Fumos: Neural Compression and Progressive Refinement for Continuous Point Cloud Video Streaming

Zhicheng Liang[†], Junhua Liu[†], Mallesham Dasari, Fangxin Wang[⋆]

Abstract— Point cloud video (PCV) offers watching experiences in photorealistic 3D scenes with six-degree-of-freedom (6-DoF), enabling a variety of VR and AR applications. The user's Field of View (FoV) is more fickle with 6-DoF movement than 3-DoF movement in 360-degree video. PCV streaming is extremely bandwidth-intensive. However, current streaming systems require hundreds of Mbps bandwidth, exceeding the bandwidth capabilities of commodity devices. To save bandwidth, FoV-adaptive streaming predicts a user's FoV and only downloads point cloud data falling in the predicted FoV. But it is difficult to accurately predict the user's FoV even 2-3 seconds before playback due to 6-DoF. Misprediction of FoV or network bandwidth dips results in frequent stalls. To avoid rebuffering, existing systems would cause incomplete FoV and degraded experience, deteriorating the user's quality of experience (QoE).
In this paper, we describe `Fumos`, a novel system that preserves interactive experience by avoiding playback stalls while maintaining high perceptual quality and high compression rate. We find a research gap in inter-frame redundant utilization and progressive mechaism. `Fumos` has three crucial designs, including (1) Neural compression framework with inter-frame coding, namely `N-PCC`, which achieves both bandwidth efficiency and high fidelity. (2) Progressive refinement streaming framework that enables continuous playback by incrementally upgrading a fetched portion to a higher quality (3) System-level adaptation that employs Lyapunov optimization to jointly optimize the long-term user QoE. Experimental results demonstrate that `Fumos` significantly outperforms Draco, achieving an average decoding rate acceleration of over 260×. Moreover, the proposed compression framework `N-PCC` attains remarkable BD-Rate gains, averaging 91.7% and 51.7% against the state-of-the-art point cloud compression methods G-PCC and V-PCC, respectively.

Index Terms—Streaming media, Point cloud compression, Deep learning, Virtual Reality (VR), User Experience

1 Introduction

The tremendous success of Internet video has led to a growing interest in point cloud video (PCV), a sequence of point cloud frames, where each frame is a set of unordered points sparsely distributed in the 3D space [1]. It provides six-degree-of-freedom (6-DoF) watching experiences in photorealistic 3D scenes. With a head-mounted display device, people can fully immerse themselves into a 3D video scene and freely change their positions as well as rotate their heads to watch the video content from any angle [2].

With such experience of full immersion and interactivity, PCV is envisioned as a killer application of the next generation of videos in the 5G/6G era [3] in various domains including entertainment, education, and e-commerce, and will empower various services such as VR, AR, MR and Metaverse [4]. Recent surveys of marketers indicate that the global VV market for industrial applications is expected to reach 22.5 billion USD by 2024 [5].

However, due to the higher dimensionality and the sparsity in nature, PCV is generally more difficult to process compared to 2D video. Its huge data volume significantly burdens its storage and transmission, hindering its future development and application. Taking a common 30 frames per second video as an example, when the number of points per frame is near 760,000, the bandwidth requirement for a VV is up to 2.9 Gbps [3], greatly exceeding the common bandwidth capabilities of commodity devices. For instance, statistics show that the standard broadband service in the U.S. is 25 Mbps [6], and Internet broadband speeds in some countries are even less than 1 Mbps [7]. Thus, optimizing the bandwidth needed for PCV delivery is imperative.

Recent research has devised several strategies to alleviate bandwidth consumption [8–11]. Among these approaches, two primary categories of solutions have emerged: compression and field-of-view (FoV)-adaptive streaming. However, during our year-scale operation of PCV streaming systems, we found that they fall far short of practical usage and continue to impose significant requirements. This is due to the dynamic and heterogeneous interactions of the users with these videos and the current inefficient compression scheme [12, 13].

FoV refers to the extent of the observable world in PCV. The FoV-adaptive scheme has been effectively utilized to curtail bandwidth consumption in 360-degree video streaming [14, 15]. As an extension, contemporary PCV systems employ analogous methodologies, which necessitates the spatial segmentation of each video chunk into 3D tiles. Each tile can then be independently downloaded and decoded at varying quality levels. FoV-adaptive schemes encompass the prediction of a user's FoV, which refers to the portion of the video visible to the user, and subsequently, tailor the transmission based on this prediction. Tiles within the predicted FoV are transmitted at a superior quality [10]. The tiles outside the FoV are transmitted at a diminished quality, or in some cases, not transmitted at all. They can significantly reduce bandwidth consumption if they correctly predict the FoV. However, it is important to note that *they will stall playback if any tile within the actual FoV is unavailable prior to the playback deadline.* Therefore, the primacy challenge of current point cloud streaming systems could be attributed to the following two factors:

Frequent Stalls. The primary focus of most FoV-adaptive strategies is to optimize the perceptual quality of the FoV and minimizing stalls. Unfortunately, under 6-DoF, the accuracy in predicting a user's FoV degrades significantly with prediction window (i.e., how much in advance the prediction is made). The prediction accuracy can be as low as 0.06 for a window of 2 seconds (§3.3). In a small window, the prediction model could accurately predict the FoV but makes the scheme vulnerable to network bandwidth dips, potentially resulting in *network-induced stalls* in video playback. For a long window, the user's FoV could be everywhere. Such large-scale FoV scope may result in *motion-induced stalls* since not all tiles relevant to the user's FoV could be fetched before playback. *Existing methods stall playback until all tiles relevant*

- † *contributed equally.* ⋆ *corresponding author.*
- *Zhicheng Liang, Junhua Liu are with the Future Network of Intelligence Institute and the School of Science and Engineering, The Chinese University of Hong Kong, Shenzhen. E-mail: zhichengliang1@link.cuhk.edu.cn, junhualiu@cuhk.edu.cn.*
- *M. Dasari is with Northeastern University and was with Carnegie Mellon University. E-mail: m.dasari@northeastern.edu.*
- *Fangxin Wang is with the School of Science and Engineering and the Future Network of Intelligence Institute, The Chinese University of Hong Kong, Shenzhen, and also with the Guangdong Provincial Key Laboratory of Future Networks of Intelligence. E-mail: wangfangxin@cuhk.edu.cn*

Manuscript received 4 Oct. 2023; accepted 26 Jan. 2024. Date of Publication xx xxx. 201x; date of current version 31 Jan. 2024. For information on obtaining reprints of this article, please send e-mail to: reprints@ieee.org. Digital Object Identifier: xx.xxxx/TVCG.201x.xxxxxxx

to a user arrive. In fact, state-of-the-art (SOTA) systems [8, 9] report significant rebuffering.

Compression Inefficiency. Developing efficient compression techniques for the underlying point cloud frames is essential to reducing the video size and deploying the system at scale. Pioneer works leverage the 3D data structure, e.g., the octree [13, 16, 17] or kd-tree [3, 18], to directly compress the point cloud, or project the points onto 2D planes followed by a traditional video codec, like H.264 or HEVC [19]. For 2D-based methods, the transformation between the 3D space and the 2D space substantially complicates the codec implementation, leading to a lengthy encoding/decoding latency. For 3D-based methods, they generally build a compact tree to geometrically represent the points of a frame. Although they could encode and decode frames in *real-time*, they only achieve a low compression ratio. Point clouds are sparsely sampled within the 3D space, while 2D images are sampled on a dense grid. Therefore, they are unable to identify and eliminate the inter-frame redundancy. Existing time-frequency transforms are not directly applicable to point clouds, which are major parts of the compression.

