Efficient Volumetric Video Streaming Through Super Resolution

Anlan Zhang, Chendong Wang
University of Minnesota, Twin Cities
{zhan6841,wang9235}@umn.edu

Bo Han
George Mason University
bohan@gmu.edu

Feng Qian
University of Minnesota, Twin Cities
fengqian@umn.edu

ABSTRACT

Volumetric videos allow viewers to exercise 6-DoF (degrees of freedom) movement when consuming fully 3D content (*e.g.,* point clouds). Due to their truly immersive nature, streaming volumetric videos is highly bandwidth-demanding. In this work, we present to our knowledge a first volumetric video streaming system that leverages 3D super resolution (SR) of point clouds to boost the video quality on commodity devices, and to facilitate the distribution of volumetric content over bandwidth-constrained wireless networks. However, directly applying off-the-shelf 3D SR models leads to unacceptably low performance (~ 0.1 FPS even on a powerful GPU). To overcome this limitation, we propose a series of optimizations to make SR efficient. Our preliminary results indicate that for an edge-assisted (standalone mobile) setup, a small subset of our proposed optimizations can already drastically improve the FPS by a factor of 131× (53×) and reduce GPU memory usage by 83% (76%), while maintaining the same or even better SR inference accuracy, compared to using an off-the-shelf SR model.

ACM Reference Format:
Anlan Zhang, Chendong Wang, Bo Han, and Feng Qian. 2021. Efficient Volumetric Video Streaming Through Super Resolution. In *The 22nd International Workshop on Mobile Computing Systems and Applications (HotMobile 2021), February 24–26, 2021, Virtual, United Kingdom.* ACM, New York, NY, USA, 6 pages. https://doi.org/10.1145/3446382.3448663

1 INTRODUCTION

Today's multimedia content is gaining not only higher resolutions, but also higher degrees of immersion, as demonstrated by, for example, the popularity of 360° panoramic videos [10, 27]. Another emerging type of multimedia content is *volumetric videos* that bear even more immersion and user interactions. In a volumetric video, every frame consists of a 3D scene that is represented by a point cloud (*i.e.,* a set of unsorted points in 3D space). Playing a volumetric video is thus essentially rendering a stream of point clouds at a fast pace such as 30 or 60 frames per second (FPS). The 3D representation offers full immersion, allowing a user to freely explore the 3D scene during video playback. The viewer can move with six degrees of freedom (6-DoF): three rotational dimensions (*i.e.,* viewing direction in yaw, pitch, and roll) and three translational dimensions (*i.e.,* viewpoint position in X, Y, and Z).

HotMobile 2021, February 24–26, 2021, Virtual, United Kingdom

ACM ISBN 978-1-4503-8323-3/21/02…$15.00
https://doi.org/10.1145/3446382.3448663

Due to their true 3D nature, volumetric videos have registered numerous applications in entertainment, education, healthcare, *etc.* that cannot be supported by 2D videos. However, a unique challenge is that streaming high-resolution[1] point clouds over the network is extremely bandwidth hungry [9, 26]. Even with compression, hundreds of Mbps of bandwidth is required to stream point clouds consisting of, for example, hundreds of thousands of 3D points per frame. The high bandwidth utilization has thus become a major hurdle preventing high-resolution volumetric videos from being distributed in particular over wireless/mobile networks.

While static point clouds have been extensively studied by the computer graphics community [14, 15, 17], efficient volumetric video streaming remains an emerging research topic. This paper presents a holistic research agenda on innovating volumetric streaming through Super Resolution (SR). To the best of our knowledge, this is the first effort to *apply SR to volumetric video streaming at line rate on consumer-class devices, with the goal of drastically reducing the bandwidth usage while maintaining a high quality-of-experience (QoE).*

Originally, SR employs deep neural networks (DNN) to improve (*i.e., upsample*) content resolution for 2D content [4, 6, 34, 36] by leveraging the overfitting property of DNN (Figure 1 Left). Recently, the CV/ML community extended SR to *static* point clouds [21, 38] (Figure 1 Right). We find that a well-designed 3D SR model (*e.g.,* PU-GAN [21]) for static point clouds is indeed effective, *e.g.,* reducing bandwidth usage by 74% with little visual quality loss. Such significant bandwidth savings greatly facilitate the wireless distribution of volumetric content. However, when applied to volumetric streaming, it suffers from an unacceptably low performance at ~0.1 FPS even on a high-end GPU. In contrast, off-the-shelf 2D SR models can effortlessly achieve line rate (>30 FPS) on a desktop GPU [36]. Also, there is a lack of visual consistency across upsampled 3D point clouds in consecutive frames (§3). To address these challenges, we judiciously design a series of novel and critical optimizations, many tailored to the unique characteristics of volumetric videos.

