds into three descriptions that enable seven possible quality representations.

The rendering application uses the Unity [43] game engine. In order to render the point clouds we use the Pcx [33] package, using the included custom shader to render the vertices as individual points. We display the Unity rendered frames in the Meta Quest 2 HMD [30].

4.2 Evaluation Metrics

Based on the discussion in Section 2 and Section 3 we use the following metrics to evaluate the performance of each component.

First, we evaluate the latency introduced in the capturing component by the Intel Realsense D455 camera. We have used a separate application to calculate the time between the rendering of a binary clock on a display and the capturing of the rendered image.

Second, we evaluate the scalability of the individual and MDC-based encoders by using multiple configurations, each having a different number of simulated clients. We use the preprocessing time and achieved FPS in order to estimate how well the systems are able to scale with an increasing number of clients.

Table 2: Capturing delay analysis of the Intel RealSense camera, model D455 for the Ubuntu 22.04 and Windows 11 operating systems.

Resolution	FPS	Latency (ms)	
		Ubuntu 22.04	Windows 11
424x240	30	70	91
	60	39	64
640x480	30	80	98
	60	45	67
848x480	30	85	104
	60	46	65
1280x720	30	100	115
	60	n/a	n/a

Third, we evaluate bandwidth usage with two metrics. Firstly, we examine the average bandwidth required to transmit a point cloud sequence at 30 FPS for the different adaptation approaches. The second metric measures the efficacy of our encoding strategies in harnessing the available bandwidth, specifically the error between the resultant bitrates and the available bandwidth.

Fourth, quality is evaluated by using the VMAF metric from a single fixed viewpoint. In order to get an accurate estimate we have employed the 4G bandwidth traces of Van Der Hooft et al. [45] to generate frames of varying qualities. Additionally, for the MDC encoders we also show the achieved VMAF score of each of the seven representations.

Fifth, we evaluate the WebRTC component by observing the achieved throughput and latency with two possible link bandwidth configurations, 50 Mbit/s and 100 Mbit/s.

Sixth, we give a breakdown of the overall end-to-end latency of our proposed implementation compared to individual encoding for multiple client configurations.

4.3 Camera Capturing Delay

An evaluation of the capturing delay introduced by the camera system has been conducted in order to assess the impact of resolution and frame rate. This experiment was performed by using the latency tool included in the SDK [25]. The tool renders a binary clock onto the display screen, that is subsequently captured by the camera. The captured frame is processed within the application utilizing the OpenCV library [32], which extracts the value of the binary clock from the frame and compares it with the current value. Table 2 presents the delay measurements for multiple resolutions at both 30 and 60 FPS. the application's performance was evaluated on two distinct operating systems, namely Ubuntu 22.04 and Windows 11, utilizing identical hardware configurations. Notably, Ubuntu 22.04 exhibits superior performance compared to Windows 11. The exact reasoning of this phenomenon is difficult to determine as it depends on a variable number of factors, such as driver and system call differences. Additionally, the Windows 11 version also exhibits inconsistent behavior, leading to increased delay throughout the application's runtime. It becomes apparent from Table 2 that the frame rate has a greater impact on the delay when compared to the resolution. However, the adaption of 60 FPS requires significantly more bandwidth and computational resources, thereby posing challenges in maintaining real-time performance.

Table 3: Individual encoding latency for multiple client compositions using five encoder threads, *CPU: Intel 12th Gen i7-1265u (4.8GHz Turbo), RAM: 16GB DDR4 (3200 MHz).*

# Clients	Latency (ms)		FPS
	Mean	Std	
5	14.87	5.78	29.12
10	32.43	16.32	18.34
20	55.99	29.16	9.65
40	97.81	51.28	5.46

4.4 Encoding Scalability

The encoding constitutes the predominant contribution to latency within the preprocessing step of our pipeline. The latency associated with the other preprocessing operations, i.e., the background removal and sampling, are negligible (mean of 3 ms) compared to the encoding. The encoding latency depends on several factors such as the number of points, the density and the color similarities of neighboring points. Figure 9 illustrates this effect, indicating that the encoding time scales linearly with the size of the point cloud.

