MuV^2: Scaling up Multi-user Mobile Volumetric Video Streaming via Content Hybridization and Sharing

Yu Liu Puqi Zhou* Zejun Zhang Anlan Zhang Bo Han* Zhenhua Li+ Feng Qian
University of Southern California *George Mason University +Tsinghua University

ABSTRACT

Volumetric videos offer a unique interactive experience and have the potential to enhance social virtual reality and telepresence. Streaming volumetric videos to multiple users remains a challenge due to its tremendous requirements of network and computation resources. In this paper, we develop MuV^2, an edge-assisted multi-user mobile volumetric video streaming system to support important use cases such as tens of students simultaneously consuming volumetric content in a classroom. MuV^2 achieves high scalability and good streaming quality through three orthogonal designs: hybridizing direct streaming of 3D volumetric content with remote rendering, dynamically sharing edge-transcoded views across users, and multiplexing encoding tasks of multiple transcoding sessions into a limited number of hardware encoders on the edge. MuV^2 then integrates the three designs into a holistic optimization framework. We fully implement MuV^2 and experimentally demonstrate that MuV^2 can deliver high-quality volumetric videos to over 30 concurrent untethered mobile devices with a single WiFi access point and a commodity edge server.

CCS CONCEPTS

• **Information systems** → **Multimedia streaming**; • **Human-centered computing** → **Ubiquitous and mobile computing systems and tools**; **Mixed / augmented reality**.

KEYWORDS

Volumetric Video Streaming, Mobile Mixed Reality, Edge Computing, Quality-of-experience (QoE).

This work is licensed under a Creative Commons Attribution International 4.0 License.
ACM MobiCom '24, November 18–22, 2024, Washington D.C., DC, USA
© 2024 Association for Computing Machinery.
ACM ISBN 979-8-4007-0489-5/24/09…$15.00
https://doi.org/10.1145/3636534.3649364

ACM Reference Format:
Yu Liu, Puqi Zhou, Zejun Zhang, Anlan Zhang, Bo Han, Zhenhua Li, Feng Qian. 2024. MuV^2: Scaling up Multi-user Mobile Volumetric Video Streaming via Content Hybridization and Sharing. In *International Conference On Mobile Computing And Networking (ACM MobiCom '24), September 30–October 4, 2024, Washington D.C., DC, USA*. ACM, New York, NY, USA, 15 pages. https://doi.org/10.1145/3636534.3649364

1 INTRODUCTION

The rapid advancement of virtual reality (VR), mixed reality (MR), and augmented reality (AR) stimulates different formats of immersive and interactive experiences [9, 27, 42, 45]. Notably, the volumetric video stands out by offering an even more interactive experience compared with other immersive video formats such as 360° video. Volumetric videos are formed by sequences of 3D representations captured from real scenes, usually via point clouds [32] or meshes [43]. Its inherent 3D nature empowers 6-degree-of-freedom (6DoF) movement in both the rotational domain (yaw, pitch, roll) and translational domain (x, y, z). The unique 3D representation of volumetric video and its immersive experience opens up novel opportunities across various applications. For instance, volumetric videos can support telepresence and facilitate immersive remote meetings [11, 24, 49].

Despite its potential, streaming volumetric videos to mobile devices faces many challenges. Due to the 3D representation of volumetric videos, their data volume significantly surpasses that of traditional video formats. Moreover, there is a lack of an efficient volumetric content compression algorithm, further increasing the challenge of volumetric video streaming. State-of-the-art systems seek to reduce the data volume to be *directly streamed* [17, 33, 42], or leverage an edge server to perform remote rendering and *transcode* the volumetric video frames into a traditional 2D video view [14, 15, 37]. Recent emerging deep learning models such as *NeRF* [39] can also be used to convert RGB images to 3D views and facilitate volumetric video streaming. There are several studies targeting real-time live volumetric content streaming with direct streaming [20, 25, 41].

The above-mentioned systems target a single-user streaming scenario, ignoring the potential of volumetric videos for larger groups. In a *multi-user* scenario, the 6DoF interaction can bring a more tailored experience to users. For instance,

System	#. Users	Visual Quality	Limited By
DirectM Unicast [17]*	Very small	Highest	Bandwidth
DirectM Multicast [56]	Small	Highest	Bandwidth
TransM [15]*	Small	Low	Computation
MuV^2 (this work)	Large	High	N/A

* Extended by us from their original single-user systems.

Table 1: Summary of different multi-user volumetric video streaming systems.

in a large classroom, students can access volumetric videos (*e.g.,* a live-streaming lecture) on their own mobile devices, allowing each student to choose their preferred viewing angles and accommodate individual learning needs. Likewise, museums can offer introductory volumetric videos about the displayed objects for a group of tourists to enhance the experience. Other use cases of multiple users watching volumetric videos simultaneously include large-group entertainment, many-to-many immersive conferencing, and product demonstrations and advertisements. It is worth mentioning that for multi-user cases, mobile devices such as untethered headsets and smartphones are preferred.

Scaling up the aforementioned single-user systems to a multi-user setting while maintaining good video quality is highly challenging. For *direct streaming*, its high bandwidth consumption hinders the practical support of multiple users even with optimizations to reduce data volumes. Within this category, M5 [56] is to our knowledge the only multi-user volumetric video streaming system. M5 leverages mmWave multicast to improve scalability. However, due to the low throughput of wireless multicast [35] and the high data rate of volumetric videos, M5 can only support at most 5 users simultaneously. For *edge-assisted transcode systems*, they often suffer from lower visual quality due to network latency, or viewport mismatch [14]. Additionally, with each connected user, the incorporation of remote rendering and encoding introduces extra computation overhead. Consequently, this accumulation of the above overhead could surpass the capacity of the edge server, preventing the edge from supporting more users. Last but not least, using known human models to reconstruct volumetric content or leveraging deep learning-based approaches such as NeRF [39] can potentially improve the visual quality and thus boost the user experience. However, they incur high computation resource usage even when processing a single user's volumetric content stream, let alone in a multi-user setting.

The goal of this paper is to use a single commodity AP and edge server to support, for the first time, at least 20 to 30 concurrent live or on-demand volumetric video streaming sessions, which suits our targeting use cases such as classrooms and museums. Our design aims to leverage limited but heterogeneous resources while maintaining high and fair visual quality across users to achieve scalability. Specifically, our system design incorporates the following key principles: First, it fully utilizes the available network and computation resources, minimizing the chance of either becoming the bottleneck. Second, it decouples the computation resource demand from the number of users. Third, even after the decoupling, the edge's workload may still be excessive; our design thus also efficiently multiplexes the multi-user workload into the limited hardware capacity. To instantiate these principles, we design and implement MuV^2, an innovative edge-assisted streaming system that scales up multi-user volumetric streaming for on-demand and live streaming.

Content hybridization (§3.4). MuV^2 achieves a balance between network and computation resource utilization through a hybrid streaming approach. Recall that either direct streaming or transcode streaming only leverages one type of resource whose limited capacity can easily throttle the whole system. To overcome this issue, MuV^2 performs edge-assisted transcoding by default, and judiciously streams volumetric frames to selected users to compensate for their visual quality loss from transcoding. This hybrid approach not only improves the overall image quality but also reduces the risk of a monolithic resource bottleneck. To determine the actual frame type assignment policy, we conducted a measurement study (§3.2) and observed that users' distance to the volumetric content is a key factor affecting visual quality. Therefore, MuV^2 adaptively assigns volumetric frames within bandwidth capacity to users who are close to the video content.

Cross-user view sharing (§3.5). MuV^2 boosts the scalability on the user level with *Transcoded View-sharing*, which decouples the computation overhead from the number of concurrent users. Specifically, according to the available computation capacity, the edge server selects and shares each transcoded view with one or more viewers. Upon receiving the view, each user performs image warping [13] if needed to render a novel view based on their real-time viewport position. Note this sharing approach differs from those of prior multi-user VR/MR systems [34, 36]: MuV^2 targets view sharing of dynamic volumetric content, while prior work [34, 36] caches and reuses previously rendered frames of static scenes. The key challenge here is to determine which view(s) to render and how to share them in real-time. We formulate the problem of view selection as a modified K-median cluster problem and design an approximation solution based on the forward greedy algorithm [12]. To efficiently determine the distortion incurred by image warping, we derive a data-driven machine learning model (§3.2) that characterizes the performance of image warping from different views on volumetric videos.

Encoder Multiplexing (§3.6). We find that even with the above two optimizations, the number of transcoding sessions

may still exceed the number of available hardware encoders. MuV^2 thus incorporates a novel design allowing multiplexing transcoding sessions into a smaller number of encoders available on the edge. Specifically, MuV^2 allows view streams (each corresponding to a view-sharing group/cluster as described above) to time-share the same encoder, with the group-of-pictures (GOP) size of each stream properly configured to balance the latency and bandwidth usage. We also design a mechanism to handle dynamic changes of transcoding groups as users' views change.

Integrated Real-time Optimization (§3.3). The above three design aspects bear different goals and balance different tradeoffs. To coordinate them, we integrate them into a holistic, principled real-time optimization framework, which incorporates our design principles with an overarching goal of maximizing resource utilization and visual quality.