Boosting the 3D SR performance from 0.1 FPS to the line rate (at least 30 FPS) on consumer-class devices faces tremendous challenges from both the algorithm and system's perspectives. Our high-level design concept consists of the following: (1) leveraging unique properties of 3D point clouds and strategically trade a small upsampling accuracy loss for significant runtime performance improvement; (2) considering not only the properties of a single point cloud belonging to a volumetric video frame, but also the correlations between point cloud instances in a series of frames in a volumetric video; (3) adapting the SR behavior to the client's processing capability and network condition. Specifically, our proposed innovations include the following.

[1]The resolution of a point cloud is defined as its point density; the resolution of a volumetric video is the average resolution of its point-cloud frames.

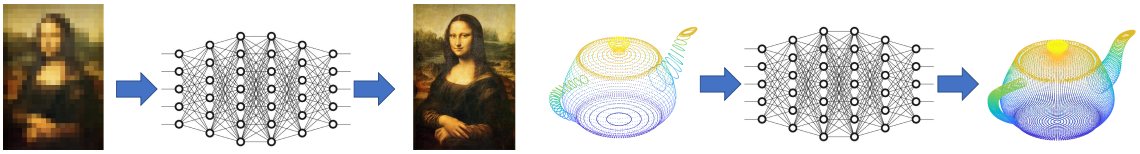

Figure 1: 2D super resolution for images (left) vs. 3D super resolution for point clouds (right).

• Through judicious model trimming and modification, we reduce the SR inference time while maintaining good inference accuracy.
• As off-the-shelf SR models are designed for only a single point cloud instance, we develop robust solutions to facilitate the visual consistency of the SR results across consecutive point cloud frames.
• Leveraging the unique characteristics of point clouds, we merge the low-resolution point cloud with the upsampled SR result to reduce the SR overhead.
• Given that off-the-shelf SR models for point clouds do not upsample colors, we develop resource-efficient schemes to color the upsampled point clouds.
• Our system is expected to work on heterogeneous client devices and over different network conditions. We design an online optimization framework that makes point cloud upsampling adaptive to the potentially scarce and fluctuating computation and network resources.

We are integrating the above components into a holistic system called VoluSR, which can be deployed in two settings. The first is for edge-assisted VR/AR systems, where a VR/AR headset is connected to an edge instance such as a gaming PC. The edge fetches volumetric content from an Internet server and performs SR and rendering. It then transmits rendered images to the headset (*e.g.*, over an HDMI cable). In this setting, VoluSR reduces the Internet-to-edge bandwidth consumption (by up to 74% in our experiments, compared to no SR). The second setting is for a standalone mobile VR/AR system, where a mobile VR/AR headset fetches volumetric content and performs SR and rendering locally. In this setting, VoluSR reduces the bandwidth consumption between the Internet server and the mobile device.

Our preliminary results indicate that for the edge-assisted setup using an Nvidia 2080 Ti GPU (the mobile-only setup using an Nvidia Jetson TX2 embedded system board), a small subset of our proposed optimizations can already drastically improve the FPS by a factor of 131× (53×) and reduce GPU memory usage by 83% (76%), while maintaining the same or even better SR inference accuracy, compared to using unoptimized, off-the-shelf SR model. VoluSR thus has the potentials of transforming SR-boosted volumetric video streaming from theory to practice on commodity client devices.

2 RELATED WORK

Volumetric Video Streaming. In the literature, there are only a few studies on volumetric video streaming. DASH-PC [11] extends dynamic adaptive streaming over HTTP (DASH) to volumetric videos. PCC-DASH [31] is another DASH-based streaming scheme of compressed point clouds with bitrate adaptation support. Park *et al.* [25] make volumetric streaming viewport-adaptive by leveraging 3D tiling. Gül *et al.* [8] propose to utilize remote cloud rendering for low-latency volumetric streaming. Recently, we also propose to leverage the edge to render volumetric videos for mobile devices [26]. Another recent work ViVo [9] introduces visibility-aware volumetric video streaming, which delivers mainly the (predicted) visible portion to the viewer to reduce resource consumption. None of the above work considers 3D SR.

Point Cloud Super Resolution (SR). We can divide the related work on SR for point clouds into two categories: optimization-based [2, 12] and learning based [21, 32, 33, 38]. The optimization-based approach proposed by Alexa *et al.* [2] upsamples a point cloud by iteratively computing Voronoi diagrams on its surface. Huang *et al.* [12] propose an upsampling method that better preserves sharp edges. Most data-driven learning approaches follow the end-to-end workflow established in PU-Net [38], which divides a point cloud into small patches, learns multilevel point features of each patch, expands these extracted features, and reconstructs the points from the expanded features. To extend PU-Net, PU-GAN [21] introduces data amendment into SR through a generative adversarial network (GAN); MPU [32] splits a large upsampling network into several small subnets, each focusing on a different level of detail; AR-GCN [33] integrates residual connections into graph convolution networks (GCN). As to be described in §3, all the above methods are designed for upsampling a *single* point cloud; they suffer from numerous limitations, in particular, unacceptable performance when applied to volumetric video streaming.

