Vues: Practical Mobile Volumetric Video Streaming Through Multiview Transcoding

Yu Liu Bo Han* Feng Qian Arvind Narayanan Zhi-Li Zhang
University of Minnesota, Twin Cities *George Mason University

ABSTRACT

The emerging volumetric videos offer a fully immersive, six degrees of freedom (6DoF) viewing experience, at the cost of extremely high bandwidth demand. In this paper, we design, implement, and evaluate Vues, an edge-assisted transcoding system that delivers high-quality volumetric videos with *low bandwidth requirement*, *low decoding overhead*, and *high quality of experience (QoE)* on mobile devices. Through an IRB-approved user study, we build a first-of-its-kind QoE model to quantify the impact of various factors introduced by transcoding volumetric content into 2D videos. Motivated by the key observations from this user study, Vues employs a novel *multiview* approach with the overarching goal of boosting QoE. The Vues edge server adaptively transcodes a volumetric video frame into multiple 2D views with the help of a few lightweight machine learning models and strategically balances the extra bandwidth consumption of additional views and the improved QoE, indicated by our QoE model. The client selects the view that optimizes the QoE among the delivered candidates for display. Comprehensive evaluations using a prototype implementation indicate that Vues dramatically outperforms existing approaches. On average, it improves the QoE by 35% (up to 85%), compared to single-view transcoding schemes, and reduces the bandwidth consumption by 95%, compared to the state-of-the-art that directly streams volumetric videos.

CCS CONCEPTS

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

KEYWORDS

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

ACM Reference Format:
Yu Liu, Bo Han, Feng Qian, Arvind Narayanan, Zhi-Li Zhang. 2022. Vues: Practical Mobile Volumetric Video Streaming Through Multiview Transcoding. In *The 28th Annual International Conference On Mobile Computing And Networking (ACM MobiCom '22), October 24–28, 2022, Sydney, NSW, Australia.* ACM, New York, NY, USA, 14 pages. https://doi.org/10.1145/3495243.3517027

ACM MobiCom '22, October 24–28, 2022, Sydney, NSW, Australia

ACM ISBN 978-1-4503-9181-8/22/10…$15.00
https://doi.org/10.1145/3495243.3517027

1 INTRODUCTION

With recent advances in capturing and streaming technologies and the increasing popularity of virtual reality (VR) and augmented reality (AR) devices on the consumer market, mobile video streaming is gaining not only higher resolutions, but also higher degrees of immersion. This is demonstrated by the emergence of 360° panoramic videos [19, 43], volumetric videos [18, 31], and more recently light field videos [12]. In this paper, we focus on volumetric videos, whose display is essentially rendering a sequence of 3D models (point clouds or 3D meshes) at a fast pace such as 30 frames per second (FPS). The 3D representation of volumetric videos offers full immersion by allowing viewers to freely explore the 3D scene during video playback.

When watching a volumetric video, users can move with six degrees of freedom (6DoF) along three rotational dimensions by changing viewing direction in yaw, pitch, and roll and three translational dimensions by altering viewpoint position in X, Y, and Z. In contrast, regular 2D videos do not support any user interaction, and 360° videos enable only 3DoF (yaw, pitch, roll). As a result, volumetric videos will empower numerous novel applications. For example, in a chemistry class, students can investigate a molecular structure from various directions by moving around its volumetric display, when the instructor is explaining the properties of a chemical element.

While static volumetric content such as point clouds and 3D meshes has been well studied in the computer graphics community [22, 25, 28, 35, 51], streaming high-quality volumetric videos on mobile networks still remains a challenging issue. The state-of-the-art approaches for volumetric video streaming can be classified into two categories: direct streaming [18, 24, 31, 41] and transcoded streaming [16, 17, 42]. When directly streaming a volumetric video, the client downloads the encoded 3D content, either in its full form or segmented parts, before decoding and rendering. Although it can offer a desirable quality of experience (QoE), direct streaming suffers from high bandwidth requirements (*e.g.,* hundreds of Mbps) and non-trivial decoding overhead on mobile devices due to the sophisticated compression algorithms such as octree [25, 38] and kd-tree [26, 32].

In a transcoded streaming system, the server or the edge proxy performs real-time transcoding, by rendering 3D scenes into 2D images based on the (predicted) viewport of the client, and streams the encoded 2D video. Existing solutions [16, 17] create a single view for each frame using a simple prediction model. They may lead to a poor QoE when viewport prediction is not accurate, which is common for volumetric video streaming due to *6DoF motion dynamics*. Note that the transcoding methods for 360° videos [44, 54] can handle prediction errors along the rotational dimension (yaw, pitch, roll) by enlarging the transcoded viewport. Robust transcoding of volumetric content is much more challenging, as it also needs to

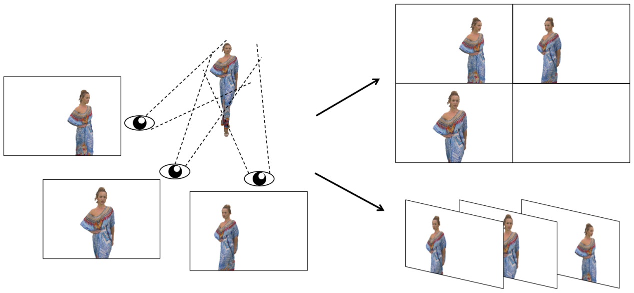

Figure 1: Generating multiple views from different viewpoints (left), spatial (upper right), and temporal (lower right) arrangements of those views.

	Direct Streaming	Single View Transcoding	Vues
Low Bandwidth		✓	✓
Low Dec. Overhead		✓	✓
High QoE	✓		✓

Table 1: Comparison of Vues **and existing approaches.**

deal with inaccurate predictions of translational motion, which is overlooked by single-view transcoding schemes [16, 17].

To address the above issues, we propose Vues, a practical mobile volumetric video streaming system that benefits from *multiview transcoding* (Figure 1). The Vues edge fetches the original (high-quality) volumetric content from the server and then adaptively transcodes each frame into multiple 2D views by utilizing a few lightweight viewport prediction models. After that, it streams to the client carefully selected candidate views, which offer more choices for display and thus improve the QoE. The Vues client chooses a view that provides the best QoE, indicated by our proposed model, among all the candidates and displays it to the viewer. We position Vues by comparing it with existing solutions using several key metrics in Table 1. Vues focuses on general video-on-demand (VoD) streaming scenarios such as online learning or YouTube-style entertainment, although the key concepts of Vues can be applied to live volumetric video streaming.

Our study for designing Vues includes the following.

QoE of Transcoding-based Volumetric Video Streaming (§4). We develop a practical model for quantifying the QoE of transcoded volumetric content. We identify several unique factors that are introduced by rendering future views based on potentially inaccurate viewport predictions, including viewport drift, viewport smoothness, viewport movement distance, and motion-to-photon latency. Our model takes into account both the above factors and those that have been studied for regular video streaming. We train this model through an IRB-approved user study that collected QoE ratings from more than 4,000 viewing sessions of transcoded volumetric videos. We also analyze the importance of these factors based on their contribution to the overall QoE.

Multiview Transcoding (§5). Based on the key observations from the above user study, we make judicious design decisions for Vues. Inspired by ensemble learning, the Vues edge intelligently combines the viewport prediction results from multiple machine learning models for enhancing the prediction accuracy and strategically adds additional views that cover a larger area than what is predicted by these models. It then ranks the candidate views based on their estimated QoE scores, dynamically decides the proper number of to-be-streamed views with a heuristic algorithm based on users' 6DoF motion, and explores two arrangement methods of encoding those views for balancing the QoE improvement and extra bandwidth consumption. The client also takes a principled, QoE-driven approach for selecting the view that maximizes the QoE.

Implementation and Evaluation (§6, §7). We implement a prototype of Vues (in 10,500 LoC) on commodity smartphones and demonstrate its efficacy through extensive evaluations. We compare the performance of Vues with both the state-of-the-art approaches that directly stream volumetric videos [18] and solutions that transcode volumetric content into a single view [16, 17]. We thoroughly evaluate Vues over WiFi, emulated LTE, and live LTE networks, with two real volumetric video datasets, as well as ratings from real subjects. We highlight key evaluation results as follows.

- Vues reduces bandwidth consumption by 95% compared to ViVo [18], the state-of-the-art direct streaming scheme, and avoids stall when streaming high-quality volumetric videos.
- When network bandwidth is limited, human subjects' ratings with mean opinion score (MOS) show that Vues attains considerably better QoE than ViVo.
- Vues achieves an average QoE improvement of 35% (up to 85%) compared to single-view, single-model transcoded streaming approaches [16, 17]. The improved QoE comes at the cost of 2.35× bandwidth usage compared to single-view systems. Given the very low bandwidth consumption of the single-view approach (around only 2 Mbps), the absolute additional bandwidth usage of Vues is acceptable. We therefore believe that Vues strikes a desired tradeoff between bandwidth usage and QoE.

2 RELATED WORK

Volumetric Video Streaming. There exist a few studies of volumetric video streaming in the literature [16–18, 24, 31, 40–42, 48]. Among them, ViVo [18] introduces several visibility-aware optimizations for streaming mainly the visible portion of a volumetric video to reduce resource consumption. GROOT [31] is another proposal that improves the efficiency of point cloud compression. The above approaches directly deliver volumetric videos. Recently, Gül *et al.* [16, 17] propose to leverage remote rendering (*i.e.,* transcoding) on cloud servers for low-latency volumetric video streaming. However, for each frame, they use only a single model to predict just a single viewport for streaming, which may degrade the QoE due to motion dynamics (as shown in §4).

