Scalable MDC-Based WebRTC Streaming for One-to-Many Volumetric Video Conferencing

MATTHIAS DE FRÉ, JEROEN VAN DER HOOFT, TIM WAUTERS, and FILIP DE TURCK,
IDLab, Ghent University - imec, Belgium

Video consumption has become central to modern life, with users seeking more immersive experiences such as virtual conferencing or concerts within virtual reality (VR). While 360° video offers rotational movement, it lacks true positional freedom. Fully immersive formats like light fields and volumetric video enable six degrees-of-freedom (6DoF), allowing both types of freedom. However, their high bandwidth and computational demands make them impractical for low-latency applications. Efforts to address these issues through compression and quality adaptation have improved quality of experience (QoE), but real-time interaction remains limited because of latency. To solve this, we introduce a novel, open-source one-to-many streaming architecture using point cloud-based volumetric video. By compressing point clouds with the Draco codec and transmitting via web real-time communication (WebRTC), we achieve low-latency 6DoF streaming. Content is adapted by employing a multiple description coding (MDC) strategy which combines sampled point cloud descriptions using the estimated bandwidth returned by the Google congestion control (GCC) algorithm. MDC encoding scales more easily to a larger number of users compared to individual encoding. Our proposed solution achieves similar real-time latency for both three and eight 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. Additionally, when compared to an approach with five fixed quality levels, our MDC-based approach scores 13% better in terms of latency, while achieving similar quality.

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

Additional Key Words and Phrases: Volumetric Video, Adaptive Streaming, WebRTC, Virtual Conferencing, MDC, Virtual Reality

ACM Reference Format:
Matthias De Fré, Jeroen van der Hooft, Tim Wauters, and Filip De Turck. 2025. Scalable MDC-Based WebRTC Streaming for One-to-Many Volumetric Video Conferencing. *ACM Trans. Multimedia Comput. Commun. Appl.* 21, XXX, Article XXX (April 2025), 23 pages. https://doi.org/XXXXXXX.XXXXXXX

1 INTRODUCTION

As remote communication platforms become more widespread, their 2D-based interaction modes expose limitations in delivering immersive experiences [22]. To bridge this gap, six degrees-of-freedom (6DoF) video solutions introduce rotational and positional freedom, enabling a more engaging interaction model [2]. However, the choice of communication system depends on the specific interaction requirements. One-on-one video calls prioritize low latency for real-time communication [28], while many-to-many virtual conferences require scalable video content for

Authors' address: Matthias De Fré, matthias.defre@ugent.com; Jeroen van der Hooft; Tim Wauters; Filip De Turck, IDLab, Ghent University - imec, Ghent, Belgium.

1551-6857/2025/4-ARTXXX $15.00
https://doi.org/XXXXXXX.XXXXXXX

multiple users [27]. Selecting the right communication method is crucial to meet interactivity needs while balancing bitrate and latency.

Immersive 6DoF content for applications on head-mounted display (HMD)s can be categorized into image-based and volumetric video formats [57]. Image-based techniques, like light fields, use numerous images to create a new view based on the user's angle, captured by either a lenslet camera or a setup with many cameras [62]. 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 [46]. In contrast, volumetric video uses 3D representations like meshes and point clouds. Meshes consist of vertices connected by edges to form polygons, while point clouds are vertices with color attributes [7]. These objects are captured by multiple cameras and stitched together, resulting in a single 3D object [3]. Compared to light fields, this approach substantially reduces the bandwidth for transmission, as it is no longer required to maintain a large collection of images; instead, it suffices to transfer a single object, increasing the maximum number of connected users. Furthermore, the computational requirements associated with rendering volumetric video are significantly lower compared to image-based solutions, which is a necessity for real-time rendering and interaction at the client side.

Even with reduced bandwidth needs, bitrates for high-resolution point clouds, like those in the 8i dataset [16], reach up to 5.2 Gbit/s, requiring specialized ultra-high bandwidth environments to use [57]. Compression and quality adaptation are essential for broader use across varying network conditions. HMDs are capable of tracking position and field-of-view (FoV) using specialized sensors [19]. By using this tracking data, video content can be adapted dynamically, transmitting only the essential details and reducing bandwidth usage. A key factor in achieving real-time communication is the choice of transport protocol. TCP-based solutions like low-latency HLS (LL-HLS) and low-latency DASH (LL-DASH) are unsuitable for virtual conferencing due to the added latency from reliability mechanisms like acknowledgments, and flow control. In contrast, UDP-based solutions, such as web real-time communication (WebRTC), prioritize real-time performance by sacrificing TCP features such as guaranteed delivery, resulting in lower latency [6].

This article serves as an extension to our previously published work [12]. The proposed approach and experiments have been extended to include an additional encoding strategy. This new strategy utilizes a number of fixed and independent quality levels, and is comparable to quality adaptation used in traditional 2D video. This article proposes a novel open-source architecture designed for one-to-many streaming [36], with the following contributions:

- 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 , and achieving 13% lower latency while maintaining similar quality compared to having five fixed quality representations.
- 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 more extensive comparative analysis between our proposed MDC-based approach and an encoding solution optimized per client, evaluating their respective performance in terms of quality, scalability and approximation of the available bandwidth during encoding.
- An extension to our previous work [12], which now also includes a detailed comparative analysis between our proposed approach and a traditional encoding method that uses a fixed number of distinct quality levels.

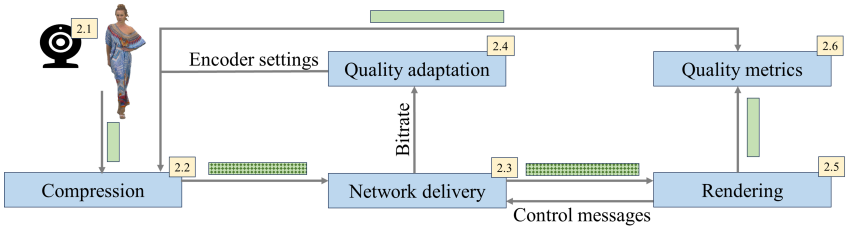


Fig. 1. A volumetric video streaming architecture [12].

The remainder of this article 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 article with an overview of the most important results and 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, followed by an overview of the state-of-the-art point cloud codecs. For each 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 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 [61]. However, point clouds employ better, optimized culling and tiling algorithms due to relying purely on vertices [8]. Additionally, the capturing process is more simplistic when utilizing point clouds, since the captured points and their colors are used directly. In contrast, meshes require the captured points to be converted into a geometric topology before rendering, this process is time consuming and negatively impacts the latency of the system [1]. Newer mesh-based systems have achieved significantly improved latency, but are still insufficient for high-quality volumetric conferencing with an acceptable frame rate [29]. As a consequence, the focus for the remainder of this paper will be on the usage of point clouds as a source of volumetric video. These point clouds can be captured through various means. LiDAR cameras use reflected lasers to attain an accurate 3D representation of a large area at the cost of increased power consumption [14]. In contrast, depth-based systems employ a stereoscopic sensor to estimate the depth for each pixel of a depth image [3]. As demonstrated by other volumetric streaming pipelines, depth cameras are often preferred due to their lower cost, reduced power consumption and ease of multi-camera synchronization, while still providing point clouds of acceptable quality [21, 27].

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 [20], 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 [18] or G-PCC [20], are designed to encode the point cloud as a 3D structure.

V-PCC employs 2D projections of the point cloud to create an image containing the different views. Subsequently, this image is encoded using existing video codecs. This method achieves high compression at the cost of non-real-time coding speeds [20]. Additionally, V-PCC supports inter-frame prediction, which groups frames into a segment. However, the used implementation has little to no impact on the resulting bitrates and instead increases the latency considerably [13]. In contrast to V-PCC, geometric encoders utilize 3D data structures, such as octrees and kd-trees, 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 [13]. For this reason, we leverage Draco to efficiently encode the point clouds.