3 BACKGROUND AND MOTIVATION

SR was initially designed for improving the visual quality of 2D images [4, 34]. Assisted by deep neural networks (DNNs), a learning-based SR consists of an offline training phase and an online inference phase. When applied to a video v, SR trains a DNN model M that upsamples low-resolution frames $L(v)$ to high-resolution ones $H(v)$, using the original (high-resolution) frames $F(v)$ as the training data. In the inference phase, the server sends M and $L(v)$ to the client, which infers $H(v) = M(L(v))$. SR leverages the overfitting property of DNN to ensure that $H(v)$ is highly similar to $F(v)$. It achieves bandwidth reduction since the combined size of M and $L(v)$ is usually much smaller than that of $F(v)$. There have been successful attempts on applying SR to 2D videos [6, 36].

Limitations of Applying Off-the-shelf 3D SR Models to Upsample Volumetric Videos. Recently, the CV/ML research community extended SR to *static* point clouds [21, 38] (§2). In this work, we explore, for the first time, the feasibility of employing SR to upsample volumetric videos. We started with a simple approach: applying PU-GAN [21], a state-of-the-art SR model, to upsample every point cloud frame of a volumetric video. PU-GAN operates by dividing a point cloud into smaller *patches* each consisting of a subset of points. Both SR training and inference are performed on a per-patch basis. Its DNN model is based on a generative adversarial network (GAN) with 29 convolution layers and 2 fully

	Feature Extraction	Feature Expansion	Point Set Generation
% Time	78.3%	19.3%	2.4%

Table 1: Profile the inference time of the PU-GAN model.

connected layers, which realize three stages: feature extraction, feature expansion, and point set generation.

Our testing video was captured by three depth cameras. It has 3,622 frames each consisting of ~100K points depicting a performing actor. We use all its frames to train a PU-GAN network. We set the SR ratio (*i.e.,* upsampling ratio) to 4, making the input and output point clouds consist of roughly 25K and 100K points, respectively. In other words, ideally the model can yield a 300% increase in video resolution.

We next describe our findings. On the positive side, the model can accurately reconstruct each individual frame, *i.e.,* each upsampled point cloud is highly similar to the original one in terms of the geometric structure. We quantify the similarity using the Earth Mover's Distance (EMD [28]) as:

$$\mathcal{L}_{EMD}(I, G) = \min_{\phi: I \to G} \frac{1}{|I|} \sum_{x \in I} ||x - \phi(x)||_2 \quad (1)$$

where I and G are the upsampled point cloud and the ground truth, respectively; $\phi : I \to G$ is a bijection from the points in I to those in G. The average EMD value across all frames is $1.47 \times 10^{-2} m$ that confirms good upsampling accuracy [21]; it is also verified by our visual inspection of each frame. Also encouragingly, we find that SR can achieve significant savings of bandwidth. For this 2-minute video, the sizes of $F(v)$, M, and $L(v)$ are 1.40 GB, 560KB, and 0.36GB, respectively, leading to a bandwidth reduction of 74.2%.

However, we notice three major issues with the vanilla PU-GAN model. First, the runtime performance is extremely poor. On a machine with a state-of-the-art GPU (Nvidia 2080 Ti), the upsampling FPS is only 0.1 – far below our desired FPS of at least 30. This presents a unique challenge for applying SR to volumetric video streaming, as off-the-shelf SR models for regular 2D videos can effortlessly achieve line rate (>30 FPS) on a desktop GPU [36]. Second, the upsampled video exhibits *visual inconsistency*: despite the good upsampling quality of each *individual* frame, the upsampled points shift randomly across consecutive frames even when the geometric shape remains the same. Although the shifted distance of each individual point is negligible, the random movements of a massive number of points cause an unpleasant "flickering" effect that impacts the QoE. Note that such visual inconsistency is introduced during upsampling, as the original point cloud videos do not exhibit this issue. Third, PU-GAN does not support color. We also examined other 3D SR models for point clouds such as MPU [32] and AR-GCN [33]. They all exhibit the same limitations.

4 THE VoluSR FRAMEWORK

As described in §1, volumetric video streaming is extremely bandwidth hungry. To overcome this challenge, in this study, we develop VoluSR, the first system that aims to perform SR at line rate for volumetric videos on commodity devices. We next describe its six main components denoted as **C1** to **C6**.