Transcoding-based Systems. The transcoding approach has been widely applied in many other streaming systems including 360 video streaming, cloud gaming, virtual reality (VR), *etc.* DeepVista[54] streams ultra high resolution (up to 16K) panoramic videos to mobile devices by leveraging an edge server to extract and transcode the views based on the client's viewport. Outatime[30] presents a low-latency mobile cloud gaming system that offloads the scene rendering to an edge server and streams the transcoded images. FlashBack[10] enables high-quality VR content rendering on mobile devices by extensively pre-rendering, transcoding and storing all possible images the users may view offline. None of the above

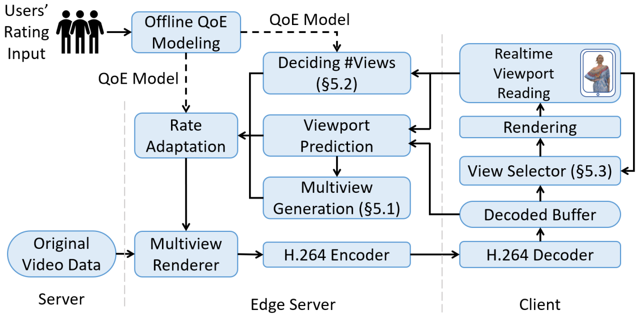

Figure 2: System architecture of Vues**.**

systems tackles volumetric video streaming that incurs unique challenges as to be described in §5.

QoE Assessment of Volumetric Videos. There has been some recent work on directly assessing the visual quality of point clouds. For example, using a data-driven approach, Meynet *et al.* [39] present a full-reference visual quality metric for colored point clouds. Viola *et al.* [49] explore global color statistics such as histograms and correlograms to analyze color-based metrics for determining the level of impairment of point clouds. A detailed review of existing subjective quality assessments and objective quality metrics for point clouds is available in Alexiou *et al.* [6]. The state-of-the-art visual quality assessment focuses on static point clouds. To the best of our knowledge, we are the first to model the QoE of transcoding-based volumetric video streaming.

3 MOTIVATION AND OVERVIEW

There are two major representations of volumetric content: point cloud and 3D mesh. A point cloud is essentially a group of unsorted points in 3D space, with attributes such as color or intensity. With a collection of vertices, edges, and faces, 3D mesh models the structure of an object with rich geometry information. These faces can be triangles, quadrangles, or even polygons with holes. We emphasize that by leveraging the transcoding of volumetric content into 2D views for streaming, the key design of Vues does not depend on the underlying representation. Although we focus on point-cloud-based volumetric videos in this work, we can extend Vues to 3D mesh-based videos by simply replacing the rendering pipeline on the edge, which is currently done by OpenGL. Moreover, since the rendering workload is offloaded to the edge server, the client-side performance of Vues is irrelevant to the complexity of 3D content.

The design of Vues is motivated by three key observations of existing solutions for volumetric video streaming. First, systems that directly stream volumetric videos [18, 31] usually lead to *high network bandwidth consumption*. For example, even with the visibility-aware optimizations proposed in ViVo [18], it may still require a bandwidth as high as 180 Mbps. Second, the performance of the above systems is typically limited by the *decoding capability of dense point clouds on mobile devices*. For instance, the average number of points per frame for all volumetric videos used in ViVo [18] is less than 250K. In contrast, each frame in the high-quality videos released by 8i [2] has ~1M points. Third, for single-view transcoding approaches [16, 17], their *QoE is in general undesirable*, mainly due to the inaccurate viewport prediction that is used to render volumetric content. They may result in an unsmooth trajectory of displayed views, a key contributor that degrades QoE (§4.4).

The overarching goal of Vues is to address the above issues by reducing the bandwidth consumption, making the decoding overhead on mobile devices independent of the quality of volumetric content, and enhancing the QoE for volumetric video streaming. We show the system architecture of Vues in Figure 2. The edge fetches the volumetric videos on demand from the original content server and adaptively transcodes a volumetric frame into multiple 2D views using a few lightweight machine learning models. It then delivers candidate views that provide more choices for the client to render and display. The Vues client selects the view that offers the best QoE among all candidates.

We face the following challenges when designing Vues. How to build a practical model for *estimating the QoE* of volumetric videos, which takes into account various factors introduced by transcoding? How to adaptively generate *multiple candidate views* for accommodating the inaccurate viewport prediction that is utilized for pre-rendering these views? How to strategically *select and encode* the most appropriate views that will be streamed to users with limited information available on the edge server? How to select from the multiple views on the client side to *maximize the perceived viewing quality* for the user? Next, we present how we address these challenges.

4 QOE MODELING FOR VUES

In this section, we discuss the key factors that can potentially affect the QoE of transcoding-based volumetric video streaming. We then conduct an IRB-approved user study to explore the suitable machine learning models for quantifying the QoE based on these factors. We also analyze the impact of each factor on user-perceived QoE.

4.1 QoE Factors

We consider both the factors introduced by remote rendering of volumetric content on the edge and traditional factors that have been extensively investigated in regular video streaming. Those traditional QoE factors include startup delay, visual quality (*i.e.,* resolution of transcoded 2D frames), quality variation caused by adaptive bitrate streaming to accommodate dynamic network bandwidth, and stall (*i.e.,* rebuffering) time [53]. We use a fixed startup delay of 2 seconds and thus it is not included in our QoE model.

We conduct a separate IRB-approved user study, referred to as `Study-Trace`, to collect viewport movement traces when users watch volumetric videos on smartphones. We recruit 16 users (8 male and 8 female) from a large university. Their age ranges from 18 to 55. Three of them are university faculty or staff members. Each participant watches volumetric videos (described in §7.1) on a smartphone while we collect the movement trajectory. We use those traces to study the users' movement patterns, measure the accuracy of various prediction models, and conduct the performance evaluation in §7. Leveraging this dataset, we apply several off-the-shelf machine learning algorithms (details in §5.1) to predict the viewport movement, and then replay the video with the predicted trajectories to perceive the visual quality. We observe that the predicted trajectories oftentimes deviate from the ground truth, with large viewport drift and unsmooth viewport movement negatively affecting the QoE. Based on those observations, we identify three

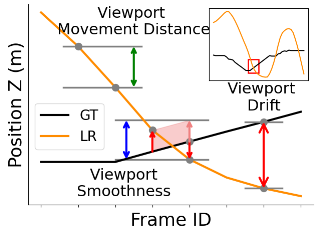

Figure 3: Illustration of three QoE factors.

viewport-related factors that affect the QoE of volumetric videos as described next. We illustrate them using the ground truth motion data (GT) from a real user and the predicted result of linear regression (LR) in Figure 3.

- **Viewport Drift**. For the ith frame of a video, the viewport drift D_V^i is defined as the vector $\overrightarrow{v^i v_p^i}$, where v^i is the center of V^i, the actual viewport, and v_p^i is the center of V_p^i, the predicted/displayed viewport. We employ the magnitude of this vector to represent the viewport drift for a single video frame in our QoE model. For a video with n frames, we define the viewport drift as

$$D_V = \frac{1}{n}\sum_{i=1}^{n} ||\overrightarrow{v^i v_p^i}||$$

When using linear regression as the prediction method, the viewport drift could be as high as 2.83 meters for the single-view approach. Even with the multiview approach adopted by Vues, it can only alleviate the viewport drift along the translational dimensions to some extent, but cannot completely avoid it.

- **Viewport Smoothness**. Besides the viewport drift, the displayed viewport trajectory based on prediction results may not be as smooth as the real trajectory, again due to inaccurate viewport prediction. The unsmooth trajectory of pre-rendered views happens especially when a user starts to move or stops moving. For example, a user from `Study-Trace` moves 1.02 meters within 300ms. However, the predicted motion moves 3.84 meters in the same direction and then moves 2.86 meters in the opposite direction before stopping. This predicted motion, which dramatically deviates from the ground truth, significantly affects the QoE.

We define the viewport smoothness S_V as the magnitude of the difference between the viewport drift vectors of two consecutive frames. For a video with n frames,

$$S_V = \frac{1}{n-1}\sum_{i=2}^{n} ||D_V^i - D_V^{i-1}||$$

- **Viewport Movement Distance**. This is the translational distance that the predicted/displayed viewport moves between two consecutive frames, which may be longer than the actual movement distance of the ground-truth viewport. For example, for the aforementioned user, the actual viewport moved only 1.02 meters. However, the predicted viewport moved 5.7 meters. For the ith video frame, suppose the center of the displayed viewport V_p^i is v_p^i. For a video with n frames, we define the viewport movement distance M_V as

$$M_V = \frac{1}{n-1}\sum_{i=2}^{n} ||\overrightarrow{v_p^i v_p^{i-1}}||$$

- **Motion-to-photon (MTP) Latency**. This latency is the required time to fully reflect user movement, either translational or rotational, on the display. It is known to be a major contributor to motion sickness for VR applications [20]. For systems that prefetch volumetric content [18], this latency is essentially the time to render a point cloud based on the current viewport. In Vues, this latency is the time to display the pre-rendered volumetric content.

By considering the above factors, we introduce the following QoE model for evaluating the viewing of transcoding-based volumetric video streaming:

$$QoE = q(D_V, S_V, M_V, L, R, R_v, B)$$

where L is the motion-to-photon latency, R is the average resolution of individual frames, R_v is the resolution variation of video frames, and B is the stall duration. R, R_v, and B are the QoE factors commonly used in QoE metrics of traditional 2D videos [53]. Next, we conduct an IRB-approved user study to identify the most suitable machine learning model for estimating the QoE.