Fumos. Existing approaches stall playback until all tiles in the actual FoV to a user arrive. This is especially undesirable in 3D settings, where a user may move during the stall event, potentially changing the FoV that is being fetched, resulting in further cascaded stalls. The existing compression techniques still have drawbacks when being deployed in practice. In this light, we propose `Fumos`, a practical PCV streaming system designed to tackle the prevalent issues of excessive bandwidth consumption and recurrent stalls in current systems. `Fumos` involves the following innovations:

Neural Point Cloud Compression with Inter-frame Coding. The major challenge of improving compression ratio is to eliminate inter-frame redundancy [9]. For conventional 2D videos [20], motion estimation, and motion compensation (ME/MC) are shown to be effective in exploiting temporal redundancy. However, the points in the neighboring frames are not spatially aligned and their numbers are not even numerically aligned, making it not straightforward to migrate the techniques in conventional 2D codecs to the 3D space. PCV needs to perform block matching to achieve ME/MC across various frames, which leads to memory and computational inefficiency. Moreover, point clouds are usually not represented as voxel grids but octrees [16]. Such octrees vary from frame to frame and it is difficult to find the correspondences between octree nodes for motion compensation. Recent studies [21–24] have showcased the remarkable compression rate achievable with static point cloud compression. The inherent properties of neural networks allow "free interpolation" when retrieving coordinates that do not coincide with existing point clouds. Therefore, we propose `N-PCC`, a novel neural compression framework that extends the current neural static point cloud compression to neural point cloud video compression and integrates ME/MC to further reduce the temporal redundancies.

Progressive Refining Streaming with Continuous Playback. The network-induced stalls and motion-induced stalls are also pervasive in current systems [25, 26]. Due to the varying network conditions, they are hard to weigh or eliminate. To ensure continuous playback without stalls, we could naively skip FoV tiles that do not arrive by the playback deadline. However, this would result in incomplete FoVs with blank areas and a sharp drop-off in user experience [8]. We observed that although `N-PCC` could greatly compress the video, it inevitably leads to extra decoding stalls and exclusively end-to-end transmission. Specifically, `N-PCC` does not support FoV-adaptive streaming and its decoding time exceeds the size of video chunk, which are prevalent issues among other neural compression methods [21, 27, 28]. Thus, we propose FoV-adaptive octree-based compression, which allows selective transmission based on the predicted FoV and fast decoding. Instead of sequentially downloading PCV frames, we propose a progressive refinement framework to gradually refine the spatial resolution of each tile as its playback time approaches. At long prediction window, as the prediction accuracy is very poor, we first prefetch the video frames encoded by `N-PCC` and decode them. When the video is about to play (short prediction window), the client requests the octree-encoded frames within the predicted FoV to refine the perceptual quality into higher densities. As they could run simultaneously, the client could download the high-quality video at low bandwidth consumption. We further adapt to varying network and compute conditions through Lyapunov optimization [29] to optimize the long-term user QoE.

In summary, we propose the first inter-frame PCV compression method and the first neural compression scheme for PCV streaming. We, for the first time, enable PCV streaming with continuous playback. We implement the above components and integrate them into `Fumos`, a holistic PCV streaming system. Our extensive evaluations indicate that `Fumos` can achieve line-rate, high-quality, bandwidth-efficient, and adaptive PCV simultaneously. We highlight key findings from our evaluations as follows:

- `Fumos` achieves an average decoding rate acceleration of over 260× than Draco, the SOTA point cloud compression library.
- `Fumos` effectively optimizes the overall *QoE* amidst wildly fluctuating available bandwidth via adaptive bit rate optimization, achieving a normalized QoE that is more than 87.5% higher than the baseline methods.
- `Fumos` achieves a throughput of 1.49Mbps on the 8iVFB dataset, marking a 92.8% and 51.3% reduction compared to Draco and G-PCC, with a minor 20.9% and 9.6% decline in $d1PSNR$.
- `N-PCC` achieves an average 91.7% and 51.7% BD-Rate gains than SOTA point cloud compression methods G-PCC and V-PCC.

2 Related Work

2.1 Inter-frame Video Compression

PCV presents significant redundancy in streaming applications, where frame differences are typically minimal [30]. ME/MC allows the storage of only the motion vectors and alterations between frames [1], but current PCV compression methods lack explicit ME/MC networks to guide the inter-prediction. Many studies [31–33] utilize entropy encoding that relies on the previous frame for compression. However, they still ignore video dynamics and FoV dynamics. 3D motion compensation methods [34, 35] exploit the inter-frame redundancy. But they are based on the intra-frame blocks rather than inter-frame chunks, which can hardly detect the motions across the block boundaries. Although V-PCC [36] could project PCV into 2D geometry and texture video and use mature video codecs [19] for inter-frame compression, it only works on small-scale PCVs and takes enormous time to transform 3D into 2D.

2.2 Neural Point Cloud Compression

Neural point cloud compression methods have emerged, which are roughly divided into point-based [21–23], voxel-based [37–39] and octree-based [24, 40, 41]. Point-based methods utilize Farthest Point Sampling (FPS) and KNN to obtain local clusters and point-wise modules like PointNet++ [42]. Voxel-based methods utilize 3D SparseCNN based network [43] to compress the voxelized point cloud as an image. Octree-based methods are octree entropy models that rely on spatial correlation exploration via ancestor or sibling nodes for compression. However, they are not specifically designed for PCV streaming, whose utility is confined to static point clouds. As such, `N-PCC` extends the current neural static point cloud compression to neural PCV compression and integrates ME/MC to reduce temporal redundancies of point cloud sequences. As the first step, we focus on geometry compression. Our proposed framework could be easily extended to color compression using traditional methods [34, 44].

2.3 Point Cloud Video Streaming

Current PCV streaming systems mainly use similar methods to VR video streaming systems [45, 46] that divide the videos into smaller tiles and only transmit the tiles inside the user's FoV or adaptively adjust the cell bitrate to optimize defined objective function [47, 48]. For example, ViVo [8] proposes three visibility-aware optimizations for video tiles to save bandwidth consumption. Li et al. [49] consider the high computation complexity of point cloud video encoding during transmission optimization. Furion [50] uses cloud-assisted VR streaming with separated foreground and background. Yuzu [9] reduces the density of point clouds and recovers them with super-resolution technology on client

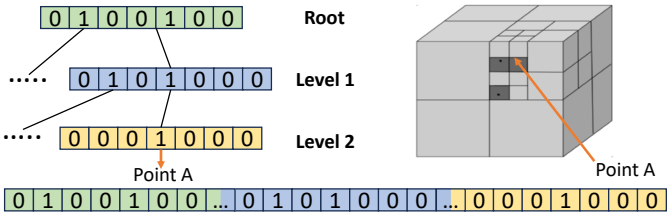

Figure 1: Octree-based Coding: How point cloud is indexed by octree.

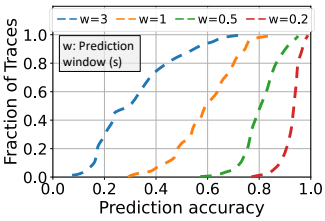

Figure 2: Prediction accuracy decreases sharply with larger windows

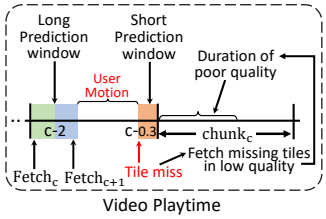

Figure 3: Inaccurate FoV prediction results in poor video quality.