Implementation and evaluation (§5). We implement MuV^2 on an edge server and commercial smartphones with more than 6K LoC and demonstrate its efficiency and scalability through comprehensive evaluations. We compare MuV^2 with four multi-user live-streaming systems, two of which are extended from single-user state-of-the-art volumetric streaming systems [15, 17] to multi-user, and the other two are variants of MuV^2. We highlight our evaluation results as follows.

- MuV^2 can support more than 30 users at 1080P with a single commercial WiFi access point of ~450 Mbps capacity, significantly improving the scalability compared to baselines.
- View-sharing across users only introduces a small visual distortion compared to the baseline.
- Jointly streaming volumetric frames and transcode frames further improves the visual quality by 47% compared to streaming transcode views alone.
- An IRB-approved user study with 11 users indicates that MuV^2 can provide a better visual quality compared with transcoding streaming baselines with 90% of users giving positive feedback.

To the best of our knowledge, MuV^2 is the first scalable multi-user volumetric video streaming framework. MuV^2 makes it feasible to leverage affordable hardware such as a single WiFi AP and an edge server with a single commodity GPU to support use cases such as tens of students' mobile volumetric content consumption in a classroom. This research does not incur any ethical issues.

2 MOTIVATION

Streaming volumetric videos to multiple users is challenging given its demanding resource requirement. In this section, we first discuss the limitation inherent in extending single-user volumetric content streaming systems to a multi-user context. We then present a series of noteworthy observations related to users' engagement with volumetric videos. These insights serve as a pivotal motivation for our design of MuV^2.

Figure 1: Video quality when supporting different numbers of users (left to right: 5, 10, 20 users).

2.1 Extend Existing Solutions to Multi-user

Direct streaming. We can extend the direct streaming system to multi-user scenarios, referred to as DirectM, by streaming compressed volumetric video frames to each client. The volumetric video is rendered locally on each client, ensuring high visual quality and minimal latency. However, due to the lack of an efficient compression algorithm, the compressed volumetric video frame still consumes high bandwidth, preventing DirectM from achieving high scalability while maintaining high visual quality. Take ViVo [17] as an example. The average bandwidth consumption to losslessly stream a volumetric video is around 90 Mbps. For a commercial AP with a bandwidth capacity of 500 Mbps, ViVo can ideally support at most **5** users. To serve additional users, DirectM needs to reduce the video quality, compromising the overall viewing experience, as demonstrated in Figure 1. In summation, simply extending the existing direct streaming systems is insufficient to stream volumetric videos to a substantial user base while preserving optimal visual quality.

Transcode streaming. We extend the edge-assisted transcode streaming to multi-user, referred to as TransM, by remotely rendering and encoding 2D views for each connected client. TransM can support more users under the same network than DirectM due to its lower bandwidth requirement. However, TransM requires extra optimizations to maintain visual quality. Currently, there are two state-of-art solutions: *multi-view [37]* streams multiple 2D views, allowing the client to select the optimal view and maximize the visual quality; *image warping [15]* streams an RGB view with a corresponding depth map. Clients then perform image warping to generate novel views based on their real-time viewport position. Since we aim to maximize system scalability, TransM opts for the image-warping approach due to its lower resource requirement. For each client, TransM needs to render and encode two 2D images: one RGB view and one depth map. However, executing the rendering and encoding tasks on the edge server introduces extra computational overhead, restricting TransM from supporting an extensive number of clients. We benchmark the rendering and encoding performance on a desktop with an NVIDIA 2080Ti GPU and find

that it can achieve 480 FPS of encoding at 1080P resolution, which means it can only support up to 480\30\2≈**8** users simultaneously at 30 FPS. The number will further reduce if the video is encoded in higher resolution. These limitations underscore that a naive extension of the transcode streaming system does not constitute the optimal solution for a scalable multi-user streaming system with high visual quality.

Exhaustive Rendering. A potential approach to mitigate the computation overhead of the transcoding system is through exhaustive rendering, which pre-renders the 3D scene from all possible viewports [8, 36]. While this strategy does not impose any real-time computation requirement, it is not a viable solution for volumetric video. The prior applications of exhaustive rendering are for virtual reality, where the users stay in a static scene with limited movement range, resulting in a finite pre-rendering data volume. However, volumetric video is a series of dynamic content, with free 6DoF movements. Therefore, the data volume for pre-rendering volumetric videos is exponentially larger compared to VR applications, therefore it's not feasible to perform in the real world. Furthermore, as discussed above, the primary utility of volumetric videos revolves around live streaming and telepresence, where pre-rendering is simply impossible.

To summarize, transcoding and direct streaming only work well when the computation and network resources are sufficient, respectively. Decreasing the transcoded resolution (for TransM) or point density (for DirectM) can reduce resource consumption and improve scalability, but it may also downgrade the visual quality and the viewing experience. Both DirectM and TransM are thus falling short of supporting large-scale multiuser volumetric video streaming. Also, although upgrading the hardware infrastructure can improve the scalability of DirectM (with better WiFi support) and TransM (with more advanced GPUs), there is always an arms race between the hardware and visual quality. Considering that the human perception limit is 16K (15360 × 8640) [57], it remains challenging for today's hardware (and possibly those in the foreseeable future) to support high-quality volumetric video streaming to a large group of users.

2.2 Other Key Observations

Users' movement pattern. Previous studies have already shown an overall similar movement pattern when watching the same volumetric video across users [17]. We further explore the movement pattern similarity across users on a frame-wise level. We evaluate the similarity of users' traces (details in §5.1) by implementing Kmeans [48] clustering, as shown in Figure 2. More specifically, we cluster all users' viewpoints on each frame and calculate the silhouette coefficient [47], which measures the degree of similarity between clusters. By analyzing the distribution of frames' silhouette coefficients,

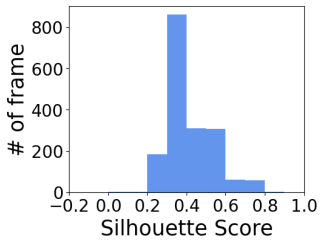

Figure 2: The distribution of frames' silhouette coefficient with number of clusters = 6 .

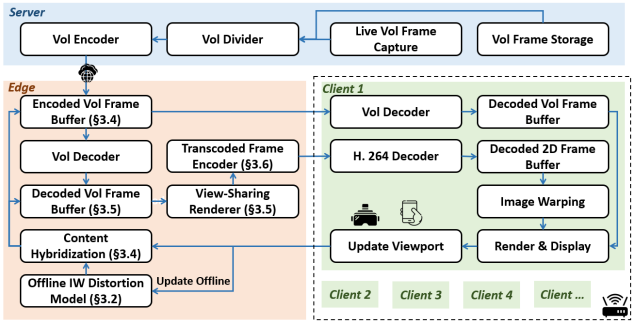

Figure 3: System Architecture Design of MuV^2

we observe that the overall average is 0.41, with more than 35% of frames having silhouette coefficients greater than 0.5. This suggests that users exhibit similar movement patterns, highlighting a frame-level similarity in watching the same volumetric video. Despite this similarity, directly using one user's view to approximate another user still yields quality issues because the shared view from another user will still be different from their actual view. Therefore, further optimization is required to improve viewing quality.

Performance of image warping. Image warping is a computer vision technique that generates novel views from the input RGB images and depth maps. With image warping, users can re-project the received transcoded view based on their real-time viewport, compensating for users' head movements during network transmission and inaccurate viewport prediction. There are also deep learning-based solutions (e.g. NeRF [10, 39]) that can generate novel views without requiring a depth map. However, these alternatives demand higher computational capacities at the client and could entail prolonged processing times. Therefore, we choose the light-weighted image warping on the client side for our study. Despite its ability to generate a novel view, image warping may introduce visual distortions and omissions along the margin of the objects in the scene, especially when the input view and targeting novel view are less similar, or when the input view is too close to the video content (§3.2, Figure 5). The distortion of image warping underscores the need for further enhancements to optimize visual quality and better support volumetric video streaming.

3 SYSTEM DESIGN

3.1 Overview

The goal of MuV^2 is to address the challenges of scalable and high-quality multi-user volumetric video streaming under resource limitations. To achieve this, we design three main optimization strategies for MuV^2: content hybridization, transcoded view-sharing, and encoder multiplexing. **Content Hybridization** improves visual quality by adaptatively streaming volumetric frames to several users who may encounter higher distortion with transcoded frames, maximizing the utilization of both resources. **View-sharing** decouples the overall computational requirement from the number of concurrent clients by only rendering a selected set of views on the edge and sharing the views across all clients. Upon receiving, each client performs image warping to re-render the received 2D view based on their real-time viewport position. Through view-sharing, MuV^2 addresses the challenge of the high computational requirement on the edge server. **Encoder multiplexing** enables multiple encoding sessions with a single encoder instance by dividing transcoded frames into smaller groups of pictures and time-sharing the same encoder.