2.3 Low-Latency Delivery

Given that volumetric video is fundamentally an extension of 2D video, it is feasible to adapt existing 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 [11]. 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 [50]. This process introduces a latency between 5 and 18 seconds [60], making it unsuitable for real-time streaming. However, even the low-latency variants, such as LL-DASH and LL-HLS, only achieve a latency between 1 and 5 seconds [60], which is still inadequate [33]. 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 [35]. UDP-based solutions such as media over QUIC (MoQ) [5] and WebRTC [15] are designed specifically to enable real-time interactive communication. Compared to other streaming formats, WebRTC achieves a sub one second delay [60]. 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 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 [41]. 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

In traditional real-time 2D video streaming solutions, the client is supplied with several representations based on different screen resolutions from which it can choose. In recent times, more and more streaming services have also started providing additional quality options that allow to user to choose between a higher 60 FPS frame rate or the default 30 FPS stream, further increasing the number of available options [44]. However, manually selecting the quality can be inconvenient and reduce the user experience. When users encounter buffering caused by packet loss or network congestion, their instinctive reaction is usually to reduce the quality. Later, they may manually increase the quality again when they believe that the network conditions have improved, which might lead to further buffering issues.

Quality adaptation is a well-known video streaming technique to ensure smooth playback without packet loss or user interaction [32]. Most encoders produce multiple quality levels by targeting different bitrates [45]. In one-to-one scenarios, 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 bitrates are chosen to ensure a low latency and acceptable quality stream for each user. Alternatively, rather than employing separate quality levels, a MDC approach can be adopted. With MDC, the content is encoded in descriptions that are combined to

achieve varying levels of quality [59]. In contrast to scalable video coding (SVC), which enhances a base layer through the use of dependent enhancement layers, MDC generates independently decodable descriptions that can be combined to improve quality [49]. These descriptions ensure frames can be rendered even in the event of the loss of one description. While these methods have been designed for 2D video, certain aspects can be integrated into volumetric video. These adaptation methods rely on bandwidth estimations to determine the quality that should be delivered. Accurate estimations enable bitrate allocation algorithms to make informed decisions, preventing buffering at the cost of lowering the quality [58]. For UDP-based solutions the estimator is commonly implemented at the application. Congestion controllers, such as Google congestion control (GCC) (general-purpose) [9] and Self-Clocked Rate Adaptation for Multimedia (SCReAM) (optimized for mobile networks) [30], are implemented in the multimedia application and require an exchange of messages between the sender and the receiver to estimate the available bandwidth. These congestion controllers independently estimate the bandwidth for each client and promote general fairness by gradually increasing the bandwidth estimation while responding to signs of congestion. In contrast, network-driven approaches leverage knowledge of the network topology and client placement to produce more accurate estimations [10]. However, their reliance on detailed network information limits their usability. For these reasons, we will utilize GCC to implement bandwidth estimation in WebRTC.

This bandwidth estimation can be supplied, together with context-aware information from the user, to the encoder with the goal of optimizing the bitrate allocation and quality. Additionally, this encoder should be optimized to support a high number of users while still maintaining the lowest possible latency [41]. In volumetric video, a user will alter their viewpoint, causing objects to fall out of the viewpoint and making it inefficient to continue transmitting these objects. van der Hooft et al. [56] encode each point cloud 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 representation for each object has to be transmitted to achieve optimal quality for the available bandwidth. However, a DASH-based approach is proposed, which concatenates the frames 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.

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. [47] use point primitives to render the point cloud. Additionally, the size of the points is changed to reduce the number of gaps, increasing the quality. To further improve performance, it is also possible to replace the object with a lower quality version whenever the user moves further away. For meshes this requires the construction of multiple level of detail (LOD) of the same mesh, which is a time-consuming process. On the other hand, the generation of multiple quality levels for point clouds is computationally less intensive as it is commonly based on sampling methods instead of mesh construction [48].

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 encoding latency, which negatively impacts the user's experience [63]. The assessment of objective quality can be conducted using different types of metrics. Tian et al. [51] introduce multiple geometric metrics which assess the deformation of the mesh or point cloud relative to the original object, independent of potential viewports. In

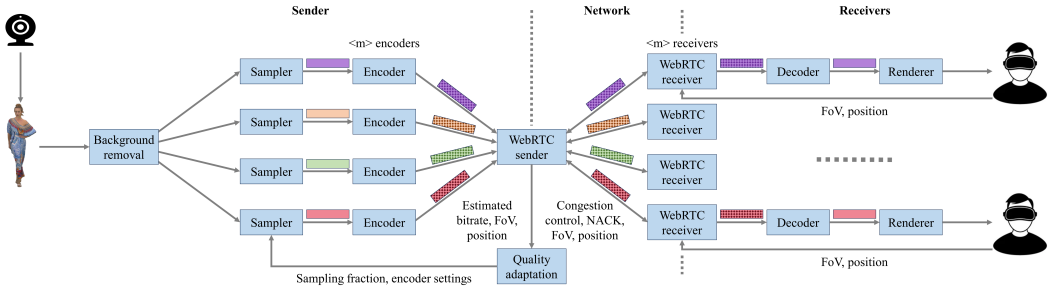


Fig. 2. The system architecture for the individually encoded one-to-many volumetric video streaming solution [12].

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 peak signal-to-noise ratio (PSNR) metrics. Both of the mentioned metrics suffer from the limitation of not being able to prioritize sections which draw more attention compared to others, such as the hands [24], thereby posing challenges in approximating user-perceived quality.

Video Multi-Method Assessment Fusion (VMAF) is a metric that combines the scores from multiple metrics in order to produce a single score, ranging from 0 to 100, that better correlates to the perceived quality [38]. Due to this superior correlation, VMAF will be used to assess the impact of the proposed approach on the visual quality of the user's FoV.

3 PROPOSED APPROACH

This section provides an overview of the proposed one-to-many MDC-based volumetric video streaming architecture. Our architecture extends an individually encoded architecture, an example of which is depicted on Figure 2. The goal of our architecture is to solve the problems that occur in terms of scalability when employing the individual architecture. 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 can then be used to map an RGB value and assign a color 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 [39]. To streamline this process, we employ sampling to generate varying levels of point cloud quality.

To prevent inter-frame dependencies, the frames are exclusively encoded through intra-coding. This ensures that each frame can be decoded independently, leading to improved quality when frames are dropped due to packet loss.

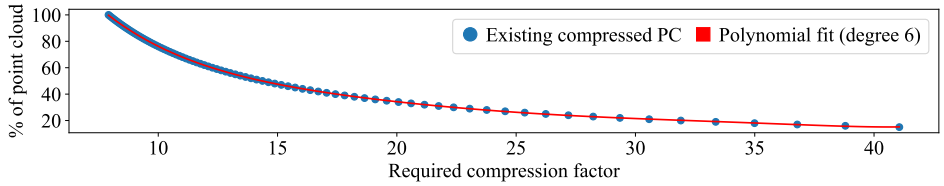


Fig. 3. The usage of a fitted polynomial together with the required compression factor allows to determine the sampling rate [12].

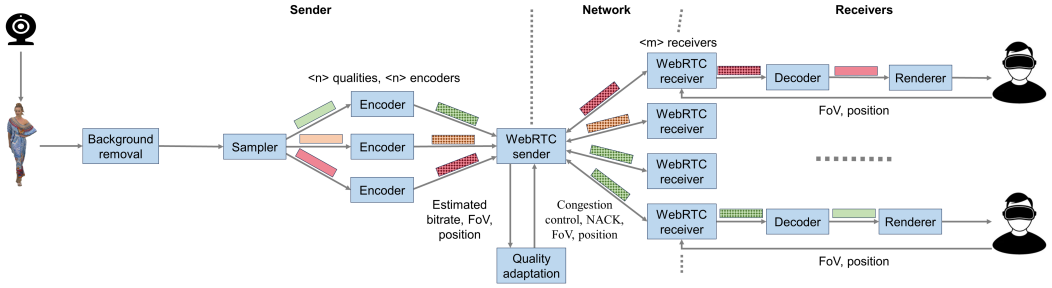


Fig. 4. The system architecture for a one-to-many volumetric video solution using n quality representations.