Figure 4: Decoding time of frames. Videos are from 8i Dataset [56].

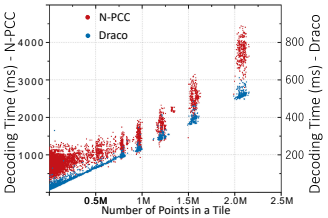

Figure 5: Decoding time under various number of points in tile

devices. Overall, these systems compress and transmit each PtCl frame independently, without consideration of the inter-frame redundancy.

3 Background, Measurement and Motivation

In this section, we present the preliminary of our design on point cloud compression, as detailed in § 3.1 and § 3.2. Moreover, our measurements and motivations behind the design of our progressive framework are expounded upon in § 3.3 and § 3.4.

3.1 Sparse Convolution

3D sparse convolution, akin to standard 3D dense convolution, efficiently processes sparse tensors, utilizing valid MP-POVs to exploit point clouds' sparse characteristics. Various implementations exist, such as Sparse 3D convolution [51], Sparse Blocks Network [52], 4D Spatiotemporal ConvNets (Minkowski CNN) [43] and Torchsparse [53]. In this work, we represent 3D sparse convolution using the notation (K^3, C, S^3)-SpConv, where $K = k \times k \times k$ denotes the kernel size, C denotes the channel size, and $S = s \times s \times s$ represents the stride. Similarly, 3D transpose convolution with analogous hyperparameters is denoted as (K^3, C, S^3)-TSpConv.

3.2 Octree-based Point Cloud Compression

Octree [16] is an efficient and compact data structure utilized for indexing points within 3D space. As shown in Fig. 1, the construction of an octree entails a recursive subdivision of the 3D space into $2 \times 2 \times 2$ subspaces until a pre-defined level-of-detail (LoD) is reached or there is no point contained. The root node symbolizes the whole space and the leaf nodes may contain any number of points. Notably, the intermediate nodes have at most eight children to enable efficient storage within a byte, where 1 indicates the corresponding subspace containing points and 0 means an empty subspace. There is no necessity to persistently store the coordinates of the points due to the approximation of the points contained by the centers of the leaf nodes. This tree structure is subsequently materialized into a byte stream following the breadth-first search strategy.

3.3 Networking Challenges of Point Cloud Video

We compare PCV streaming with conventional video streaming in Table. 1. The PCV supports a 6-DoF experience, which differs from other video types in terms of data volume, latency tolerance, and processing time. For instance, the standard broadband service in the U.S. is 25 Mbps [6], but the PCV streaming requires extreme bandwidth consumption (∼1 Gbps). Even with several strategies [9, 10, 54], the medium-quality PCV is still high (∼100 Mbps [8]).

Table 1: Comparisons among four types of video streamings

Video Types	2D Video	360° Video	VR Video	Point Cloud Video
User Freedom	N/A	2-DoF	3-DoF	6-DoF
Data Volume	∼1 Mbps	∼100 Mbps	∼100 Mbps	∼1 Gbps
Delay Sensitivity	medium	high	high	very high
Processing Time	short	medium	high	very high

These strategies, however, also engender issues. For example, FoV-adaptive streaming should deal with motion and network stalls. Prediction inaccuracy can trigger motion-induced stalls, which increase sharply with prediction window. As shown in Fig. 2, median accuracy is 93.1% for a window of 0.2 seconds and 29.6% for 3 seconds. Some systems [3, 8] fetch tiles for each chunk at different qualities based on predicted FoV. At long prediction window, inaccurate prediction enhances the quality of the wrong set of tiles and leads to poor QoE. As shown in Fig. 3, some systems [10, 55] fetch tiles within the predicted FoV. Due to user motion, they must fetch the missing tiles when the video is about to play, leading to much lower quality that persists for several frames or network-stalls. While they can prefetch the tiles at short prediction window, the limited download time significantly degrades the video quality.

3.4 Observation and Motivation of Neural Compression

For neural compression methods, decoding speed is the key bottleneck for PCV streaming rather than bandwidth consumption. As such, none of existing PCV systems adopt neural compression, thus burying its significant compression rate. However, `N-PCC` integrates the ME/MC, which separates the frames into keyframes and predicted frames. As shown in Fig. 4, keyframe decoding is the most time-consuming. The predicted frames, which are compressed based on differences relative to adjacent frames, could be decoded quickly at low PCV quality. Therefore, for each chunk encoded by `N-PCC`, we can finish the decoding before playback by decoding a few seconds in advance (long prediction window). Current octree-based methods can decode immediately, but have a low compression rate and ignore the advantages of FoV. Observations in § 3.3 and the above motivate us to propose the progressive refinement framework. As neural compression is not FoV-adaptive, the client prefetches low-quality PCV encoded by `N-PCC` at long prediction window. At short window, the client requests the remaining octree-encoded PCV within the predicted FoV to refine the video quality, which can be downloaded and decoded immediately due to accurate prediction, without degrading the quality as Fig. 3. As keyframe decoding of each chunk doesn't occur at the same time, the decoding of each chunk could run simultaneously, the client could use `N-PCC` to download most of the video data with a small amount of bandwidth and achieve real-time decoding. As shown in Fig. 5, the decoding time is proportional to the number of point clouds, thus `Fumos` can adjust adaptively according to varying network/compute conditions.

4 System Overview of Fumos

Based on the core idea proposed in § 3.4, we propose `Fumos`. Fig. 6 shows the system architecture of `Fumos`. A PCV is initially segmented sequentially into video chunks, each containing multiple frames. Point cloud (PtCl) within a chunk share the same hyperparameters for processing. Each frame is downsampled via an octree, with a maximum depth of LoD. Subsequently, a dynamic distribution mechanism apportions a fraction γ of the PCV to the FoV-adaptive codec and the remainder to `N-PCC`. `N-PCC` excels in compression and is employed to condense full-scene PCV into compact data for transmission. Initially, it decodes the prefetched, compactly compressed PtCl at the client end, forming a coarse backbone. Subsequently, the FoV-adaptive codec decodes

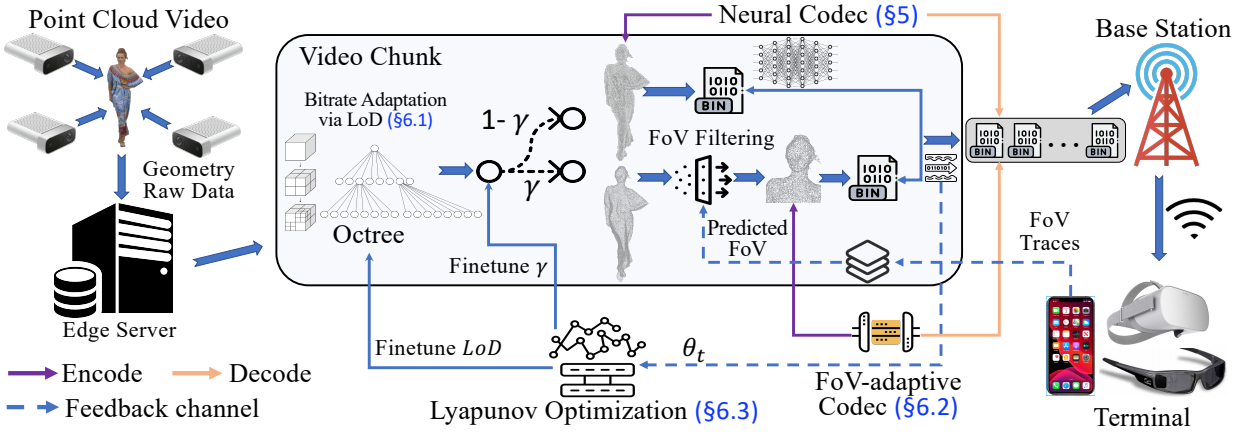

Figure 6: Overview of our proposed system - Fumos.