We present the system design of MuV^2 in Figure 3. As shown in the figure, MuV^2 consists of a content server, an edge server, and multiple clients that are geometrically close to each other and wirelessly connected to the edge server through the same Wi-Fi Access Point (AP). Note that despite our system focusing on the scenario where all users are connected to the same AP, the main algorithms and concepts of MuV^2 can also be modified to apply to situations where users are connected to different APs. The content server supports both on-demand and live content streaming, and streams encoded volumetric videos to the edge. The edge server performs view-sharing and hybrid streaming while optimizing the overall visual quality for clients. Upon receiving, the client performs image warping on transcoded views or directly renders the volumetric data based on their real-time viewport.

3.2 Image Warping Efficiency Profiler

One of the key features of MuV^2 for improving the system scalability is **view-sharing**. This is realized via image warping (IW). As discussed in §2.2, IW will incur inevitable visual distortion and impact users' perceived quality. Therefore, MuV^2 needs to carefully select the shared views to ensure overall high visual quality is maintained. To achieve an optimal selection, we need to enable an efficient and accurate way to profile the visual distortion of IW, denoted as δ_{IW}, to guide the view-sharing decision in §3.5. We measure the distortion δ_{IW} with the SSIM value [51] drop of the display

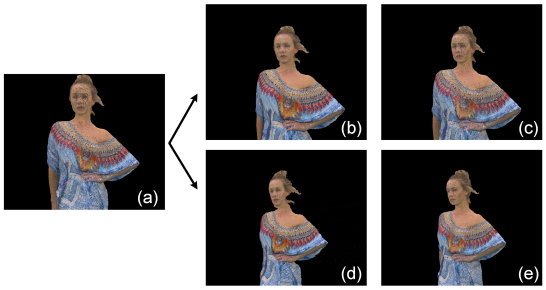

Figure 4: Image warping performance with reference viewport being 2.5m from video content. a) reference view; b) IW result of a closer display view; c) ground truth of b; d) IW result of a further display view; e) ground truth of d.

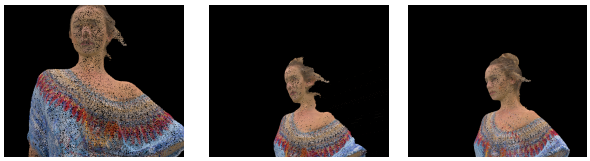

Figure 5: Image warping performance, reference viewport is 1.5m from video content.

view compared to its ground truth (where SSIM ≡ 1):

$$\delta_{IW} = 1 - SSIM(V_d, V_{r\to d}) \quad (1)$$

We use V_d to denote the ground truth of the targeting display view, V_r to denote the input reference view, and $V_{r\to d}$ to denote the output view using IW to generate V_d from V_r. Due to the limited rendering capacity of the edge server, we cannot perform IW on the edge server to calculate the distortion on time. Alternatively, we seek to generate an offline model to profile the distortion.

We identify two features impacting visual distortion: between **viewport distance** (denoted as $dist_{r,d}$) and **video radius** (R). We demonstrate the impact of viewport distance in Figure 4 with the same reference view. As shown in the figure, when V_d and V_r are closer to each other (Figure 4 b, c), the output view is less distorted with $\delta_{IW} = 0.048$. When $dist_{r,d}$ becomes larger, (Figure 4 d, e), the distortion value also increases to 0.052. Video radius (R_v) captures the feature of the video content. For the videos that feature one or more humans, we approximate the video as a cylinder with more than 95% of the points scattered within a certain radius R_v.

Based on the above two features, we measure the IW distortion with all potential combinations of the features ($V_r \times V_d \times R_v$). The viewports are within a 6m range from the video content with a stepsize of 0.1m, based on previous studies [17, 37, 53]. In total, we collected more than 500K data points with three different volumetric videos. We will introduce more details of the videos in §5.1. From the collected

Model	Accuracy (1-MAPE)	Pearson	Spearman	R2
LR	0.8226	0.5266	0.4975	0.2773
PR-2	0.8724	0.7567	0.7787	0.5725
PR-3	0.9098	0.8720	0.8511	0.7604
KNN	0.9582	0.9344	0.9316	0.8729
DTR	0.9921	0.9958	0.9968	0.9917

Table 2: Performance of different learning models.

data, we observe that the distance to the video content also has a significant impact on the visual quality. We illustrate this impact with another example in Figure 5 with the reference view closer to the video. Compared to d, e in Figure 4, the IW distortion of Figure 5 increases to 0.090 even though they have the same $dist_{r,d}$.

Next, we test the following machine-learning models with cross-validation and compare their performances for evaluating δ_{IW}: Linear regression (LR), Polynomial regression (PR-2 and PR-3), K-nearest neighbor regression (KNN), and decision tree regression (DTR). We use V_r, V_d, and R_v as the features, and δ_{IW} as the output. We show their qualitative performance in Table 2. We use the mean absolute percentage error (MAPE) to indicate the accuracy of each model. We also present the Pearson [7] and Spearman [5] values to show the correlation between models' output and ground truth, and R^2 score [30] to measure each model's ability to interpolate the data. For all metrics, a value closer to 1 indicates a more preferred model. We find that the DT regression yields the best score across all metrics and therefore is selected in MuV^2 to evaluate the efficiency of image warping. Since our model contains a feature for the video content, our model can be applied to videos with similar content formats.

3.3 Generic Optimization Formulation

One of the key challenges for MuV^2 is how to intelligently utilize the limited resources and maximize the visual quality and scalability. This problem is similar to rate adaptation in regular video streaming systems, where the server needs to decide the resolution of each video clip for the user based on the network condition and the user's buffer capacity [21, 26]. However, the resource allocation algorithm in MuV^2 is more complicated. MuV^2 needs to balance the different bandwidth consumption and visual quality of volumetric and transcoded frames to maximize resource utilization. Moreover, for the transcoded views, due to the limited computation capacity, the edge of MuV^2 needs to select an optimal set of views to share across users such that the visual quality of all users is maximized. This is a non-trivial problem to solve, as we will introduce in §3.5. Finally, MuV^2 needs to encode the views for multiple users under the hardware encoding capacity. We present a generic cost function that characterizes the visual quality and bandwidth requirement for MuV^2, then present our three optimizations based on the cost function.

Notation. We use $\delta(i)^t$ to denote the decision for hybrid streaming: $\delta(i)^t = 1$ represents sending a transcoded frame to user i on frame t, and $\delta(i)^t = 0$ represents a volumetric frame. We denote V_r^t as the set of views selected to render and encode at the edge server that shares among users with $\delta(i)^t = 1$. V_p^t represents the set of viewports from users at frame t, where each user's viewport is denoted as $V_{p,i}^t$. Note that due to network transmission latency, V_p^t is not the ground truth viewport but a predicted viewport. For user i, We chose $V_{r\to i}^t \in V_r^t$ as its reference viewport.

Cost Function Factors. To derive the resource allocation algorithm, we first define the optimization problem to be solved with a cost function that evaluates the visual quality. Our cost function contains the following components:

Visual Distortion. Visual distortion measures the quality drop compared to local rendering. For 2D views, the distortion is caused by image warping and is evaluated with the model we generated at §3.2. For 3D views, the visual distortion is always 0 since this is equivalent to $\delta_{IW}(V_d, V_d) \equiv 0$. We then define the visual distortion as follows:

$$DSTR_i^t = \delta(i)^t * \delta_{IW}(V_{r\to i}^t, V_{d,i}^t) + (1 - \delta(i)^t) * 0 \quad (2)$$

Quality Switch. The quality switch evaluates the quality difference between consecutive frames. It is a commonly considered factor in the quality function of regular video streaming as well [37, 54]. The quality of consecutive frames should not differ too much, as it will also lower the viewing experience. In MuV^2, we denote the quality switch as the difference in visual distortion:

$$SWI_i^t = DSTR_i^t - DSTR_i^{t-1} \quad (3)$$

Fairness. For multi-user systems, maintaining fairness across users is also crucial [29, 36, 46]. In MuV^2, we would like to maintain a similar visual quality across all users, with no user experiencing large visual distortion while others are having a lower distortion. We evaluate fairness with Jain's fairness index [23]:

$$FAIR^t = \frac{(\sum_{i=1}^n DSTIR_i)^2}{n\sum_{i=1}^n DSTR_i^2} \quad (4)$$

Bandwidth consumption As described in §2.1, one of the bottlenecks for direct streaming is the high bandwidth consumption. Therefore, it's crucial that we profile the bandwidth consumption of both 3D and 2D frames and allocate the available bandwidth accordingly. We denote the bandwidth consumption of each user at frame t as

$$BW_i^t = \delta(i)^t * Sizeof(2D) + (1 - \delta(i)^t) * Sizeof(3D) \quad (5)$$

Optimization problem. We formulate our optimization problem for the resource allocation algorithm with our cost function as follows:

$$\min Cost(V_r^t, \delta^t) = \alpha \sum_{i=1}^{n} DSTR_i^t + \beta \sum_{i=1}^{n} SWI_i^t + \gamma FAIR^t \quad (6)$$
$$s.t. \sum_{i}^{n} BW_i^t < BW_{total} \quad \& \quad |V_r^t| \leq m$$

where α, β, and γ are weights for each cost factor. BW_{total} is the total available bandwidth for all users. m is the computation capacity limit, *i.e.*, the maximum number of concurrent encoding tasks.

