P^{2}VS: Progressive Partition-Based Volumetric Video Streaming under Network Dynamics

Jingrou Wu
Department of Computer Science and Engineering, Research Institute of Trustworthy Autonomous Systems
Southern University of Science and Technology
Shenzhen, Guangdong, China
University of Technology Sydney
Sydney, New South Wales, Australia
jingrou.wu@student.uts.edu.au

Haoxian Liu
Department of Computer Science and Engineering
Southern University of Science and Technology
Shenzhen, Guangdong, China
liuhx2021@mail.sustech.edu.cn

Jin Zhang*
Department of Computer Science and Engineering, Research Institute of Trustworthy Autonomous Systems
Southern University of Science and Technology
Peng Cheng Laboratory
Shenzhen, Guangdong, China
zhangj4@sustech.edu.cn

Dan Wang
The Hong Kong Polytechnic University
Hong Kong, Hong Kong SAR, China
csdwang@comp.polyu.edu.hk

Jing Jiang
Australian Artificial Intelligence Institute, Faculty of Engineering and Information Technology
University of Technology Sydney
Sydney, New South Wales, Australia
jing.jiang@uts.edu.au

Abstract

Volumetric videos are essential for immersive applications due to their engaging and realistic experiences. However, streaming them in real time over constrained, fluctuating networks remains challenging. Progressive streaming is an effective method to mitigate this issue by gradually enhancing video quality through incremental data transmission. However, existing progressive volumetric streaming solutions often rely on specific compression algorithms or require codec modifications, leading to poor compatibility with standard codecs. In this paper, we propose P^{2}VS, a progressive partition-based volumetric video streaming framework, to achieve codec-independent progressive streaming. Specifically, P^{2}VS leverages the unique structure of point cloud-based volumetric video to incrementally enhance video quality without being constrained by specific compression algorithms. Moreover, we propose adaptive streaming algorithms under this framework to enhance the quality of experience (QoE). Extensive simulations demonstrate that P^{2}VS improves QoE by 21% on average compared to non-progressive streaming schemes. It also achieves better bandwidth efficiency and full compatibility with standard codecs. A prototype is built to verify the feasibility of P^{2}VS.

*Corresponding author.

MM '25, Dublin, Ireland

ACM ISBN 979-8-4007-2035-2/2025/10
https://doi.org/10.1145/3746027.3755403

CCS Concepts

• **Information systems** → **Multimedia streaming**; • **Networks** → **Application layer protocols**; • **Computing methodologies** → **Virtual reality**.

Keywords

Volumetric video; Progressive streaming; Adaptive streaming

ACM Reference Format:
Jingrou Wu, Haoxian Liu, Jin Zhang, Dan Wang, and Jing Jiang. 2025. P^{2}VS: Progressive Partition-Based Volumetric Video Streaming under Network Dynamics. In *Proceedings of the 33rd ACM International Conference on Multimedia (MM '25), October 27–31, 2025, Dublin, Ireland.* ACM, New York, NY, USA, 10 pages. https://doi.org/10.1145/3746027.3755403

1 Introduction

Volumetric videos are becoming increasingly important in immersive applications [7, 23] because they allow users to be fully engaged by providing 6 degrees of freedom movement. This movement is enabled by the unique 3D format of volumetric videos, which are often represented as point clouds. While this 3D structure enhances immersion, it also demands a large amount of bandwidth resources to support volumetric video transmission. For example, the size of a 300-frame video sequence in the 8iVFB dataset is up to 6 GB [5].

Various compression algorithms support multiple encoding versions for adaptive volumetric streaming. Adaptive streaming algorithms [8, 13, 22, 36] adjust quality based on bandwidth conditions and buffer levels, delivering higher or lower-quality chunks accordingly. When conditions improve, these low-quality chunks are either kept unchanged or replaced by high-quality versions [31, 32]. However, both methods fail to fully utilize previously transmitted data, resulting in degraded quality or bandwidth wastage from chunks that are transmitted but never viewed.

Progressive streaming mitigates these issues by gradually enhancing video quality through incremental data transmission, rather than fully replacing previously transmitted chunks. While progressive streaming is commonly used in pixel-based videos through scalable video coding (SVC) [29], SVC is not applicable to volumetric videos due to fundamental differences in video formats. Recently, some progressive volumetric video streaming frameworks [2, 44] are proposed by leveraging scalable octree-based representations. However, these methods require modifications to existing volumetric codecs, reducing their compatibility with standard codecs and limiting practical deployment.

In this paper, we propose P^{2}VS, a progressive partition-based volumetric video streaming framework that enables progressive streaming using standard codecs without modification. The key insight behind this design is the unique data structure of point cloud-based volumetric videos, where quality heavily depends on point cloud density. Hence, points can be naturally partitioned and then combined to enhance quality. We partition the original volumetric video into multiple disjoint subsets of points. Because each partition can be independently encoded by standard volumetric video codecs, our design does not require any modification of current codecs and can be applied to various types of codecs. However, we face two main challenges when designing this approach.

First, the partition design brings a new decision dimension to the progressive video streaming framework. Unlike prior approaches [2, 29, 44] where quality is solely controlled by encoding parameters, our method must consider the point cloud density determined by partitions as well. These two factors jointly affect quality, decoding time, and data size, making it more difficult to determine the best combination under dynamic network conditions. To address this, we investigate how different combinations of partition densities and encoding versions influence visual quality, bandwidth consumption, and decoding efficiency. Based on these insights, we design a partition-aware compression module that selects the most beneficial partition-version pair, ensuring progressive quality improvement without unnecessary overhead.

Second, progressive streaming requires selecting new encoded partitions based on previously received data. However, it is not as straightforward as expected, as integrating encoded partitions sometimes fails to improve quality and may even degrade it (detailed in Sec 4.4). While combining original partitions improves quality by adding more points, encoding introduces compression loss, which can negate these gains. When the compression loss dominates the benefit of additional points, the quality will be degraded instead of improved. To address this, we thoroughly investigate the effects of partition combination under different encoding parameters. Based on these observations, we formulate a quality of experience (QoE) optimization problem that explicitly captures the trade-off between density and compression loss. Furthermore, we propose a buffer-based progressive streaming algorithm to solve this problem while maintaining robustness against network fluctuations.

In summary, we propose P^{2}VS, a codec-independent progressive partition-based streaming framework for volumetric video. It includes partition-aware compression and adaptive buffer-based scheduling to optimize QoE under dynamic network conditions. Extensive evaluations and a working prototype demonstrate its effectiveness and feasibility in real-world scenarios.

2 Related Work

Volumetric video compression aims to reduce the size of 3D data. *Geometry-based* compression such as octree-based [9, 11, 17, 39] and k-d tree-based [4, 10] algorithms, reorganize volumetric videos as tree structures and compress them accordingly. *Video-based* point cloud compression (V-PCC) [15] projects 3D geometries onto 2D video tracks and then leverage mature 2D video compression standards. However, it still suffers from high computing overhead. Some works [28, 30, 35] make efforts to V-PCC for real-time streaming. Recently, *machine learning-based* approaches [25, 26, 41, 43] demonstrate the potential for volumetric compression but are hindered by high computational costs. While our work leverages Google Draco for its real-time decoding capability, the proposed framework can adapt to other compression methods.