the short-term predicted region within the FoV to refine this region by merging it with the coarse backbone, thereby enhancing the visual quality within FoV. This progressive refining process is illustrated in Fig. 7. A Lyapunov optimization-based adapter, upon receiving the system status Θ_t at time t, fine-tunes *LoD* and γ to optimize the overall *QoE*. Broadly speaking, `Fumos` consists of the following two components:

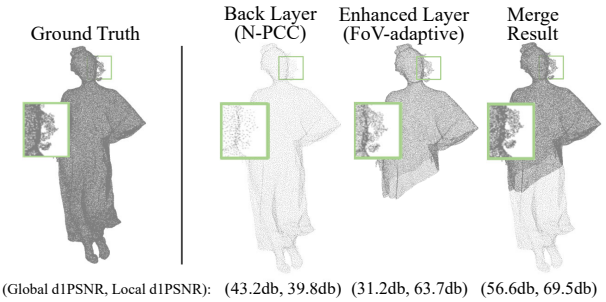

Figure 7: Progressive Refining Process

Neural Point Cloud Compression with Inter-frame Coding (§5). This component utilizes the implicit dependency to encode frames on the server side. An entropy-minimized motion compensation module is designed to generate a reference frame by transforming the previous frame to minimize conditional entropy. With the reference frame as context, the inter-frame entropy encoding module encodes the current frame, which enables reducing the video volume sharply.

Progressive Refining Streaming with Continuous Playback (§6). This component adapts to dynamic FoVs by transmitting encoded tiles with two encoding algorithms to ensure a complete and fluent display. On the server side, the encoder selection module adaptively selects the inter-frame entropy encoding module to encode the tiles.

5 N-PCC: Neural Point Cloud Compression

5.1 Overview of N-PCC

In this section, we elaborate on the proposed Neural PtCl Compression (`N-PCC`). The detailed architecture of `N-PCC` is illustrated in Fig. 8. Given two consecutive PtCl frames $P_t = \{C_t, F_t\}$ and $P_{t-1} = \{C_{t-1}, F_{t-1}\}$, where $C_t \in \mathbb{R}^3$ represent the coordinate matrices and $F_t \in \mathbb{R}^1$ signifies the associated feature matrices with all-one vectors indicating voxel occupancy. The network seeks to exploit the motion estimation between P_t and the previously reconstructed frame $\hat{P}_{t-1}$ to optimize bit rate consumption through inter-frame prediction. Initially, a multi-scale feature extraction encoder is employed to capture the density and topological information D through a series of downsampling blocks for current frame P_t and previously decoded frame $\hat{P}_{t-1}$, deriving $y_t = \{C_t^3, D_t\}$ and $\hat{y}_{t-1} = \{C_{t-1}^3, \hat{D}_{t-1}^3\}$, where C_t^3 represents the frame C_t post threefold downsampling. Futhermore, the ME module analyzes the motion vectors V_t from $\{y_t, \hat{y}_{t-1}\}$ and MC module adjusts V_t to $\hat{y}_{t-1}$ to deduce the predicted feature of the current frame $\tilde{y}_t$. The motion V_t and residual r_t between $\tilde{y}_t$ and y_t are compressed for subsequent reconstruction $\hat{P}_t$.

5.2 Feature Extraction

When presented with two consecutive PCV frames $\{P_{t-1}, P_t\}$, the objective of the feature extraction module is to capture both the inherent geometric properties and localized density information. Fig. 9 depicts how encoder extracts these features.

Multi-Scale Feature Aggregation. The previous frame P_{t-1} undergoes a multi-scale encoding process to acquire multi-scale topological information. The Multi-Scale Encoder (MS Encoder) consists of two encoders and some SpConv layers. Each encoder is composed of three Downsampling (DS) blocks. Initially, P_{t-1} is subjected to serial downsampling within the first encoder, yielding P_{t-1}^1, P_{t-1}^2, and P_{t-1}^3, denoting the progressively downsampled versions of P_{t-1} after i downsampling iterations. Subsequently, the second encoder integrates these four multi-scale features and a series of SpConv layers futher extracts the fused feature to generate $\hat{D}_{t-1}^3$, thereby deriving the predicted feature of previous frame $\hat{y}_{t-1} = \{C_{t-1}^3, \hat{D}_{t-1}^3\}$. Meanwhile, a single encoder generates the predicted feature of the current frame $y_t = \{C_t^3, D_t\}$.

Downsampling And Density Learning. Inspired by Wang et al. [38], we adopt a $(K^3, C, 2^3)$-SpConv as a downsampler to reduce the spatial redundancies of PtCls and learn hierarchical features from PtCls, capturing the complex structures and patterns in 3D space. Afterward, the remaining points aggregate the topological and density information from the vanished points, which can be leveraged to aid the PtCl reconstruction. To capture these features, we then introduce a Density-Aware Unit. Let P^i and P^{i+1} represent the Pre- and Post-downsampled versions of the PtCl P, respectively, following the $(K^3, C, 2^3)$-SpConv operation. These two sets, P^i and P^{i+1}, are subsequently processed by a density-aware unit designed to extract density features. To elucidate this process, we introduce the concept of a grouping point set denoted as $G(p_n)$, where each point $p_n \in P^{i+1}$ is a member of this set.

In the determination of the downsampled point set, each discarded point is exclusively assigned to its nearest downsampled point. Consequently, all points grouped with a particular downsampled point p_n collectively constitute a grouping points set $G(p_n)$. We denote the density feature of point p_n as D_n, which is calculated as follows:

$$D_n = \frac{1}{\|G(p_n)\|} \sum_{p_j \in G(p_n)} \|p_n - p_j\|_2 \tag{1}$$

While this strategy is effective in capturing density information, it may lead to significant computational overhead. To mitigate this, we employ a K-nearest neighbors (KNN) approach to select the k closest neighbors of p_n from within $G(p_n)$, denoted as p_n^{kn}. Subsequently, the density D_n is computed as follows:

$$D_n = \frac{1}{k} \sum_{j=1}^{k} \|p_n - p_j\|_2, \text{ where } p_j \in p_n^{kn} \tag{2}$$

This modification reduces the computational burden while still effectively capturing density features for point P_n. Subsequently, this computed density information is embedded into a latent space, producing density feature embedding. The density embedding F_{D_n} captures the density of $G(P_n)$ by mapping the D_n to a d-dimensional embedding via Multilayer Perceptrons (MLPs).

$$F_{D_n} = MLP(D_n), F_{D_n} \in \mathbb{R}^d \tag{3}$$

5.3 Motion Estimation (ME)

Given the latent representation of previous frame and current frame, i.e., $\hat{y}_{t-1} = \{C^3_{t-1}, \hat{D}^3_{t-1}\}$ and $y_t = \{C^3_t, D^3_t\}$, the ME module analyzes the temporal and spatial correlation and predict the motion vector V_t. The motion module first concatenate D^3_t and $\hat{D}^3_{t-1}$ to get the concatenation feature $\bar{D}^3_t$. Then, $\bar{D}^3_t$ is passed through a $(1^3, C^3, 1^3)$-SpConv block to generate motion vector V_t. Increasing the downsampling factor to exploit the inherent sparsity of PtCls does widen the perceptual field, but at the cost of significant information loss. In light of this, we adopt the multi-scale motion aggregation (MSMA) strategy similar to that described in § 5.2. The core idea behind MSMA is to capture motion information at multiple scales, ensuring that both coarse and fine motion patterns are effectively represented. This is achieved by processing the feature representations through a series of SpConv layers with varying kernel sizes, and subsequently fusing the outputs. This method enhances motion embedding, serving as a key tool to strengthen the connection between sparse PtCls across sequential frames. It helps overcome the challenges of their natural variability and the information loss caused by extensive downsampling. The MSMA module computes the fused multi-scale motion flow V_t as:

$$V_t = \mathcal{F}_v(\hat{y}_{t-1} \oplus y^3_t), \tag{4}$$

where $\mathcal{F}_v$ represents the MSMA operation. V_t is subsequently compressed through a deep entropy model (§ 5.5) for future reconstruction.