Fast search for solution. Due to the fast-changing nature of users' head movements, the resource allocation algorithm needs to make a real-time decision for each user on each video frame based on their viewport movement. It is essential that the algorithm can perform a fast search for the optimal solution. However, considering the large solution space, the dynamic nature of human movement, and the real-time constraint, it is difficult to analyze and solve the optimization problem as a whole. Therefore, we consider decomposing the resource allocation problem into three sub-problems (§3.4, §3.5, and §3.6) and solving each subproblem separately. Note that this decomposition strategy may not achieve a "global optimum". Instead, it makes MuV^2 scalable and practically effective.

3.4 Content Hybridization

MuV^2 maximizes resource utilization by hybrid streaming volumetric video frames to compensate for the default transcoded frames. Since volumetric frames inhibit higher visual quality but also higher bandwidth requirements, the overarching goal of content hybridization is to intelligently assign volumetric frames to some users under limited bandwidth while maximizing the overall visual quality. In other words, we need to solve the decision of $\delta(i)^t$. Note that MuV^2 performs content hybridization before other optimizations because we should always strictly enforce the bandwidth constraint to avoid any potential stall [45].

MuV^2 applies a similar tile-based visibility optimization as ViVo [17] to reduce bandwidth requirements for volumetric frames. Each volumetric frame is divided into several tiles with different quality levels. The edge will select tiles and quality levels based on users' visibility of the volumetric video while maintaining the same visual quality.

To maximize the overall visual quality, MuV^2 assigns volumetric frames to the users that are more likely to encounter higher distortion if receive transcoded frames. Finding those users can be non-trivial: to achieve an accurate search, we need to first decide the distortion of each user by finding the optimal reference view set, then identify those high-distortion users. However, this will require extra calculation steps as we need to re-calculate the optimal reference viewport set for the rest of the users. To perform a faster search, we apply a heuristic approach based on the observation we made in §3.2: Users who are closer to the video content are more likely to experience higher distortions and therefore should be assigned volumetric frames.

We conclude our algorithm for network allocation as follows: First, we decide the tiles needed for each user based on their viewport and calculate the corresponding bandwidth requirement; then, we search from the closest user to the furthest user and find the top q users ($q \leq n$) such that $\sum_{i=1}^{q} Sizeof(3D)_i + \sum_{i=q+1}^{n} Sizeof(2D)_i < BW_{total} - BW_{enc}$. Here the BW_{enc} is a small bandwidth budget we reserve for the encoding optimization (§3.6). The size of the transcoded frame is profiled with average 2D frame sizes. After performing the bandwidth allocation algorithm, the top q users are assigned volumetric frames and are excluded from the following optimizations. In a rarer case where the BW_{total} is too low, even sending transcoded frames to all users ($q = 0$) will still exceed the total bandwidth, we should consider reducing the resolution of the transcoded frame. Note that this is the last resort that we can try to fulfill the bandwidth requirement, as it further downgrades the visual quality.

3.5 User-level View Sharing

After assigning volumetric frames, the next question is to find the optimal set of views to render and encode on the edge server, denoted as V_r, for **view-sharing**. The size of V_r, denoted as m, is limited by the rendering and encoding capacity of the edge server. To reduce the overall bandwidth consumption, each user will only receive one view from m views. In other words, for **view-sharing**, we "cluster" users into m groups and select one reference view for each cluster.

Despite the simple formulation, finding the solution is non-trivial and faces the following challenges: first, the search space for this problem is theoretically infinite. Anywhere in the 3D space and any rotational angles should be in our search space. To make the optimization feasible, we chose to reduce the search space to the viewports of users that will receive transcoded views. This could potentially downgrade the final visual quality: the global optimal solution may not necessarily include users' viewports. For instance, for two views V_A and V_B in the same cluster, the optimal reference view for both of them may be somewhere else, denoted as V_{opt}. However, we argue that this will have minimal impact on the final visual quality: Since the overarching goal is to maximize the visual quality for users, each user will receive the reference view that is most similar to their ground truth view. Therefore, the view of V_A and V_B should both be similar to V_{opt}, meaning that there won't be a significant difference between $\delta_{IW}(V_A, V_{opt}) + \delta_{IW}(V_B, V_{opt})$ and $\delta_{IW}(V_A, V_B)$.

Second, searching for the optimal solution can take a long time. There are $\binom{n-q}{m}$ different combinations of a potential

reference viewport set V_r^t. With each combination, we also need to find the optimal reference view for each user i that is not selected, such that the visual quality is maximized for each user. The search for an optimal reference viewport for every single user then takes $(n-q) * m$ time. As a result, performing an exhaustive search for the optimal solution would take approximately $\binom{n-q}{m} * (n-q) * m$ time, which is not ideal if n is too large.

To perform a faster search, we modify the forward greedy algorithm from the K-median clustering problem [18, 40]. The K-median clustering problem is a variation of the K-means clustering. It finds the optimal cluster partition which minimizes the sum of the distances from each point to its closest cluster centroid. The K-Median problem is NP-hard, and there exist several approximation algorithms to find a solution fast, including the forward-greedy algorithm [6, 12]. We find our optimization problem for computation resource allocation similar to the K-median clustering problem: the distortion function of each user can be considered as the distances in the K-median problem, and the centroid of each group is the reference viewports we want to search for. In addition to considering the distortion as in the K-median formulation, we also need to optimize the quality switch and fairness across all users.

We formulate our modified forward-greedy algorithm for selecting the set of reference viewports as follows: We first get the set of candidate viewports V_p from users with $\delta^t(i) = 1$, and remove the duplicate viewports from it. We then check if there are already less than m candidates in the set. If so, we do not need any further search. Otherwise, we start with an empty set V_o which stores our final selected viewports. For each $V_{p,i}$ in the candidate set, we test its cost as a potentially selected reference viewport, *i.e.*, $Cost(V_o \cup V_{p,i})$ (6), and add the viewport with the lowest cost to V_o. We repeat this step until we have m viewports in V_o, which is the final set of reference viewports. Finally, we decide the reference viewport for each user i not in V_o such that the overall cost is minimized. Note that the forward greedy algorithm is an approximation algorithm. We will show the validation of the forward greedy algorithm in our system in §5.

Note that view sharing in MuV^2 adapts to the available computation resources: when the edge server can support encoding and rendering for all users ($m \geq n$), no view-sharing is performed; when the computation resource becomes limited, MuV^2 will share views across users to reduce the computation resource usage. Also, under limited computation resources, MuV^2 applies view sharing even when the viewport similarity across users is small. Otherwise, the incurred stalls will lead to an even more significant degradation of the viewing experience compared to that caused by the distortion due to image warping.

3.6 Encoder Multiplexing

After view-sharing, we have generated a set of m views (V_r^t) to be encoded on the edge server. Users are also divided into m groups, with each group assigned one reference view. Our next question is how to encode those views efficiently.

Most video encoders, such as H.264, HEVC, etc, leverage the technique of inter-frame compression to achieve high compression ratios [22, 38]. The inter-frame compression divides the video stream into small groups of frames, referred to as a "group of pictures" (GOP) [52]. The first frame of a GoP is the keyframe, which is encoded with only its own information and can be decoded independently. The other frames following a keyframe are P-frames or B-frames. Those frames are encoded with a reference to the previous keyframe and can be decoded only after decoding the corresponding keyframe. The keyframes have a lower compression ratio than P-frames but are necessary for encoding because they provide key references for other frames [19]. Normally, the video encoders are designed to encode a single video stream, and the user is supposed to decode all the frames sequentially. However, in MuV^2, the encoder needs to encode the same video frame into multiple video sequences. If we simply encode all views into a single video stream, then each user would need to receive and decode all the views to ensure a correct decoding result. This will increase the overall bandwidth consumption and reduce the scalability of our system. To maintain the high scalability of our system, we should intelligently arrange the frames and insert keyframes so that each user only receives one view for each volumetric frame.

One naive solution to achieve this is to encode every view as a keyframe. Each view can be decoded independently, but since the keyframe is much larger in size, it will require higher bandwidth and therefore reduce scalability. Another solution is to create m concurrent encoding sessions that each encode one video stream for one group of users. However, although some GPUs may support creating multiple encoding sessions, the number of sessions is limited[1].

We design the following encoding scheme for MuV^2 to support multi-user encoding with one single encoder. We start with a simple scenario where the group division for reference viewports is fixed throughout the entire video session. For each group of users, we encode its reference view, $V_{r,i}^t$ and the following k frames, up to $V_{r,i}^{t+k}$, into a GOP, then switch to the next group and encode $V_{r,j}^t$ to $V_{r,j}^{t+k}$ into a GOP. With this approach, each group of users only needs to receive its own group of frames. Note that the GOP size k needs to be carefully tuned: A larger k value will result in a lower bandwidth requirement since it reduces the number of keyframes.

[1]For most Nvidia GPUs that support concurrent sessions, the upper limit is 3: https://developer.nvidia.com/video-encode-and-decode-gpu-support-matrix-new

However, it will decrease fairness across users: other groups need to wait longer for the same video frame. A smaller k will make all users synchronous better but at the cost of a higher bandwidth requirement. With a carefully tuned k, our method can reduce bandwidth consumption compared to encoding all frames into keyframes while also maintaining high scalability by ensuring each user only receives one view for each volumetric frame.