3.3 Quality Adaptation

The first step of our adaptation algorithm is the creation of diverse quality representations. To facilitate this process, we propose three methods capable of generating these qualities. The goal of these methods is to produce a representation for each user, while ensuring that preprocessing remains below the inter-frame time for real-timeliness. All 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 method ensures a balance between the sampling time and resulting quality compared to a more FoV-aware sampling which achieves higher quality at the cost of latency [13].

The first method generates a unique representation for each connected client. This is accomplished by first calculating the required compression factor, which is equal to the raw frame size divided by the available bitrate for that frame. The compression factor is then used together with the fitted polynomial in Figure 3 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 cloud 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 shows the integration of this method into a WebRTC streaming pipeline, which selects the quality at the server, resulting in faster quality switching at the cost of requiring slightly more computational power at the server [4]. A 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 number of clients.

The second approach, as depicted in Figure 4, is similar to how traditional video works. The application produces n representations by sampling the point cloud at n distinct rates. Each quality is independent of the other qualities and multiple qualities will contain the same points.

The third method, which is shown in Figure 5, 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

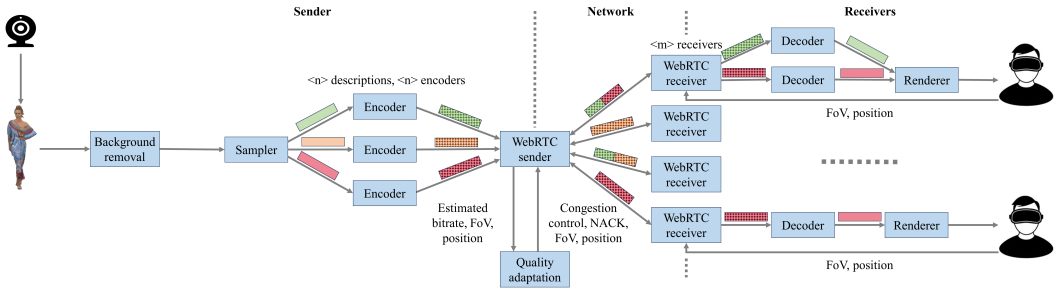


Fig. 5. The system architecture for the proposed MDC-based one-to-many volumetric video solution, requiring encoders equal to the amount of descriptions, the number of local decoders for each client scales with the number of received descriptions [12].

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

```
viewAngles ← calculateAngles(yaw, pitch, roll)
distance ← getDistanceToUser(userPos, clientPos)
category ← getClientCategory(fov, viewAngles, distance)
descriptions ← getAllowedDescriptions(category)
for d ← descriptions.max() to descriptions.min() do
    if bitrate >= getDescriptionSize(d) then
        bitrate ← bitrate − getDescriptionSize(d)
        addDescriptionToFrame(d)
    end if
end for
```

Fig. 6. Quality category is assigned based on distance and field of view [12].

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 3, to estimate n sampling percentages necessary for creating n descriptions with varying bitrate targets

Compared to traditional video encoding, which encodes a video into multiple quality representations, these MDC-based approaches perform better at point cloud encoding due to the lack of inter-frame prediction and having more quality levels with different bitrates.

Using Algorithm 1, we first calculate the rotation matrix and viewing angles by utilizing the yaw, pitch and roll of the camera. Following this, the distance of the user to the streamed object is calculated. Subsequently, the viewing angle, FoV and distance are used to impose restrictions on the maximum achievable quality that can be transmitted to the client, even if the available bandwidth

Table 1. Example implementation of the possible quality levels of MDC encoding through the combination of descriptions (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%
	2, 3	~85%
	1, 2, 3	~100%

allows for a higher quality. Figure 6 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, 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. The fixed quality encoder utilizes the same table as the MDC-based approach, classifying qualities with a sampling rate of 60% or higher as high quality, those with a rate of 25% or higher as medium quality, and all others as low quality. Finally, we iterate over the allowed descriptions, starting with the highest quality description, and validate if enough bandwidth remains 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 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. 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. Furthermore, as a measure to address potential packet loss, lost packets are retransmitted. This mechanism ensures 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, requiring the full frame to be received. 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 quality adaptation.

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 upscaling can be observed in Figure 7. 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. Displaying this content can be accomplished in one of two ways: either by rendering it locally on the HMD, or by offloading the rendering to a remote system that processes the point cloud and streams the rendered video frames to the HMD.. In Section 4, the latter approach will be considered, for increased computational resources at the decode side.

(a) 100% sampled point cloud (b) 40% sampled point cloud (c) Upscaled point cloud

Fig. 7. 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 [12].

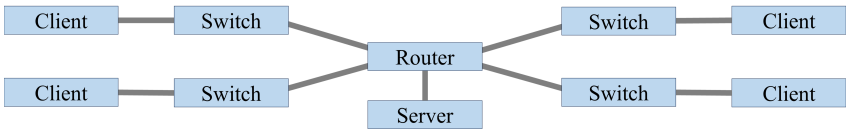


Fig. 8. Hardware configuration used in the experimental setup [12].

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 [54]. In contrast to PSNR-based metrics, VMAF thus produces a metric that corresponds to the imagery as perceived by the user [34].

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 capturing latency, preprocessing latency, encoder scalability and renderer quality using VMAF, ranging from one to forty clients, 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, with three and eight clients, are carried out on the imec Virtual Wall [25], a testbed with a large number of interconnected bare metal nodes. We use several nodes with the following hardware specifications; *CPU: Intel Xeon E5520 (2.27GHz), RAM: 12GB DDR3 (1066 MHz), NIC:* 1 Gbit/s. Each node operates as one of following three roles: a headless WebRTC client (without decoding or rendering), a WebRTC server, or a network switch. 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 8.

To evaluate the suggested approaches with the same data, we use a captured point cloud sequence of 1800 frames [43]. This sequence contains point clouds generated from a single camera, with alternating periods of little to no movement and gestures to mimic behavior during a conversation. Additionally, to achieve realistic results, we employ a dataset of 4G traces [55] 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 9.

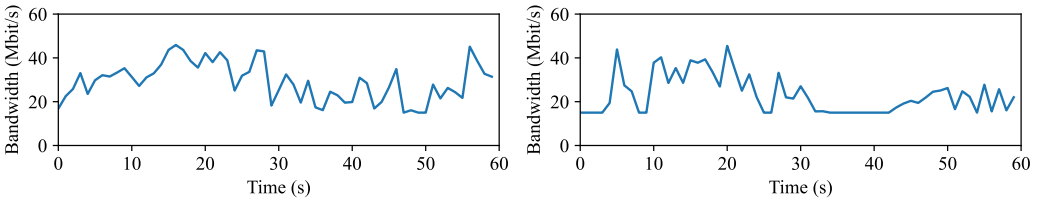


Fig. 9. Two example 4G bandwidth traces [55].

To capture this sequence, we have used an Intel RealSense D455 depth camera [26] 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. Frames are encoded using the CPU-based Draco codec with QP=11, which ensures real-time encoding latency [13].

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 [17] together with the Pion package [42]. 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. The default and minimum bitrate was set to 15 Mbit/s, while the maximum bitrate value was set to 100 Mbit/s. For retransmission of lost packets, a negative acknowledgment (NACK)-based system is used [23].

For the adaptation methods we utilize the polynomial depicted in Figure 3 for estimating the required sampling rate of the individual and bitrate-based MDC encoders. The fixed quality encoding approach utilizes n equally spaced sampling rates, ranging from 15% to 100%, to produce the different quality representations. In the experiments, values for the parameter n range from three to seven, and are used to demonstrate the impact of varying the number of quality levels. 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 sampling rates for the base descriptions (15%, 25% and 60%) used in the experiments enable a wide range of available bitrates, and were chosen for the following reasons:

- The 15% rate, due to lower rates resulting in significant quality degradation.
- The 60% rate, enables a higher quality, compared to lower rates, when using the upscaling described in Section 3.5.
- The 25% rate, to allow for 100% of the points when combining all three descriptions.