5.4 Motion Compensation (MC)

Motion Compensation utilizes motion vector V_t and the latent representation of the previous frame $\hat{y}_{t-1}$ to predict the current frame's latent representation $\tilde{y}_t$. In parallel to MSMA, a multi-scale reconstruction module amalgamates dimensionally-scaled, multi-solution motion vectors to form a fused motion $\hat{V}_t$. Subsequently, for a point p with estimated motion v, the new position p' is derived as:

$$p' = p + v \tag{5}$$

Given the non-uniform distribution of points, direct interpolation may yield inaccuracies. To address this, we employ a weighted interpolation algorithm for motion compensation, integrating a penalty coefficient λ to selectively diminish the influence of distant neighbors during interpolation. Hence, $\tilde{y}_t$ is formulated as:

$$\tilde{y}_t = \mathcal{F}_{wi}(\hat{V}_t, \hat{y}_{t-1}), \tag{6}$$

where $\mathcal{F}_{wi}$ denotes the weighted interpolation function, generating adaptive weights for a refined frame representation. The residual r_t is obtained by subtracting y_t from $\tilde{y}_t$, and is compressed using a deep entropy model, as elaborated in the following § 5.5.

5.5 Deep Entropy Model

Ballé et al. [27] employed deep learning methodologies to gauge the entropy of data targeted for compression. In the realm of information theory, entropy quantifies the inherent uncertainty or randomness in a dataset, acting as a constraint on the maximal average compression rate attainable by any lossless compression algorithm applied to specific data. Once the entropy is discerned, an Algorithmic Encoder (AE) and Algorithmic Decoder (AD) proceed to compress and decompress the data, adhering to the entropy values.

In our study, we implement the compression framework, where the motion vector V_t and the residual r_t undergo quantization are subsequently compressed utilizing a specialized deep entropy model, incorporating multiple MLPs. In contrast, the coordinate C^3_t is subjected to lossless compression via an octree-based encoder.

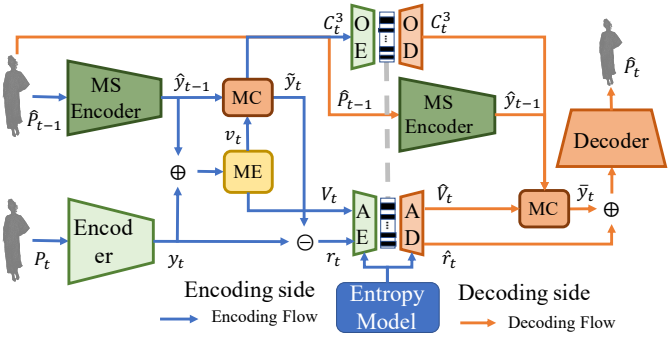

Figure 8: **N-PCC**: The OE and OD denote Octree-based Encoder and Decoder. The AE and AD denote Algorithmic Encoder and Decoder.

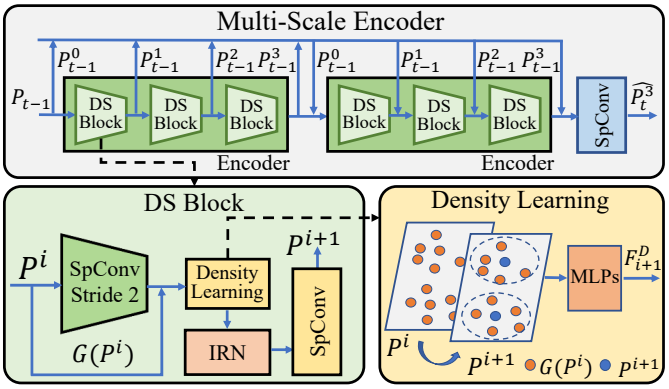

Figure 9: **Feature Extraction module:** The Multi-Scale Encoder comprises six Downsampling (DS) Blocks and a $(1^3, C, 1^3)$-SpConv operation for multi-scale density feature learning.

5.6 Point Cloud Reconstruction (Video Decoding)

Fig. 10 depicts the architecture of the decoder. It accepts a thrice-downsampled PtCl tensor and strategically upsamples it to reconstruct the original PtCl tensor, utilizing transpose convolution in the process. Following each upscaling via $(K^3, C, 2^3)$-TSpConv, an Inception-Residual Network (IRN) [57] block learns and aggregates local features, while a pruning layer reconstructs the geometry. This pruning layer works to eliminate incorrect voxels and to isolate the genuinely occupied ones through binary classification. The symmetrical structure of the decoder is crucial as it enables efficient reconstruction of the input, mirroring the encoder's architecture to ensure consistency and allow for accurate retrievals.

5.7 Design of Loss Function

During the end-to-end training, we utilize the standard rate-distortion loss function, represented as:

$$L = D + \lambda R \tag{7}$$

to achieve an optimal trade-off. Here, D acts as a penalty for distortion, and R serves to penalize the bitrate. This approach ensures a balanced compromise between maintaining fidelity and minimizing the amount of data required for representation, contributing to the overall effectiveness and efficiency of the model.

Rate Loss. Due to the discrete and conditional nature of entropy encoding, along with the quantization step which involves rounding off or truncating values, a non-differentiability issue arises in the compression process. Throughout the training phase, a differentiable surrogate is employed [27, 28]. This surrogate replaces the quantization step with additive uniform noise, leading to the approximation of the number of bits through the rate loss, denoted as R.

Distortion loss. Distortion measures the fidelity of the reconstructed data relative to the original data. It quantifies the loss of quality or the errors introduced during compression and subsequent decompression. We design two components to minimize the distortion of reconstruction:

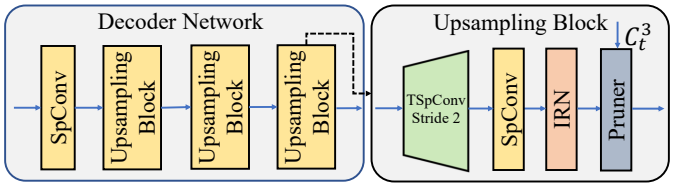

Figure 10: **Decoder Network:** The architecture of the decoder network reflects the encoder network, which includes a $(1^3, C, 1^3)$-SpConv and three distinct upsampling blocks. Each upsampling block upsamples the PtCl and learns to prune points not present in C_i^3.

Distortion loss: Optimization for Pruning. The pruner discussed in §. 5.6 utilizes the occupation probability p_v of each voxel v in the decoded PtCl to decide whether to eliminate it. Therefore, we apply binary cross entropy (BCE) loss to measure the distortion:

$$\mathcal{D}_{BCE} = \frac{1}{N}\sum_{v} -\left(\mathcal{O}_v \log p_v + (1-\mathcal{O}_v)\log(1-p_v)\right) \tag{8}$$

where $\mathcal{O}_v$ is the ground truth that v is either occupied or unoccupied.