The above scheme works well for a fixed group division. However, in reality, the group division will not always be the same because of users' dynamic movement. Forcing a fixed division will result in users receiving less similar views and increase visual distortion. When the group division is dynamic, we cannot simply apply the above scheme because users may receive mismatched keyframes and P-frames. For example, a user may be in group i at frame t but switch to group j at frame $t+k$. If we apply the above scheme, the user will be missing the keyframe from group j and therefore encounter decoding distortion. Therefore, we need to adjust our encoding scheme to adapt to changing group divisions. There are two ways to address this problem: enforce the group division or enforce the GOP division. To enforce group division, we insert a keyframe whenever a user changes to another group (for the above example, we will insert a keyframe for group j at frame $t+k$). It will maintain the visual quality for that user but at the cost of higher bandwidth consumption. To enforce the GOP division, we fix the group division for every GOP (e.g. force this user to stay in the group i from frame t to $t+k$). This will maintain the bandwidth requirement but potentially introduce higher visual distortion. We can denote this extra visual quality loss at frame $t+k$ as:

$$DSTR_{enc}^{t+k} = |Cost(V_r^{t+k}, V_p^{t+k})_{\delta=1} - Cost(V_r^{t}, V_p^{t+k})_{\delta=1}| \quad (7)$$

where frame t is the last keyframe. In MuV^2, we consider both methods and make the decision dynamically based on users' viewport and available bandwidth. Note that we should prioritize enforcing the group division if possible as it does not reduce the visual quality. We then formulate the optimization problem that quantifies the trade-off between the visual quality and bandwidth consumption as follows:

$$\min DSTR_{enc}^{t+k} \quad (8)$$
$$s.t. \sum_i Sizeof(2D)_{\delta=1} < BW_{aval}$$

Recall from §3.4, we have saved some bandwidth for the encoding scheme when allocating the bandwidth. Therefore, we do not need to change the group of users that will receive volumetric frames. The $Band_{aval}$ refers to the remaining bandwidth after reserving for volumetric frames. In cases where there is not enough bandwidth to insert keyframes for all groups, we insert keyframes for the groups that will incur the highest $DSTR_{enc}$.

4 IMPLEMENTATION

We implement the client of MuV^2 on Android with Java. The edge and server of MuV^2 are both implemented on Linux with C++. The transcoded frames are encoded using Nvidia Codec on the edge server and are decoded with MediaCodec API [1] on clients. We implement image warping with OpenGL shaders and use OpenGL ES to render both 2D frames and point clouds on clients. For each 2D view, we render the corresponding depth map as a gray-scale image and concat the depth map to the bottom of the RGB image for encoding. The RGB image and depth map are encoded in 1080P (1920×1080) and the final transcoded view is 1920×2160. The volumetric video compression and decoding are both implemented with the open-source library, Draco [2]. To achieve the best efficiency, we use multi-threading to encode and decode point cloud data, and asynchronous encoding and decoding for transcoded views. In total, we implement the client with more than 2K LoC, and the edge server with more than 4K LoC.

We discuss two limitations of our implementation. First, MuV^2's viewing sharing and volumetric frame delivery are implemented with unicast due to the poor multicast performance on today's Wi-Fi infrastructures [50]. Note that both features are on the content level and therefore can be implemented by multicast in the future to further reduce the bandwidth consumption and boost the scalability. Second, when deciding between transmitting 2D transcoded vs. 3D volumetric content, the content hybridization module (§3.4) considers the AP's total bandwidth capacity as the major network resource constraint. This is largely acceptable in our setup (WiFi shared by co-located users). In other wireless environments, such as cellular networks, additional network resource constraints may exist (*e.g.,* per-user bandwidth constraints imposed by the cellular base station's scheduler). These additional constraints can be incorporated into MuV^2's framework.

5 EVALUATION

5.1 Setup

Hardware Devices: We evaluate our system on 10 smartphones with different computational capacities: Samsung S21+ (5), Samsung S22 (2), Samsung S8 (1), Samsung S9 (1), ROG Phone 2 (1). The smartphones run a mix of OS versions from Android 9 to 13. For the rest of the clients, we run simulated client applications on three desktops with Ubuntu 22.02. We implement our edge server on a desktop server with Intel Core i7-9700K CPU @ 3.60GHz, GeForce RTX 2080Ti GPU, and Ubuntu 20.04. Our server is implemented on a separate desktop server with Intel(R) Xeon(R) E-2186G CPU @ 3.80GHz and Ubuntu 18.04.

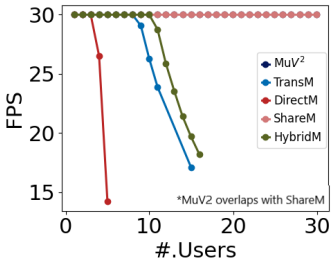

Figure 6: Frame rate of the worst-performing user.

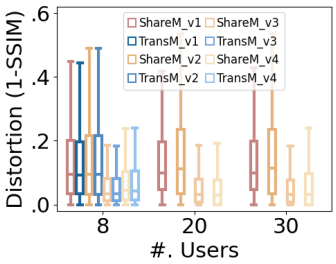

Figure 7: Visual distortion of TransM **and** ShareM**.**

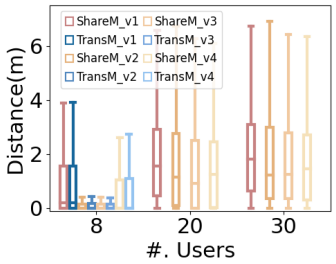

Figure 8: Viewport distance of TransM **and** ShareM

Network condition: The content server and edge server are co-located and connected through a high-throughput cable to prevent transmission delay. All clients (smartphones and emulated clients) are connected wirelessly through a single commercial 802.11ac AP at 5GHz to the edge server. The average bandwidth of the AP is 450 Mbps with a ping latency of less than 10ms.

Volumetric Video Dataset and Traces: Although there exist several public volumetric video datasets (*e.g.,* 8i [3]), most datasets only feature short video clips (around 10 seconds) and similar video content (stationary or moving portraits of a single human). To capture more meaningful movement traces, we recorded four longer volumetric videos (V1 - V4) by merging captured point clouds from several synchronized stereo cameras. These volumetric videos feature multiple people standing close together and have an average length of 77 seconds. The average point density is 120K~140K points per frame and the average bitrate after compression is around 150 Mbps for all four videos. We collected 33 users' trajectory traces while watching those videos on smartphones and replayed the traces for more reproducible experiment results.

Test Environment: We evaluate our system in two different live-streaming environments. *Simulated live-streaming.* To make our results reproducible, we implement a simulated live-streaming environment where the content server reads a local volumetric video file at a fixed frame rate of 30 FPS. This emulates a video being captured by a live camera. *Real live-streaming.* To evaluate the performance of MuV^2 under a real live-streaming environment, we implement a live-streaming server with three Zed [4] cameras. The live content server performs a one-time offline camera calibration before streaming the live content. During the live-streaming session, the content server receives raw point cloud frames from each camera at 30 FPS and merges the point clouds in real time. The merged point cloud is then divided into smaller tiles and encoded with Draco compression.

System to compare: We implement the four systems for comparison evaluation:

DirectM: We extend ViVo [17] to multi-user as a prototype of DirectM. The volumetric video is divided into several 3D tiles. For each user, the server dynamically decides the quality of each tile to be streamed based on the user's real-time viewport position: the tiles that fall out of the user's view or are occluded by other tiles will be streamed on a lower-quality level to reduce the overall bandwidth consumption.

TransM: We extend the transcode streaming system with image warping [14, 15] for TransM. The edge server encodes one RGB view along with the depth map for each user and applies the same GOP encoding method in §3.6. The user will then leverage image warping to generate the desired view based on their real-time viewport.

HybridM: We implement a hybrid streaming system named HybridM by removing the view-sharing from MuV^2. HybridM implements the same content hybridization algorithm as MuV^2 to decide which users receive volumetric frames. For the rest of the users, HybridM performs transcoding in the same manner as TransM.

ShareM: We implement ShareM as a variation of MuV^2 that only applies view-sharing. ShareM applies the same reference view selection and encoding optimization are the same as MuV^2.

5.2 Scalability Comparison

We evaluate each system's scalability with the frame rate of each system when streaming to different numbers of users under the emulated live streaming. The scalability of each system is lower-bounded by the worst-performing user for each system. The server of MuV^2 runs at 30 FPS. We show our results in Figure 6. As shown in the figure, DirectM has the lowest scalability with only 3 users, limited by its high bandwidth requirement. TransM can support at most 8 users. As discussed in §2.1, TransM is limited by its computational overhead. We observe that the GPU encoding capacity reached 95% when supporting 8 concurrent users, therefore cannot support more users. The upper limit of HybridM is 10 users, limited by both resources. None of the above systems achieves the desired scalability.

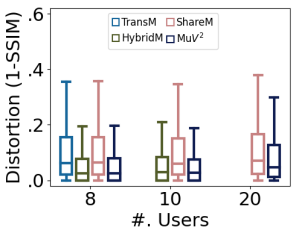

Figure 9: Visual distortion comparing all four systems.