C1: Simplify SR Model and Patch Generation. To satisfy the key timing requirement of point cloud streaming, we optimize the SR model by reducing its inference time while maintaining good inference accuracy. As shown in Table 1, By profiling the runtime performance of the vanilla PU-GAN model, we find that the bottleneck of the inference is feature extraction (accounting for 78% of the inference time), which is thus our optimization target. Through careful model trimming in the feature extraction stage, we find that it is feasible to speed up the model inference with little impact on the inference accuracy. Specifically, our current model tuning consists of the following three approaches. (1) We remove the last two dense layers of feature extraction as well as several heavy-weight convolution layers in the feature expansion stage, as they are found to contribute little to the upsampling accuracy; (2) we replace the original feature extraction function with a spherical kernel [20] for more efficient convolution operations; (3) we judiciously remove a small number of features to reduce the GPU memory consumption. At a high level, our experiences and results (§5) indicate that off-the-shelf models developed by the CV/ML community still have considerable room for system performance improvement when being integrated into volumetric video streaming. We plan to refine the above results using more systematic optimization techniques and diverse point cloud videos, as well as to investigate how to simplify other SR models.

VoluSR also optimizes the patch generation. Recall from §3, to ensure a small model size, a 3D SR model divides a point cloud into small patches as basic units for upsampling. We discover that the patch generation process incurs a high overhead. Again take PU-GAN as an example. It generates the patches by applying k-nearest neighbors (kNN) to the seeds created by downsampling. Since the generated patches may overlap, after upsampling, PU-GAN applies the furthest point sampling [24] to remove duplicated points. We consider two directions to mitigate the above overhead. First, we can generate the patches offline since the point cloud frames are known for video on demand (VOD) streaming. Second, we can simplify the patches' geometry shapes to make them easy to be manipulated at runtime. Specifically, the shapes of patches generated by kNN and the furthest point sampling do not have closed-form mathematical expressions. This makes it difficult to perform many operations in VoluSR such as viewport intersection tests in **C5**. We will explore the Voronoi diagram [7] for efficiently generating the patches. In a Voronoi diagram, the area (patch) that each vertex (seed) s belongs to consists of points whose distances to s are less than or equal to those to any other vertex. This key property naturally ensures that each patch consists of a "continuous" region of points, and the patch sizes tend to be homogeneous (assuming the seeds are uniformly selected). An even faster approach is to divide the space into cubic grids, with each non-empty grid (which contains points) corresponding to a patch. Both approaches above bring no overlap among patches, thus eliminating the overlap removal step.

C2: Consider Consistency across Frames. As mentioned in §3, independently applying an SR model to individual point cloud frames can cause visual inconsistency – a problem that is overlooked in 2D SR. Some generic solution ensures the output consistency of image processing algorithms by training an RNN model using consecutive 2D frames [19]. However, this is too computationally expensive in 3D SR. Instead, we take two methods to mitigate this problem. First, we modify the loss function of the SR model

to ensure cross-frame consistency. Specifically, we add to the loss function a penalty term based on the inconsistency between the current patch and the corresponding patches in the previous frames. The penalty term is formulated using the EMD metric (Eq. 1).

The second approach we adopt is to reuse SR results. Let A and B be two patches at the same location of two consecutive frames. Let $l(A)$ and $h(A)$ be the low-resolution (input to SR) and high-resolution (output of SR) versions of A, respectively. Similar notions $l(B)$ and $h(B)$ are also defined. We can calculate a "motion vector" $\overrightarrow{V_l}$ from $l(A)$ to $l(B)$ that approximates how the points in $l(A)$ are transformed to the points in $l(B)$. We then use $h(A)+\overrightarrow{V_l}$ to approximate $h(B)$. This brings two benefits. First, the transition from $h(A)$ to approximated $h(B)$ is smooth, thus eliminating the inconsistency problem; second, we do not need to run SR on $l(B)$, thus reducing the computation overhead. The downside is that the approximation of $h(B)$ may not be accurate if A and B are geometrically very different. Therefore, we only apply the above approximation if $\overrightarrow{V_l}$ is small (*e.g.,* when A and B belong to a static object or the lower body of a standing speaker). For VOD streaming, since the content is known, $\overrightarrow{V_l}$ can be computed offline using methods such as bipartite graph matching, or point cloud registration [30].

C3: Merge SR Input with the SR Output. The inference stage of almost all machine learning models are conceptually the same and straightforward: feed the input features to the model, which then subsequently produces the output label. After the output label is generated, the input features are no longer useful. The same procedure applies to our investigated 3D SR models [21, 32, 33]. We instead make two interesting observations regarding SR for point clouds. First, a point cloud is a set of unstructured points, meaning that multiple point clouds can be combined into a single point cloud by simply merging their points. Second, SR's output points are different from the input due to the complex, non-linear operations taken by the SR model. Based on these two observations, VoluSR overlays the SR model's input (*i.e.,* low-resolution point clouds) onto the SR model's output (*i.e.,* high-resolution point clouds). This helps improve the upsampling quality when the SR model is fixed, or reduce the computation overhead while maintaining a similar upsampling quality. For example, in order to achieve 3× upsampling, we can use a 2× SR model and merge the input with the output, instead of directly using a 3× SR model that is more computationally heavy. We experimentally confirm that both methods (*e.g.,* 3× SR and 2× SR with merging) yield similar upsampling accuracy. In general, to achieve an n× upsampling ratio, we can merge the input with the output of an $(n-1)$× SR model.