In the context of individual encoding, a unique quality representation has to be encoded for each client. Thus, additional clients increase the total time spent preprocessing a single frame. Naturally, by encoding individual streams in parallel, this impact can be reduced. Table 3, shows the impact of using five worker threads with a varying number of clients. In this system configuration, we wait until all clients are done encoding before capturing the next frame, resulting in a lower frame rate when the number of clients increases. To ensure that the system remains at a consistent 30 FPS, it is imperative to scale the number of worker threads proportionately to the number of clients. Additionally, the implementation of a circular buffer for worker threads enables the encoding of consecutive frames whilst certain clients are still encoding the previous frame. On a machine with the following specifications: *CPU: Intel 12th Gen i7-1265u (4.8GHz Turbo), RAM: 16GB DDR4 (3200 MHz)* and with five simulated clients, we achieve similar preprocessing times by employing only three threads, with an average of 18.27 ms (std=4.56 ms). From this, we infer that an equal amount of workers is not requisite for the number of clients and that the number of required workers can be estimated with the following equation:

$$n_{\text{threads}} = n_{\text{clients}} \times \frac{\text{Mean Encoding Time}}{\text{Interframe Time}} \tag{1}$$

However, under the assumption of a singular machine with a limited capacity of twelve threads, the system is still only able to service a modest number of clients. Additionally, as shown in Figure 9, the encoding time scales linearly with the requested sampling rate, making the number of required workers highly correlated with the requested number of quality representations.

MDC encoding alleviates this problem by having a fixed number of descriptions that are used to construct additional quality levels. In addition, the utilization of sampled descriptions reduces the maximum number of points used in a single encoding operation, lowering the overall encoding time. These descriptions are encoded in parallel in order to reduce the total required encoding time. As a result, the encoding time of a frame is restricted to the size of the largest description compared to the number of clients.

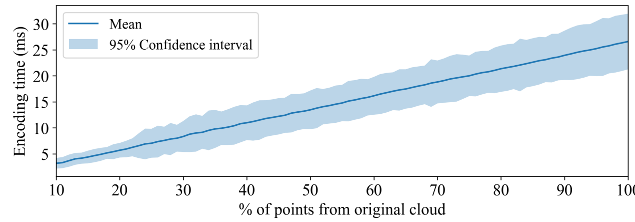


Figure 9: Linear scaling of encoding time with sampling rate.

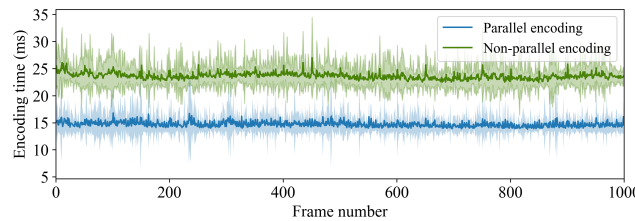


Figure 10: MDC encoding approaches are improved by encoding the three descriptions in parallel, using one worker for each description.

Figure 10 depicts the increased encoding performance when applying parallelization. MDC limits the number of required worker threads to the number of descriptions, making it more suited for a system with reduced computational resources or a large number of clients. Similar performance gains are achieved at the client side by applying parallel decoding to the descriptions.

Comparing the average preprocessing time (30 iterations, sequence of 1800 frames) of the individual encoder against the MDC encoder, we observe that the individual encoder (five clients, five workers) attains an average time of 14.87 ms (std=5.78 ms) versus the MDC encoder (three descriptions, three workers) which achieves an average of 16.16 ms (std=1.82 ms). Evidently, the preprocessing time of the individual encoding approach suffers greatly from the large range of possible sampling rates, with the higher rates significantly increasing the maximum encoding time, creating a much more variable preprocessing time. One approach to mitigate this problem would be the implementation of MDC into the individual adaptation method. A viable strategy could entail imposing a limit which only allows 50% of the point cloud to be encoded in a single thread, while the remainder of the points is encoded in a separate thread. Additionally, this would allow reuse of the 50% description by multiple clients, further reducing the encoding time in favor of compression efficiency.

4.5 Bandwidth Usage

The MDC encoder produces seven possible representations for each frame, these are created from three base descriptions which correspond to approximately 15%, 25% and 60% of the points in the original cloud. Table 4 shows the resulting bitrates for each MDC approach. It is clear from this table that the MDC approach satisfies a range of varying bandwidths between 12 Mbit/s and 75 Mbit/s.