Volumetric video streaming aims to improve user experience by adaptive transmission under network dynamics. Some works use *point cloud density* to represent various quality levels. Frameworks such as DASH-PC [13] and VSAS [37] extend dynamic adaptive streaming concepts to volumetric data, enabling point cloud density adjustments to balance quality and bandwidth. ViVo [12] optimizes streaming efficiency by dynamically filtering non-visible tiles and adjusting densities for occluded or distant objects. Others use different *encoding versions*. Approaches like PCC-DASH [34] provide multiple compression levels for different objects within the same scene, enabling fine-grained adaptation. Authors in [20] propose hybrid saliency-based tiling schemes and determine the encoding version for each tile, while POT [18] mitigates viewport prediction errors by employing rolling optimization windows and then adjust encoding versions adaptively. Our work considers both point cloud density and encoding versions for better flexibility.

Progressive video streaming enables quality improvement of prefetched video content by transmitting and combining additional data with it. This framework has been explored in pixel-based video streaming [6, 24, 29, 32], where an initial coarse version of the video is transmitted first, and enhancement data is sent later to improve quality as bandwidth permits. Recently, authors in [2, 44] propose a progressive video streaming based on progressive octree-based compression where volumetric videos are encoded into octree with different depths. Fumos [21] also adopts a progressive refinement framework for volumetric video streaming, relying on neural compression and octree representations. Authors in [3] propose a scalable multiple description coding quality adaptation solution in a real-time conferencing scenario. Moreover, HPC [41] proposes a hierarchical progressive coding framework for Neural Radiance Field-based volumetric video. Unlike prior approaches that achieve progressive refinement through codec-specific hierarchies, our method decouples the streaming logic from compression, enabling flexible partition-based adaptation with standard decoders.

3 Motivation

In this section, we highlight three key limitations in existing systems that motivate the design of P^{2}VS: (1) Non-progressive streaming schemes struggle under network dynamics; (2) Existing codecs lack support for progressive streaming; (3) Adaptive streaming schemes suffer from suboptimal quality selection.

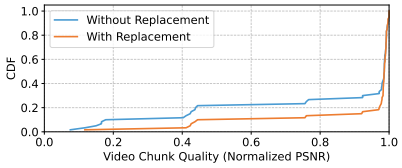


Figure 1: Quality comparison with/without replacement

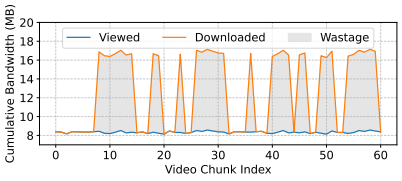


Figure 2: Bandwidth wastage of streaming with replacement

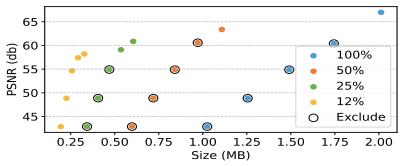


Figure 3: Quality and size comparison with densities and encoding versions

Table 1: Frame setting in the toy example

	Encoding Version 1		Encoding Version 2	
	Size (Mb)	Quality	Size (Mb)	Quality
Whole (100%)	1	0.6	1.5	1
Partition 1 (50%)	0.6	0.3	0.8	0.5
Partition 2 (50%)	0.6	0.3	0.8	0.5

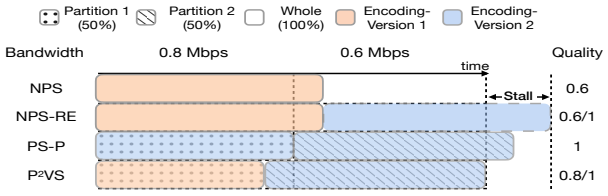


Figure 4: Toy example for motivation

To demonstrate limitations, we conduct preliminary experiments on the *longdress* sequence from the 8iVFB dataset [5], which is encoded using the Draco codec with default parameters. The bandwidth trace is selected from a 5G dataset [27]. The quality is measured by normalized Point Cloud PSNR [19, 33].

Non-progressive volumetric video streaming schemes limitations under network dynamics. Most existing volumetric video streaming schemes [12, 13, 34] follow a non-progressive design, where different-quality video chunks are independent and can not be combined together to achieve higher quality. Moreover, most streaming schemes prefetch low-quality chunks when buffer level and bandwidth are limited to avoid playback stalls. However, once a chunk is transmitted, it is rarely upgraded, even if subsequent bandwidth conditions improve. This leads to degraded video quality and inefficient use of available network resources. In contrast, replacement-based solutions attempt to address this by re-sending high-quality versions of previously streamed chunks. We conduct experiments to demonstrate their benefit, as illustrated in Fig. 1. While replacement improves video quality, it introduces extra bandwidth wastage as indicated in Fig. 2. In this example, 267.55MB of data, about 52.6% of the final viewed size, is wasted on unviewed chunks. These findings motivate the design of a progressive volumetric video streaming framework to enhance video quality and improve bandwidth utilization.

Codec limitations in supporting progressive streaming. Existing volumetric video codecs, such as Draco [10], G-PCC, and V-PCC [11], do not inherently support scalable coding, making it challenging to directly implement progressive streaming. While some works attempt to address this issue by modifying existing codecs [2, 44], these approaches compromise compatibility. Fortunately, the unique structure of volumetric videos, i.e., point clouds, enables gradual quality enhancement by partitioning points into disjoint subsets and progressively combining them. It motivates us to propose a partition-based progressive streaming framework that leverages this feature without codec modifications.

Quality selection limitation in adaptive volumetric video streaming. Most adaptive volumetric video streaming frameworks [12, 13, 34] focus on either point cloud density or encoding versions to support different video qualities. However, such single-dimensional adaptation limits the flexibility and adaptability of the system under varying network conditions. As shown in Fig. 3, different combinations of density (indicated by color) and encoding version (indicated by points) lead to diverse trade-offs between quality and encoded size. Neither dimension alone provides sufficient flexibility to cover the full spectrum of streaming conditions. These findings motivate us to jointly consider both density and encoding versions in our framework design to expand the quality–bitrate adaptation space and support more efficient progressive streaming.

Toy example. To illustrate the limitations of existing strategies and the benefits of our approach, we present a toy example comparing four streaming strategies: (1) non-progressive without replacement (NPS), (2) non-progressive with replacement (NPS-RE), (3) progressive with partitioning (PS-P), and (4) **P²VS**, which considers partitions and encoding versions. Under a two-slot network setting (0.8 Mbps then 0.6 Mbps), each strategy aims to stream a single frame, detailed in Table 1. The encoding version 2 achieves higher quality but a larger size than the encoding version 1.

As shown in Fig. 4, P²VS achieves a final frame quality of 0.8 without incurring playback stalls, outperforming NPS (quality of 0.6) and NPS-RE (quality 1 with stalls). Moreover, it offers more flexible adaptation strategies than PS-P. It can behave like PS-P to reach the highest quality of 1 with short stalls, or slightly sacrifice quality to maintain smooth playback. Overall, P²VS better balances quality and smoothness, demonstrating strong potential for improving streaming performance under dynamic network conditions.

However, we face several challenges when designing P²VS: How to partition videos and how to determine encoding versions for each partition? How to design a streaming strategy to optimize QoE? In the following sections, we present the detailed design of P²VS, addressing these issues.