C4: Color the SR Results. VoluSR supports coloring the upsampled volumetric videos. To achieve this goal, we are exploring two approaches. One is to modify the SR model itself by also inferring the color components. This approach is expected to accurately reconstruct the original colors but at the cost of nearly doubling the SR workload, because the model needs to infer not only each point's position (X, Y, Z), but also its color (R, G, B). The other approach is to approximate a point's color using the color information in the low-resolution frame (*i.e.,* the input to the SR model) through, for example, interpolation. This approach can be made lightweight but may also be less accurate. To balance the performance and visual quality, the above two approaches can be combined: using interpolation-based approximation for patches with simple, homogeneous colors, and using SR with color support for patches with complex color patterns. In addition, we can relax the color fidelity requirement as the distance between the viewer and the content increases.

C5: Viewport-adaptive Streaming. With 6-DoF movements, viewers can freely change the viewing position and direction to the displayed volumetric content. Since the viewer oftentimes only perceives a small portion of the entire point cloud, the client can choose to only upsample patches that are predicted to fall into the view frustum (*i.e.,* the viewport in 3D); for outside-viewport patches, the client can use a low SR ratio or not perform upsampling at all to reduce the resource footprint. This viewport-adaptive optimization requires 6-DoF prediction of the viewport in the near future, whose feasibility was demonstrated in our prior work [9].

C6: Make SR Adaptive to Computation and Network Resources. For traditional video streaming, a major challenge is to design an adaptive bitrate (ABR) algorithm that adapts the video quality to the network condition. While the same adaptation for volumetric video streaming is important (and new), we emphasize that adapting to the available *computation resources* is also critical to SR-enhanced volumetric video streaming because upsampling is computationally heavy. An even bigger challenge faced by VoluSR is that the consumption of computation and network resources incurs a critical tradeoff: increasing the SR ratio reduces the network bandwidth consumption but increases the computation resource usage, while decreasing the SR ratio goes the other way. VoluSR therefore employs a discrete optimization framework to formally balance this tradeoff and to handle situations where both types of resources are insufficient. We next describe our current formulation. For each frame i, a possible way to model its QoE is:

$$QoE_i = Q_i - \alpha(I_{1,i} + I_{2,i}) - \beta \cdot T_i^{stall} \tag{2}$$

where Q_i represents the visual quality of all the patches belonging to frame i; $I_{1,i}$ is the quality change from frame $i-1$ to frame i; $I_{2,i}$ is quality variation across the patches within frame i. A large inter-frame quality change or infra-frame patch quality change will degrade the viewer's QoE. T_i^{stall} denotes the incurred stall duration due to the excessive network delay or local computation delay of SR. α and β are the weights penalizing the quality variation and stall, respectively. We define Q_i, $I_{1,i}$, and $I_{2,i}$ as follows.

$$l_{i,j} = d_{i,j}u_{i,j}(1 - E_{SR}(u_{i,j})), \qquad Q_i = \sum_{\text{patch } j} v_{i,j} l_{i,j} \tag{3}$$

$$I_{1,i} = \left\| \frac{\sum_j v_{i,j} l_{i,j}}{\sum_j v_{i,j}} - \frac{\sum_j v_{i-1,j} l_{i-1,j}}{\sum_j v_{i-1,j}} \right\|, I_{2,i} = \textbf{StdDev}\left(\{l_{i,j} | \forall j, v_{i,j} > 0\}\right) \tag{4}$$

In Eq. 3, $l_{i,j}$ is the quality of patch j in frame i. For each patch, the client needs to make two decisions: (1) select quality level $d_{i,j} \in \{1, ..., k\}$ and fetch it from the server; (2) apply SR ratio $u_{i,j} \in \{1, ..., k'\}$ to the fetched patch. k and k' are the number of quality levels stored on the server and the number of possible SR ratios, respectively ($u_{i,j} = 1$ means no SR is applied). The resulting quality of the patch is $l_{i,j} = d_{i,j}u_{i,j}(1 - E_{SR}(u_{i,j}))$ where $E_{SR}(u_{i,j})$ denotes the quality loss due to SR. Q_i is defined as the sum of the $l_{i,j}$ values across all the visible patches of frame i. $v_{i,j} \in \{0, 1\}$ is a

O_1	The vanilla PU-GAN model [21].
O_2	O_1 and optimizing patch generation: divide the space into cubic grids, with each non-empty grid corresponding to a patch.
O_3	O_2 and trimming some layers of PU-GAN: remove the last two dense layers of feature extraction and several heavy-weight convolution layers in the feature expansion stage.
O_4	O_3 and accelerating the convolution operations: replace the original feature extraction function with a spherical kernel function (SKF) in the feature extraction stage.
O_5	O_4 and generating fewer features: remove a small number of features to reduce the GPU memory consumption.