In order to protect against an underestimation of the required sampling rate and the non-instant update nature of the bandwidth estimator we opted to use a safeguard which limits the bitrate available to the encoder to 90% of the estimated bandwidth. Without

Table 4: Resulting bitrates (Mbit/s) for the different MDC quality representations, with 15%, 25% and 60% being the base descriptions used to generate the other representations.

	15%	25%	40%	60%	75%	85%	100%
Fixed Size	11.9	18.6	30.6	39.9	51.8	58.5	70.5
Fixed Bitrate	12.6	19.6	32.1	41.2	53.6	60.8	73.3
Percentage	12.7	19.8	32.0	42.4	55.0	62.1	74.8

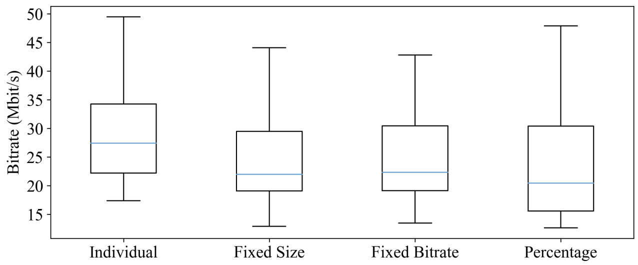


Figure 11: Resulting bitrates for all encoding approaches.

this safeguard bandwidth usage hovers above 100%, causing intolerable packet loss. Although the MDC methods never supply a bitrate greater than the estimate, we opt to use the same safeguard to protect against the non-instant update. When applying this to the bandwidth traces from Van Der Hooft et al. the individual encoder achieves a bandwidth usage of 90.4% of the estimated bandwidth, which results in an average of 3.17 Mbit/s of bandwidth being wasted. Due to having only a limited number of representations the MDC approaches only reach an average bandwidth usage of 75%, which comes down to 7.88 Mbit/s being wasted. Figure 11 depicts the average resulting bitrate for each encoding approach when employing the 4G traces.

4.6 VMAF Quality

VMAF allows us to measure impact of increasing the point size in order to increase the perceived quality. Table 5 presents the results for varying point size across all possible quality representations obtained from the MDC encoder. The comparison is made between sampled frames and rendered frames of the original point cloud. In rendering the original cloud, a point size of 0.3 (Unity units) was adopted, as lower values introduce gaps in the point cloud when viewing it from the fixed viewpoint. From Table 5, we discern the significant impact of increasing the point size on the resulting quality. However, it is imperative to exercise caution to only increase the point size when necessary to avert a compromise in quality. The impact of this is notably visible in the higher quality representations where an approximate 30% decline in quality occurs upon doubling the point size. For the individual encoder, we employ a similar upscaling methodology. However, for this encoder we pick the MDC point size of which the sampling rate is the closest to the individual sampling rate. For example, if a individual client requires a sampling rate of 20%, a point size of 0.06 is chosen, and for a sampling rate of 43%, a point size of 0.05 is selected.

Table 6 shows the achieved VMAF score for each of the seven quality representations. We observe that the three variations offer similar performance when it comes down to the visual quality of the resulting FoV. When compared to the VMAF scores of the individual encoding approach, for a varying number of clients, we

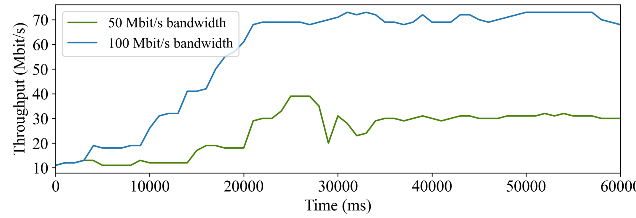


Figure 12: Throughput obtained when using the fixed size MDC approach for 50 Mbit/s and 100 Mbit/s.

can see that the MDC-based approach scores better when at least twenty clients are involved (see Table 7, sequence of 1800 frames). This phenomenon can be explained by the fact that the used point cloud sequence also has segments with little to no movement which result in a high VMAF score, even at lower frame rates. The standard deviation of the experiments below indicate this as well. From the data in Table 7, we conclude that the MDC approach works well in environments with many clients or for volumetric videos that suffer from at a low frame rate.