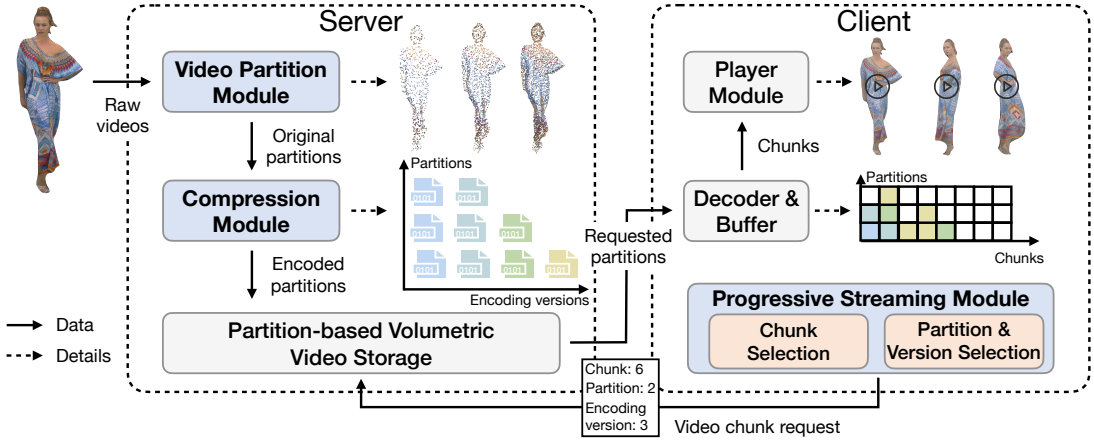


Figure 5: System overview of P^{2}VS

4 P^{2}VS Design

4.1 System Overview

The goal of P^{2}VS is to enable progressive volumetric video streaming using standard codecs without modification. To achieve this, we leverage the unique structure of volumetric videos, the point cloud, which can be partitioned and combined to improve video quality gradually. As illustrated in Fig. 5, the system includes a server for partitioning and compression, and a client for adaptive streaming. The server partitions each frame into disjoint subsets of points via the **Video Partitioning Module**, and encodes them with appropriate quantization parameters using the **Compression Module**, producing multiple quality levels for adaptive streaming. On the client side, the **Progressive Streaming Module** selects the chunk, the partition, and the encoding version to transmit based on bandwidth and buffer states. Received partitions are buffered, decoded, and integrated directly in 3D space to reconstruct full point cloud frames, without requiring any codec-specific merging.

4.2 Video Partition Module

The **Video Partition Module** is designed to divide raw volumetric video frames into multiple partitions, i.e., a subset of points. Since the video quality highly depends on the point density, the combination of different partitions allows progressive quality improvement. We generate partitions at different densities using a binary tree structure, where each node contains half the points of its parent. Points are randomly sampled to form each level of the binary tree. The root node holds the full frame (100% of points), and the following levels represent 50%, 25%, 12.5%, and 6.25% densities. This hierarchical structure not only offers flexible qualities but also reduces transmission overhead compared to naive uniform partitioning. As shown in Table 2, transmitting sixteen 6.25% partitions to reconstruct a 100% frame leads to up to 34.8% more data with Google Draco. In contrast, the binary tree structure allows for more efficient combinations. Partitions are decoded sequentially here, but they can also be decoded in parallel to further reduce latency. Therefore, this design is well-suited for adaptive and progressive streaming under bandwidth-constrained scenarios.

Table 2: Overhead brought by video partition (Draco)

	100%	50%	25%	12.5%	6.25%
Encoded size (MB)	2.3	2.5	2.7	2.9	3.1
Decoding time (ms)	71.9	70.1	68.2	64.1	64.1

Table 3: QP values for different codecs

Codec	Parameter	Low	Medium	High
Draco	Quantization Bits	6	8	10
G-PCC	Position Quant. Scale	0.1	0.25	0.8
V-PCC	Geometry QP	96	64	32

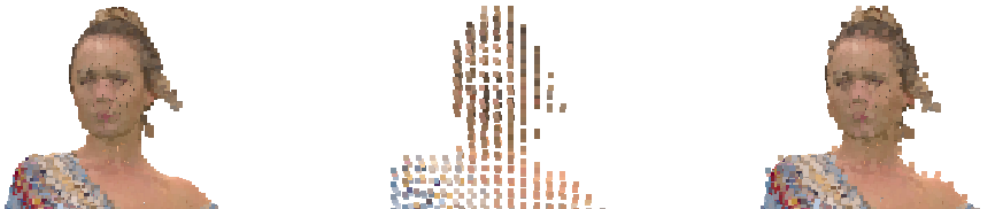
4.3 Partition-Aware Compression Module

Compression is essential for volumetric video streaming due to the large size of raw videos. Standard codecs use quantization parameters (QP) to control quality. [1] We evaluate three popular codecs: Draco [10], G-PCC, and V-PCC [11], on the *longdress* sequence [5]. As shown in Table 3 and Table 4, higher-quality settings increase data size and decoding latency. Despite differences in QP semantics, trends are consistent across codecs. Draco offers the lowest decoding latency and is used as default.

We also observe that at low-quality QPs, increasing point density offers minimal visual improvement but significantly increases compression size and decoding time (see Section 3). This implies that applying the same QP across all partitions leads to inefficiencies.

To address this inefficiency, the **Partition-aware Compression Module** adjusts QP values based on point density. We avoid configurations that produce low quality with large size—for instance, only 6.25% partitions use QP=6. Low-density partitions span a wider QP range, while high-density ones are restricted to high-QP settings, ensuring better trade-offs between quality, size, and decoding cost.

[1] We focus on geometry QP, but attribute QP shows similar trends.



(a) 12.5%-Partition with QP=8 (b) 12.5%-Partition with QP=6 (c) Combination of two partitions

Figure 6: Quality degradation from partition combination

Table 4: The impact of QP values under different codecs

		Draco	G-PCC	V-PCC
Low Quality	Encoded Size (Mb)	1.1	0.03	0.77
	Decoding Time (ms)	60	20	1793
	Quality (db)	42	46	50
Medium Quality	Encoded Size (MB)	1.6	0.2	0.79
	Decoding Time (ms)	66	87	2005
	Quality (db)	55	54	57
High Quality	Encoded Size (MB)	2.2	1.2	0.9
	Decoding Time (ms)	74	705	2295
	Quality (db)	67	64	61

4.4 Progressive Streaming Module

The **Progressive Streaming Module** adaptively requests chunks, partitions, and encoding versions. The goal of this module is to maximize the user QoE during progressive streaming. To achieve this, the module must balance two key decisions: (1) **Partition & Version Selection** to ensure efficient quality improvement, and (2) **Chunk Selection** to balance playback smoothness and video quality. Since the partition design introduces new decision dimensions beyond traditional streaming systems, a new QoE formulation and adaptive streaming algorithm are needed. In the next section, we present the formal formulation of our QoE optimization problem and introduce a partition-based progressive streaming algorithm.

5 Problem Formulation and Algorithm Design

5.1 QoE Optimization Problem Formulation

As in many prior works [38, 40, 42], we adopt a simplified, model-based definition of QoE that combines **video quality** Q_{video}, **quality switch impairment** I_{switch}, and **video stall impairment** I_{stall}.