4.2 User Study for Building the Model

To build a practical model, we conduct an IRB-approved user study, referred to as `Study-QoE`, for collecting real users' ratings of viewing transcoded volumetric content. We recruit 33 participants, including 18 males and 15 females. Most of the participants are students, faculty, and staff from a large university, with their ages ranging from 18 to 40. The goal of our user study is to create a model that can accurately estimate the QoE ratings based on the factors mentioned in §4.1. We develop a volumetric video player for this purpose, which renders and displays the pre-generated video content on smartphones.

We employ the Double Stimulus Comparison Scale (DSCS) method for our user study, which requires participants to watch two video clips (*i.e.,* the original one as the ground truth and its impaired counterpart) that are displayed side by side in random order. During playback, participants can freely make translational movements on smartphones using screen interactions. Afterwards, we ask them to compare the QoE of the two video clips. The clips are pre-generated by applying 2 to 3 levels of impairments introduced by the QoE factors discussed in §4.1 to two volumetric video sources, *Long Dress* and *Loot*, from the 8i dataset [2]. The DSCS method is recommended by ITU [3] to minimize the impact of video content and bias towards unimpaired videos. The high-level approach was also used by Netflix to build the VMAF QoE model [33, 34] that is widely used in the industry. Also note that, in this user trial, we focus on studying participants' translational motion that is unique to volumetric content. For the rotational motion dynamics, Vues can tackle them by leveraging existing methods (§5.1).

We list the parameters to reflect different levels of impairments in Table 2. Their values are determined from our impairment experiments conducted on `Study-Trace` based on our domain knowledge.

In total, we create 144 impaired videos using the settings in Table 2. For videos with low-level (high-level) of viewport drifts, we set D_V^i (in meters) for each pre-rendered video frame to be in the range of [0, 0.08] ([0, 0.36]), which results in an average viewport drift of 0.029 (0.124) meters. We control the difference of viewport drift between consecutive frames to be less than 0.014 (0.056) meters, on average, for videos with low (high) values of viewport smoothness (the lower, the better). The time for rendering each

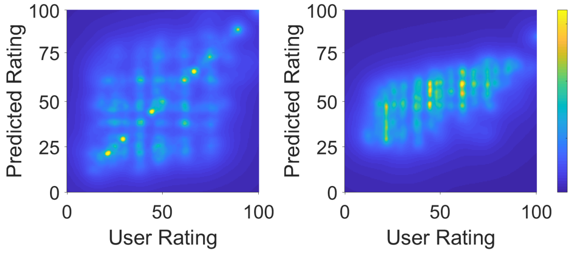

Figure 4: Accuracy heatmaps QoE models: decision tree (left) and linear regression (right).

	Low	Medium	High
Viewport Drift (m)	0.029	-	0.124
Viewport Smoothness (m)	0.014	-	0.056
Motion-to-Photon Latency (ms)	0	100	200
Resolution	2.020	-	2.313
Resolution Variation	0.014	-	0.055
Stall (s)	0	0.033	0.066

Table 2: Different impaired levels of QoE factors.

video frame is fixed to be 100 (200) ms, in order to generate videos with medium-level (high-level) of MTP latency. We consider three resolution levels: 1 (540p), 2 (720p), and 3 (1080p). For videos with low resolution, the level for each frame is randomly and uniformly selected. For videos with high resolution, the level is 3 (1080p) for about 1/3 frames and is 2 (720p) for the rest frames. We re-select the resolution every 40 (10) frames for videos with a low (high) resolution variation. A resolution switch occurs if the re-selected resolution level is different from the previous one. The probability of stalling each frame is set to be 3%. When a stall happens, it lasts for 33ms (66ms) for videos with a low-level (high-level) of stall.

During the user study, the participants watch 144 pairs of volumetric videos on their smartphones in random order, generated using either *Long Dress* or *Loot*, and rate the QoE using seven scores: {left looks much better, better, slightly better, similar to, slightly worse, worse, much worse than right}. *We allow participants to pause and replay the video if needed.* In total, we collect 4,752 ratings.

4.3 A Practical QoE Model

With the data collected from the above user study, we explore different machine learning models for estimating the QoE of transcoding-based volumetric video streaming. We convert all ratings from our participants to the same scale before training the models.

For each video clip pair, the possible scores rated by participants belong to $\{-3, -2, -1, 0\}$, where -3 indicates that the impaired clip incurs much worse QoE than the ground truth clip, whereas 0 indicates the two clips yield the same QoE. Note that the positive ratings (+1, +2, and +3), which indicate that the ground truth has a worse viewing experience compared to the impaired video, do exist in our user study results. We exclude them from our analysis because (1) these ratings are very rare (around 1.6% of all ratings), and (2) most of them appear to be the cases where users made mistakes by unintentionally selecting the opposite option (*e.g.*,-3 vs. +3); including them will thus introduce noises. This issue was also mentioned in the standardization documentation of the DSCS method [3]. We then further standardize and normalize the ratings. The goal of standardization is to eliminate the bias of the data collected from each user and make the data from different users comparable. Users may have different standards for what counts as better and much better. Standardization helps eliminate this difference between users. We leverage the Z-score standardization [29] to rescale the data to have a mean of 0.0 and a standard deviation of 1.0. After the standardization, we merge the ratings from all users, cluster them with similar impairment levels, and normalize the data by rescaling the ratings into the range of [0.0, 100.0].

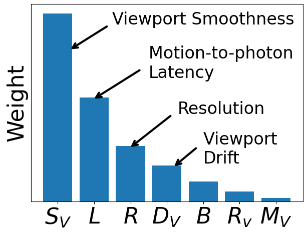

Figure 5: Importance of QoE factors.

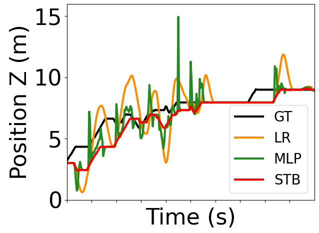

Figure 6: Predicted viewport trajectories.

Model	MAE	RMSE	Pearson	Spearman
SVR	14.1036	17.5473	0.6035	0.5761
LR	14.2638	17.5452	0.6010	0.5968
KNN	15.9504	21.0479	0.5203	0.5034
DT	16.5025	22.5515	0.4706	0.4464

Table 3: Results of four QoE models.

We train four machine learning models, by applying the 10-fold cross-validation, and evaluate their accuracy for estimating the QoE. We consider two classification models, decision tree (DT) and k-nearest neighbor (KNN), and two regression models, linear regression (LR) and support vector regression (SVR). When training these models, we use the values of the identified QoE factors for creating impaired videos as the features and the corresponding ratings from the users as the labels.

We quantitatively compare the prediction errors of these four models and present the results in Table 3. We also report the Pearson [8] and Spearman [7] correlation coefficients (the higher, the better) that measure the strength of the relationship between the actual user ratings and the predicted ratings using these models. Ideally, an accurate model would lead to an error close to 0 and a correlation coefficient close to 1.0. As we can see from this table, although the mean absolute error (MAE) of DT and KNN is only slightly higher than that of LR and SVR (11.6% increase, DT vs. LR), the difference of the root mean squared error (RMSE) between the classification and regression models is significant (28.2% increase, DT vs. LR). The better accuracy of LR and SVR is further verified by their higher Pearson and Spearman correlation coefficients, compared to KNN and DT. Among the two regression models, the accuracy of LR is comparable to that of SVR.

We also infer the model accuracy using the heatmap of user ratings and ratings predicted by these models in Figure 4. The x-axis of the heatmap is the standardized and normalized ratings collected from the user study. The y-axis is the predicted QoE ratings using those models. Each point in the map shows the user rating (on x-axis) and the predicted rating (on y-axis) for the same video watched by the same user. For an accurate model, a majority of the points should be scattered along the diagonal line connecting (0, 0) and (100, 100). The heatmap of KNN (SVR) looks similar to that of DT (LR), and thus is not shown. The heatmaps also suggest that LR has better accuracy compared to DT, which has more samples (albeit with lower intensities) scattered across the heatmap. Thus, we select the lightweight LR to model the QoE in Vues.

It is worth mentioning that the QoE model derived here is specifically designed for assessing the quality of transcoded volumetric videos. It may not be directly applicable to the direct streaming scheme (Table 1). Nevertheless, the high-level data-driven approach we adopted is generic and can be applied to other streaming paradigms for building their QoE models.

4.4 Importance of Different QoE Factors

We investigate the impact of different factors on the overall QoE. Figure 5 plots the coefficient of each component in the trained LR model. The coefficients, which are properly scaled to ensure they are comparable, measure the contributions of their corresponding components to the final predicted QoE. Figure 5 shows that viewport smoothness S_V contributes the most to the QoE, followed by motion-to-photon latency L and video resolution R. Surprisingly, this figure demonstrates that viewport drift has a much less contribution to the QoE, compared to viewport smoothness. A key reason is that viewport drift is usually less perceivable than viewport smoothness. For example, even if all the predicted viewports drift from the ground truth by 0.5 meters, as long as their trajectory is smooth, the impact of the drifted display on the QoE is limited.

5 SYSTEM DESIGN

By leveraging the QoE model in §4 and our observation from **Study-QoE**, we propose Vues for boosting the QoE of transcoding-based volumetric video streaming.