4.7 Throughput and WebRTC Latency

The goal of our pipeline is being able to adapt its content based on the available bandwidth, while still maintaining a latency suited for real-time communication. In order to test the achievable throughput and WebRTC latency we use two different link capacities, 50 Mbit/s and 100 Mbit/s. We motivate the use of the 50 Mbit/s capacity due to it being below the maximum required bandwidth, while still allowing for a sufficient number of quality representations. In this experiment, we employ the fixed-size MDC encoder due to it having the most consistent resulting bitrates, as a stable bandwidth makes it easier for the congestion controller to converge. Figure 12 depicts the results of this experiment, we observe that the 100 Mbit/s link reaches a stable bitrate of 70.6 Mbit/s. Using Table 4, we observe that this corresponds to the bitrate of the highest quality representation. Contrary, the 50 Mbit/s link stabilizes at 30.6 Mbit/s, corresponding with the 40% representation. However, even with the 90% safeguard this link should be able to achieve bitrate of the 60% representation. Figure 12 indicates that this behavior is due to the GCC algorithm not being able to recover after an overestimation which subsequently caused packet loss. By analyzing both figures, we can conclude that the GCC algorithm takes a significant time before converging to a stable estimation. This phenomenon is related to the initial bitrate of GCC, which was set to the minimum required bandwidth. Increasing the initial bitrate speeds up the convergence at the cost of causing packet loss for low bandwidth links. In the case of the 50 Mbit/s link, convergence is observed around 30 Mbit/s. Referring to Table 4, this corresponds to a quality level of 40% with an average VMAF score of 67.56.

In terms of latency, 100 Mbit/s link achieves an average transport latency of 30.89 ms (std=10.22 ms) compared to the result of the 50 Mbit/s link which achieves 24.42 ms (std=5.68 ms). Both results are acceptable to use in a real-time environment. The lower latency of the 50 Mbit/s link is attributed to the lower throughput and indicates that the GCC algorithm ensures a good balance between estimated bandwidth, latency and potential packet loss.

Table 5: VMAF scores for all possible quality and size combinations for point sizes 0.03 to 0.09.

	15%		25%		40%		60%		75%		85%		100%	
	Mean	Std	Mean	Std	Mean	Std	Mean	Std	Mean	Std	Mean	Std	Mean	Std
0.03	4.99	2.72	24.06	2.93	47.38	2.68	68.58	2.17	**78.69**	**1.6**	**83.42**	**1.26**	**88.06**	**0.84**
0.04	22.03	2.93	44.46	2.66	65.28	1.87	**75.82**	**1.08**	77.65	0.82	77.74	0.73	77.18	0.75
0.05	33.98	2.85	55.26	2.13	**67.56**	**1.14**	68.98	1.03	67.9	1.01	67.0	1.0	65.64	1.06
0.06	41.53	2.58	**57.22**	**1.44**	61.16	1.11	59.05	1.22	57.38	1.21	56.46	1.32	55.18	1.3
0.07	**44.72**	**2.08**	52.31	1.49	50.88	1.34	47.71	1.5	45.96	1.58	44.99	1.56	43.84	1.58
0.08	42.89	1.68	44.32	1.54	40.81	1.58	37.66	1.72	36.02	1.75	35.14	1.78	34.12	1.85
0.09	37.87	1.58	35.82	1.65	31.5	1.72	28.7	1.83	27.34	1.99	26.52	1.85	25.66	1.9

Table 6: VMAF scores for MDC approaches with fixed quality.

	15%		25%		40%		60%		75%		85%		100%	
	Mean	Std	Mean	Std	Mean	Std	Mean	Std	Mean	Std	Mean	Std	Mean	Std
Fixed Size	44.52	2.13	57.05	1.7	67.69	1.16	75.84	1.11	78.73	1.58	83.24	1.31	88.05	0.8
Fixed Bitrate	49.99	1.84	61.01	1.29	69.35	1.04	72.98	1.42	76.25	1.57	83.87	1.16	89.32	0.61
Percentage	46.62	1.93	58.64	1.48	68.61	1.08	76.71	0.82	81.71	0.69	85.9	0.59	89.76	0.47

Table 7: VMAF scores for a sequence of 1800 frames for both MDC-based approaches versus individual encoding, using three descriptions for the MDC-based approaches.