Video quality Q_{video} is defined as the average quality of visible tiles across all N_c chunks,

$$Q_{\text{video}} = \frac{1}{N_c}\sum_{i}^{N_c} Q_{\text{chunk}}(c_i). \tag{1}$$

Each chunk c_i is divided into tiles $\mathcal{T}_i = \{t_1, ...t_{ij}, ...t_{iN_t}\}$, and only visible tiles contribute to chunk quality:

$$Q_{\text{chunk}}(c_i) = \sum_{j=1}^{N_t} x_{ij} Q_{\text{tile}}(t_{ij}), \tag{2}$$

where $x_{ij} = 1$ denotes a visible tile and $x_{ij} = 0$ otherwise. The tile quality $Q_{\text{tile}}(t_{ij})$ depends on the received encoded partitions $\mathcal{R}_{ij}$. Since not all combinations improve quality due to compression loss (see Fig. 6), we select the subset with the highest quality:

$$Q_{\text{tile}}(t_{ij}) = \max_{\mathcal{R}' \subseteq \mathcal{R}_{ij}} Q(\mathcal{R}'). \tag{3}$$

Video quality switch impairment I_{switch} is the average quality switch impairment between successive chunks $I_{\text{switch}}(c_i)$,

$$I_{\text{switch}} = \frac{1}{N_c - 1}\sum_{i=2}^{N_c} |Q_{\text{chunk}}(c_i) - Q_{\text{chunk}}(c_{i-1})|. \tag{4}$$

Video stall impairment I_{stall} is defined as the average playback delay across all chunks:

$$I_{\text{stall}} = \frac{1}{N_c} I_{\text{stall}}(c_i) = \frac{1}{N_c}\sum_{i=1}^{N_c} (t_{\text{actual}}(c_i) - t_{\text{expected}}(c_i)), \tag{5}$$

where $t_{\text{expected}}(c_i)$ denotes the expected playback time of chunk c_i. It is recursively defined as the previous chunk's actual playback time plus the chunk duration τ_{chunk},

$$t_{\text{expected}}(c_i) = t_{\text{actual}}(c_{i-1}) + \tau_{\text{chunk}}. \tag{6}$$

The actual playback time $t_{\text{actual}}(c_i)$ is the maximum value between its expected play time and the time when the first received data is buffered: $t_{\text{actual}}(c_i) = \max\{t_{\text{expected}}(c_i),\ t_{\text{buffer}}(r_{i,1})\}$.

Hence, the final **QoE model** is the weighted sum of all these three components,

$$QoE = \omega_1 Q_{\text{video}} - \omega_2 I_{\text{switch}} - \omega_3 I_{\text{stall}}, \tag{7}$$

where ω_1, ω_2 and ω_3 are weighted parameters.

The QoE maximization problem aims to select transmission data at each time slot s, denoted as $\mathcal{D} = \{d_1, d_2, \cdots, d_s, \cdots\}$, to optimize QoE. Each d_s represents a set of encoded partitions for tiles in the chunk c_s, i.e., $d_s = \{(p_{sjk}, v_{sjk}) \mid t_{sj} \in \mathcal{T}_s\}$, where p denotes the partition and v denotes the encoding version,

$$\max_{\mathcal{D}} QoE = \omega_1 Q_{\text{video}} - \omega_2 I_{\text{switch}} - \omega_3 I_{\text{stall}} \tag{8}$$

$$\text{s.t.} \quad c_s \in C,\ t_{sj} \in \mathcal{T}_s,\ p_{sjk} \in \mathcal{P}_{sj},\ v_{sjk} \in \mathcal{V}_{sj},\ \forall d_s \in \mathcal{D} \tag{9a}$$

$$\mathcal{R}_i = \{d_s \mid c_s = c_i,\ t_{\text{buffer}}(d_s) \le t_{\text{actual}}(c_i)\}. \tag{9b}$$

Constraint (9a) defines the valid options for transmission, while constraint (9b) specifies the received data before a chunk plays.

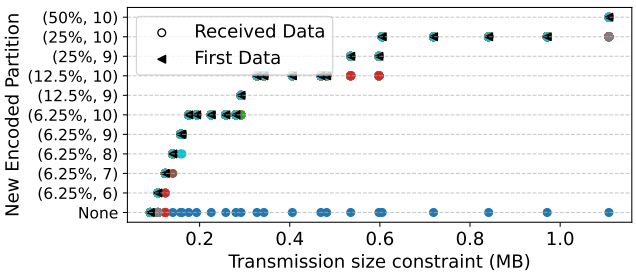


Figure 7: Optimal selection for received data

5.2 P^{2}VS Streaming Algorithm Design

Due to the large decision space, we decompose the problem into a series of per-slot decisions d_s, rather than solving for the complete solution $\mathcal{D}$ at once. Each decision d_s involves selecting a chunk c_s and determining its optimal partition p_s and encoding version v_s.

To further simplify the problem, we consider a subproblem: given a target chunk c_s and a transmission size constraint S_s, what is the optimal partition p_s and encoding version v_s to transmit? We address this via an offline profiling approach. Once the **Partition & Version Selection** mechanism is established, we introduce a **Chunk Selection** algorithm to decide which chunk to transmit.

5.2.1 Partition & Encoding Version Selection. Since additional data does not always improve quality, selecting the appropriate partition and encoding version is critical to avoid transmission wastage. A brute-force search can yield the optimal choice under a transmission size constraint, but it is computationally infeasible due to the large search space. Instead, we propose a greedy algorithm that selects the next optimal encoded partition based on the current transmission size constraint and previously received data.

To guide selection, we precompute an offline mapping from transmission size and prior data to the optimal partition-version pair. As shown in Fig. 7, it captures consistent optimal choices across different received data. Circles and triangles denote selections with and without prior data, respectively, while color indicates downloaded content. A "None" label means no further improvement is possible. We formalize this mapping as $\phi_i : \mathbb{R}^+ \rightarrow (p_{ijk}, v_{ijk}) \mid x_{ij} = 1$ for each visible tile. If no better combination exists, the transmission is skipped. Since our system targets on-demand volumetric video, this profiling is performed once per video asset before deployment, eliminating the need for runtime generalization.

To apply this mapping, we need to determine the transmission size constraint S_s, as illustrated in Alg. 1. We first calculate an allocated time slot duration τ_s based on the classical Buffer-based Approach (BBA) [14] (lines 1-8), and then compute S_s as the product of τ_s and the predicted bandwidth $\hat{b}_s$ (line 9).

5.2.2 Chunk selection. Based on the transmission budget S_s, we propose the **Chunk Selection Algorithm** (Alg. 1) to determine which chunk to transmit. The algorithm first decides whether to transmit a new chunk based on the current buffer level (lines 1–4). Then, the transmission size constraint is obtained via Alg. 1 (line 5). The algorithm evaluates and sorts candidates by their current quality (line 6). For each candidate c_i, if a feasible partition-version pair exists, the algorithm selects c_i for transmission (lines 7–10). If there is no valid combination, the algorithm defaults to transmit the newest chunk (line 11).