The FoV is equally divided into the three quality categories shown in Figure 6. The distance levels are determined by the height of the point cloud within the viewport (80%-100% of the viewport for high quality, 50%-80% for medium quality, 20-50% for low quality). The rendering application uses the Unity [53] game engine. In order to render the point clouds we use the Pcx [31] package, using the included shader to render the vertices as individual points. The Unity rendered frames are displayed in the Meta Quest 2 HMD [37].

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 and the capturing of the rendered image.

Table 2. Capturing delay analysis of the Intel RealSense camera, model D455 for the Ubuntu 22.04 and Windows 11 operating systems based on 10 iterations with each having 100 frames.

Resolution	FPS	Latency (ms)							
		Ubuntu 22.04				Windows 11			
		Min	Max	Avg	Std	Min	Max	Avg	Std
424x240	30	68.0	71.0	70.4	1.1	91.0	93.0	91.1	1.9
	60	39.0	40.0	39.3	0.6	64.0	65.0	64.3	1.6
640x480	30	80.0	82.0	80.1	1.6	97.0	99.0	98.0	1.2
	60	44.0	48.0	45.1	1.9	66.0	69.0	67.2	2.3
848x480	30	83.0	87.0	85.2	1.8	102.0	109.0	104.0	2.9
	60	42.0	50.0	46.1	2.2	62.0	67.0	65.4	1.5
1280x720	30	99.0	102.0	100.2	2.6	113.0	122.0	115.2	3.2
	60	n/a	n/a	n/a	n/a	n/a	n/a	n/a	n/a

Second, we evaluate the scalability of the individual, fixed quality 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 the systems' scalability.

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 three 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 to achieve an objective score that correlates to the perceived quality. To get an accurate estimate, we have employed the 4G bandwidth traces of van der Hooft et al. [55] to generate varying qualities. Additionally, for the MDC encoder we show the achieved VMAF score of each of the seven qualities.

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 and fixed quality 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 [26]. 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 [40], 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)* [12].

# Clients	Latency (ms)		FPS
	Avg	Std	
5	14.9	5.8	29.1
10	32.4	16.3	18.3
20	56.0	29.2	9.7
40	97.8	51.3	5.5

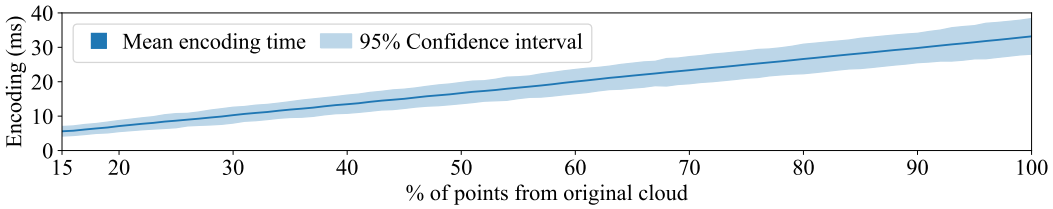


Fig. 10. Linear scaling of encoding time with sampling rate for point clouds of 125.000 points [12, 43].

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 10 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 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 workers 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 workers proportionately to the number of clients. Additionally, the implementation of a circular buffer for workers enables the encoding of consecutive frames whilst certain clients are still encoding the previous frame. On a machine with the following specifications: *CPU: Intel i7-1265u (4.8GHz Turbo), RAM: 16GB DDR4 (3200 MHz)* and five simulated clients, we achieve similar encoding times by employing only three threads, with an average of 18.3 ms (std=4.6 ms). From this, we infer that an equal amount of workers is not requisite for the number of clients and that the number of workers can be estimated with the following equation:

$$n_{\text{threads}} = n_{\text{clients}} \times \frac{\text{mean encoding time}}{\text{inter-frame time}} \tag{1}$$

However, under the assumption of a singular machine with a capacity of twelve threads, the system is still only able to service a modest number of clients. Additionally, as shown in Figure 10, the encoding scales linearly with the sampling rate, making the number of workers highly correlated with the requested number of quality representations.

The fixed quality encoding employs a limited number of quality levels to satisfy the needs of all clients. This greatly improves the scalability of the encoding side, as it is no longer restricted by the number of active clients. However, the fixed quality encoder is still limited by two different factors. A first constraint is the time required to encode the highest quality level, which in these

Table 4. FPS and encoding latencies for the fixed encoder with the number of qualities varying from 3 to 7.

#Qualities	FPS		Encoding Time		10% Peak Encoding		1% Peak Encoding	
	Avg	Std	Avg	Std	Avg	Std	Avg	Std
3	30.5	1.8	33.0	2.1	37.5	2.7	43.6	4.1
4	30.5	1.8	33.0	2.1	37.5	2.5	43.4	3.9
5	30.1	2.0	33.4	2.6	38.8	3.4	46.3	6.1
6	25.5	1.7	39.5	2.9	45.5	3.7	53.2	7.4
7	20.7	1.2	48.6	3.3	54.8	4.4	63.1	9.8

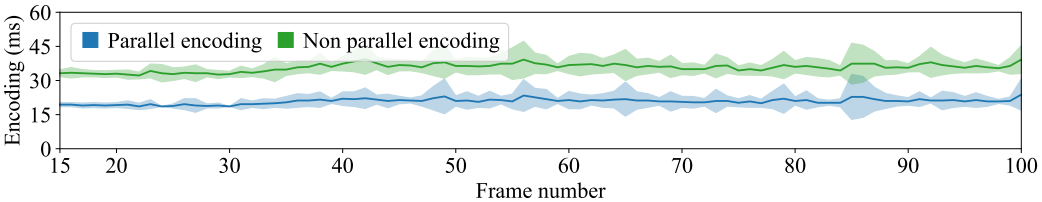


Fig. 11. MDC encoding approaches are improved by encoding the three descriptions in parallel, using one worker for each description.

experiments uses a sampling rate of 100%. As shown in Figure 10, with the given hardware and camera configuration, it takes an average of 33.2 ms (std=2.68 ms) to encode the highest quality level. Although this duration matches the inter-frame interval, maintaining a consistent 30 FPS can be challenging as larger frames may take longer to encode. The second constraint is the number of available workers. Similar to individual encoders, the use of only three workers limits either the maximum number of quality levels or the highest achievable FPS, as indicated in Table 4. Given the linear scalability of the encoding time and the use of equally spaced sampling rates, the pipeline can encode $2w - 1$ (where w represents the number of workers) qualities without adding extra latency. In these experiments, this configuration allows for five quality levels to be encoded at approximately 30 FPS. However, this results in nearly all workers being occupied, thus unavailable for other tasks if the pipeline would be extended.

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. The maximum description now encodes considerably quicker than the top quality level utilized by the fixed quality encoder, requiring only 19.9 ms as opposed to 33.2 ms. 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 11 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.

Figure 12 illustrates the encoding time (30 iterations, sequence of 1800 frames) required for all used encoding approaches. Comparing the average preprocessing time of the individual encoder against the MDC encoder, we observe that the individual encoder (five clients, three workers) attains an average time of 14.9 ms (std=5.8 ms) versus the MDC encoder (three descriptions, three workers) which achieves an average of 19.9 ms (std=1.7 ms). However, 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 the moment the number of clients starts increasing. Using Figure 10,

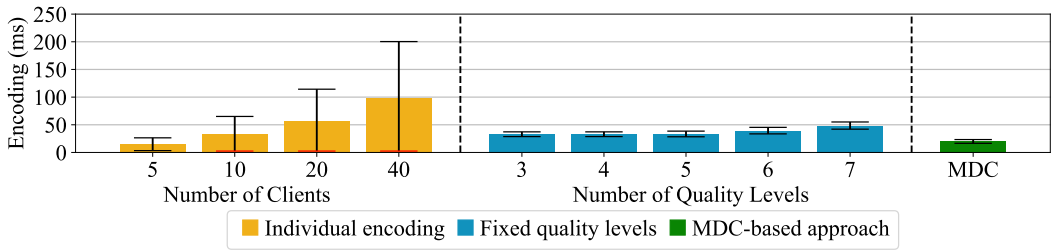


Fig. 12. The encoding time of the individual encoder depends on the number of clients whilst the fixed quality encoder depends on the number of representations. The MDC-based approach depends on neither. Confidence intervals are indicated by the vertical line, red caps indicating that the value was clipped to 0.