	# Clients	FPS	VMAF Score	
			Mean	Std
Individual Adaptive Frame Rate	5	30	67.98	11.93
	10	20	67.67	13.50
	20	10	65.90	16.48
	40	5	58.47	19.20
Fixed Size	n	30	65.98	9.01
Fixed Bitrate	n	30	65.12	11.63
Percentage	n	30	66.64	7.39

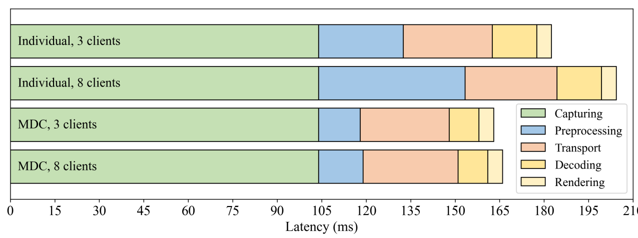


Figure 13: Composite latency of each described component in the proposed architecture, with no artificial link delays, using three worker threads.

4.8 End-to-End Latency Breakdown

Figure 13 illustrates the complete end-to-end latency for both encoding approaches, with no artificially added link delays. For this experiment we have used the fixed frame sequence with three workers. However, we have added the camera latency for completeness. We have deployed two instances of the client application on each node when increasing the number of clients to eight. Each client is assigned 100 Mbit/s bandwidth using tc.

Evidently, the capturing component has the most substantial influence, constituting between 50% and 64% of the total latency. However, the use of a different, more optimized camera would yield immediate improvements in the capturing component of the pipeline. Overall, we observe that our pipeline is able to stream a volumetric video captured at a frame rate of 30 FPS and a resolution of 848x480 pixels with a low enough latency to enable real-time communication. When increasing the number of clients the MDC-based approach is able to maintain a stable latency compared to individual encoding, which has an increased encoding time due to the limited number of workers.

5 CONCLUSION AND FUTURE WORK

In this paper, we proposed a novel one-to-many architecture for facilitating the streaming of volumetric videos. We conducted a comparative analysis of various encoding and adaptation strategies. Our findings indicate that an individually encoded quality representation for each client yields the best results for a limited number of clients. To address scalability concerns, we introduce a MDC-based approach which utilizes several distinct descriptions to construct multiple quality representations. Despite exhibiting comparatively inferior ideal bandwidth utilization and objective quality, the MDC approach demonstrates superior scalability, achieving equivalent preprocessing times with a substantially reduced number of workers. Compared to per client encoding, we observe 9% lower end-to-end latency (163 ms vs 182 ms) with three clients and 19% (166 ms vs 204 ms) when increasing the number of clients to eight. We conclude that our pipeline is able to produce an acceptable latency with a visually quality comparable to the individual encoder in low client environments, and is able to produce higher average quality in environments with more than 20 clients.

In future work, we will focus on extending the architecture to allow for many-to-many streaming of volumetric video, facilitating the need to implement an MCU WebRTC architecture. Furthermore, the extension of multiple input clients will require us to improve the encoder and adaptation algorithms in order to achieve acceptable bitrates and visual quality in scenes with a large number of clients. An extension will also be made to allow for the use of multiple cameras, which will be evaluated through VMAF scores for multiple viewpoints.

REFERENCES

[1] Sun Joo Ahn, Laura Levy, Allison Eden, Andrea Stevenson Won, Blair MacIntyre, and Kyle Johnsen. 2021. Ieeevr2020: exploring the first steps toward standalone virtual conferences. *Frontiers in Virtual Reality*, 2, 648575.

[2] Dimitrios S Alexiadis, Dimitrios Zarpalas, and Petros Daras. 2012. Real-time, full 3-d reconstruction of moving foreground objects from multiple consumer depth cameras. *IEEE Transactions on Multimedia*, 15, 2, 339–358.

[3] [SW], Anonymous (will change to repo when accepted) 2023. URL: Anonymous.

[4] [SW], Anonymous 2023. URL: Anonymous.

[5] [SW], Anonymous Point Cloud Sequence 2023. URL: https://github.com/anonymous4674/sequence.

[6] Anonymous. [n. d.] Anonymous.