Distortion loss: Optimization for Density Learning. We introduce a density term intended to advocate the recovery of local density. In stage s of the decoder, a point $\hat{p}$ is upsampled to a newly selected set of points, denoted as $\hat{\mathcal{C}}(\hat{p})$. Subsequently, its nearest counterpart, p, on the encoder side is identified. This point, p, emerges from a set $\mathcal{C}(p)$. The density values D_p and $D_{\hat{p}}$ can be inferred from $\mathcal{C}(p)$ and $\hat{\mathcal{C}}(\hat{p})$ respectively, abiding by the density calculation protocol delineated in §. 5.2. We can articulate the density loss, $\mathcal{D}_{\text{den}}$, as:

$$\mathcal{D}_{\text{den}} = \sum_{s=0}^{S-1}\sum_{\hat{p}\in\hat{P}_{s+1}} \frac{|D_p - D_{\hat{p}}| + \gamma|\overline{D_p} - \overline{D_{\hat{p}}}|}{|\hat{P}_{s+1}|} \tag{9}$$

In Equ. 9, the numerator's first term computes the density disparity between the two sets, while the second term evaluates the deviation between the mean densities of all points in the sets relative to the central points p or $\hat{p}$. Here, γ functions as the weight regulating the contribution of the mean density difference to the overall density loss. Finally, the overall distortion loss is as follows:

$$\mathcal{D} = \mathcal{D}_{BCE} + \beta\mathcal{D}_{\text{den}} \tag{10}$$

where β is the weight of the density term.

6 FUMOS: PROGRESSIVE REFINING STREAMING

6.1 Pruning: Bitrate Adaptation via Level of Detail

The perceptual quality of a PCV is usually indicated by the peak signal-to-noise ratio (PSNR) [58]. The enhancement in quality and volumetric detail is concomitant with an escalation in bandwidth consumption. Octree is a hierarchical structure where each level represents a different LoD. Therefore, we propose that the *LoD* serves as a crucial parameter, enabling the adaptive bitrate adaptation. By varying the maximum permissible depth of the octree, we can orchestrate the video quality. The higher level of the octree presents a more generalized and coarse appearance. In contrast, the lower levels provide a more detailed and precise appearance.

The current streaming systems [8, 10] directly alter the point cloud density for bitrate adaptation. Our proposed LoD-based adaptation could mitigate visual artifacts arising from density adjustments, enabling precise manipulation of the point cloud's appearance while maintaining its overall integrity. The system's overarching objective is to judiciously modulate the *LoD* in congruence with the prevailing network conditions. This adaptive adjustment seeks to harmonize the trade-off between the fidelity of the PCV and the associated bandwidth consumption, thereby optimizing the overall QoE. Such a balanced approach is pivotal in scenarios where network resources are constrained, and the efficient utilization of available bandwidth is paramount. We elaborate the optimization process in the § 6.3.

6.2 FoV-adaptive Octree-based Compression

At short prediction window, `Fumos` achieves very accurate FoV prediction (depicted in Fig. 2). We design a FoV-adaptive Octree-based Codec, namely `FoV-adaptive codec`, to compress the tiles within the predicted FoV through our octree-based codec. This region reduction facilitatefs a much faster encoding, decoding and downloading compared to standard codecs. Therefore, `Fumos` does not lead to cascaded stalls or quality degradation like other systems. The detailed analysis on efficiency improvement and visual quality enhancement are shown in § 7.2. The details are shown below:

FoV Prediction. To predict future FoV, we design a model that leverages historical FoV traces alongside the content of the current frame. Utilizing an LSTM-based network, we capture features from N historical FoV traces to predict a coarse FoV region, denoted as $\tilde{S}_t$. Following this, an encoder consists of a series of SpConv layers with varying kernels efficiently extract and fuse content features from the previous N FoV regions $\{S_{t-N-1}, S_{t-N}, \cdots, S_{t-1}\}$, forming F_{t-1}. Then the encoder extracts the feature F_t from an expanded area around the initially predicted region $\tilde{S}_t$. Finally, a content-aware trace estimator, constituted of a $(1^3, C, 1^3)$-SpConv and an IRN network, analyzes the inter-frame content feature to refine the coarse region $\tilde{S}_t$ to an accurate one $\hat{S}_t$.

FoV Filtering. Upon acquiring an accurate predicted FoV region, a FoV filter is deployed for downsampling, pruning points outside the FoV. Given a point $P(x,y,z)$ denoting the observer's position in a 3D coordinate space, a 3D unit vector $\vec{G} = (x_1, y_1, z_1)$ representing the gaze direction, a set of 3D points $S = \{P_1, P_2, ..., P_N\}$ denoting the PtCl, and a FoV angle θ, the set of points within the FoV is determined by computing the normalized vectors $\vec{U}_i = \frac{P_i - P}{||P_i - P||}$ from the observer's position to each point P_i in S, and subsequently comparing the cosine of the angle α_i between the gaze direction vector $\vec{G}$ and each normalized vector $\vec{U}_i$ to the cosine of half of the FoV angle $\frac{\theta}{2}$. The resultant set, $\{P_i \in S \,|\, \cos(\alpha_i) \geq \cos\left(\frac{\theta}{2}\right)\}$, encompasses all points within the observer's FoV.

6.3 Lyapunov Optimization for Video Streaming

In the realm of video streaming, optimizing the Quality of Experience (QoE) is crucial. The challenge lies in ensuring high video quality and low latency while adhering to computational constraints, especially in volumetric video streaming scenarios.

6.3.1 Problem Formulation of QoE Maximization

The QoE is principally determined by two components: the perceptual quality of the video and delay. We assess perceptual quality by determining the distortion error between the reconstructed PtCl and the original PtCl. In a Video on Demand (VoD) scenario, latency is primarily a result of data transmission and data decoding by the codec at the client's end. The latency incurred due to data transmission through the network is considered a constant value, which is not incorporated into the optimization process, hence, the latency under consideration is exclusively the decoding time. Different pairs of (LoD_t, γ_t) yield different compression ratios affecting reconstructed quality and latency. We evaluate the reconstructed quality and latency using function $f(LoD_t, \gamma_t)$ and $g(LoD_t, \gamma_t)$, respectively. The function $f(LoD_t, \gamma_t)$ employs the MPEG evaluation tool [59] to compute d_1PSNR (point-to-point Peak Signal-to-Noise Ratio) and d_2PSNR (point-to-plane geometry Peak Signal-to-Noise Ratio) between the original and reconstructed PtCl. Subsequently, a comprehensive *PSNR* value is calculated considering a weighted combination of different specific *PSNR* values. More explicitly, the formula is given by:

$$PSNR = d_1PSNR_{glb} + \alpha_1 d_2PSNR_{glb} + \alpha_2(d_1PSNR_{loc} + \alpha_1 d_2PSNR_{loc}) \tag{11}$$

Here, the subscript glb denotes the *PSNR* calculated over the entire frame, and loc denotes the *PSNR* calculated on the parts in the Field of View (FoV). The parameters α_1 and α_2 represent the significance of d_2PSNR and the perceptual visual quality in the FoV, respectively.

The function $g(LoD_t, \gamma_t)$ represents the latency arising from the codec's data decoding on the client's side. Following the data transmission and decoding scheme outlined in § 3.4, the latency can primarily be ascribed to the decoding time necessitated by the `FoV-adaptive codec`.