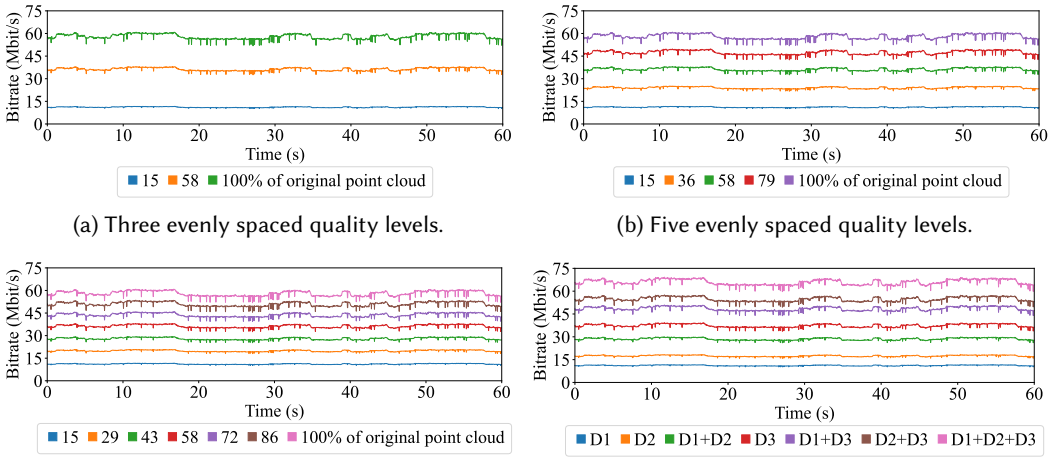
this maximum encoding time can increase up to 33 ms, a latency significantly more than MDC-based solution. One approach to mitigate this problem would be the implementation of MDC into the individual adaptation method. A viable strategy could entail imposing a limit which only allows 50% of the point 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.

In contrast, fixed quality encoding does not depend on the number of clients but rather on the quantity of required quality levels. The MDC-based encoder scales significantly better with both the number of clients and the required number of quality levels. Being able to provide seven quality levels using three distinct descriptions with an average encoding time of 19.9 ms (std=1.7 ms), whereas having seven fixed quality levels would take 48.6 ms (std=3.3 ms) when utilizing three workers. Additionally, by splitting the point cloud into smaller, faster encodable descriptions, the MDC-based method remains faster than the fixed quality encoder even for a larger number of workers ($n >= 7$). Furthermore, the MDC-based encoder is able to encode larger point clouds while still maintaining 30 FPS, unlike the fixed quality encoder, which struggles with larger point clouds.

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. As shown in Figure 13d, the MDC approach produces a well-distributed bitrate ladder that is comparable to using seven separately encoded quality levels. Nevertheless, this strategy incurs extra overhead on the total bandwidth since Draco can encode denser point clouds more efficiently with the fixed encoding approach. However, assuming that the number of workers is limited to three and that the volumetric video needs to be processed in real time at 30 FPS, it is only possible to support a maximum of five quality representations, of which the resulting bitrates are shown in Figure 13b. Figure 13a illustrates that when both the MDC-based approach and the fixed quality encoder utilize three base qualities, the gap between the different quality levels is significantly wider than for the MDC-based approach. These large jumps result in coarser quality transitions and can cause sudden data bursts within the network, increasing queuing times for other competing flows. However, with five quality levels (as depicted in Figure 13b), these gaps narrow and become more comparable to the behavior of the MDC-based approach. Table 5 shows the resulting bitrates for each MDC approach. It is clear from this table that each 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 this safeguard bandwidth



(a) Three evenly spaced quality levels.

(b) Five evenly spaced quality levels.

(c) Seven evenly spaced quality levels.

(d) MDC-based encoding with D1=15%, D2=25% and D3=60% of the original point cloud.

Fig. 13. Having only three fixed quality representations causes large jumps in the bitrate ladder compared to the smooth behavior of having seven fixed representations. Available bitrates for the MDC-based approach have a similar spread as the seven qualities.

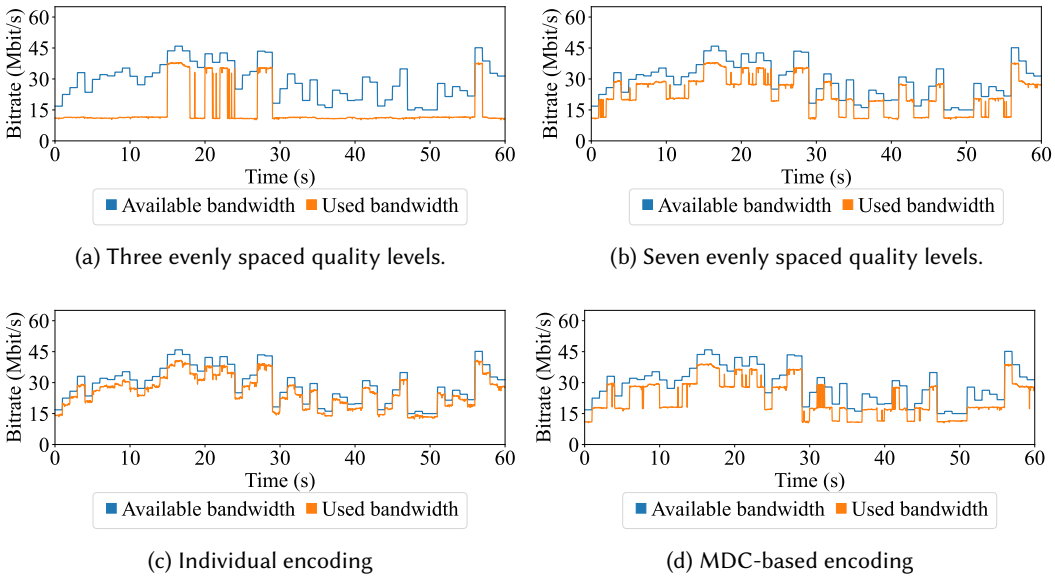
Table 5. Resulting average bitrates (Mbit/s) for the different MDC quality representations, with 15%, 25% and 60% being the base descriptions used to generate the other representations [12].

	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

usage hovers above 100%, causing intolerable packet loss. Although the fixed quality and 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 3.2 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 75.0%, which comes down to 6.9 Mbit/s being wasted. This result is comparable to the usage of seven fixed quality levels, leading to a utilization of 75.1% and a waste of 6.8 Mbit/s. Figure 14 illustrates this behavior, demonstrating that both the MDC approach and the seven fixed layers approximate the performance of the individual encoder. Additionally, the figure shows that employing only three fixed qualities would cause a significant underutilization of available bandwidth and potential quality loss, highlighting the necessity of providing a larger number of quality representations. Decreasing the number of quality levels to five results in comparable outcomes, with a usage of 67.8% and a waste of 8.9 Mbit/s. However, it is important to note that these results do not factor in the overhead caused by the MDC-based encoding. To fully understand how effectively the available bandwidth is being used, the visual quality must also be considered.

4.6 VMAF Quality

VMAF allows us to measure impact of increasing the point size in order to increase the perceived quality. Table 6 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



(a) Three evenly spaced quality levels.

(b) Seven evenly spaced quality levels.

(c) Individual encoding

(d) MDC-based encoding

Fig. 14. Resulting bitrate for a bandwidth trace using the encoders, illustrating their ability to approximate the available bandwidth.

frames of the original point cloud. In rendering the original cloud, a point size of 0.30 (Unity units) was adopted, as lower values introduce gaps in the point cloud when viewing it from the fixed viewpoint. From Table 6, 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. Due to the sampling rates of the fixed quality encoders being close to those of the MDC-based approach, the closest sampling rate from Table 6 is used to determine the point size for the fixed quality representations.