5.1 Multiview Generation

Unlike existing approaches [16, 17] that transcode the viewport into a single 2D video frame using a single prediction model, which may lead to remarkable viewport drift and unsmooth trajectory of displayed content, Vues creates multiple candidate views of a future viewport. The sets of candidate views are generated in real-time based on the output of multiple 6-DoF viewport prediction models. Compared to a single view, generating multiple candidate views allows the client to select the most appropriate view at the playback time. This helps tolerate users' sudden movement and inaccurate prediction of each individual algorithm.

Motivation and Challenges: To demonstrate the limitations of existing single-view single-model transcoding solutions, we plot in Figure 6 the ground-truth viewport trajectory along the Z dimension (backward or forward) and the trajectories predicted by two machine learning models, linear regression (LR) and multilayer perceptron (MLP). We choose these two models because they are lightweight and have been extensively used for viewport prediction of 360° video streaming and volumetric video streaming [18, 19, 43, 52]. We set the history window to be 500ms and the prediction window to be 1000ms. That is to say, at time t seconds, we use the viewport trajectory in the window $[t - 0.5, t]$ to train a machine learning model for predicting the viewport at time $t + 1.0$. We also plot the trajectory generated by a simple method, called stable (STB), which assumes the viewport will not change within the prediction window. While Vues may benefit from more sophisticated deep learning algorithms such as LSTM [52] to yield more accurate viewport prediction, those models usually require substantial efforts for offline training and incur considerable overheads for online inference. We plan to explore this direction in our future work. Note that the Vues framework does not rely on specific choices of prediction methods. Other prediction models can be easily plugged in with little changes to the multiview generation pipeline.

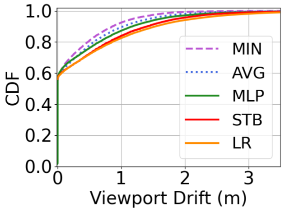

Figure 7: CDF of viewport drifts.

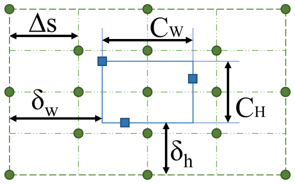

Figure 8: Selecting locations for creating additional views with a large area.

As we can see from Figure 6, none of the methods can accurately predict future viewports. Both LR and MLP lead to significant viewport drift, compared to the ground truth. Moreover, the spikes and fluctuations indicate that the predicted trajectories are less smooth and incur unnecessary motions compared to the ground truth, thus bearing a significantly lower QoE. For the trajectory generated by STB, it bears the same level of smoothness as the ground truth and introduces a moderate level of viewport drifts when the viewers keep moving around. However, it introduces a noticeable latency for updating the displayed view when users start to move or stop moving, which increases motion-to-photon latency and degrades the overall QoE perceived by users.

To quantitatively illustrate the prediction errors of these three methods, we plot in Figure 7 the CDF of viewport drifts for their created trajectories. On average, the viewport drift is 0.35, 0.40 and 0.45 meters for MLP, STB, and LR, respectively. Around 60% of predicted viewports are accurate without any drift. The main reason is that viewers are stationary when watching these video frames. However, for 20% of the predictions, the viewport drift is at least 0.65, 0.80, and 0.86 meters for MLP, STB, and LR, respectively. This indicates that the single-model, single-view solution also incurs larger viewport drifts compared to the ground truth.

Given the deficiency of the single-model, single-view solution, we propose to generate multiple views using multiple models to shield the inaccuracy of individual view/motion, which is inevitable due to the random nature of the viewer's motion. However, the large solution space, the heterogeneity of the prediction models, the complex QoE metrics (§4.3), and the highly dynamic motion all make the multi-view approach challenging.

Our Solution: In order to address the above challenges, we (1) intelligently leverage the prediction results from multiple models, (2) strategically select the views that will be pre-rendered by the edge server and delivered to the client, and (3) carefully choose the final view that will be displayed by the client-side player.

Generating Multiple Candidate Views. Unlike existing approaches that use a single machine learning for predicting future viewports, Vues leverages multiple prediction models that have different accuracies under different scenarios for creating multiple views. A simple solution to generate multiple views is to just combine the prediction results from different models. For instance, if we use the above three models, MLP, STB, and LR, we will create three views for a single frame (one for each model). At the client-side, the player selects a view from the three candidates that is closest to the ground-truth viewport. We show the CDF of viewport drifts introduced by this solution in Figure 7, illustrated by the MIN curve. This simple multiview scheme achieves an average viewport drift of 0.24 meters (a 31.4% reduction from MLP, the most accurate model among the three).

After analyzing the viewport movement traces collected from our **Study-Trace**, we find two issues of the above approach. First, if a user suddenly changes the movement speed (*e.g.,* starts to move faster/slower, or stops moving), all predicted viewports will dramatically deviate from the ground truth. Note that when the movement of users is relatively stable with low speed or when the users have been stationary for a while, the predicted translational positions (*i.e.,* viewpoints) are usually close to the ground truth. Second, if users have been keeping moving around to explore the volumetric video content, the prediction accuracy of the three aforementioned models is much worse than that for the stationary or low movement speed scenarios, leading to substantial viewport drifts.

To solve the above two issues, we need to add more views to cover a larger area than the prediction results. We use the following method to dynamically expand the coverage area of the predicted results and uniformly add more candidate views. We show a sketch of these candidates in Figure 8, where the blue squares are the translational positions predicted by the three models and the green circles are the locations of the additional views created by Vues. We first determine the size of the blue box that is the convex hull of the predicted positions (a rectangle for the case with three models). We then expand the blue box into the outer green box that covers a larger area, to handle the sudden changes of movement pattern. We increase the width of the blue box by δ_w and its height by δ_h to create the outer green box. Finally, we divide the outer green box into a 4×4 grid. To reduce viewport drift, we select all 9 green circles inside the grid and only 8 green circles on the outer boundary box for generating additional candidate views. In total, we identify 20 candidate views in Vues (17 views from the expanded area plus 3 views from prediction models).

Next, we explain how to get δ_w and δ_h. Intuitively, the distance of two neighboring grid points should be approximately the step size Δs that a user could move between two consecutive frames, to accommodate viewport prediction errors. We derive the step size Δs from the viewport traces collected from our **Study-Trace**, which is around 0.1 meters. With Δs known, we can calculate δ_W and δ_H based on the convex hull determined by the prediction results:

$$\Delta s = \frac{1}{4}(2 \times \delta_w + C_W) = \frac{1}{4}(2 \times \delta_h + C_H)$$

where C_W and C_H are the width and the height of the convex hull. By solving the above equations, we have

$$\delta_w = 2 \times \Delta s - \frac{1}{2}C_W, \delta_h = 2 \times \Delta s - \frac{1}{2}C_H$$

Similar to other systems that use viewport prediction [43], Vues dynamically changes the size of the prediction window, for balancing potential video stalls caused by a small window and larger viewport drifts and unsmooth viewport trajectory introduced by a large window. Essentially, the prediction window T_W needs to cover the processing time T_P on both the edge and the client, the network transfer time T_N, and the buffering time T_B introduced by the client-side video player. Hence, $T_W \geq T_P + T_N + T_B$. Vues profiles offline the processing time T_P in advance and keeps monitoring the network latency T_N and buffering time T_B. It then dynamically tunes the prediction window T_W based on those parameters.

When creating multiple candidate views for Vues, we consider mainly translational movements along the X (leftward or rightward) and Z (backward or forward) dimensions, due to two reasons. First, the analysis of viewport trajectories by existing work shows that the movement along the Y (upward or downward) dimension is usually limited, as it is inconvenient for users to crouch down and jump up [18]. Second, even when users move along the Y dimension, the prediction of translational position in that dimension is much more accurate than the X and Z dimensions, mainly because of the low movement speed along the Y dimension [18].

Handling Rotational Movement. The design of the above multiview generation mitigates the impact of inaccurate viewport prediction along the translational dimensions on the QoE. Instead of predicting user movements along the rotational dimensions, a simple solution would be to create a 360° panoramic view for each candidate viewport. In this case, even if there are rotational movements within the prediction window, users can still watch their intended content. However, this solution significantly increases bandwidth consumption by delivering redundant video content.

We carefully select the size of the pre-rendered panoramic view to accommodate potential rotational movement within the prediction window. Previous work has shown that when watching volumetric videos on smartphones, the movement speed along the yaw and pitch dimensions is, on average, less than 10° per second, and is less than 20° per second for 80% of the cases [18]. As a result, Vues extends the pre-rendered viewport from the default 90° ×45° field of view (FoV) to a 180° ×90° panoramic view for avoiding the side effect caused by the rotational movements. By doing this, we can correctly render, at the client side, the view expected by users based on the current viewing direction.

For other devices such as headsets, the rotational movement pattern and the viewport prediction accuracy may differ from those on smartphones [18]. Therefore, when applying Vues to headsets, the above FoV values may be adjusted accordingly.

Discussion. Another possible solution to address the inaccurate viewport prediction along the translational dimensions is to leverage image-based rendering [46] such as view interpolation [14] and 3D warping [37]. Image-based rendering has been applied to virtual reality [10], 3D video and graphics streaming [13, 45], and cloud gaming [30], to name a few. However, the integration of image-based rendering into Vues may increase computation overhead on mobile devices for reconstructing the expected view and degrade QoE due to the distortion incurred by the reconstruction. We plan to explore image-based rendering in our future work.

5.2 Megaframe Formulation

After identifying the candidate positions to generate multiple views, we next explain how to merge these views into a megaframe for encoding and streaming.