[7] Muhammad Ajmal Azad, Rashid Mahmood, and Tahir Mehmood. 2009. A comparative analysis of dccp variants (ccid2, ccid3), tcp and udp for mpeg4 video applications. In *2009 International Conference on Information and Communication Technologies*. IEEE, 40–45.

[8] Sung-Ho Bae, Jaeil Kim, Munchurl Kim, Sukhee Cho, and Jin Soo Choi. 2013. Assessments of subjective video quality on hevc-encoded 4k-uhd video for beyond-hdtv broadcasting services. *IEEE Transactions on Broadcasting*, 59, 2, 209–222.

[9] Maarten Bassier, Maarten Vergauwen, and Florent Poux. 2020. Point cloud vs. mesh features for building interior classification. *Remote Sensing*, 12, 14, 2224.

[10] 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, 92–97.

[11] Michael Broxton et al. 2020. Immersive light field video with a layered mesh representation. *ACM Transactions on Graphics (TOG)*, 39, 4, 86–1.

[12] Keming Cao, Yi Xu, and Pamela Cosman. 2020. Visual quality of compressed mesh and point cloud sequences. *IEEE Access*, 8, 171203–171217.

[13] Neal Cardwell, Stefan Savage, and Thomas Anderson. 2000. Modeling tcp latency. In *Proceedings IEEE INFOCOM 2000. Conference on Computer Communications. Nineteenth Annual Joint Conference of the IEEE Computer and Communications Societies (Cat. No. 00CH37064)*. Vol. 3. IEEE, 1742–1751.

[14] Gaetano Carlucci, Luca De Cicco, Stefan Holmer, and Saverio Mascolo. 2016. Analysis and design of the google congestion control for web real-time communication (webrtc). In *Proceedings of the 7th International Conference on Multimedia Systems*, 1–12.

[15] Yan Cui, Sebastian Schuon, Derek Chan, Sebastian Thrun, and Christian Theobalt. 2010. 3d shape scanning with a time-of-flight camera. In *2010 IEEE Computer Society Conference on Computer Vision and Pattern Recognition*. IEEE, 1173–1180.

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

[17] Qin Dai and Ralf Lehnert. 2010. Impact of packet loss on the perceived video quality. In *2010 2nd International Conference on Evolving Internet*. IEEE, 206–209.

[18] César Debeunne and Damien Vivet. 2020. A review of visual-lidar fusion based simultaneous localization and mapping. *Sensors*, 20, 7, 2068.

[19] Boni García, Luis López-Fernández, Francisco Gortázar, and Micael Gallego. 2019. Practical evaluation of vmaf perceptual video quality for webrtc applications. *Electronics*, 8, 8, 854.

[20] [SW], Golang 2023. URL: https://go.dev/.

[21] Michael J Gourlay and Robert T Held. 2017. Head-mounted-display tracking for augmented and virtual reality. *Information Display*, 33, 1, 6–10.

[22] Danillo Graziosi, Ohji Nakagami, Shinroku Kuma, Alexandre Zaghetto, Teruhiko Suzuki, and Ali Tabatabai. 2020. An overview of ongoing point cloud compression standardization activities: video-based (v-pcc) and geometry-based (g-pcc). *APSIPA Transactions on Signal and Information Processing*, 9, e13.

[23] Janine Hacker, Jan vom Brocke, Joshua Handali, Markus Otto, and Johannes Schneider. 2020. Virtually in this together–how web-conferencing systems enabled a new virtual togetherness during the covid-19 crisis. *European Journal of Information Systems*, 29, 5, 563–584.

[24] [SW], Intel Realsense 2023. URL: https://www.intelrealsense.com/.

[25] [SW], Intel RealSense SDK 2 2023. URL: https://www.intelrealsense.com/sdk-2/.

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

[27] Ingemar Johansson. 2014. Self-clocked rate adaptation for conversational video in lte. In *Proceedings of the 2014 ACM SIGCOMM workshop on Capacity sharing workshop*, 51–56.

[28] Alan B Johnston and Daniel C Burnett. 2012. *WebRTC: APIs and RTCWEB protocols of the HTML5 real-time web*. Digital Codex LLC.

[29] Leonid Keselman, John Iselin Woodfill, Anders Grunnet-Jepsen, and Achintya Bhowmik. 2017. Intel realsense stereoscopic depth cameras. In *Proceedings of the IEEE conference on computer vision and pattern recognition workshops*, 1–10.