Table 7 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 can see that the MDC-based approach scores better when at least twenty clients are involved (see Table 8, sequence of 1800 frames). This phenomenon is explained by the fact that the used point cloud sequence 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 the same behavior. From the data in Table 8, we conclude that the MDC approach works well in environments with many clients or for volumetric videos that suffer from a low FPS.

Figure 15a shows the VMAF scores for the fixed quality encoder. These scores are calculated independently of the time required to encode the frames. As such, when limiting the workers to three, the resulting FPS for the fixed quality has to be decreased to ensure real-timeliness. This results in a slight decrease in the VMAF scores, which brings the performance of the MDC-based encoder closer to the fixed quality encoder. Figure 15b illustrates the performance of both encoders in comparison to an ideal encoding method that always selects the highest possible sampling rate

Table 6. VMAF scores for all quality and size combinations for point sizes 0.03 to 0.09. In bold, the highest score for each sampling rate is highlighted, indicating the optimal point size for the sampling rate [12].

	15%		25%		40%		60%		75%		85%		100%	
	Avg	Std	Avg	Std	Avg	Std	Avg	Std	Avg	Std	Avg	Std	Avg	Std
0.03	5.0	2.7	24.1	2.9	47.4	2.7	68.6	2.2	**78.7**	**1.6**	**83.4**	**1.3**	**88.1**	**0.8**
0.04	22.0	2.9	44.5	2.7	65.3	1.9	**75.8**	**1.1**	77.7	0.8	77.7	0.7	77.2	0.8
0.05	34.0	2.9	55.3	2.1	**67.6**	**1.1**	69.0	1.0	67.9	1.0	67.0	1.0	65.6	1.1
0.06	41.5	2.6	**57.2**	**1.4**	61.2	1.1	59.1	1.2	57.4	1.2	56.5	1.3	55.2	1.3
0.07	**44.7**	**2.1**	52.3	1.5	50.9	1.3	47.7	1.5	46.0	1.6	45.0	1.6	43.8	1.6
0.08	42.9	1.7	44.3	1.5	40.8	1.6	37.7	1.7	36.0	1.8	35.1	1.8	34.1	1.9
0.09	37.9	1.6	35.8	1.7	31.5	1.7	28.7	1.8	27.3	2.0	26.5	1.9	25.7	1.9

Table 7. VMAF scores for MDC approaches with fixed quality [12].

	15%		25%		40%		60%		75%		85%		100%	
	Avg	Std	Avg	Std	Avg	Std	Avg	Std	Avg	Std	Avg	Std	Avg	Std
Fixed Size	44.5	2.1	57.1	1.7	67.7	1.2	75.8	1.1	78.7	1.6	83.2	1.3	88.1	0.8
Fixed Bitrate	50.0	1.8	61.0	1.3	69.4	1.0	73.0	1.4	76.3	1.6	83.9	1.2	89.3	0.6
Percentage	46.6	1.9	58.6	1.5	68.6	1.0	76.7	0.8	81.7	0.7	85.9	0.6	89.8	0.5

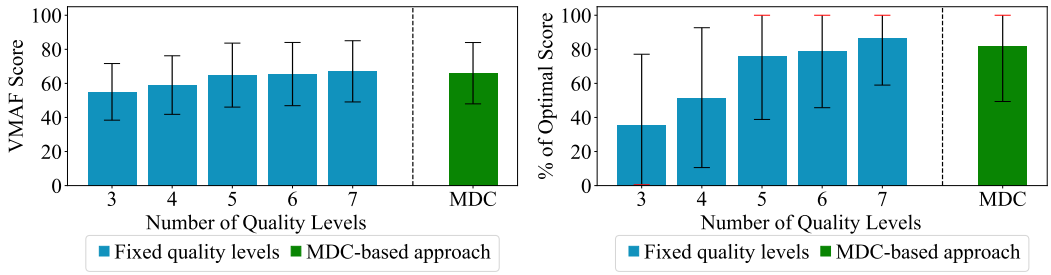
Table 8. VMAF scores for a sequence of 1800 frames for both MDC-based approaches versus individual encoding, using three descriptions for the MDC-based approaches [12].

	# Clients	FPS	VMAF Score	
			Avg	Std
Individual Adaptive Frame Rate	5	30	68.0	11.9
	10	20	67.7	13.5
	20	10	65.9	16.5
	40	5	58.5	19.2
Fixed Size	n	30	66.0	9.0
Fixed Bitrate	n	30	65.1	11.6
Percentage	n	30	66.6	7.4

based on the available bandwidth, thereby maximizing bandwidth utilization. While this method is not due to the unknown sizes for each sampling rate, it provides a good benchmark for how well the tested encoders can approximate this ideal scenario and utilize the available bandwidth. As depicted in Figure 14a, the limited granularity of the bitrate ladder with three qualities severely limits the encoder's ability to utilize the available bandwidth. This limitation is also evident in the results shown in Figure 15b, indicating that using three fixed quality levels performs much worse compared to the MDC-based approach. The resulting VMAF scores for the MDC-based approach is comparable to the scores of the fixed quality encoders with six and seven distinct qualities.

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



(a) VMAF Score. (b) Percentage of Optimal VMAF Score.

Fig. 15. VMAF scores for fixed quality and MDC encoders, indicating that the MDC encoder with three distinct descriptions approximates the quality of an encoder with seven fixed qualities. Confidence intervals are indicated by the vertical line, with red caps indicating that the value was clipped between 0 and 100.

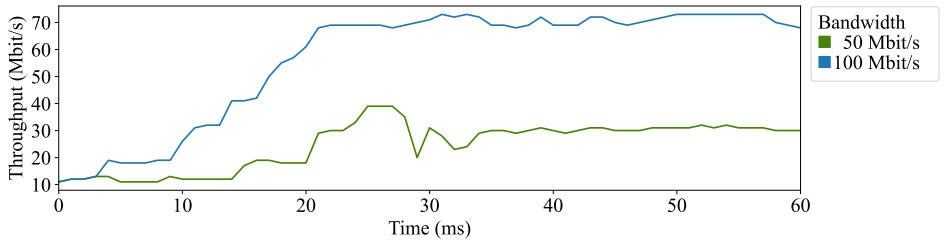


Fig. 16. Throughput obtained when using the fixed size MDC approach for 50 Mbit/s and 100 Mbit/s [12].

bandwidth makes it easier for the congestion controller to converge. Figure 16 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 5, we observe that this result corresponds to the bitrate of the highest quality. 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 achieve the bitrate of the 60% representation. Figure 16 indicates that this behavior is due to the GCC algorithm not recovering after an overestimation which subsequently caused packet loss. By analyzing both figures, we 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 5, this corresponds to a quality level of 40% with an average VMAF score of 67.56.

In terms of latency, the 100 Mbit/s link averages a transport latency of 30.9 ms (std=10.2 ms), compared to 24.4 ms (std=5.7 ms) for the 50 Mbit/s link. 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, indicating that the GCC algorithm ensures a good balance between estimated bandwidth, latency and potential packet loss.