We define a megaframe to be a group of views (rendered from the same volumetric video frame) that will be streamed to the client. The resolution of a candidate view could be 540p (960×540) or 720p (1280×720). We exclude the 1080p (1920×1080) resolution based on an observation from our **Study-QoE**, which reveals that the improvement introduced by 1080p on the overall QoE is marginal, compared to 720p.[1] Note that the above resolution configurations are chosen based on typical smartphone screen sizes. For other devices such as headsets and tablets, the resolution configurations may differ, but the multiview approach remains generic.

Challenges: There are several challenges when formulating a megaframe, including selecting the most suitable views, arranging chosen views into a megaframe, and determining the resolution for each to-be-delivered view to accommodate the network dynamics. Compared to single-view transcoding schemes, Vues can improve the user-perceived QoE at the cost of extra bandwidth consumption caused by streaming additional views. Thus, it is crucial to limit the number of candidate views and select those that could potentially maximize the QoE. However, it is non-trivial to determine the QoE improvement introduced by a candidate view with the limited information available on the edge server. For example, to calculate the viewport drift and viewport smoothness that are required by the QoE model, we need to know the ground truth viewport, which is the key task of viewport prediction and is not obtainable when formulating a megaframe.

Our Solution: To address the above challenges and balance the QoE improvement and bandwidth consumption, we (1) explore different arrangements of candidate views to encode megaframes with different communication overhead, (2) effectively combine the prediction results from multiple models to estimate the ground-truth viewport when predicting the QoE score of candidate views for ranking, (3) design an intelligent heuristic algorithm that decides the proper number of candidate views based on users' motion, and (4) develop an efficient rate adaptation algorithm for multiview.

Arranging Candidate Views. In Vues, we propose two methods to form a megaframe: spatial arrangement and temporal arrangement. Spatial arrangement encodes candidate views into a single larger frame. We set the resolution of a megaframe using the spatial arrangement to be 4K (3840×2160). Although there are smartphones that can decode 8K videos, this high resolution is unnecessary for Vues due to its efficient algorithm for optimizing the number of candidate views. Thus, a megaframe includes at most nine 720p, or sixteen 540p views, resulting in 10 possible layouts. These layouts include zero to nine 720p views and the rest, if any, will be filled by one or more 540p views.

Temporal arrangement encodes the candidate views sequentially without merging them into a single frame. Different from the spatial arrangement, the number of candidate views in the temporal arrangement is mainly limited by the decoding capability of mobile devices. If the client can decode at most N 720p frames every second, Vues will generate up to $\lfloor N/30 \rfloor$ views to guarantee a 30 FPS decoding rate. Although the temporal arrangement may include fewer views in a megaframe, as we will show in §7, it reduces the consumed bandwidth for the same number of candidate views, compared to its spatial counterpart. We also experiment with other codecs such as the H.265 and get similar results. To select the most suitable views, we need to first rank them by their contribution to the QoE improvement, which we will explain next.

Ranking Candidate Views. The key challenge for ranking the candidate views is to estimate the ground truth viewport that is used in our QoE model. Although the viewport prediction of each individual model may not be accurate, we find that simply averaging the results of these models can improve the prediction accuracy, offering a good estimation of the ground truth. We call this prediction method AVG. The AVG method is inspired by ensemble learning [11, 15, 21] that utilizes multiple learning algorithms for obtaining better predictive performance than what could be achieved by any of the basic learning algorithms alone. As we can see from Figure 7, the AVG prediction method indeed decreases the average viewport drift to 0.30 meters (14.3% reduction compared to MLP). The result of AVG should be close to the center of the bounding box in Figure 8, and thus it is not considered as one of the candidate views to save a spot.

With the estimated ground-truth viewport by AVG, we can compute the score of each candidate view using the QoE model in §4.3. When ranking these views, we use the viewport drift, viewport smoothness, and viewport movement distance as the factors. We use the view that has the highest ranking in the previous megaframe as the viewport of the previously displayed frame. We will determine the resolution and resolution variation in the rate adaptation algorithm as to be detailed next.

Deciding the Number of Views. After ranking the candidate views, we should determine the proper number of views that will be streamed to the client, with the goal of minimizing the bandwidth consumption while not affecting user-perceived QoE. As shown in Figure 6, when the user is stationary or slowly moves, the three prediction models will generate similar results. In this case, sending multiple views with almost no difference will waste the network bandwidth. On the other hand, when the user is moving fast, the predictions are less accurate, requiring more views to compensate the prediction error. Thus, instead of fixing the number of candidate views, we propose a heuristic algorithm to decide the proper number of views based on users' motion. For each frame, the initial number of views is set to be 1. The reason is that when the user is stationary, the three prediction models give the same result, and thus there is no need to send extra views. When the user is moving slowly (indicated by $\delta_W = 0$ and $\delta_H = 0$), we add two more views for the prediction results from LR and MLP. When the user is moving too fast or random (indicated by larger C_W for x-axis or C_H for z-axis shown in Figure 8), we add two more views for a larger C_W or C_H, leading to a maximum of 7 views per frame. As discussed previously, the movement on y-axis is slower than x-axis and z-axis, and thus it is not considered in our algorithm.

We also compare our heuristic algorithm to a deep learning model. The reinforcement learning (RL) model would fit our needs since it selects a series of actions based on the environment and states. RL has been used in video streaming [36]. However, we find that RL does not exhibit a better QoE compared to our heuristic

[1] We divide the user ratings of impaired videos into three groups based on the average resolution level: less than 2 (the average resolution is around 720p), between 2 and 2.27, and higher than 2.27 (at least 1/3 of the views are 1080p). The median user ratings of the three groups are 35.06, 43.70, and 44.21, respectively, indicating the limited QoE improvement of 1080p (mainly due to the small screen size of smartphones).

algorithm due to the high randomness of users' motion. Thus, we apply the heuristic algorithm to decide the number of views.

Adapting to Network Dynamics. Our rate adaptation algorithm is based on the throughput-based ABR algorithms of traditional 2D videos [27], which selects the highest content quality level under the constraint of the (predicted) available bandwidth. Despite a simple concept, the challenge in our case is to determine the highest content quality level: we need to determine not only the number of views, but also each view's resolution; these two dimensions incur a tradeoff that is unique in our multi-view representation and is difficult to balance even if we know the precise network bandwidth.

The two dimensions above form an exponential solution space with respect to the number of views. To make the solution efficient, we develop an approximate algorithm based on the gradient descent concept. In our approach, we start with the list of views determined by the aforementioned heuristic algorithm, with each view having the highest resolution. We then iteratively decrease either the number of views or a view's resolution until the bandwidth constraint is satisfied. Since the views are already ranked by their QoE scores, in each iteration, we take a greedy action by either removing the very last (least important) view, or reducing the resolution of the least important view that is higher than 540p (the lowest quality). Since all the views incur statistically similar bandwidth consumption, manipulating the least important view (in terms of its QoE contribution) ensures bandwidth reduction at the cost of the minimum QoE reduction. We repeat the above process until the bandwidth usage of the remaining views is less than the predicted available bandwidth. We find the above solution works well in practice despite its heuristic-driven nature, as to be evaluated in §7.2. The same content selection approach could be plugged into many other bitrate adaptation algorithms.

5.3 Selection of Displayed View

The above design is for the Vues edge. Next, we explain the method for choosing the most suitable view from the megaframe for the client-side display. A naive solution is to select the view that is the closest to the ground-truth viewport expected by the user (to avoid the drift). However, as demonstrated in our `Study-QoE`, the contribution of viewport drift to the overall QoE is much less than viewport smoothness. Hence, the view chosen by the above approach may not lead to the best QoE.

Vues leverages our proposed QoE model in §4.3 to calculate the QoE score of each candidate view and displays the view with the highest score. In order to do that, Vues keeps tracking the ground-truth viewport trajectory and the viewport of each displayed frame. The client computes the viewport smoothness with the ground truth of the current and previous frames and the displayed viewport of previous frames. The above information enables the client to calculate the QoE score for each candidate and choose the one with the best QoE for display.

6 IMPLEMENTATION

We implement the Vues client on Android devices and the Vues edge on a Linux server. The client is written in Java using Android SDK for networking, decoding, and rendering. The megaframe decoding is implemented with the Android MediaCodec API [1] in asynchronous mode for achieving the best decoding performance. Client-side buffer is realized by storing the decoded frames in FrameBufferObject (FBO) provided by OpenGL. The panoramic frame is rendered with GPU-accelerated OpenGL ES. Our client can decode and render 4K megaframes at 30 FPS on commodity smartphones. The edge is written in C++ on Ubuntu for rendering, encoding, and networking. It renders volumetric content using OpenGL. The megaframe encoding is implemented by the Nvidia codec API [4] and NvPiPe [9]. Our client implementation consists of 4700 LoC in Java and the edge consists of 5800 LoC in C++.

7 EVALUATION

7.1 Experiment setup

Devices and Network: We run our client on Samsung Galaxy S8 (Android 9.0, Snapdragon 835, 4G RAM), and the edge on a desktop PC with Intel Core i7-9700K CPU @ 3.60GHz, GeForce RTX 2080 Ti GPU, and Ubuntu 19.10. We experiment with three network settings. (1) Home WiFi: we connect the edge and client using a commodity 802.11ac AP at 5GHz, with around 300 Mbps bandwidth and `ping` latency <10ms. This represents a good and stable network condition. (2) Emulated LTE networks: we use the `tc` tool [5] to replay 10 network bandwidth traces collected from a commercial LTE network at multiple locations with poor to medium signal strength. The average bandwidth of these traces ranges from 9 to 15 Mbps. They represent limited, fluctuating network conditions. (3) Real-world LTE network. We conduct live experiments over a commercial LTE network in the U.S. with good signal strength. The average bandwidth is around 30 Mbps. Also, we co-locate the edge and the video content server (implemented as a simple TCP server) to ensure that the server-edge path does not become the bottleneck, as this paper focuses on the edge-side transcoding and client-side view selection.