Finally, we combine $f(\cdot)$ and $g(\cdot)$ to formulate the QoE model:

$$QoE(t) = f(LoD_t, \gamma_t) - \lambda g(LoD_t, \gamma_t) \tag{12}$$

where λ is the important factor of latency.

6.3.2 Constraints on Resource

Given the dynamic nature of network conditions and the constraints of computational resources, both the time-varying available bandwidth and the available computational resource serve as crucial parameters that play a pivotal role in our optimization problem.

Bandwidth Constraints. We denote B_t as the available bandwidth at time t, and formulate current bandwidth consumption with function $h(LoD_t, \gamma_t)$. Then we have

$$h(LoD_t, \gamma_t) \leq B_t \tag{13}$$

Computational Resource Constraints. `N-PCC`, with higher compression rate, exhibits increased complexity compared to `FoV-adaptive codec`. To balance decoding complexity and compression rate within client-side computational resource limits, we monitor system status to derive resource usage metrics, U_{CPU}, U_{GPU}, and U_{Mem}, assembled into a vector U. A vector W assigns weight to each resource, with Resource Consumption Score (*RCS*) computed as $RCS = U \cdot W^T$. The Computational Resource Requirement Ratio (*CR*) between `N-PCC` and `FoV-adaptive codec` is obtained by $CR = \frac{RCS_N}{RCS_F}$, representing the *RCS* obtained by `N-PCC` and `FoV-adaptive codec` respectively. Therefore, the computational consumption, $C(LoD_t, \gamma_t)$, can be formulated as follows:

$$C(LoD_t, \gamma_t) \triangleq v(LoD_t) \times (CR_t \times (1-\gamma_t) + \gamma_t) \tag{14}$$

Here, $v(LoD_t)$ signifies a proportional relationship concerning the data volume of a PtCl encoded in an octree structure of depth LoD_t, directly influencing the resource usage metrics U. We treat the computational resource on the client side as a constant value, denoted as $\mathcal{C}$. To simplify the problem, we only consider memory usage. Consequently, the constraint can be expressed as:

$$C(LoD_t, \gamma_t) \leq \mathcal{C} \tag{15}$$

Objective. The system aims to optimize the overall QoE. Therefore, by integrating the aforementioned constraints, the problem can be formulated as:

$$\text{Obj: } \max_{LoD_t, \gamma_t} \lim_{T\to\infty} \frac{1}{T} \sum_{t=0}^{T-1} [f(LoD_t, \gamma_t) - \lambda g(LoD_t, \gamma_t)] \tag{16}$$

$$\text{s.t.} \quad 0 \leq \gamma \leq 1 \tag{17}$$

$$LoD_t \in \mathbb{Z}^+ \tag{18}$$

$$h(LoD_t, \gamma_t) \leq B_t \tag{19}$$

$$C(LoD_t, \gamma_t) \leq \mathcal{C} \tag{20}$$

6.3.3 Lyapunov Optimization

This section illustrates the intricacies involved in the optimization of QoE. Initially, Lyapunov optimization is employed to transmute the extensive long-term optimization problem into a singular time slot problem, which is subsequently resolved by evolutionary algorithms.

Problem Formulation. We commence by introducing a concatenated vector, represented as the system state Θ_t:

$$\Theta_t \triangleq [f_t(\cdot), g_t(\cdot), B_t] \tag{21}$$

Here, $f_t(\cdot)$ symbolizes $f(LoD_t, \gamma_t)$ and $g_t(\cdot)$ represents $g(LoD_t, \gamma_t)$. Subsequently, the Lyapunov function is defined as:

$$L(\Theta_t) \triangleq \frac{f_t(\cdot)^2}{2} + \frac{g_t(\cdot)^2}{2} + \frac{B_t^2}{2} \tag{22}$$

Algorithm 1 Differential Evolution for Mixed-Integer Nonlinear

Input: Population size N, Scaling factor F, Crossover probability CR, Maximum generations $G_{\max}$
Output: Optimal solution X_{opt}
Initialize population P with N individuals with random integer LoD and random γ within bounds
Evaluate the objective function J for each individual in P
for $g = 1$ to $G_{\max}$ **do**
  for each target vector X in P **do**
    Elect three distinct vectors X_{r1}, X_{r2}, X_{r3} from P randomly
    Mutate: $V = X_{r1} + F \cdot (X_{r2} - X_{r3})$
    Ensure LoD component of V is an integer by rounding
    Crossover: For each component i in V
    if $rand(0,1) < CR$ or i is a randomly chosen index **then**
      $U_i = V_i$
    else
      $U_i = X_i$
    end if
    Ensure LoD component of U is an integer by rounding
    Evaluate $J(U)$ considering constraints
    Selection: If $J(U) < J(X)$, replace X with U in P
  end for
  Check for convergence and break if converged
end for
Return the individual in P with the lowest objective function value as X_{opt} =0

We then introduce the concept of Lyapunov drift $\Delta(\Theta_t)$, serving as a measure for the anticipated elevation in the Lyapunov function across a singular time slot:

$$\Delta(\Theta_t) = \mathbb{E}[L(\Theta_{t+1}) - L(\Theta_t) | \Theta_t] \tag{23}$$

In the pursuit of optimizing the system's performance whilst maintaining stability, the decision pertaining to LoD and γ is made at each time t to minimize the following expression:

$$J(LoD_t, \gamma_t) \triangleq \Delta(\Theta_t) + V \times \mathbb{E}[QoE(t) | \Theta_t] \tag{24}$$

Herein, V is a non-negative control parameter arbitrating a compromise between system stability and the aspired performance metric. Finally, integrating the constraints described in § 6.3.2, the optimization problem can be rewritten as:

$$\text{Minimize } J(LoD_t, \gamma_t) \tag{25}$$

$$\text{s.t.} \quad 0 \leq \gamma \leq 1 \tag{26}$$

$$LoD_t \in \mathbb{Z}^+ \tag{27}$$

$$h(LoD_t, \gamma_t) \leq B_t \tag{28}$$

$$C(LoD_t, \gamma_t) \leq \mathcal{C} \tag{29}$$

Evolutionary Algorithm Solver. The optimization problem is identified as a mixed-integer nonlinear problem due to integer constraints and potential non-linearity of objective and constraint functions. This non-convex problem, characterized by multiple local optima, often arises from discrete variables like integer constraints. To tackle this, an evolutionary algorithm solver is utilized for its ability to find superior solutions by iteratively refining a set of potential solutions based on the quality measure. The solution process is detailed in Algorithm 1. In our experiment, we set N to 15, F to a range of [0.5, 1], and CR to 0.7.

7 Evaluation

7.1 Experiment Settings

7.1.1 Dataset

Point cloud dataset For this experiment we use the 8i Voxelised Full Body (8iVFB) Dataset [56] provided by MPEG for testing. The user FoV trace is from [60], including the traces of users watching the

8iVFB dataset. It involves 26 participants viewed 150 looped frames of PCV in the Unity engine, recording viewport location and orientation at each frame. Fig. 11 illustrates the distribution of historical FoV traces, elucidating user interest and engagement across varied content segments and spotlighting areas capturing predominant user attention. The bandwidth trace is from NYU Mobile Bandwidth Trace [61], includes 4G bandwidth traces.