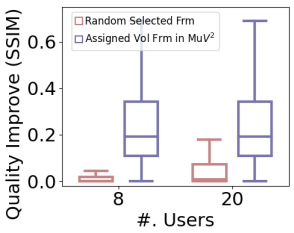

Figure 10: Visual quality improvement of different hybridization algorithms.

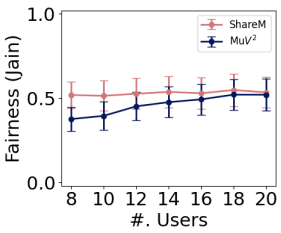

Figure 11: Jain's Fairness across all users of all systems.

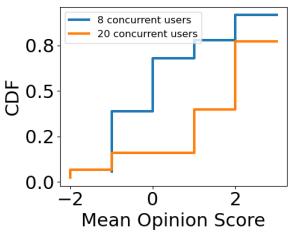

Figure 12: Mean opinion score comparing TransM and MuV².

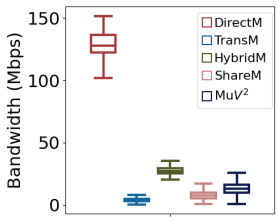

Figure 13: Per-user bandwidth consumption of all systems.

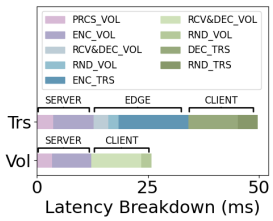

Figure 14: End-to-end latency breakdown of MuV².

ShareM and MuV² bear the same level of scalability, which is more than 30 users, achieving our desired scalability for classrooms or museums. Because both systems apply the view-sharing method and have a fixed number of views to be encoded on the edge server, the scalability of both systems is no longer limited by the encoding capacity of the GPU. MuV² applies a dynamic content hybridization and streams under the bandwidth limit, therefore is not throttled by network capacity as well. We will show later that, despite achieving the same scalability as ShareM, MuV² improves the viewing quality significantly compared to ShareM. Note that the above scalability results are evaluated with a mix of real devices and simulated clients where the ratio of the real versus simulation is kept the same.

5.3 Efficiency of Cross-user View Sharing

Verification of forward-greedy. As described in §3.5, we use the forward-greedy algorithm to approximate the optimal solution for view selection. Therefore, we need to verify that the forward-greedy algorithm can produce a good approximation of the global optimal solution. We verify this by comparing the solution of the greedy algorithm with the global optimal solution given by an exhaustive search. Our comparison results show that the difference between the forward-greedy solution and the global optimal solution in terms of the final cost function value is less than 0.5%, which means that the solution given by the forward-greedy algorithm is a sufficiently accurate approximation of the optimal solution. Moreover, the exhaustive search takes 100ms, which makes it impossible for live-streaming, whereas the forward greedy only takes less than 10ms.

Visual distortion evaluation. We next evaluate the efficiency of our view-selection and view-sharing algorithms by comparing the visual quality of ShareM to TransM with emulated live streaming. Both systems only stream transcoded frames and apply a single encoder instance with the encoding optimization described in §3.6. We evaluate the visual quality by comparing the visual distortion caused by image warping. A lower visual distortion represents a better viewing experience.

We show the visual distortion value in Figure 7. The performance of TransM and ShareM is the same when connected with 8 users. Compared to TransM with 8 concurrent users, ShareM only increases the distortion by 6.0% when connected to 20 users, and 7.2% when connected to 30 users for V4. We observe similar results for other videos as well. We also evaluate the viewport distance between the received view and the users' ground truth view, as shown in Figure 8. Compared with 8 concurrent users, ShareM encounters a much larger distance, with an increase of 1.50m when connected to 20 users and 1.58m for 30 users (V2). Despite the large viewport distance, ShareM achieves a similar level of visual quality as TransM. This is because our view selection algorithm targets selecting the reference views that can minimize the visual distortion, not the viewport distance. Our results validate the effectiveness of the view-sharing algorithm: When the number of users increases, the selected reference views are representative enough and ensure that users can maintain good visual quality.

5.4 Efficiency of Content Hybridization

Next, we evaluate the effectiveness of jointly streaming volumetric frames and transcoded frames. We evaluate this by pairwisely comparing the visual distortion of HybridM to TransM, and MuV² to ShareM. We show our results in

Figure 9. When connected to 8 users, TransM, HybridM, ShareM, and MuV^2 incur a visual distortion of 0.121, 0.063, 0.121, and 0.064, respectively. Compared to TransM, HybridM improves the visual quality by 48%, and MuV^2 improves the quality by 47% compared to ShareM We observe a similar improvement of MuV^2 over ShareM when connected to 10 and 20 users. As discussed in §2.1, volumetric video frames incur no visual distortion caused because of local rendering. Our results indicate that jointly streaming volumetric video frames can have a significant improvement in visual quality.

We evaluate quality fairness with Jain's fairness index value, as described in §3.3. A value closer to 1 indicates a more fair system. We present our fairness evaluation result in Figure 11. Our results indicate that ShareM can maintain the same level of fairness for different numbers of users. The fairness of MuV^2 is lower when connected to fewer users because a higher ratio of frames is assigned volumetric frames, which has no visual distortion and leads to a less fair result. The quality switch (SWI) of all systems is similar and close to 0, therefore is not shown in the figures.

We further explore the efficiency of our volumetric frame allocation algorithm by showing the quality improvement of MuV^2 over ShareM and comparing it with a random selection approach. The result is shown in Figure 10. The random selection approach randomly selects the frames to be streamed as volumetric frames with the same ratio as MuV^2. Our results show that our view selection algorithm can reduce a visual distortion of 0.231 (0.230), whereas the random selection only reduces 0.059 (0.068) for 8 users (20 users). The results further demonstrate that our network allocation algorithm can better improve visual quality.

5.5 Viewing Experience Evaluation

To evaluate the viewing experience and quality of MuV^2, we conduct a user study and compare MuV^2 with TransM. We generate 16 video clips with 4 randomly selected users' traces with 8 and 20 concurrent users scenarios and present each pair of videos generated from TransM and MuV^2 side-by-side for users to watch and rate. The order of the two videos is random. Users' ratings range from -3 (the left video is much better) to 3 (the right video is much better), with 0 meaning "mutual quality". We show the users' rating results in Figure 12, where the positive score means MuV^2 has a higher visual quality than TransM. As shown, when 8 users are connected, the visual quality of MuV^2 is slightly better than TransM, with 34% positive scores and 52% mutual scores. The visual improvement is caused by the better visual quality of volumetric frames. When 20 users are connected, the visual quality of MuV^2 is much better, with 90% of users giving positive scores for MuV^2. This is because when 20 users are connected, TransM exceeds its capacity and will incur a much lower playback frame rate, leading to a much worse viewing experience.

5.6 Bandwidth Usage Comparison

We show the per-user bandwidth consumption of each system in Figure 13. Because DirectM, TransM, and HybridM can support much fewer users than ShareM and MuV^2, comparing the overall bandwidth consumption is meaningless. Instead, we compare the per-user bandwidth consumption. The average per-user bandwidth consumption of DirectM, TransM, HybridM, ShareM, and MuV^2 is 134.8, 4.1, 29.4, 8.0, and 12.2Mbps, respectively. DirectM incurs the highest bandwidth consumption because it only streams volumetric videos. HybridM and MuV^2 both have slightly higher bandwidth consumption than TransM and ShareM. ShareM consumes less bandwidth than MuV^2 because ShareM only streams transcoded video frames, while MuV^2 streams volumetric video frames to several users if bandwidth permits. Our results echo our findings in §5.2: The high per-user bandwidth consumption of DirectM and HybridM is one of the bottlenecks that hinder supporting more users. Also, note that the average bandwidth consumption of DirectM is higher than ViVo [17] (not shown in Figure 13). This is because DirectM uses a different encoding scheme compared to ViVo. [2]

5.7 End-to-end Latency Breakdown

We record the running time under the simulated live-streaming environment and break down the running time latency of each system function in Figure 14. As shown in the figure, the processing latency of transcoded frames and volumetric frames are 49.7ms and 25.9ms, respectively. The server latency is the same for both frames, being 12.8ms for volumetric frames processing and encoding. The transcoded frames yield a longer process time because they require extra process procedures on the edge side, being the receiving and decoding of volumetric frames (3.16ms), 2D rendering, and encoding (18.2ms). On the client side, the 2D decoding and rendering takes 15.7ms on average. On the other hand, the volumetric frames only require receiving, decoding, and rendering on the client side, with an average latency of 13.7ms. Note that for the transcoded frames, the 2D encoding and decoding are performed on the hardware encoder that handles its own queue. Therefore, its latency time only reflects

[2] Given a volumetric frame, DirectM splits its point cloud into 5 equal-sized sets $\{P_1, ..., P_5\}$ through uniform sampling and encodes each set separately. The i-th point density level ($1 \leq i \leq 5$) consists of (encoded) $P_1 \cup ... \cup P_i$. In contrast, ViVo generates the i-th point density level by directly sampling $i * 20\%$ of points from the original volumetric frame and encoding them. While ViVo's scheme is more bandwidth-efficient, it incurs a much higher transcoding overhead compared to DirectM since more points need to be encoded at runtime when $i > 1$.

the time during which that frame is in the encoder, instead of the actual encoding time.