Algorithm 1 Slot Budget Allocation Algorithm

Input: Buffer level B_s, thresholds $B_{\text{res}}, B_{\text{cus}}$, durations $\tau_{\min}, \tau_{\max}$, adjustment factor α, bandwidth estimate $\hat{b}_s$
Output: Allocated transmission budget S_s
1: **if** $B_s \le B_{\text{res}}$ **then**
2: $\quad \tau_s \leftarrow \tau_{\min}$
3: **else if** $B_s \ge B_{\text{cus}}$ **then**
4: $\quad \tau_s \leftarrow \tau_{\max}$
5: **else**
6: $\quad$ Calculate a download duration rate factor $F \leftarrow \frac{B_s - B_{\text{res}}}{B_{\text{cus}}}$
7: $\quad$ Calculate the duration $\tau_s \leftarrow \tau_{\min} + F \times (\tau_{\max} - \tau_{\min})$
8: $\quad$ Adjust the duration $\tau_s \leftarrow \tau_s \times \alpha$
9: Calculate budget $S_s = \tau_s \cdot \hat{b}_s$
10: **return** Allocated transmission budget $\S_s$

Algorithm 2 Chunk Selection Algorithm

Input: Current buffer B_s, thresholds $B_{\text{res}}, B_{\text{cus}}$
Output: Chunk to transmit c_s
1: Compute buffer ratio $r_b \leftarrow \max\{0, \frac{B_s - B_{\text{res}}}{B_{\text{cus}} - B_{\text{res}}}\}$
2: Sample replacement flag $RT \sim \text{Random}(1 - P(r_b))$
3: **if** RT is False **then**
4: $\quad$ **return** New chunk to transmit c_{new}
5: Get transmission budget S_s from Alg. 1
6: Identify valid chunks in buffer: $C_s \leftarrow \{c_i \mid t + \tau_s \ge t_{\text{actual}}(c_i)\}$ and sort C_s by increasing chunk quality
7: **for** each c_i in sorted C_s **do**
8: $\quad$ Select data to transmit $d_s \leftarrow \phi_{ij}(S_s)$
9: $\quad$ **if** d_s exists and size < budget S_s **then**
10: $\quad\quad$ **return** Old chunk to transmit c_i
11: **return** New chunk to transmit c_{new}

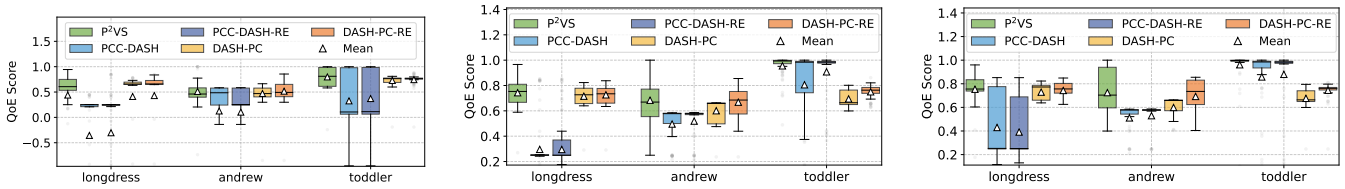
The final P^{2}VS streaming algorithm first performs the **Chunk Selection Algorithm**. Given the transmission budget and the set of received encoded partitions for the selected chunk, it then uses the **Partition & Encoding Version** mapping to determine the optimal partition-version pair to transmit.

6 Evaluation and Implementation

6.1 Experiment Setup

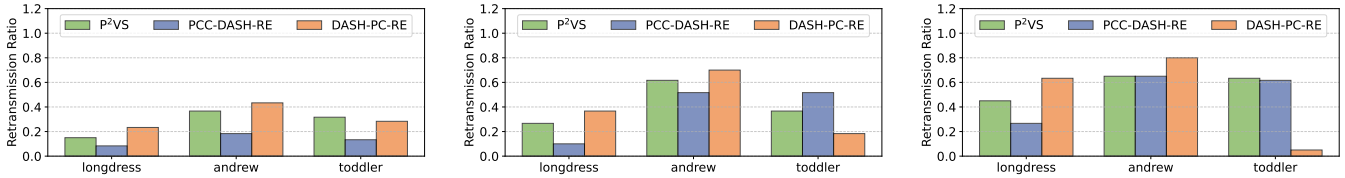
Hardware and dataset. We conduct evaluations on a Mac mini with an Apple M1 chip and 8 GB of memory. We use three volumetric video datasets: 8iVFB [5], CMU Panoptic [16], and Microsoft Voxelized Upper Bodies [1], selecting one representative sequence from each (see Table 5). All videos are recorded at 30 FPS and partitioned using a binary tree with a maximum depth of 5, yielding a minimum point cloud density of 6.25%. Each partition is encoded using Google Draco with five QPs, $v \in \{6, 7, 8, 9, 10\}$.

Communication resource. We select three network traces from a 5G dataset [27]. The first trace has an average bandwidth of 64 Mbps and exhibits a low-high pattern. The second trace has an average bandwidth of 70 Mbps with high throughput in the initial period and low throughput in the later time. The third trace has an average bandwidth of 142 Mbps and has a periodic pattern.



(a) QoE comparison under BW1 (b) QoE comparison under BW2 (c) QoE comparison under BW3

Figure 8: Comparison of QoE performance of different videos under different network traces



(a) Replacement count under BW1 (b) Replacement count under BW2 (c) Replacement count under BW3

Figure 9: Comparison of replacement ratio comparison of different videos under different network traces

Table 5: Video information

	8iVFB	CMU Panoptic	Upper Bodies
Selected Video	Longdress	Toddler	Andrew
Video Duration (Second)	10	54	10
# Points (K/Frame)	834	225	284

Evaluation metrics and algorithm settings. We evaluate volumetric video quality using Point Cloud PSNR [19, 33], normalized to [0,1] via min-max scaling: the minimum corresponds to a 6.25% partition with QP=6, and the maximum to the original frame with QP=10. Each video chunk contains 10 frames. The buffer holds 6 chunks, with B_{res} and B_{cus} set to $\frac{1}{4}$ and $\frac{3}{4}$ of the buffer size, respectively. The adjustment factor α is 0.9. Bandwidth is predicted by the average bandwidth of the past 5 slots.

Baselines. We compare P²VS with four existing volumetric video systems, all adapted for tile-based streaming: (1) **PCC-DASH** [34] selects encoding versions based on bandwidth but does not support replacement. We adopt the same QPs as P²VS for a fair comparison; (2) **PCC-DASH-RE** extends PCC-DASH with quality replacement when the buffer exceeds B_{res}. Due to the high bandwidth requirements of partitions with high point cloud density, the partition density is restricted to 25%; (3) **DASH-PC** [13] supports multiple densities but lacks adaptive logic, so we reuse the PCC-DASH strategy. It does not reuse previously transmitted chunks for quality enhancement; (4) **DASH-PC-RE** enables progressive enhancement with QP fixed at 9 to ensure quality gain. The retransmission strategy follows the same logic as P²VS.

6.2 Experimental Results

Overall QoE performance. In Fig. 8, we compare the QoE values of five algorithms under three traces and three videos. Among these algorithms, P²VS consistently achieves the highest and most stable QoE, thanks to its progressive streaming that enables smooth quality upgrades under network fluctuations. PCC-DASH and PCC-DASH-RE exhibit unstable QoE due to large quality gaps between encoding versions despite small size differences. Therefore the video quality varies dramatically from frame to frame. DASH-PC and DASH-PC-RE demonstrate more stable QoE because point cloud density variations have lower quality gaps but higher size gaps, as indicated in Sec. 4.3. However, their performance remains lower than P²VS, as they lack encoding version flexibility. In summary, P²VS improves QoE by 6%-37% on average.

Impact of videos. Performance varies with video characteristics. PCC-DASH and PCC-DASH-RE perform poorly on high-size videos like *longdress*, suggesting that encoding versions alone cannot efficiently handle large data volumes. In contrast, P²VS achieves higher QoE on smaller videos by offering more flexible combinations of partitions and encoding versions, enabling finer adaptation to bandwidth variation and better overall quality.