Videos and User Traces: We create two volumetric video datasets for our experiments. The first dataset is referred to as `Video-Regular`, which includes four videos (captured by us) with an average point density of ~195K points per frame. In a point cloud frame, each point occupies 15 bytes (4 bytes each for X, Y, and Z, and 1 byte each for R, G, and B). So the raw bitrate of these videos ranges from 500 to 870 Mbps. The duration of these videos is from 60 seconds to 120 seconds. The second dataset is referred to as `Video-Dense`, which includes two videos with higher point densities (250K and 456K). We use the raw videos released by 8i [2] to generate those two videos. For example, we create the second video by merging two 8i videos (each depicting a single person), leading to a raw bitrate of ~1.6 Gbps. The duration of both videos is 10 seconds.

We employ 2 videos (V1, V2) from `Video-Dense` for the high point density evaluation compared to ViVo [18], and 4 videos (V3–V6) from `Video-Regular` for other experiments. For most evaluations, we employ `Video-Regular` because the videos are longer, and the viewport movement traces (described below) can thus capture more diverse user movement. We replay viewport traces of 16 users collected from `Study-Trace` (§4.1) to make the experiments automated and reproducible. To better illustrate the user experience of Vues, we captured several videos comparing Vues, SingleView, and ViVo.[2]

[2] Vues vs. LR: https://youtu.be/R4DQaSKIGRg;
Vues vs. MLP: https://youtu.be/cwrCQ96isVI;
Vues vs. ViVo: https://youtu.be/zTrLggBXTv8.

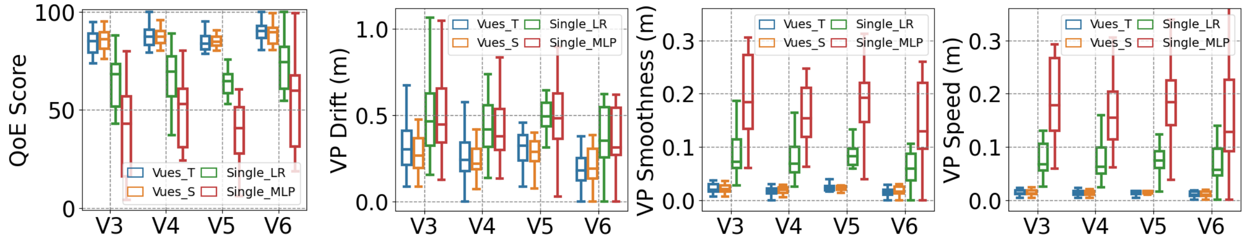

Figure 9: QoE of Vues **and SingleView over WiFi: overall QoE, viewport drift, smoothness, and movement distance (left to right).** Vues **improves QoE by 35% compared to SingleView with viewport smoothness contributing the most to QoE improvement.**

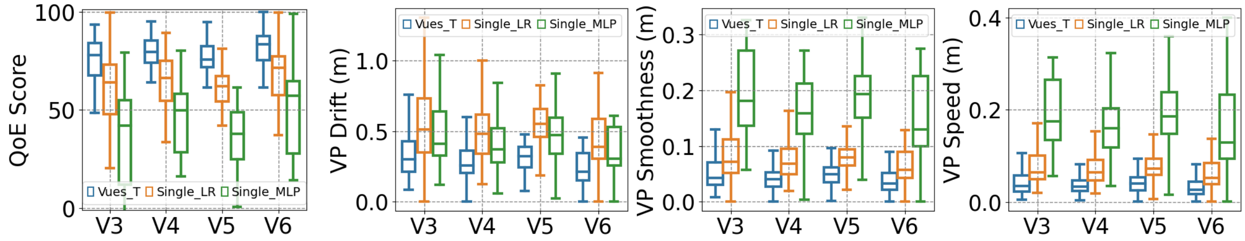

Figure 10: QoE of Vues **and SingleView under emulated LTE: overall QoE, viewport drift, smoothness, and movement distance.** Vues **improves QoE by 22% compared to SingleView, with viewport smoothness contributing the most to QoE improvement.**

Systems to Compare with: We compare two view arrangement schemes of Vues: spatial (denoted as Vues_S) and temporal (denoted as Vues_T). Recall from §5.2 that Vues_S combines multiple views into a 4K megaframe whereas Vues_T encodes each view sequentially without spatial combination. For brevity, in some experiments we only show the results of Vues_T if both arrangement schemes exhibit similar performance. We also compare Vues with two existing volumetric video streaming systems: ViVo [18] and SingleView [16, 17]. ViVo is a recently proposed system where the server directly sends a point cloud stream (without transcoding) to the client; the client performs point cloud decoding and renders the 3D scene for each frame. SingleView is the approach taken by Gül *et al.* [16, 17] where the 3D point cloud stream is transcoded to 2D video frames. The main difference between Vues and SingleView is that SingleView only transcodes and transmits one view per frame as opposed to multiple views in Vues. Other transcoding-based streaming schemes [44, 54] are similar to SingleView: they only apply one single prediction model and send one candidate view for each video frame. We thus only compare Vues with SingleView, which serves as the 2D remote rendering baseline. To ensure apple-to-apple comparison, SingleView also uses two resolutions for bitrate adaptation: 720p for high bandwidth and 540p for low bandwidth, and applies the expanded FoV to handle the rotational movement as described in §5.1. We choose two prediction models for SingleView: LR and MLP, denoted as Single_LR and Single_MLP respectively. The rate adaptation algorithm of SingleView works as follows. When the estimated bandwidth drops below the bandwidth requirement of streaming 720p frames, SingleView will send a 540p frame instead.

7.2 Comparing QoE of Vues and SingleView

We first evaluate the QoE score of Vues compared to SingleView under three network conditions.

Unlimited WiFi. We first run the experiments using WiFi with no bandwidth limit. The client-side buffer ranges from 300ms to 1200ms. On average, the client stores 750ms of frames inside the buffer. Figure 9 shows the QoE scores and the three viewport-related QoE factors (the lower, the better) for SingleView and Vues. The median QoE score of Vues_T, Vues_S, Single_LR, and Single_MLP is 87, 86, 67, and 47, respectively. On average, Vues_T improves the QoE by 35% (up to 85%) compared to Single_LR, and no significant QoE difference compared to Vues_S. We also show three main factors: viewport drift, viewport smoothness, and viewport movement distance. Under good WiFi conditions, there is no stall for all four systems. The resolution, resolution variance, and motion-to-photon latency are similar for the four systems, and thus are not shown in the figure. As shown, Vues significantly improves the viewport smoothness, which contributes the most to the overall QoE improvement. It also results in a much shorter viewport movement distance, compared to SingleView. Overall, the result indicates that with the help of multi-view, Vues can significantly improve the QoE by improving viewport smoothness and reducing viewport movement distance.

Emulated LTE. We next compare the QoE of Vues against SingleView when the network bandwidth is limited. Figure 10 shows the QoE scores and the three viewport-related QoE factors. The results are generated by the 10 low-bandwidth traces described in §7.1. We only show the result of Vues_T in the figure because Vues_S yields similar results to Vues_T. The median QoE score of Vues_T, Single_LR, and Single_MLP is 79, 65 and 44, respectively. On average, under limited bandwidth, Vues improves the QoE score by 22%. The main contributions of the improvement are still from viewport smoothness and movement distance, due to Vues's multi-view approach. Notice that the viewport drift of the two systems is similar. This is because when bandwidth is low, Vues will adaptively select fewer views to send. Those views are chosen to achieve

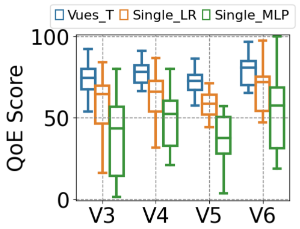

Figure 11: QoE over live LTE. Vues **improves QoE by 21% compared to SingleView.**

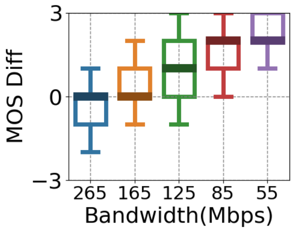

Figure 12: Subjects’ ratings comparing Vues **and ViVo (median values are marked in thick lines).** Vues **achieves higher user-rated scores under lower bandwidth.**

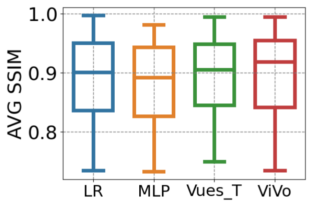

Figure 13: SSIM values of SingleView, Vues**, and ViVo. The small differences are attributed to limitations of traditional image quality metrics.**

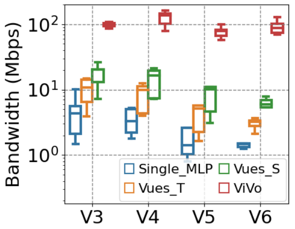

Figure 14: BW usage among three systems. Vues **reduces BW usage by 95% compared to ViVo while incurring higher (yet acceptable) BW usage than SingleView.**

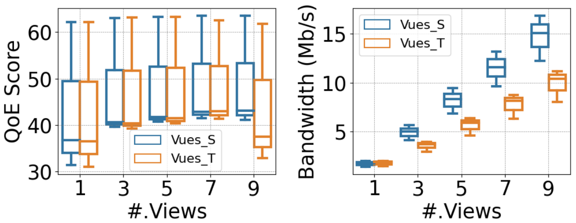

Figure 15: Impact of the number of views on QoE and data usage. As the number of views increases, the QoE improves while the bandwidth usage also increases.

better QoE instead of simply reducing the viewport drift. Therefore, some of the selected viewports may have slightly higher viewport drift but can achieve better smoothness.