We next evaluate the live end-to-end latency under the real live-streaming environment. We measure the streaming latency by live-streaming a millisecond digital clock [3] and comparing the time difference between the clock and the displayed video on the client. Since volumetric frames and transcoded frames have different process pipelines and latency, we evaluate the end-to-end latency of the two frame types separately with DirectM for volumetric frames and ShareM for the latency of transcoded frames. We compare the latency with two single-user streaming systems: DirectS and TransS, where DirectS streams single volumetric frames and TransS streams a single transcoded frame to the client with image warping. On average, DirectS, TransS, DirectM, and ShareM achieve 135, 151, 171, and 217ms latency respectively. Compared to the corresponding baselines, MuV^2 only introduces an extra **36**ms for volumetric frames and **66**ms for transcoded frames. Considering the significantly higher scalability MuV^2 can provide with the edge server, we believe that this is a tolerable latency to achieve.

5.8 Energy Consumption

We measure the GPU usage with Android Studio Performance Monitor while replaying V3 on all smartphones. On average, MuV^2 incurs less than 7% CPU usage when receiving transcoded frames, and less than 25% CPU usage when receiving volumetric frames. We measure the battery usage by replaying V3 for 30 min and evaluating the battery drop. The average battery usage is 9% (3D) and 6% (2D). Overall, we believe MuV^2 achieves an acceptable energy consumption.

6 RELATED WORK

On-demand volumetric content streaming. There are two types of on-demand volumetric streaming systems: direct streaming and transcode streaming. Direct streaming systems stream compressed volumetric videos to users. ViVo [17] streams compressed volumetric videos based on users' visibility of the volumetric content and reduces the bandwidth requirement. Yuzu [55] utilizes a super-resolution model to boost the quality of volumetric video and reduce the bandwidth requirement. Transcode streaming systems leverage an edge to transcode video into 2D views before streaming [14, 16, 44]. Vues [37] leverages an edge server to transcode one volumetric video frame into multiple views with viewports from different viewport prediction models. [15] utilizes image warping to reduce the viewport drift between the rendered view and the user's actual viewport.

Live volumetric content streaming. Existing live volumetric streaming systems mainly focus on single-user experience [20, 28]. Holoportation [41] presents a real-time 3D reconstruction and streaming system of an entire 3D scene to create a more immersive sense of presence on an MR headset. Starline [31] introduces a real-time bidirectional 3D communication system that captures the 3D representation of both clients and streams to each other. HyperVR [25] proposes a camera-agnostic live volumetric video capturing system and an efficient mesh compression algorithm, which enables using 50 Mbps to deliver photorealistic volumetric content at 24 FPS.

Multi-user Volumetric Content Streaming. Several studies have been studying streaming VR content to multiple users. Firefly [36] enables more than 10 players to use VR together by applying a series of techniques such as offline content preparation, viewport-adaptive streaming with motion prediction, and adaptive content quality control among users. M5 [56] uses mmWave network to enhance multi-user volumetric content delivery. It proactively adapts mmWave beams and pre-fetches frames to mitigate signal blockage effects and uses multicast transmission to stream the overlapped content to minimize the bandwidth requirement. MuVR [34] caches and opportunistically reuses static VR background scenes (as opposed to dynamic volumetric contents in MuV^2) among multiple users via image warping. In addition, MuVR puts all the rendering workload on the edge without any local rendering on user devices and does not consider balancing visual quality among users.

7 CONCLUDING REMARKS

In this paper, we present MuV^2, an edge-assisted live volumetric video streaming system that jointly streams volumetric video frames and transcoded 2D views. MuV^2 achieves high scalability through a novel view-sharing algorithm facilitated by image warping. For each frame, MuV^2 dynamically selects a reference view set and shares those views across users. MuV^2 also intelligently assigns volumetric video frames for users that may incur lower visual quality. Through an extensive evaluation, we demonstrate that MuV^2 can support a significantly larger group of users compared to extending state-of-the-art direct streaming and transcode streaming systems to multi-users. MuV^2 also improves visual quality by 47% through the hybrid of volumetric frames.

ACKNOWLEDGMENTS

We thank the reviewers and our shepherd for their insightful comments. This research was supported in part by NSF Award 2106090, 1915122, 2212298, 2128489, and a Cisco Research Grant. The research of Bo Han was funded by NSF Awards CNS-2212296 and CNS-2235049.

[3] https://www.youtube.com/watch?v=RJfAkfY2Ifo

REFERENCES

[1] Android mediacodec api document. https://developer.android.com/reference/android/media/MediaCodec.

[2] Draco 3D Data Compression. https://google.github.io/draco/.

[3] Eugene d'eon, bob harrison, taos myers, and philip a. chou, "8i voxelized full bodies - a voxelized point cloud dataset," iso/iec jtc1/sc29 joint wg11/wg1 (mpeg/jpeg) input document wg11m40059/wg1m74006, geneva, january 2017.

[4] Zed camera sdk. https://www.stereolabs.com/developers/release/.

[5] R. Artusi, P. Verderio, and E. Marubini. Bravais-pearson and spearman correlation coefficients: meaning, test of hypothesis and confidence interval. *The International journal of biological markers*, 17(2):148–151, 2002.

[6] V. Arya, N. Garg, R. Khandekar, A. Meyerson, K. Munagala, and V. Pandit. Local search heuristic for k-median and facility location problems. In *Proceedings of the thirty-third annual ACM symposium on Theory of computing*, pages 21–29, 2001.

[7] J. Benesty, J. Chen, Y. Huang, and I. Cohen. *Pearson Correlation Coefficient*, pages 1–4. Springer Berlin Heidelberg, Berlin, Heidelberg, 2009.

[8] K. Boos, D. Chu, and E. Cuervo. Flashback: Immersive virtual reality on mobile devices via rendering memoization. In *Proceedings of the 14th Annual International Conference on Mobile Systems, Applications, and Services*, MobiSys '16, page 291–304, New York, NY, USA, 2016. Association for Computing Machinery.

[9] F. Broxton, J. Flynn, R. Overbeck, D. Erickson, P. Hedman, M. DuVall, J. Dourgarian, J. Busch, M. Whalen, and P. Debevec. Immersive Light Field Video with a Layered Mesh Representation. In *Proceedings of ACM SIGGRAPH*, 2020.

[10] Z. Chen, T. Funkhouser, P. Hedman, and A. Tagliasacchi. Mobilenerf: Exploiting the polygon rasterization pipeline for efficient neural field rendering on mobile architectures. In *The Conference on Computer Vision and Pattern Recognition (CVPR)*, 2023.

[11] A. Clemm, M. T. Vega, H. K. Ravuri, T. Wauters, and F. De Turck. Toward Truly Immersive Holographic-type Communication: Challenges and Solutions. *IEEE Communications Magazine*, 58(1):93–99, 2020.

[12] D. Dohan, S. Karp, and B. Matejek. K-median algorithms: theory in practice. Technical report, Working paper, Princeton, Computer Science, 2015.

[13] C. A. Glasbey and K. V. Mardia. A review of image-warping methods. *Journal of applied statistics*, 25(2):155–171, 1998.

[14] S. Gül, D. Podborski, J. Son, G. S. Bhullar, T. Buchholz, T. Schierl, and C. Hellge. Cloud rendering-based volumetric video streaming system for mixed reality services. In *Proceedings of the 11th ACM Multimedia Systems Conference*, MMSys '20, page 357–360, New York, NY, USA, 2020. Association for Computing Machinery.

[15] S. Gül, C. Hellge, and P. Eisert. Latency compensation through image warping for remote rendering-based volumetric video streaming. In *2022 IEEE International Conference on Image Processing (ICIP)*, pages 2026–2030, 2022.

[16] S. Gül, D. Podborski, T. Buchholz, T. Schierl, and C. Hellge. Low-latency cloud-based volumetric video streaming using head motion prediction, 2020.

[17] B. Han, Y. Liu, and F. Qian. Vivo: Visibility-aware mobile volumetric video streaming. In *Proceedings of the 26th Annual International Conference on Mobile Computing and Networking*, MobiCom '20, New York, NY, USA, 2020. Association for Computing Machinery.

[18] S. Har-Peled and S. Mazumdar. On coresets for k-means and k-median clustering. In *Proceedings of the thirty-sixth annual ACM symposium on Theory of computing*, pages 291–300, 2004.

[19] J. He, M. A. Qureshi, L. Qiu, J. Li, F. Li, and L. Han. Rubiks: Practical 360-Degree Streaming for Smartphones. In *Proceedings of ACM MobiSys*, 2018.

[20] J. Hu, A. Shaikh, A. Bahremand, and R. LiKamWa. Characterizing real-time dense point cloud capture and streaming on mobile devices. In *Proceedings of the 3rd ACM Workshop on Hot Topics in Video Analytics and Intelligent Edges*, pages 1–6, 2021.