As illustrated by Figure 17, it is necessary to have at least five evenly distributed quality levels to achieve the throughput needed for the highest quality. This requirement arises from significant jumps in bitrates, which cause sudden latency spikes. The GCC implementation cannot manage these spikes effectively, because it relies on inter-packet latency to detect sudden congestion and cannot discern whether the increase is caused by the application or congestion. Consequently, having a higher number of quality levels gradually increases the latency, ensuring the inter-packet latency increase remains stable, as shown in Figure 18d. Reducing the available bandwidth to 50 Mbit/s shows similar behavior and requires at least five fixed quality levels before acceptable bitrates are achieved. This behavior can be derived from Figure 13 which indicates that the granularity of the

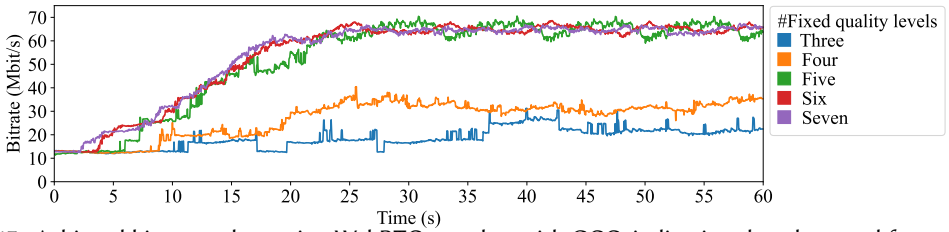
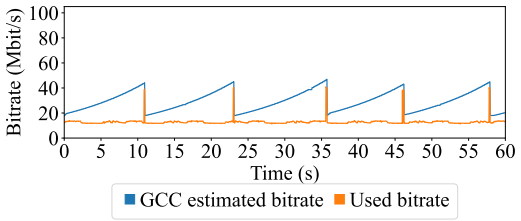
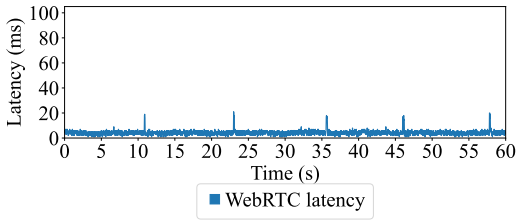


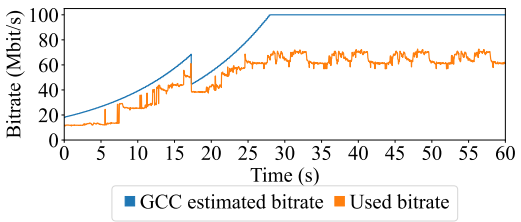
Fig. 17. Achieved bitrates when using WebRTC together with GCC, indicating that three and four quality levels are insufficient to achieve the throughput required for high quality.



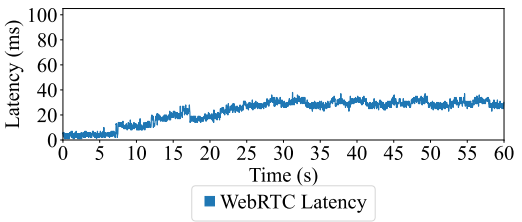
(a) Estimated + Used Bitrate for Three Quality Levels.



(b) WebRTC Latency for Three Quality Levels.



(c) Estimated + Used Bitrate for Five Quality Levels.



(d) WebRTC Latency for Five Quality Levels.

Fig. 18. Resulting bitrate, estimated bitrate and latency for a single WebRTC run, illustrating the impact of the latency spikes on the throughput and bitrate estimated by GCC when using a 100 Mbit/s network link.

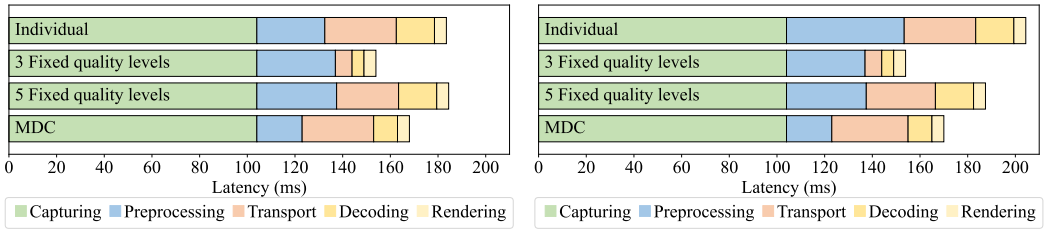
bitrate ladder with three qualities is insufficient. However, for encoders with five or more qualities the gaps between the qualities have decreased enough to allow a smoother increase in latency.

With the 100 Mbit/s link, the encoder with five fixed quality levels achieves a throughput of 65.2 Mbit/s (std=3.5 Mbit/s) with an average latency of 29.4 ms (std=2.6 ms). These results are comparable to the results of the MDC-based approach. This is also observed when reducing the bandwidth to 50 Mbit/s, which results in a throughput of 43.8 Mbit/s (std=1.5 Mbit/s), which corresponds to a sampling rate of 60% or a VMAF score of 75.8.

4.8 End-to-End Latency Breakdown

Figure 19 illustrates the complete end-to-end latency for all encoding approaches, with no artificially added link delays. For this experiment we have used the fixed frame sequence with three workers. 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 more optimized camera would yield immediate improvements for the overall latency. Overall, we observe that our pipeline streams point clouds captured at a frame rate of 30 FPS and a resolution of 848x480 pixels with a low enough latency to enable real-time communication. For an increased number of clients, the MDC-based approach maintains a stable latency compared to individual encoding, which has an increased encoding time



(a) Three clients (b) Eight clients

Fig. 19. Composite latency of each component in the proposed architecture, with no artificial link delays, using three worker threads.

due to the limited number of workers. It is noteworthy that the fixed encoder with three qualities achieves reduced latency because of the lower throughput. Compared to the fixed encoder with five qualities, the MDC-based encoder performs better in terms of latency. Additionally, both the MDC-based and fixed quality encoders scale similarly when the number of clients is increased.

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. Regarding encoding performance, the MDC-based encoder achieves a steady 30 FPS for 40 clients compared to 6 FPS of the individual encoder, and maintains this independent of the number of clients. Compared to per client encoding, we observe 9% lower end-to-end latency (167 ms vs 182 ms) with three clients and 19% (170 ms vs 204 ms) when increasing the number of clients to eight.

The fixed quality encoder with seven levels offers the same number of quality representations as the MDC-based approach with three distinct descriptions. Consequently, both encoders deliver comparable visual quality. When applied to a series of captured bandwidth traces, the MDC-based approach achieves a VMAF score of 65 compared to a score of 67 when using a fixed encoder with seven quality representations. Nevertheless, the fixed quality encoder demands more computational power to maintain real-time performance at 30 FPS. As a result, the seven-level fixed quality encoder only performs at 20 FPS (regardless of the number of clients). However, by reducing the number of qualities to five, it becomes feasible to stream at 30 FPS, resulting in an end-to-latency of 187 ms. In contrast, the MDC-based method achieves a consistent 30 FPS with an end-to-end latency of 163 ms, implying that the MDC-based encoder ensures better end-to-end latency by allowing the point cloud to be divided into smaller, more quickly encodable segments. We conclude that our pipeline produces an acceptable latency with quality comparable to the individual encoder in low-client environments, and produces higher average quality with more than 20 clients. Additionally, it achieves similar quality as having seven fixed qualities at the cost of some bandwidth overhead.

In future work, we will extend the architecture to allow for many-to-many streaming as well as supporting other protocols such as MoQ. Furthermore, the extension of multiple input clients will require improved encoder and adaptation algorithms to achieve acceptable bitrates and quality in scenes with a large number of clients. An extension will also be made to enable multiple cameras, which will be evaluated through VMAF scores for multiple viewpoints.

ACKNOWLEDGMENTS

This work has been funded by the European Union (SPIRIT project, Grant Agreement 101070672, https://www.spirit-project.eu/).