Live LTE. We repeat the same experiment under the live LTE network. Figure 11 shows the QoE of Vues and SingleView. The median QoE score of Vues_T, Single_LR, and Single_MLP is 76, 63 and 46, respectively. On average, Vues improves the QoE score by 21% over the real-world LTE network.

7.3 Comparing QoE of Vues and ViVo

We compare the viewing quality of Vues and ViVo with **`Video-Regular`**. A challenge we face here is that ViVo directly streams the volumetric content without transcoding. Thus, we cannot use our model derived in §4 to assess the QoE for ViVo. To address this problem, we conduct a separate IRB-approved user study. The high-level procedure of this user study is similar to **`Study-QoE`** in §4.2, which uses the DSCS method for comparison. We recruit 32 voluntary participants (16 females). Their ages range from 20 to 40+. We invite each of them to watch 45 pairs of videos (video {V4, V5, V6} × 3 representative viewport traces × 5 bandwidth traces described below) side-by-side, in random order. In each pair, one video is generated by Vues, and the other is generated by ViVo. We ask the participants to rate the QoE of the two videos using the same choices for **`Study-QoE`**.

We generate the 5 bandwidth traces using the following method. We notice that ViVo supports five levels of video quality, corresponding to five average point density levels: 160K, 134K, 102K, 80K, 57K points per frame. To comprehensively assess the QoE of ViVo, we thus generate five bandwidth traces, each being just able to support one of the five point cloud density levels. Their average bandwidth is 265 Mbps, 165 Mbps, 125 Mbps, 85 Mbps, and 55 Mbps, respectively. The end-to-end RTT is <10ms. Figure 12 shows the rating distributions, in terms of the Mean Opinion Score (MOS) difference, given by the 32 subjects. A MOS difference of +3 (-3) means that Vues achieves a much better (worse) QoE than ViVo based on the subjects’ feedback; 0 suggests a similar QoE between Vues and ViVo. As shown, under high network bandwidth that allows ViVo to stream high-quality volumetric content without a stall, ViVo’s QoE is similar to Vues with an average MOS difference of 0. For 35% of the cases under the highest bandwidth, ViVo achieves a better QoE than Vues. This is because ViVo can maintain a high QoE by directly streaming dense 3D point clouds at the cost of high bandwidth usage – a desirable approach when the bandwidth is sufficiently high. However, when the bandwidth drops, ViVo has to reduce its point cloud density to adapt to the scarce network resources, leading to significantly reduced QoE. In contrast, Vues’s transcoding approach is much more bandwidth efficient, making it a preferred approach when the bandwidth is limited.

Next, we compare the viewing experience of Vues and ViVo with **`Video-Dense`**. Recall from §7.1, there are two videos in **`Video-Dense`** with point densities of around 250K and 456K per frame respectively. We find that, even under the unlimited WiFi network, the average stall percentage of ViVo for the two videos in **`Video-Dense`** is 95% and 255%, respectively, and makes the video not watchable. This is caused by the excessive decoding and rendering overhead of the high-density point clouds on smartphones with limited processing capability. In contrast, no stall is observed with Vues, making the viewing experience much better than ViVo with higher point density videos. This echos our discussion in §3 of one key benefit of transcoding-based streaming over direct streaming: for the former scheme, the client-side processing overhead is independent of the complexity of volumetric content (the number of points per frame) – the computational workload is offloaded to the edge/cloud.

7.4 Comparing with Traditional Quality Metrics

We are also interested in understanding whether traditional objective image quality metrics can accurately reflect the QoE for Vues. We use SSIM (Structural Similarity Index [50]), a widely adopted perceptual metric measuring image distortion and quality degradation. When calculating the SSIM for each frame, we use the frame transcoded using the viewer’s true movement as the ground truth image. For ViVo, the ground truth image is obtained by transcoding the original point cloud frame according to the viewer's true movement without ViVo's visibility-aware optimizations [18]. Figure 13 shows the SSIM distributions for Single_LR, Single_MLP, Vues_T, and ViVo under unlimited bandwidth, using `Video-Regular` with five representative users' motion traces. As shown, the difference is overall small: the median SSIM values for the four schemes are 0.90, 0.89, 0.91, and 0.92, respectively. ViVo achieves slightly higher SSIM under unlimited bandwidth due to the reasons explained in §7.3. Vues also slightly outperforms the single-view transcoding approaches. We observe qualitatively similar results for PSNR (Peak Signal-to-Noise Ratio [23]). The Spearman's correlation coefficients between SSIM and PSNR are 0.949, 0.914, 0.944, and 0.926 for LR, MLP, Vues_T, and ViVo, respectively.

The SSIM and PSNR results contrast our results in §7.2 where Vues can improve the QoE (quantified using our derived QoE model) by 35%. This is because traditional image quality metrics such as SSIM and PSNR can only capture the visual quality dimension such as viewport drift within each individual transcoded frame. Other important cross-frame factors, in particular those relevant to user's movement such as viewport smoothness and viewport movement distance, cannot be captured by traditional quality metrics. This is why we develop a custom QoE model for Vues.

7.5 Bandwidth Consumption

We next compare the bandwidth consumption of Vues_T, Vues_S, Single_MLP, and ViVo. We evaluate all four systems under the unlimited WiFi network with `Video-Regular`. Note that for SingleView, the prediction model does not affect the bandwidth consumption. Thus, we only show the results of Single_MLP here.

Figure 14 shows the bandwidth consumption of four systems. Each boxplot is generated from 5 representative viewport movement traces. On average, the required bandwidth is 2.01, 4.72, 9.02, and 94.50 Mbps for Single_MLP, Vues_T, Vues_S, and ViVo, respectively. Vues_T reduces the average bandwidth consumption by 40% compared to Vues_S, and reduces the bandwidth consumption by 95% compared to ViVo, but requires 2.35× more bandwidth than SingleView. Vues_S requires 4.49× more bandwidth than Single_MLP. Compared to SingleView, Vues consumes more bandwidth because it streams multiple views instead of a single 720p view. The reason of Vues_T being more bandwidth-efficient than Vues_S is likely attributed to the video codec (we use H.264). Newer video codecs such as the multiview extension (MV-HEVC [47]) may further improve the encoding efficiency for spatial view arrangement. Overall, the results reveal a fundamental tradeoff that Vues aims to better balance: Vues trades more bandwidth for far better QoE. We believe the increase in bandwidth usage is reasonable in most situations considering the significant QoE improvement brought by Vues.

We also compare the bandwidth consumption of Vues and ViVo with higher point density videos from `Video-Dense`. The bandwidth consumption of Vues_T and ViVo is 2.96 and 225.08 Mbps, respectively. Compared to ViVo, Vues's bandwidth consumption for high point-density videos remains almost the same compared to lower point-density videos (V3, V5). The reason is that Vues streams transcoded 2D content instead of raw 3D point clouds. The above results confirm the advantages of Vues when handling high-complexity volumetric content.

7.6 Comparing Arrangement Methods

To compare the accessibility and performance of the two arrangement methods, we benchmark the encoding and decoding performance of two resolutions on our devices (described in §7.1). On average, the edge server can encode 4K frames at 30 FPS and 720P frames at 180 FPS. The client can decode 4K frames at 90 FPS and 720P frames at 550 FPS.

We also compared the two arrangement methods on the QoE score and bandwidth requirement with different numbers of views in Figure 15. There is a QoE drop with 9 views for the temporal arrangement caused by stall: it requires 270 encoding FPS for this setting, which is beyond the encoding capability and results in a stall. With less than 9 views, the temporal arrangement achieves lower bandwidth usage than the spatial arrangement with similar QoE. The right plot in Figure 15 also suggests that Vues_T is more bandwidth-efficient than Vues_S, echoing our finding in Figure 14.

Figure 15 demonstrates the trade-off between bandwidth consumption and QoE gain: with only one view, the bandwidth consumption is minimized but the QoE also becomes the lowest. On the other hand, having more views improves the QoE at the cost of additional bandwidth usage. Given such a complex tradeoff, Vues adaptively decides the number of views to be sent to the client based on the user's movement to achieve a desirable balance between the bandwidth usage and QoE improvement.

7.7 Energy and CPU Utilization

To profile the energy consumption, we play V5 repeatedly over the WiFi network on an SGS8 smartphone for 30 minutes. We start each experiment on a fully-charged phone. After 30 minutes of playback, the battery level drops from 100% to 85% for Vues, 86% for SingleView, and 83% for ViVo. The average CPU utilization on SGS8 is 22% for Vues, 21% for SingleView, and 27% for ViVo. Compared to SingleView, the additional energy and CPU usage of Vues is only 1%. Overall, we believe the resource consumption of Vues is acceptable.

8 CONCLUDING REMARKS

We presented Vues, an edge-assisted system that delivers truly immersive volumetric content with high QoE, low bandwidth consumption, and low decoding overhead on mobile devices. Vues prerenders a volumetric video frame into several 2D views using multiple lightweight viewport prediction models. Compared to direct streaming, it significantly reduces the bandwidth consumption and makes the decoding overhead independent of the quality of volumetric content. Compared to single-view, single-model transcoding, Vues substantially improves the smoothness of displayed viewport trajectory, leading to a much better user experience. Using a practical QoE model that we proposed for transcoding-based volumetric video streaming, we demonstrate that Vues dramatically enhances the QoE by up to 85% (35% on average).