[21] T.-Y. Huang, R. Johari, N. McKeown, M. Trunnell, and M. Watson. A buffer-based approach to rate adaptation: Evidence from a large video streaming service. In *Proceedings of the 2014 ACM conference on SIGCOMM*, pages 187–198, 2014.

[22] Y.-W. Huang, B.-Y. Hsieh, T.-C. Chen, and L.-G. Chen. Analysis, fast algorithm, and vlsi architecture design for h. 264/avc intra frame coder. *IEEE Transactions on Circuits and systems for Video Technology*, 15(3):378–401, 2005.

[23] R. K. Jain, D.-M. W. Chiu, W. R. Hawe, et al. A quantitative measure of fairness and discrimination. *Eastern Research Laboratory, Digital Equipment Corporation, Hudson, MA*, 21, 1984.

[24] J. Jansen, S. Subramanyam, R. Bouqueau, G. Cernigliaro, M. M. Cabré, F. Pérez, and P. Cesar. A pipeline for multiparty volumetric video conferencing: transmission of point clouds over low latency dash. In *Proceedings of the 11th ACM Multimedia Systems Conference*, pages 341–344, 2020.

[25] B. Ji, W. Pi, W. Liu, Y. Liu, Y. Cui, X. Zhang, and S. Peng. HyperVR: a hybrid deep ensemble learning approach for simultaneously predicting virulence factors and antibiotic resistance genes. *NAR Genomics and Bioinformatics*, 5(1):lqad012, 2023.

[26] J. Jiang, V. Sekar, and H. Zhang. Improving Fairness, Efficiency, and Stability in HTTP-based Adaptive Video Streaming with FESTIVE. In *Proceedings of ACM CoNEXT*, 2012.

[27] Q. Jin, Y. Liu, P. Zhou, B. Han, S. Yarosh, and F. Qian. Volumivive: An authoring system for adding interactivity to volumetric video. In *2023 IEEE Conference on Virtual Reality and 3D User Interfaces Abstracts and Workshops (VRW)*, pages 569–570. IEEE, 2023.

[28] B. Jones, Y. Zhang, P. N. Wong, and S. Rintel. Vroom: virtual robot overlay for online meetings. In *Extended Abstracts of the 2020 CHI Conference on Human Factors in Computing Systems*, pages 1–10, 2020.

[29] V. Joseph and G. de Veciana. Jointly optimizing multi-user rate adaptation for video transport over wireless systems: Mean-fairness-variability tradeoffs. In *2012 Proceedings IEEE INFOCOM*, pages 567–575. IEEE, 2012.

[30] E. Kasuya. On the use of r and r squared in correlation and regression. Technical report, Wiley Online Library, 2019.

[31] J. Lawrence, D. B. Goldman, S. Achar, G. M. Blascovich, J. G. Desloge, T. Fortes, E. M. Gomez, S. Häberling, H. Hoppe, A. Huibers, et al. Project starline: A high-fidelity telepresence system. 2021.

[32] K. Lee, J. Yi, Y. Lee, S. Choi, and Y. M. Kim. GROOT: A Real-Time Streaming System of High-Fidelity Volumetric Videos. In *Proceedings of ACM MobiCom*, 2020.

[33] K. Lee, J. Yi, Y. Lee, S. Choi, and Y. M. Kim. Groot: A real-time streaming system of high-fidelity volumetric videos. In *Proceedings of the 26th Annual International Conference on Mobile Computing and Networking*, New York, NY, USA, 2020. Association for Computing Machinery.

[34] Y. Li and W. Gao. Muvr: Supporting multi-user mobile virtual reality with resource constrained edge cloud. In *2018 IEEE/ACM Symposium on Edge Computing (SEC)*, pages 1–16. IEEE, 2018.

[35] X. Liu, C. Vlachou, F. Qian, and K.-H. Kim. Supporting untethered multi-user vr over enterprise wi-fi. In *Proceedings of the 29th ACM Workshop on Network and Operating Systems Support for Digital Audio and Video*, pages 25–30, 2019.

[36] X. Liu, C. Vlachou, F. Qian, C. Wang, and K.-H. Kim. Firefly: Untethered multi-user vr for commodity mobile devices. In *Proceedings of the 2020*

USENIX Conference on Usenix Annual Technical Conference, pages 943–657, 2020.

[37] Y. Liu, B. Han, F. Qian, A. Narayanan, and Z.-L. Zhang. Vues: Practical Volumetric Video Streaming through Multiview Transcoding. In *Proceedings of ACM MobiCom*, 2022.

[38] D. Marpe, T. Wiegand, and G. J. Sullivan. The h. 264/mpeg4 advanced video coding standard and its applications. *IEEE communications magazine*, 44(8):134–143, 2006.

[39] B. Mildenhall, P. P. Srinivasan, M. Tancik, J. T. Barron, R. Ramamoorthi, and R. Ng. Nerf: Representing scenes as neural radiance fields for view synthesis. *Communications of the ACM*, 65(1):99–106, 2021.

[40] M. Moshkovitz, S. Dasgupta, C. Rashtchian, and N. Frost. Explainable k-means and k-medians clustering. In *International conference on machine learning*, pages 7055–7065. PMLR, 2020.

[41] S. Orts-Escolano, C. Rhemann, S. Fanello, W. Chang, A. Kowdle, Y. Degtyarev, D. Kim, P. L. Davidson, S. Khamis, M. Dou, et al. Holoportation: Virtual 3d teleportation in real-time. In *Proceedings of the 29th annual symposium on user interface software and technology*, pages 741–754, 2016.

[42] J. Park, P. A. Chou, and J.-N. Hwang. Volumetric Media Streaming for Augmented Reality. In *Proceedings of IEEE GLOBECOM*, 2018.

[43] J. Peng, C.-S. Kim, and C.-C. J. Kuo. Technologies for 3D mesh compression: A survey. *Journal of Visual Communication and Image Representation*, 16(6):688–733, 2005.

[44] F. Qian, B. Han, J. Pair, and V. Gopalakrishnan. Toward Practical Volumetric Video Streaming On Commodity Smartphones. In *Proceedings of ACM HotMobile*, 2019.

[45] F. Qian, B. Han, Q. Xiao, and V. Gopalakrishnan. Flare: Practical Viewport-Adaptive 360-Degree Video Streaming for Mobile Devices. In *Proceedings of ACM MobiCom*, 2018.

[46] L. Qian, Z. Cheng, Z. Fang, L. Ding, F. Yang, and W. Huang. A qoe-driven encoder adaptation scheme for multi-user video streaming in wireless networks. *IEEE Transactions on Broadcasting*, 63(1):20–31, 2016.

[47] P. J. Rousseeuw. Silhouettes: a graphical aid to the interpretation and validation of cluster analysis. *Journal of computational and applied mathematics*, 20:53–65, 1987.

[48] K. P. Sinaga and M.-S. Yang. Unsupervised k-means clustering algorithm. *IEEE access*, 8:80716–80727, 2020.

[49] E. C. Strinati, S. Barbarossa, J. L. Gonzalez-Jimenez, D. Ktenas, N. Cassiau, L. Maret, and C. Dehos. 6G: The Next Frontier: From Holographic Messaging to Artificial Intelligence Using Subterahertz and Visible Light Communication. *IEEE Vehicular Technology Magazine*, 14(3):42–50, 2019.

[50] Y. Sun, Z. Chen, M. Tao, and H. Liu. Bandwidth Gain From Mobile Edge Computing and Caching in Wireless Multicast Systems. *IEEE Transactions on Wireless Communications*, 19(6):3992–4007, 2020.

[51] Z. Wang, A. Bovik, H. Sheikh, and E. Simoncelli. Image quality assessment: from error visibility to structural similarity. *IEEE Transactions on Image Processing*, 13(4):600–612, 2004.

[52] J. Xu, B. Zhou, C. Zhang, N. Ke, W. Jin, and S. Hao. The impact of bitrate and gop pattern on the video quality of h. 265/hevc compression standard. In *2018 IEEE International Conference on Signal Processing, Communications and Computing (ICSPCC)*, pages 1–5. IEEE, 2018.

[53] M. Yang, Z. Luo, M. Hu, M. Chen, and D. Wu. A comparative measurement study of point cloud-based volumetric video codecs. *IEEE Transactions on Broadcasting*, 2023.

[54] X. Yin, A. Jindal, V. Sekar, and B. Sinopoli. A Control-Theoretic Approach for Dynamic Adaptive Video Streaming over HTTP. In *Proceedings of ACM SIGCOMM*, 2015.

[55] A. Zhang, C. Wang, B. Han, and F. Qian. YuZu: Neural-enhanced Volumetric Video Streaming. In *Proceedings of USENIX NSDI*, 2022.

[56] D. Zhang, P. Zhou, B. Han, and P. Pathak. M5: Facilitating multi-user volumetric content delivery with multi-lobe multicast over mmwave. 2022.

[57] W. Zhang, F. Qian, B. Han, and P. Hui. Deepvista: 16k panoramic cinema on your mobile device. In *Proceedings of the Web Conference 2021*, WWW '21, page 2232–2244, New York, NY, USA, 2021. Association for Computing Machinery.