Table 2: Our currently implemented optimizations in C1.

	O_1	O_2	O_3	O_4	O_5
GPU Memory Usage (MB)	7065	7065	7065	1811	1171
Frames per second (FPS)	0.1	2.5	4.0	7.9	13.1
Average Accuracy (cm)	1.47	0.71	0.68	0.93	0.80

Table 3: Memory usage, upsampling FPS, and upsampling accuracy of O_1 to O_5 (edge-assisted setup).

	O_1	O_2	O_3	O_4	O_5
Memory Usage	4760	3981	4173	1521	1138
Frames per second (FPS)	0.2	1.0	1.7	5.0	10.7
Average Accuracy (cm)	3.50	3.08	3.11	3.11	2.55

Table 4: Memory usage, upsampling FPS, and upsampling accuracy of O_1 to O_5 (mobile-only setup).

binary function indicating whether patch j in frame i is visible in the viewport. In Eq. 4, $I_{1,i}$ is the difference of the average quality of visible patches between frames i and $i-1$, and $I_{2,i}$ is the standard deviation of all the visible patches in frame i. In Eq. 2, T_i^{stall} can be estimated by comparing the frame's playback deadline and its total downloading/processing latency, which consists of downloading, decompressing, (optional) upsampling, and rendering the patches. We will derive the parameters such as α and β through user studies where participants give subjective QoE scores to volumetric video snippets with different Q_i, $I_{\{1,2\},i}$, and T_i^{stall}.

When running on a client for the first time, VoluSR conducts one-time performance profiling of the device's upsampling speed, which is needed to estimate T_i^{stall}. At runtime, the client performs online optimizations to maximize Eq. 2 for frames in a forward-looking window. The decisions to make consist of which patches to fetch and what SR ratios to apply to the patches.

5 RESULTS OF OPTIMIZATION C1

Our proposed design of VoluSR indicates that it is far from trivial to incorporate (even a well established) deep learning model into a networked system. We next present the preliminary results of our ongoing implementation of VoluSR.

We have conducted a preliminary implementation of **C1** (simplifying SR model and patch generation). Table 2 summarizes our currently implemented optimizations. O_1 denotes the vanilla PU-GAN model as the comparison baseline; O_2 to O_5 are our currently implemented optimizations belonging to **C1**. They are presented in a *cumulative* fashion, *i.e.,* O_i includes every feature of O_{i-1} plus some new feature.

We consider two evaluation setups described in §1: edge-assisted streaming and mobile-only streaming. For the edge-assisted setup, we use a PC with an Nvidia 2080 Ti GPU to upsample the video used in §3 from 25k points/frame to 100k points/frame; for the mobile-only setup, we use a Jetson TX2 embedded system board with a 256-core Nvidia Pascal GPU to upsample the same video from 5k points/frame to 20k points/frame. Both setups use the same SR ratio of 4×. We use a lower resolution in the mobile-only setup because the mobile GPU is much weaker than the desktop GPU.

We consider three metrics shown in Table 3 and 4: (1) maximum GPU memory usage (on Jetson TX2 we measure the system memory shared by GPU and CPU), (2) average upsampling speed (in FPS), and (3) inference accuracy measured in EMD between each upsampled frame and the ground truth. The results of edge-assisted setup and mobile-only setup are shown in Tables 3 and 4, respectively. For the edge-assisted setup, compared to O_1, O_5 reduces the GPU memory usage by 83%, accelerates the upsampling by 131×, and improves the average upsampling accuracy by 46%. Also as shown, each optimization (O_2 to O_5) individually improves the upsampling speed and possibly other metric(s). The mobile-only setup shows a similar trend. Compared to O_1, O_5 reduces the memory usage by 76%, accelerates the upsampling by 53×, and improves the average upsampling accuracy by 27%. The two setups differ in inference accuracy mainly because of their different point cloud densities (the former's video has a 4× point density compared to the latter). We will develop other components proposed in §4, which are expected to further boost the SR performance to line rate.

6 DISCUSSION

3D SR vs. Other Video Compression Schemes. The compression (encoding) of volumetric videos is very different from that of conventional 2D videos. Volumetric videos are usually compressed by special data structures such as Octree [29] and k-d tree [5] on a per-frame basis. It is worth mentioning that SR is orthogonal to and can work in conjunction with them. In fact, in the pilot experiment in §3, we apply both PU-GAN and k-d tree based compression (implemented using Draco [1]) to the point cloud stream. Also, there are very few studies on inter-frame compression of volumetric videos [16, 23], and we find them to be either too complex, or incompatible with 3D SR that works on a per-patch basis. We therefore propose new schemes that reuse SR results across frames (**C2**); they can be regarded as new inter-frame compression methods for volumetric videos.