Impact of bandwidth traces. The three typical bandwidth traces show varying impacts on QoE values. BW1 features high fluctuation with a low-high pattern. P²VS buffers during low throughput and gradually improves quality as bandwidth increases. In contrast, baselines without replacement miss the later high-bandwidth opportunity, resulting in low QoE. Even replacement-enabled schemes struggle due to persistently low buffer levels in the early phase. BW2 follows a high-low pattern, allowing replacement-based schemes to briefly improve QoE by quickly filling the buffer. P²VS sustains high QoE across videos by leveraging early bandwidth for progressive upgrades, while PCC-DASH variants only perform well on the smaller *toddler* video. BW3 offers stable, periodic bandwidth with the highest average throughput. Although all methods benefit, P²VS still leads due to its finer-grained and more flexible adaptation.

Impact of replacement. We analyze the replacement behavior of P²VS, PCC-DASH-RE, and DASH-PC-RE in Fig. 9. All three exhibit similar replacement ratios across scenarios due to sharing

Table 6: Comparison of bandwidth wastage between P^{2}VS and PCC-DASH-RE

	Longdress	Andrew	Toddler
P^{2}VS	15.54 MB	0 MB	4.23 MB
PCC-DASH-RE	931.04 MB	895.02 MB	659.62 MB

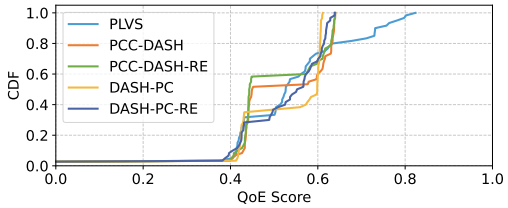


Figure 10: QoE comparison under G-PCC

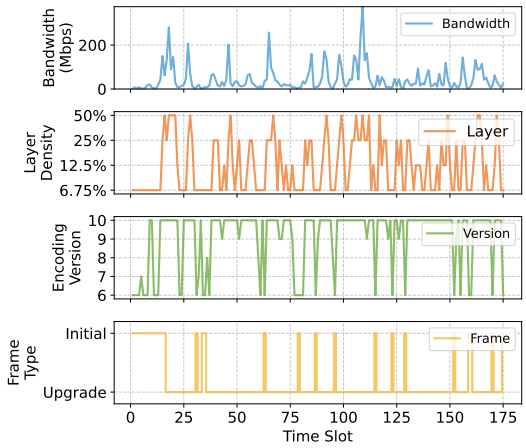


Figure 11: Run-time streaming request of P^{2}VS

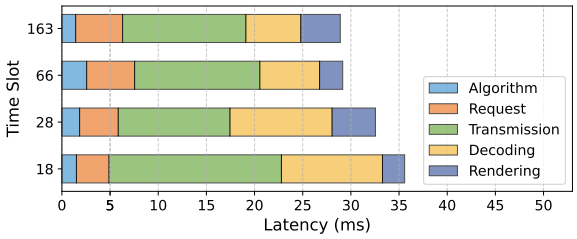


Figure 12: End-to-end latency breakdown

the same replacement strategy. Replacements are more frequent under BW2 and BW3, where high bandwidth accelerate buffer growth. They also occur more often with low-size videos, which allow greater potential for quality improvement. However, in high-bandwidth conditions with small videos, DASH-PC-RE shows fewer replacements since it often transmits the highest-quality version upfront, leaving little room for enhancement.

Bandwidth wastage. Replacement can cause bandwidth wastage when downloaded data is later discarded. We compare P^{2}VS and PCC-DASH-RE in Table 6 under three traces. PCC-DASH-RE keeps only the highest-quality chunk and discards others, leading to significant waste. In contrast, P^{2}VS reuses partitions and incurs wastage mainly when low-quality data is fetched early and later overridden by higher-quality versions (see Sec. 4.4). Although P^{2}VS introduces slightly more overhead than DASH-PC-RE, it achieves significantly better QoE, making the tradeoff worthwhile.

Evaluation with G-PCC. To further validate that our proposed framework is independent of the choice of codec, we conduct an additional QoE comparison using G-PCC as the compression codec instead of Draco. As shown in Fig. 10, P^{2}VS outperforms baselines as the previous evaluations do. This result confirms that our design is not tied to a specific codec and can generalize well to other standard volumetric video compression algorithms. We do not include V-PCC in this comparison due to its significantly longer decoding time, which is unsuitable for real-time streaming.

6.3 Implementation and Case Study

We implement a prototype of P^{2}VS to validate its feasibility using Google Draco for real-time decoding. We analyze the runtime behavior and end-to-end latency through a case study.

Case study. We evaluate the P^{2}VS prototype by streaming the *andrew* video at 30fps over Wi-Fi. As shown in Fig.11, partition and encoding selections adapt dynamically to bandwidth changes. Under low bandwidth, P^{2}VS chooses minimal-size partitions (e.g., 6.25%, QP=6) to avoid stalls. When bandwidth improves, it progressively upgrades quality with higher densities and QPs. Once the buffer exceeds a threshold, P^{2}VS also refines previously streamed frames, e.g., using 50% density and QP=10 during time slots 30–60, as shown in the bottom plot of Fig. 11.

End-to-end latency breakdown. We analyze end-to-end latency in the case study shown in Fig. 12, using representative time slots covering typical network and playback conditions. The progressive streaming algorithm introduces less than 2 ms overhead, indicating minimal latency impact. While request delay (4 ms) is excluded from simulation, it varies slightly with bandwidth. Transmission delay dominates total latency (28–72%), depending on network conditions. Decoding and rendering take 10–15 ms, varying by video content. Overall, these results confirm P^{2}VS's potential for real-time volumetric streaming.

7 Conclusion

We propose P^{2}VS, a progressive partition-based volumetric video streaming system that improves bandwidth efficiency under network dynamics. By enabling quality enhancement through reusable partitions, P^{2}VS supports codec-independent streaming without modifying existing codecs. We further optimize encoding parameters based on compression characteristics and formulate a QoE-driven scheduling algorithm. Experiments show that P^{2}VS enhances QoE and reduces bandwidth waste. Future work includes extending to multi-user and AR/VR settings, integrating perceptual metrics and user studies, and exploring hybrid codec strategies to balance efficiency and compatibility.

Acknowledgments

This work was supported by the National Natural Science Foundation of China (NSFC) under Grant T2495254 and Fang Keng Fellowship.

References

[1] Sergio Orts Escolano Charles Loop, Qin Cai and Philip A. Chou. 2016. Microsoft Voxelized Upper Bodies – A Voxelized Point Cloud Dataset. *ISO/IEC JTC1/SC29 Joint WG11/WG1 (MPEG/JPEG) input document m38673/M72012* (2016).

[2] Hao Chen, Bowei Xu, Shaowei Wang, Xun Cao, and Zhan Ma. 2024. Toward Adaptive Volumetric Video Streaming: A Joint Network-Viewport Adaptation Framework. *Comm. Mag.* 63, 3 (Aug. 2024), 197–203. https://doi.org/10.1109/MCOM.001.2400053

[3] Matthias De Fré, Jeroen van der Hooft, Tim Wauters, and Filip De Turck. 2024. Scalable MDC-Based Volumetric Video Delivery for Real-Time One-to-Many WebRTC Conferencing. In *Proceedings of the 15th ACM Multimedia Systems Conference* (Bari, Italy) *(MMSys '24)*. ACM, New York, NY, USA, 121–131. https://doi.org/10.1145/3625468.3647617

[4] O. Devillers and P.-M. Gandoin. 2000. Geometric Compression for Interactive Transmission. In *Proceedings of the IEEE Visualization 2000 (VIS 2000)*. IEEE, 319–326. https://doi.org/10.1109/VISUAL.2000.885711

[5] Eugene d'Eon, Bob Harrison, Taos Myers, and Philip A Chou. 2017. 8i Voxelized Full Bodies - a Voxelized Point Cloud Dataset. *ISO/IEC JTC1/SC29 Joint WG11/WG1 (MPEG/JPEG) input document WG11M40059/WG1M74006* (2017).