ACKNOWLEDGMENTS

We thank the reviewers and our shepherd for their insightful comments. This research was supported in part by NSF Award 1901103, 1915122, 2106090, 2106771, 2128489, and a Cisco research award. The research of Bo Han was funded in part by 4-VA, a collaborative partnership for advancing the Commonwealth of Virginia.

REFERENCES

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

[2] 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.

[3] ITU-P.913: Methods for the subjective assessment of video quality, audio quality and audiovisual quality of Internet video and distribution quality television in any environment. https://www.itu.int/rec/T-REC-P.913.

[4] Nvidia video codec sdk. https://developer.nvidia.com/nvidia-video-codec-sdk.

[5] tc(8) - linux man page. https://linux.die.net/man/8/tc.

[6] E. Alexiou, I. Viola, T. M. Borges, T. A. Fonseca, R. L. de Queiroz, and T. Ebrahimi. A comprehensive study of the rate-distortion performance in MPEG point cloud compression. *APSIPA Transactions on Signal and Information Processing*, 8:e27, 2019.

[7] 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.

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

[9] T. Biedert, P. Messmer, T. Fogal, and C. Garth. Hardware-Accelerated Multi-Tile Streaming for Realtime Remote Visualization. In H. Childs and F. Cucchietti, editors, *Eurographics Symposium on Parallel Graphics and Visualization*. The Eurographics Association, 2018.

[10] 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.

[11] L. Breiman. Bagging predictors, 1994.

[12] 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.

[13] C.-F. Chang and S.-H. Ger. Enhancing 3d graphics on mobile devices by image-based rendering. In *Proceedings of the Third IEEE Pacific Rim Conference on Multimedia: Advances in Multimedia Information Processing*, PCM '02, page 1105–1111, Berlin, Heidelberg, 2002. Springer-Verlag.

[14] S. E. Chen and L. Williams. View interpolation for image synthesis. In *Proceedings of the 20th annual conference on Computer graphics and interactive techniques*, pages 279–288, 1993.

[15] T. G. Dietterich et al. Ensemble learning. *The handbook of brain theory and neural networks*, 2:110–125, 2002.

[16] 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.

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

[18] 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.

[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] L. J. Hettinger and G. E. Riccio. Visually induced motion sickness in virtual environments. *Presence: Teleoperators & Virtual Environments*, 1(3):306–310, 1992.

[21] J. A. Hoeting, D. Madigan, A. E. Raftery, and C. T. Volinsky. Bayesian model averaging: a tutorial (with comments by m. clyde, david draper and e. i. george, and a rejoinder by the authors. *Statist. Sci.*, 14(4):382–417, 11 1999.

[22] H. Hoppe. Progressive meshes. In *Proceedings of the 23rd Annual Conference on Computer Graphics and Interactive Techniques*, SIGGRAPH '96, page 99–108, New York, NY, USA, 1996. Association for Computing Machinery.

[23] A. Horé and D. Ziou. Image quality metrics: Psnr vs. ssim. In *2010 20th International Conference on Pattern Recognition*, pages 2366–2369, 2010.

[24] M. Hosseini and C. Timmerer. Dynamic Adaptive Point Cloud Streaming. In *Proceedings of ACM Packet Video*, 2018.

[25] Y. Huang, J. Peng, C.-C. J. Kuo, and M. Gopi. A Generic Scheme for Progressive Point Cloud Coding. *IEEE Trans. on Vis. and Computer Graphics*, 14(2):440–453, 2008.

[26] E. Hubo, T. Mertens, T. Haber, and P. Bekaert. The Quantized kd-Tree: Efficient Ray Tracing of Compressed Point Clouds. In *Proceedings of IEEE Symposium on Interactive Ray Tracing*, 2006.

[27] J. Jiang, V. Sekar, and H. Zhang. Improving fairness, efficiency, and stability in http-based adaptive video streaming with festive. In *Proceedings of the 8th International Conference on Emerging Networking Experiments and Technologies*, CoNEXT '12, page 97–108, New York, NY, USA, 2012. Association for Computing Machinery.

[28] S. Katz and A. Tal. On the Visibility of Point Clouds. In *Proceedings of IEEE ICCV*, 2015.

[29] R. Kochendörffer. Kreyszig, e.: Advanced engineering mathematics. j. wiley & sons, inc., new york, london 1962. ix + 856 s. 402 abb. preis s. 79.—. *Biometrische Zeitschrift*, 7(2):129–130, 1965.

[30] K. Lee, D. Chu, E. Cuervo, J. Kopf, Y. Degtyarev, S. Grizan, A. Wolman, and J. Flinn. Outatime: Using speculation to enable low-latency continuous interaction for mobile cloud gaming. In *Proceedings of the 13th Annual International Conference on Mobile Systems, Applications, and Services*, MobiSys '15, page 151–165, New York, NY, USA, 2015. Association for Computing Machinery.

[31] 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.

[32] J.-M. Lien, G. Kurillo, and R. Bajcsy. Multi-camera tele-immersion system with real-time model driven data compression. *The Visual Computer*, 26(3):3–15, 2010.

[33] J. Y. Lin, T. Liu, E. C. Wu, and C. . J. Kuo. A fusion-based video quality assessment (fvqa) index. In *Signal and Information Processing Association Annual Summit and Conference (APSIPA), 2014 Asia-Pacific*, pages 1–5, 2014.

[34] T.-J. Liu, Y.-C. Lin, W. Lin, and C.-C. J. Kuo. Visual quality assessment: recent developments, coding applications and future trends. *APSIPA Transactions on Signal and Information Processing*, 2:e4, 2013.

[35] A. Maglo, G. Lavoué, F. Dupont, and C. Hudelot. 3D Mesh Compression: Survey, Comparisons, and Emerging Trends. *ACM Computing Surveys*, 47(3), 2015.

[36] H. Mao, R. Netravali, and M. Alizadeh. Neural adaptive video streaming with pensieve. In *Proceedings of the Conference of the ACM Special Interest Group on Data Communication*, SIGCOMM '17, page 197–210, New York, NY, USA, 2017. Association for Computing Machinery.

[37] W. R. Mark, L. McMillan, and G. Bishop. Post-rendering 3d warping. In *Proceedings of the 1997 symposium on Interactive 3D graphics*, pages 7–ff, 1997.

[38] R. Mekuria, K. Blom, and P. Cesar. Design, Implementation and Evaluation of a Point Cloud Codec for Tele-Immersive Video. *IEEE Trans. on Circuits and Systems for Video Technology*, 27(4):828–842, 2017.

[39] G. Meynet, Y. Nehmé, J. Digne, and G. Lavoué. PCQM: A Full-Reference Quality Metric for Colored 3D Point Clouds. In *Proceedings of the 12th International Conference on Quality of Multimedia Experience (QoMEX)*, 2020.

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

[41] J. Park, P. A. Chou, and J.-N. Hwang. Rate-Utility Optimized Streaming of Volumetric Media for Augmented Reality. *IEEE Journal on Emerging and Selected Topics in Circuits and Systems*, 9(1):149–162, 2019.

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

[43] 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.

[44] S. Shi, V. Gupta, and R. Jana. Freedom: Fast Recovery Enhanced VR Delivery Over Mobile Networks. In *Proceedings of ACM MobiSys*, 2019.

[45] S. Shi, W. Jeon, K. Nahrstedt, and R. Campbell. Real-time remote rendering of 3d video for mobile devices. pages 391–400, 01 2009.

[46] H. Shum and S. B. Kang. Review of image-based rendering techniques. In *Visual Communications and Image Processing 2000*, volume 4067, pages 2–13. International Society for Optics and Photonics, 2000.

[47] G. Tech, Y. Chen, K. Müller, J.-R. Ohm, A. Vetro, and Y.-K. Wang. Overview of the multiview and 3d extensions of high efficiency video coding. *IEEE Transactions on Circuits and Systems for Video Technology*, 26(1):35–49, 2016.

[48] J. van der Hooft, T. Wauters, F. De Turck, C. Timmerer, and H. Hellwagner. Towards 6dof http adaptive streaming through point cloud compression. In *Proceedings of the 27th ACM International Conference on Multimedia*, pages 2405–2413, 2019.

[49] I. Viola, S. Subramanyam, and P. Cesar. A color-based objective quality metric for point cloud contents. In *Proceedings of the 12th International Conference on Quality of Multimedia Experience (QoMEX)*, 2020.

[50] 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.

[51] J. Xia, P. L. Dubin, T. Izumi, M. Hirata, and E. Kokufuta. Dynamic and electrophoretic light scattering of poly (dimethyldiallylammonium chloride) in salt-free solutions. *Journal of Polymer Science Part B: Polymer Physics*, 34(3):497–503, 1996.

[52] T. Xu, B. Han, and F. Qian. Analyzing viewport prediction under different vr interactions. In *Proceedings of the 15th International Conference on Emerging Networking Experiments And Technologies*, CoNEXT '19, page 165–171, New York, NY, USA, 2019. Association for Computing Machinery.

[53] X. Yin, A. Jindal, V. Sekar, and B. Sinopoli. A control-theoretic approach for dynamic adaptive video streaming over http. *SIGCOMM Comput. Commun. Rev.*, 45(4):325–338, Aug. 2015.

[54] 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.