3D SR vs. 2D SR. There are several existing studies on applying SR to conventional 2D videos [6, 18, 35, 36]. Compared to 2D SR, 3D SR is different and unique because pixel-based 2D videos (including 360° videos) fundamentally differ from volumetric videos whose frames consist of unstructured points or polygons. As a result, 3D SR for volumetric videos has vastly different model structures, and incurs much higher computation overhead compared to 2D SR. We find that merely downscaling an SR model (as performed in several 2D SR studies [6, 36]) is far from achieving 3D SR at line rate. We therefore develop for VoluSR unique optimizations tailored to 3D SR, such as **C2**, **C3**, and **C5**.

Energy Consumption and Heat Dissipation. When VoluSR runs on a mobile device, it may incur non-trivial energy consumption and temperature increase. This is attributed to the high computation overhead of 3D SR as described above. However, while incurring higher CPU/GPU energy usage, applying SR helps reduce the energy consumed by the network (in particular over the energy-hungry cellular networks [13]). This is because SR allows the player to download less data for achieving a similar level of visual quality compared to no SR. Furthermore, the adaptation framework (**C6**) can be enhanced to take into consideration the energy and heat factors. We leave it as our future work.

QoE Model for SR-enhanced Volumetric Videos. There is a plethora of work on understanding and modeling the quality-of-experience (QoE) for conventional 2D videos [3, 22, 37]. For example, a widely accepted QoE metric is a weighted sum of video bitrate, stall duration, video quality switches, and startup delay [37]. However, QoE metrics for volumetric videos still remain an open problem, not to mention the QoE impact of 3D SR. Compared to 2D videos' QoE, the QoE of SR-enhanced volumetric video streaming can be affected by a much wider range of factors such as the point density, viewing distance, SR ratio/distortion, artifacts incurred by patches, to name a few. We plan to construct comprehensive QoE models for SR-enhanced volumetric videos. To get the QoE ground truth, we will play volumetric video segments with strategically crafted impairments to subjects recruited from IRB-approved user studies and ask the subjects to give subjective scores. The resource adaptation algorithm (**C6**) will use the derived QoE model to guide the selection of volumetric content resolutions and SR ratios.

7 CONCLUDING REMARKS

We conduct a first study of reducing the computation overhead for SR-boosted volumetric video streaming through a series of novel optimizations: simplifying SR model and patch generation; reusing SR results across frames; and strategically leveraging the input to SR models. We also develop robust methods to efficiently color the SR results, and make SR adaptive to computation and network resources. The above optimizations show case how methods from computer vision, machine learning, graphics, and mobile systems can be adapted to benefit volumetric video streaming. We are in the progress of developing VoluSR. We also plan to conduct extensive evaluations using real volumetric videos and real users' 6DoF motion traces under diverse network conditions.

ACKNOWLEDGMENTS

We thank the anonymous reviewers for their comments. We also thank Dr. Mi Zhang for shepherding our paper.