[6] Anis Elgabli, Vaneet Aggarwal, Shuai Hao, Feng Qian, and Subhabrata Sen. 2018. LBP: Robust Rate Adaptation Algorithm for SVC Video Streaming. *IEEE/ACM Trans. Netw.* 26, 4 (Aug. 2018), 1633–1645. https://doi.org/10.1109/TNET.2018.2844123

[7] Peter Fasogbon, Surarshan Bisht, Jaakko Kernen, Ugurcan Budak, Lauri Ilola, and Lukasz Kondrad. 2024. Volumetric Video Use Cases for XR Immersive Streaming. In *Proceedings of the 2024 8th International Conference on Education and Multimedia Technology* (Tokyo, Japan) *(ICEMT '24)*. ACM, New York, NY, USA, 1–8. https://doi.org/10.1145/3678726.3678754

[8] Yu Gao, Pengyuan Zhou, Zhi Liu, Bo Han, and Pan Hui. 2022. *Fras: Federated Reinforcement Learning Empowered Adaptive Point Cloud Video Streaming.* arXiv:2207.07394 https://doi.org/10.48550/arXiv.2207.07394

[9] Diogo C. Garcia and Ricardo L. de Queiroz. 2018. Intra-Frame Context-Based Octree Coding for Point-Cloud Geometry. In *Proceedings of the 25th IEEE International Conference on Image Processing (ICIP 2018)*. 1807–1811. https://doi.org/10.1109/ICIP.2018.8451802

[10] Google. 2018. Draco. https://github.com/google/draco

[11] D. Graziosi, O. Nakagami, S. Kuma, A. Zaghetto, T. Suzuki, and A. Tabatabai. 2020. An Overview of Ongoing Point Cloud Compression Standardization Activities: Video-Based (V-PCC) and Geometry-Based (G-PCC). *APSIPA Transactions on Signal and Information Processing* 9 (2020), e13. https://doi.org/10.1017/ATSIP.2020.12

[12] Bo Han, Yu Liu, and Feng Qian. 2020. ViVo: Visibility-Aware Mobile Volumetric Video Streaming. In *Proceedings of the 26th Annual International Conference on Mobile Computing and Networking* (London, United Kingdom) *(MobiCom '20)*. ACM, New York, NY, USA, Article 11, 13 pages. https://doi.org/10.1145/3372224.3380888

[13] Mohammad Hosseini and Christian Timmerer. 2018. Dynamic Adaptive Point Cloud Streaming. In *Proceedings of the 23rd Packet Video Workshop* (Amsterdam, Netherlands) *(PV '18)*. ACM, New York, NY, USA, 25–30. https://doi.org/10.1145/3210424.3210429

[14] Te-Yuan Huang, Ramesh Johari, Nick McKeown, Matthew Trunnell, and Mark Watson. 2014. A buffer-based approach to rate adaptation: evidence from a large video streaming service. In *Proceedings of the 2014 ACM Conference on SIGCOMM* (Chicago, Illinois, USA) *(SIGCOMM '14)*. ACM, New York, NY, USA, 187–198. https://doi.org/10.1145/2619239.2626296

[15] Euee S. Jang, Marius Preda, Khaled Mammou, Alexis M. Tourapis, Jungsun Kim, Danillo B. Graziosi, Sungryeul Rhyu, and Madhukar Budagavi. 2019. Video-Based Point-Cloud-Compression Standard in MPEG: From Evidence Collection to Committee Draft [Standards in a Nutshell]. *IEEE Signal Processing Magazine* 36, 3 (2019), 118–123. https://doi.org/10.1109/MSP.2019.2900721

[16] Hanbyul Joo, Hao Liu, Lei Tan, Lin Gui, Bart Nabbe, Iain Matthews, Takeo Kanade, Shohei Nobuhara, and Yaser Sheikh. 2015. Panoptic Studio: A Massively Multiview System for Social Motion Capture. In *2015 IEEE International Conference on Computer Vision (ICCV)*. 3334–3342. https://doi.org/10.1109/ICCV.2015.381

[17] Kyungjin Lee, Juheon Yi, Youngki Lee, Sunghyun Choi, and Young Min Kim. 2020. GROOT: a Real-Time Streaming System of High-Fidelity Volumetric Videos. In *Proceedings of the 26th Annual International Conference on Mobile Computing and Networking* (London, United Kingdom) *(MobiCom '20)*. ACM, New York, NY, USA, Article 57, 14 pages. https://doi.org/10.1145/3372224.3419214

[18] Jie Li, Huiyu Wang, Zhi Liu, Pengyuan Zhou, Xianfu Chen, Qiyue Li, and Richang Hong. 2023. Toward Optimal Real-Time Volumetric Video Streaming: A Rolling Optimization and Deep Reinforcement Learning Based Approach. *IEEE Trans. Cir. and Sys. for Video Technol.* 33, 12 (Dec. 2023), 7870–7883. https://doi.org/10.1109/TCSVT.2023.3277893

[19] Jie Li, Xiao Wang, Zhi Liu, and Qiyue Li. 2021. *A QoE Model in Point Cloud Video Streaming.* arXiv:2111.02985 https://doi.org/10.48550/arXiv.2111.02985

[20] Jie Li, Cong Zhang, Zhi Liu, Richang Hong, and Han Hu. 2023. Optimal Volumetric Video Streaming With Hybrid Saliency Based Tiling. *Trans. Multi.* 25 (Jan. 2023), 2939–2953. https://doi.org/10.1109/TMM.2022.3153208

[21] Zhicheng Liang, Junhua Liu, Mallesham Dasari, and Fangxin Wang. 2024. Fumos: Neural Compression and Progressive Refinement for Continuous Point Cloud Video Streaming. *IEEE Trans. Vis. Comput. Graph.* 30, 5 (May 2024), 2849–2859. https://doi.org/10.1109/TVCG.2024.3372096

[22] Junhua Liu, Boxiang Zhu, Fangxin Wang, Yili Jin, Wenyi Zhang, Zihan Xu, and Shuguang Cui. 2023. CaV3: Cache-assisted Viewport Adaptive Volumetric Video Streaming. In *Proceedings of the 30th IEEE Conference Virtual Reality and 3D User Interfaces (VR 2023)*. 173–183. https://doi.org/10.1109/VR55154.2023.00033

[23] Zhi Liu, Qiyue Li, Xianfu Chen, Celimuge Wu, Susumu Ishihara, Jie Li, and Yusheng Ji. 2021. Point Cloud Video Streaming: Challenges and Solutions. *Netwrk. Mag. of Global Internetwkg.* 35, 5 (Sept. 2021), 202–209. https://doi.org/10.1109/MNET.101.2000364

[24] Afshin Taghavi Nasrabadi, Anahita Mahzari, Joseph D. Beshay, and Ravi Prakash. 2017. Adaptive 360-Degree Video Streaming using Scalable Video Coding. In *Proceedings of the 25th ACM International Conference on Multimedia* (Mountain View, California, USA) *(MM '17)*. ACM, New York, NY, USA, 1689–1697. https://doi.org/10.1145/3123266.3123414