[30] [SW] Meta, Meta Quest 2 Virtual Reality Headset 2023. URL: https://www.meta.com/quest/products/quest-2.

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

[32] [SW], OpenCV 2023. URL: https://opencv.org/.

[33] [SW] keijiro, https://www.meta.com/be/en/quest/products/quest-2/ 2023. URL: https://github.com/keijiro/Pcx.

[34] [SW], Pion WebRTC 2023. URL: https://github.com/pion/webrtc.

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

[36] Reza Rejaie, Mark Handley, and Deborah Estrin. 1999. Quality adaptation for congestion controlled video playback over the internet. *ACM SIGCOMM Computer Communication Review*, 29, 4, 189–200.

[37] David Roberts, Toby Duckworth, Carl Moore, Robin Wolff, and John O'Hare. 2009. 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*. IEEE, 89–94.

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

[39] Markus Schütz et al. 2016. Potree: rendering large point clouds in web browsers. *Technische Universität Wien, Wiedeń*.

[40] Michael Seufert, Sebastian Egger, Martin Slanina, Thomas Zinner, Tobias Hoßfeld, and Phuoc Tran-Gia. 2014. A survey on quality of experience of http adaptive streaming. *IEEE Communications Surveys & Tutorials*, 17, 1, 469–492.

[41] Dong Tian, Hideaki Ochimizu, Chen Feng, Robert Cohen, and Anthony Vetro. 2017. Geometric distortion metrics for point cloud compression. In *2017 IEEE International Conference on Image Processing (ICIP)*. IEEE, 3460–3464.

[42] Eric M Torlig, Evangelos Alexiou, Tiago A Fonseca, Ricardo L de Queiroz, and Touradj Ebrahimi. 2018. A novel methodology for quality assessment of voxelized point clouds. In *Applications of Digital Image Processing XLI*. Vol. 10752. SPIE, 174–190.

[43] [SW], Unity 2023. URL: https://unity.com.

[44] Sam Van Damme, Maria Torres Vega, and Filip De Turck. 2021. A full-and no-reference metrics accuracy analysis for volumetric media streaming. In *2021 13th International Conference on Quality of Multimedia Experience (QoMEX)*. IEEE, 225–230.

[45] Jeroen Van Der Hooft, Stefano Petrangeli, Tim Wauters, Rafael Huysegems, Patrice Rondao Alface, Tom Bostoen, and Filip De Turck. 2016. Http/2-based adaptive streaming of hevc video over 4g/lte networks. *IEEE Communications Letters*, 20, 11, 2177–2180.

[46] Jeroen Van der Hooft, Tim Wauters, Filip De Turck, Christian Timmerer, and Hermann Hellwagner. 2019. Towards 6dof http adaptive streaming through point cloud compression. In *Proceedings of the 27th ACM International Conference on Multimedia*, 2405–2413.

[47] Jeroen van der Hooft, Hadi Amirpour, Maria Torres Vega, Yago Sanchez, Raimund Schatz, Thomas Schierl, and Christian Timmerer. 2023. A tutorial on immersive video delivery: from omnidirectional video to holography. *IEEE Communications Surveys & Tutorials*.

[48] Wowza. 2021. 2021 Video Streaming Latency Report. Tech. rep. Wowza.

[49] Emin Zerman, Cagri Ozcinar, Pan Gao, and Aljosa Smolic. 2020. Textured mesh vs coloured point cloud: a subjective study for volumetric video compression. In *2020 Twelfth International Conference on Quality of Multimedia Experience (QoMEX)*. IEEE, 1–6.

[50] Yizhong Zhang, Zhiqi Li, Sicheng Xu, Chong Li, Jiaolong Yang, Xin Tong, and Baining Guo. 2023. Remotetouch: enhancing immersive 3d video communication with hand touch. In *2023 IEEE Conference Virtual Reality and 3D User Interfaces (VR)*. IEEE, 1–10.

[51] David J Zielinski, Hrishikesh M Rao, Mark A Sommer, and Regis Kopper. 2015. Exploring the effects of image persistence in low frame rate virtual environments. In *2015 IEEE Virtual Reality (VR)*. IEEE, 19–26.