REFERENCES

[1] Draco 3D Data Compression. https://google.github.io/draco/.
[2] M. Alexa, J. Behr, D. Cohen-Or, S. Fleishman, D. Levin, and C. T. Silva. Computing and Rendering Point Set Surfaces. *IEEE Trans. on Visualization and Computer Graphics*, 9(1):3–15, 2003.
[3] A. Balachandran, V. Sekar, A. Akella, S. Seshan, I. Stoica, and H. Zhang. Developing a Predictive Model of Quality of Experience for Internet Video. In *Proceedings of ACM SIGCOMM*, 2013.
[4] H. Chang, D.-Y. Yeung, and Y. Xiong. Super-Resolution through Neighbor Embedding. In *Proceedings of CVPR*, 2004.
[5] D. Chen, Y.-J. Chiang, and N. Memon. Lossless Compression of Point-Based 3D Models. In *Proceedings of the 13th Pacific Conference on Computer Graphics and Applications*, 2005.
[6] M. Dasari, A. Bhattacharya, S. Vargas, P. Sahu, A. Balasubramanian, and S. Das. Streaming 360 degree Videos using Super-resolution. In *IEEE INFOCOM*, 2020.
[7] S. Fortune. Voronoi diagrams and delaunay triangulations. In *Comp. in Euclidean geometry*, pages 225–265. World Sci., 1995.
[8] 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 *ACM MMSys*, 2020.
[9] B. Han, Y. Liu, and F. Qian. ViVo: Visibility-Aware Mobile Volumetric Video Streaming. In *ACM MobiCom*, 2020.
[10] J. He, M. A. Qureshi, L. Qiu, J. Li, F. Li, and L. Han. Rubiks: Practical 360° Streaming for Smartphones. In *MobiSys*, 2018.
[11] M. Hosseini and C. Timmerer. Dynamic Adaptive Point Cloud Streaming. In *Proceedings of ACM Packet Video*, 2018.
[12] H. Huang, S. Wu, M. Gong, D. Cohen-Or, U. Ascher, and H. Zhang. Edge-Aware Point Set Resampling. *ACM Transactions on Graphics*, 32(1):9:1–9:12, 2013.
[13] J. Huang, F. Qian, A. Gerber, Z. M. Mao, S. Sen, and O. Spatscheck. A close examination of performance and power characteristics of 4G LTE networks. In *ACM MobiSys*, 2012.
[14] 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.
[15] E. Hubo, T. Mertens, T. Haber, and P. Bekaert. The Quantized kd-Tree: Efficient Ray Tracing of Compressed Point Clouds. In *IEEE Symposium on Interactive Ray Tracing*, 2006.
[16] J. Kammerl, N. Blodow, R. B. Rusu, S. Gedikli, M. Beetz, and E. Steinbach. Real-time Compression of Point Cloud Streams. In *Proceedings of International Conference on Robotics and Automation*, 2012.
[17] S. Katz and A. Tal. On the Visibility of Point Clouds. In *IEEE ICCV*, 2015.
[18] J. Kim, Y. Jung, H. Yeo, J. Ye, and D. Han. Neural-enhanced live streaming: Improving live video ingest via online learning. In *ACM SIGCOMM*, 2020.
[19] W.-S. Lai, J.-B. Huang, O. Wang, E. Shechtman, E. Yumer, and M.-H. Yang. Learning blind video temporal consistency. In *ECCV*, 2018.
[20] H. Lei, N. Akhtar, and A. Mian. Spherical kernel for efficient graph convolution on 3D point clouds. *IEEE Transactions on Pattern Analysis and Machine Intelligence*, 2020.
[21] R. Li, X. Li, C.-W. Fu, D. Cohen-Or, and P.-A. Heng. PU-GAN: A Point Cloud Upsampling Adversarial Network. In *ICCV*, 2019.
[22] Z. Li, A. Aaron, I. Katsavounidis, A. Moorthy, and M. Manohara. Toward a practical perceptual video quality metric. *The Netflix Tech Blog*, 6(2), 2016.
[23] 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.
[24] C. Moenning and N. A. Dodgson. Fast marching farthest point sampling. Technical report, University of Cambridge, 2003.
[25] 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.
[26] F. Qian, B. Han, J. Pair, and V. Gopalakrishnan. Toward Practical Volumetric Video Streaming On Commodity Smartphones. In *ACM HotMobile*, 2019.
[27] F. Qian, B. Han, Q. Xiao, and V. Gopalakrishnan. Flare: Practical Viewport-Adaptive 360-Degree Video Streaming for Mobile Devices. In *ACM MobiCom*, 2018.
[28] Y. Rubner, C. Tomasi, and L. J. Guibas. The earth mover's distance as a metric for image retrieval. *International journal of computer vision*, 40(2):99–121, 2000.
[29] R. Schnabel and R. Klein. Octree-based Point-Cloud Compression. In *Proceedings of the 3rd Eurographics / IEEE VGTC conference on Point-Based Graphics*, 2006.
[30] G. Tam, Z.-Q. Cheng, Y.-K. Lai, F. C. Langbein, Y. Liu, D. Marshall, R. R. Martin, X.-F. Sun, and P. L. Rosin. Registration of 3D Point Clouds and Meshes: A Survey From Rigid to Non-Rigid. *IEEE Trans. on Vis. and Comp. Graphics*, 19(7):1199–1217, 2013.
[31] J. van der Hooft, T. Wauters, F. D. Turck, C. Timmerer, and H. Hellwagner. Towards 6DoF HTTP Adaptive Streaming Through Point Cloud Compression. In *ACM Multimedia*, 2019.
[32] Y. Wang, S. Wu, H. Huang, D. Cohen-Or, and O. Sorkine-Hornung. Patch-based Progressive 3D Point Set Upsampling. In *CVPR*, pages 5958–5967, 2019.
[33] H. Wu, J. Zhang, and K. Huang. Point cloud super resolution with adversarial residual graph networks. In *arXiv preprint arXiv:1908.02111*, 2019.
[34] J. Yang, J. Wright, T. S. Huang, and Y. Ma. Image Super-Resolution Via Sparse Representation. *IEEE Transactions on Image Processing*, 19(11):2861–2873, 2010.
[35] H. Yeo, C. J. Chong, Y. Jung, J. Ye, and D. Han. Nemo: enabling neural-enhanced video streaming on commodity mobile devices. In *ACM MobiCom*, 2020.
[36] H. Yeo, Y. Jung, J. Kim, J. Shin, and D. Han. Neural Adaptive Content-aware Internet Video Delivery. In *USENIX OSDI*, 2018.
[37] X. Yin, A. Jindal, V. Sekar, and B. Sinopoli. A Control-Theoretic Approach for Dynamic Adaptive Video Streaming over HTTP. In *ACM SIGCOMM*, 2015.
[38] L. Yu, X. Li, C.-W. Fu, D. Cohen-Or, and P.-A. Heng. PU-Net: Point Cloud Upsampling Network. In *IEEE CVPR*, 2018.