[25] Dat Thanh Nguyen, Maurice Quach, Giuseppe Valenzise, and Pierre Duhamel. 2021. Learning-Based Lossless Compression of 3D Point Cloud Geometry. In *Proceedings of the IEEE International Conference on Acoustics, Speech and Signal Processing (ICASSP 2021)*. 4220–4224. https://doi.org/10.1109/ICASSP39728.2021.9414763

[26] Maurice Quach, Giuseppe Valenzise, and Frederic Dufaux. 2019. Learning Convolutional Transforms for Lossy Point Cloud Geometry Compression. In *Proceedings of the IEEE International Conference on Image Processing (ICIP 2019)*. 4320–4324. https://doi.org/10.1109/ICIP.2019.8803413

[27] Darijo Raca, Dylan Leahy, Cormac J. Sreenan, and Jason J. Quinlan. 2020. Beyond Throughput, the Next Generation: a 5G Aataset with Channel and Context Metrics. In *Proceedings of the 11th ACM Multimedia Systems Conference (MMSys '20)*. ACM, New York, NY, USA, 303–308. https://doi.org/10.1145/3339825.3394938

[28] Michael Rudolph, Stefan Schneegass, and Amr Rizk. 2024. Transcoding V-PCC Point Cloud Streams in Real-time. *ACM Trans. Multimedia Comput. Commun. Appl.* (Aug. 2024). https://doi.org/10.1145/3682062

[29] Heiko Schwarz, Detlev Marpe, and Thomas Wiegand. 2007. Overview of the Scalable Video Coding Extension of the H.264/AVC Standard. *IEEE Transactions on Circuits and Systems for Video Technology* 17, 9 (2007), 1103–1120. https://doi.org/10.1109/TCSVT.2007.905532

[30] Yuang Shi, Bennett Clement, and Wei Tsang Ooi. 2024. QV4: QoE-based Viewpoint-Aware V-PCC-encoded Volumetric Video Streaming. In *Proceedings of the 15th ACM Multimedia Systems Conference* (Bari, Italy) *(MMSys '24)*. ACM, New York, NY, USA, 144–154. https://doi.org/10.1145/3625468.3647619

[31] Kevin Spiteri, Ramesh Sitaraman, and Daniel Sparacio. 2018. From Theory to Practice: Improving Bitrate Adaptation in the DASH Reference Player. In *Proceedings of the 9th ACM Multimedia Systems Conference* (Amsterdam, Netherlands) *(MMSys '18)*. Association for Computing Machinery, New York, NY, USA, 123–137. https://doi.org/10.1145/3204949.3204953

[32] Liyang Sun, Fanyi Duanmu, Yong Liu, Yao Wang, Yinghua Ye, Hang Shi, and David Dai. 2019. A Two-Tier System for On-Demand Streaming of 360 Degree Video over Dynamic Networks. *IEEE Journal on Emerging and Selected Topics in Circuits and Systems* 9, 1 (2019), 43–57. https://doi.org/10.1109/JETCAS.2019.2898877

[33] Dong Tian, Hideaki Ochimizu, Chen Feng, Robert Cohen, and Anthony Vetro. 2017. Geometric Distortion Metrics for Point Cloud Compression. In *Proceedings of the IEEE International Conference on Image Processing (ICIP)* (Beijing, China). IEEE, 3460–3464. https://doi.org/10.1109/ICIP.2017.8296925

[34] Jeroen van der Hooft, Tim Wauters, Filip De Turck, Christian Timmerer, and Hermann Hellwagner. 2019. Towards 6DoF HTTP Adaptive Streaming Through Point Cloud Compression. In *Proceedings of the 27th ACM International Conference on Multimedia* (Nice, France) *(MM '19)*. ACM, New York, NY, USA, 2405–2413. https://doi.org/10.1145/3343031.3350917

[35] Yihan Wang, Yongfang Wang, Tengyao Cui, and Zhijun Fang. 2024. Fast Video-Based Point Cloud Compression Based on Early Termination and Transformer Model. *IEEE Transactions on Emerging Topics in Computational Intelligence* 8, 3 (2024), 2336–2348. https://doi.org/10.1109/TETCI.2024.3360290

[36] Lai Wei, Yanting Liu, Fangxin Wang, and Dan Wang. 2024. FewVV: Few-Shot Adaptive Bitrate Volumetric Video Streaming With Prompted Online Adaptation. *IEEE Internet of Things Journal* (2024). https://doi.org/10.1109/JIOT.2024.3424977

[37] Lai Wei, Yanting Liu, Fangxin Wang, Dayou Zhang, and Dan Wang. 2023. VSAS: Decision Transformer-Based On-Demand Volumetric Video Streaming With Passive Frame Dropping. *IEEE Internet of Things Journal* (2023). https://doi.org/10.1109/JIOT.2023.3340642

[38] Yuhong Xie, Yuan Zhang, and Tao Lin. 2023. Deep Curriculum Reinforcement Learning for Adaptive 360° Video Streaming With Two-Stage Training. *IEEE Transactions on Broadcasting* (2023). https://doi.org/10.1109/TBC.2023.3334137

[39] Dongxiao Yu, Ruopeng Chen, Xin Li, Mengbai Xiao, Guanghui Zhang, and Yao Liu. 2023. A GPU-Enabled Real-Time Framework for Compressing and Rendering Volumetric Videos. *IEEE Trans. Comput.* (2023). https://doi.org/10.1109/TC.2023.3343104

[40] Anlan Zhang, Chendong Wang, Bo Han, and Feng Qian. 2022. YuZu: Neural-Enhanced Volumetric Video Streaming. In *Proceedings of the 19th USENIX Symposium on Networked Systems Design and Implementation (NSDI 2022)*. USENIX Association, Renton, WA, 137–154. https://www.usenix.org/conference/nsdi22/presentation/zhang-anlan

[41] Zihan Zheng, Houqiang Zhong, Qiang Hu, Xiaoyun Zhang, Li Song, Ya Zhang, and Yanfeng Wang. 2024. HPC: Hierarchical Progressive Coding Framework for Volumetric Video. In *Proceedings of the 32nd ACM International Conference on Multimedia* (Melbourne VIC, Australia) *(MM '24)*. ACM, New York, NY, USA, 7937–7946. https://doi.org/10.1145/3664647.3681107

[42] Gangqiang Zhou, Zhenxiao Luo, Miao Hu, and Di Wu. 2022. Presr: Neural-Enhanced Adaptive Streaming of VBR-Encoded Videos with Selective Prefetching. *IEEE Transactions on Broadcasting* 69, 1 (2022), 49–61. https://doi.org/10.1109/TBC.2022.3227419

[43] Yuanwei Zhu, Yakun Huang, Xiuquan Qiao, Zhijie Tan, Boyuan Bai, Huadong Ma, and Schahram Dustdar. 2022. A Semantic-Aware Transmission with Adaptive Control Scheme for Volumetric Video Service. *IEEE Transactions on Multimedia* 25 (2022), 7160–7172. https://doi.org/10.1109/TMM.2022.3217928

[44] Tongyu Zong, Yixiang Mao, Chen Li, Yong Liu, and Yao Wang. 2023. *Progressive Frame Patching for FoV-based Point Cloud Video Streaming*. arXiv:2303.08336 https://doi.org/10.48550/arXiv.1403.1349