REFERENCES

[1] H. Afzal et al. 2018. Full 3D reconstruction of non-rigidly deforming objects. *ACM Transactions on Multimedia Computing, Communications, and Applications (TOMM)* 14, 1s (2018).
[2] S. J. Ahn et al. 2021. IEEEVR2020: Exploring the first steps toward standalone virtual conferences. *Frontiers in Virtual Reality* 2 (2021).
[3] D. Alexiadis et al. 2012. Real-time, full 3D reconstruction of moving foreground objects from multiple consumer depth cameras. *IEEE Transactions on Multimedia* 15, 2 (2012).
[4] S. Altamimi et al. 2020. QoE-fair DASH video streaming using server-side reinforcement learning. *ACM Transactions on Multimedia Computing, Communications, and Applications (TOMM)* 16, 2s (2020), 1–21.
[5] S. Arisu and A. Begen. 2018. Quickly starting media streams using QUIC. In *Proceedings of the 23rd Packet Video Workshop*. 1–6.
[6] M. A. Azad et al. 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.
[7] M. Bassier et al. 2020. Point cloud vs. mesh features for building interior classification. *Remote Sensing* 12, 14 (2020).
[8] K. Cao et al. 2020. Visual quality of compressed mesh and point cloud sequences. *IEEE Access* 8 (2020).
[9] G. Carlucci et al. 2016. Analysis and design of the google congestion control for web real-time communication (WebRTC). In *Proceedings of the 7th ACM Multimedia Systems Conference*.
[10] M. Catalan-Cid et al. 2020. FALCON: Joint fair airtime allocation and rate control for DASH video streaming in software defined wireless networks. In *Proceedings of the 30th ACM Workshop on Network and Operating Systems Support for Digital Audio and Video*. 14–20.
[11] Q. Dai and R. Lehnert. 2010. Impact of packet loss on the perceived video quality. In *2010 2nd International Conference on Evolving Internet*. IEEE.
[12] M. De Fré et al. 2024. Scalable MDC-Based Volumetric Video Delivery for Real-Time One-to-Many WebRTC Conferencing. In *Proceedings of the 15th ACM Multimedia Systems Conference*.
[13] M. De Fré et al. 2023. Master Thesis Dissertation: Low-Latency Volumetric-Video Delivery for Real-Time Conferencing. http://lib.ugent.be/catalog/rug01:003150149
[14] C. Debeunne and D. Vivet. 2020. A review of visual-LiDAR fusion based simultaneous localization and mapping. *Sensors* 20, 7 (2020).
[15] S. Dutton et al. 2012. Getting started with WebRTC. *HTML5 Rocks* 23 (2012).
[16] E. d'Eon et al. 2017. 8i voxelized full bodies-a voxelized point cloud dataset. *ISO/IEC JTC1/SC29 Joint WG11/WG1 (MPEG/JPEG) input document WG11M40059/WG1M74006* 7, 8 (2017).
[17] Golang 2023. https://go.dev/
[18] Google. 2023. Draco. https://google.github.io/draco/
[19] M. Gourlay and R. Held. 2017. Head-Mounted-Display Tracking for Augmented and Virtual Reality. *Information Display* 33, 1 (2017).
[20] D. Graziosi et al. 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 (2020).
[21] S. Gunkel et al. 2018. Virtual Reality Conferencing: Multi-user immersive VR experiences on the web. In *Proceedings of the 9th ACM Multimedia Systems Conference*. 498–501.
[22] J. Hacker et al. 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 (2020).
[23] S. Holmer et al. 2013. Handling packet loss in WebRTC. In *2013 IEEE International Conference on Image Processing*.
[24] X. Huang et al. 2024. Double Reference Guided Interactive 2D and 3D Caricature Generation. *ACM Transactions on Multimedia Computing, Communications and Applications* (2024).
[25] imec Virtual Wall 2023. https://doc.ilabt.imec.be/ilabt/virtualwall/
[26] Intel Realsense 2023. https://www.intelrealsense.com/
[27] J. Jansen et al. 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*.
[28] X. Jiang et al. 2018. Low-latency networking: Where latency lurks and how to tame it. *Proc. IEEE* 107, 2 (2018), 280–306.
[29] T. Jin et al. 2024. Meshreduce: Scalable and bandwidth efficient 3d scene capture. In *2024 IEEE Conference Virtual Reality and 3D User Interfaces (VR)*. IEEE, 20–30.
[30] I. Johansson and Z. Sarker. 2017. *Self-clocked rate adaptation for multimedia*. Technical Report.

[31] keijiro. 2023. Point cloud shader for Unity. https://github.com/keijiro/Pcx
[32] C. Koch et al. 2019. Transitions of viewport quality adaptation mechanisms in 360 degree video streaming. In *Proceedings of the 29th ACM Workshop on Network and Operating Systems Support for Digital Audio and Video*. 14–19.
[33] B. Li et al. 2013. Two decades of internet video streaming: A retrospective view. *ACM transactions on multimedia computing, communications, and applications (TOMM)* 9, 1s (2013).
[34] Z. Li et al. 2018. VMAF: The journey continues. *Netflix Technology Blog* 25, 1 (2018).
[35] J. Luo et al. 2017. Standardization of low-latency TCP with explicit congestion notification: A survey. *IEEE Internet Computing* 21, 1 (2017), 48–55.
[36] MDC-Based WebRTC Point Cloud Streaming 2023. https://github.com/idlab-discover/pc-webrtc-o2m
[37] Meta Quest 2 Virtual Reality Headset 2023. https://www.meta.com/quest/products/quest-2
[38] C. Müller et al. 2023. Machine-learning based VMAF prediction for HDR video content. In *Proceedings of the 14th Conference on ACM Multimedia Systems*. 328–332.
[39] M. Nguyen et al. 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.
[40] OpenCV 2023. https://opencv.org/
[41] S. Petrangeli et al. 2019. A scalable WebRTC-based framework for remote video collaboration applications. *Multimedia Tools and Applications* 78 (2019).
[42] Pion WebRTC 2023. https://github.com/pion/webrtc
[43] Point Cloud Sequence Used in Experiments 2023. https://users.ugent.be/~madfr/pc-streaming-mmsys2024/frames.zip
[44] K. Ragimova et al. 2019. Analysis of YouTube dash traffic. In *2019 IEEE International Black Sea Conference on Communications and Networking (BlackSeaCom)*. IEEE.
[45] I. Richardson. 2011. *The H. 264 advanced video compression standard*. John Wiley & Sons.
[46] O. Schreer et al. 2019. Capture and 3D video processing of volumetric video. In *2019 IEEE International conference on image processing (ICIP)*. IEEE.
[47] M. Schütz et al. 2016. Potree: Rendering large point clouds in web browsers. *Technische Universität Wien, Wiedeń* (2016).
[48] M. Schütz et al. 2023. GPU-Accelerated LOD Generation for Point Clouds. In *Computer Graphics Forum*, Vol. 42. Wiley Online Library.
[49] H. Schwarz et al. 2007. Overview of the scalable video coding extension of the H. 264/AVC standard. *IEEE Transactions on circuits and systems for video technology* 17, 9 (2007), 1103–1120.
[50] M. Seufert et al. 2014. A survey on quality of experience of HTTP adaptive streaming. *IEEE Communications Surveys & Tutorials* 17, 1 (2014).
[51] D. Tian et al. 2017. Geometric distortion metrics for point cloud compression. In *2017 IEEE International Conference on Image Processing (ICIP)*. IEEE.
[52] E. Torlig et al. 2018. A novel methodology for quality assessment of voxelized point clouds. In *Applications of Digital Image Processing XLI*, Vol. 10752.
[53] Unity 2023. https://unity.com
[54] S. Van Damme et al. 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.
[55] J. van der Hooft et al. 2016. HTTP/2-based adaptive streaming of HEVC video over 4G/LTE networks. *IEEE Communications Letters* 20, 11 (2016).
[56] J. van der Hooft et al. 2019. Towards 6dof http adaptive streaming through point cloud compression. In *Proceedings of the 27th ACM International Conference on Multimedia*.
[57] J. van der Hooft et al. 2023. A Tutorial on Immersive Video Delivery: From Omnidirectional Video to Holography. *IEEE Communications Surveys & Tutorials* (2023).
[58] C. Wang et al. 2017. Design and analysis of QoE-aware quality adaptation for DASH: A spectrum-based approach. *ACM Transactions on Multimedia Computing, Communications, and Applications (TOMM)* 13, 3s (2017).
[59] Y. Wang et al. 2005. Multiple description coding for video delivery. *Proc. IEEE* 93, 1 (2005), 57–70.
[60] Wowza. 2021. *2021 Video Streaming Latency Report*. Technical Report. Wowza.
[61] E. Zerman et al. 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.
[62] S. Zhou et al. 2021. Review of light field technologies. *Visual Computing for Industry, Biomedicine, and Art* (2021), 29.
[63] D. Zielinski et al. 2015. Exploring the effects of image persistence in low frame rate virtual environments. In *2015 IEEE Virtual Reality (VR)*.

Received 15 September 2024; revised 26 June 2025; accepted XXXX