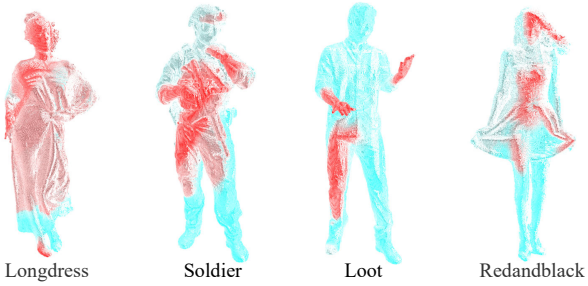

Figure 11: Distribution of historical FoV traces from multiple users. The more frequently the user views an area, the deeper the red color becomes; otherwise, the deeper the cyan color.

7.1.2 Training Strategy of N-PCC

The N-PCC is trained utilizing five distinct values of λ, ensuring the encompassing of a broad spectrum of bit rates. An Adam optimizer [62], characterized by $\beta = (0.9, 0.999)$, is employed in conjunction with a learning rate scheduler that exhibits a decay rate of 0.8 post every 20 epochs. In the overall distortion loss (Equ. 10), β is set to 0.5, and in the density loss (Equ. 9), γ is set to 1. The model is trained over 1000 epochs with an early stopping strategy. The batch size is set at 16 throughout the training process.

Training leverages the *basketball* sequences sourced from the MPEG's Owlii Dataset [63], while the 8iVFB sequences are utilized for testing purposes. All the experimental undertakings are executed on a server equiped with two 10-core Intel Xeon Silver 4210 CPUs, 128GB RAM, and an NVIDIA A100 GPU with 80GB of memory.

7.1.3 Parameter Setting

In Equ. 11, the parameters are set as follows: $\alpha_1 = 1$, $\alpha_2 = 3$, and λ is set to 3000 in Equ. 12. The *LoD* is constrained within a range of (5, 12), while γ is bounded within a range of (0.2, 0.6). The available bandwidth for optimization is simulated through random generation, determined by the dataset size, which is specified within a range of 0.75 to 1 times 5‰ of the size of 30 frames (assuming a frame rate of 30 FPS). The long prediction window is 3 seconds, the short window is 0.3 seconds.

7.1.4 Evaluation Metric

The codec performance is assessed by bit rate (bits per point, bpp) and distortion, measured via d_1PSNR and d_2PSNR,in accordance with the MPEG CTC standards. For a comprehensive evaluation of the streaming system, the decoding rate is derived as the inverse of decoding time, throughput is measured in Mbps, and *QoE* is derived from Equ. 12.

7.1.5 Baseline

Codec. To ascertain the performance superiority of N-PCC, we consider three rule-based codecs: G-PCC, Draco, V-PCC, along with a learning-based codec PCGCV2 as baseline methods.

System. Vivo, Groot, Vues are main PCV system used at present.

- **G-PCC** [64]: A standard point cloud compression method provided by MPEG, used by Groot [55]. In lossy compression via trisoup mode, it's denoted as G-PCC (T).
- **V-PCC** [36]: A patch-based method provided by MPEG that leverages existing video codecs for compressing the geometry and texture information of a point cloud video, used by Vues [9].
- **Draco** [12]: A library for compressing and decompressing 3D geometric meshes and PtCl provided by Google, used by ViVo [8]
- **PCGV2** [38]: A state-of-the-art, sparse convolution based autoencoder for PtCl compression and reconstruction.

Streaming system. To evaluate the system performance, balancing *QoE* against codec complexity, we devise three basic baseline methods utilizing the codecs G-PCC, Draco, and N-PCC for PtCl compression and decompression without employing bandwidth-aware optimization.

7.2 Performance Analysis

7.2.1 Overall Performance

Fig. 12, 13, 14, and 16 illustrate the trend of observations on the dataset *Longdress* with a rolling average of window size equal to 10. The x-axis represents time, with the unit being video trunk, and the size of each video trunk is 10 frames. The result for each trunk is an average of 10 frames within the trunk. It's noteworthy that we choose to showcase the visual results of *Longdress* over other datasets due to its considerably larger Field of View (FoV) (refer to Fig. 11), which results in higher bandwidth consumption and necessitates a superior level of optimization for data transmission. The results obtained from other datasets are reported in Table 2. These results demonstrate an enhanced performance speed, while simultaneously preserving nearly identical visual quality.

Observations for Bandwidth Aware Fine-tuning. In a resource-constrained scenario requiring high compression, we simulate available bandwidth B_t (see § 7.1.3) to assess our method. Fig. 12 shows bandwidth fluctuations and Fig. 13 displays normalized average values of *LoD*, γ, and *QoE*. Higher γ and *LoD* improve perceptual quality (Fig. 15) but increase bandwidth use. The trend of these parameters mirrors bandwidth fluctuations, indicating that Fumos adaptively adjusts them based on network conditions.

Observations for Effectiveness of Downsampling. Fig. 14 displays the proportion of points remaining post-Octree and FoV filter downsampling. Octree nearly halves the points, and FoV filter further excludes regions outside the FoV, minimizing transmission data size. The upward trend in FoV in indicative of a gradual shift in user focus towards the upper part of the body over time, increasing the point volume for transmission and decompression, which lowers the decoding rate (Fig. 16), and thus, degrades *QoE* (Fig. 13).

Observations for Perceptual Quality. Fig. 15 depicts the PSNR trend, showing a lower global (glb) PSNR compared to the local (loc) PSNR, as anticipated. The glb PSNR evaluates distortion over the entire frame, whereas loc PSNR targets the FoV region. Initially, N-PCC reconstructs a base layer, which FoV-adaptive codec later refines, yielding a higher loc PSNR than glb PSNR. This progressive refinement is demonstrated in Fig. 7.

Comparison with Baseline. Fig. 17 highlights the superior *QoE* of Fumos compared to baseline methods, with Table 2 detailing the performance across metrics. Fumos surpasses baseline methods by more than 87.5% in terms of *QoE*. Baseline methods' low *QoE* stems from their slow decoding, whereas Fumos, powered by N-PCC, efficient Octree downsampling, and accurate FoV prediction, boasts a much faster decoding rate. However, in low-bandwidth settings, aggressive downsampling leads to significant information loss and poorer reconstruction quality. In higher bandwidth scenarios, Fumos shows adaptability, improving quality by about 15% in PSNR while maintaining reasonable bandwidth consumption.

7.2.2 Performance of N-PCC

Effectiveness. The rate-distortion curves, derived from various methods tested on the *Longdress* dataset, are depicted in Fig. 18. These curves illustrate that our proposed method, N-PCC, consistently surpasses other methods, attaining 91.7%, 99%, 51.7%, 99.9%, and 29.2% BD-Rate (Bjontegaard Delta Rate) gains over G-PCC, G-PCC (T), V-PCC, Draco, and PCGV2, respectively. Table 3 further reveals a significant performance degradation from all perspectives without the N-PCC. Moreover, excluding the density loss function (denoted as "Den. Loss" in Table 3) from N-PCC training leads to performance degradation, evident from the d1psnr and d2psnr metrics, indicating its role in improving reconstruction quality.

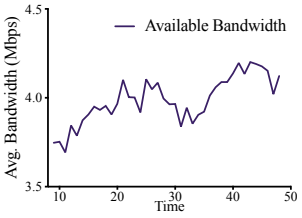

Figure 12: The Avg. Available bandwidth over time

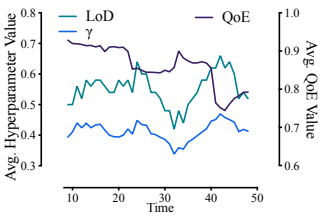

Figure 13: The Avg. normalized hyperparameters and QoE over time

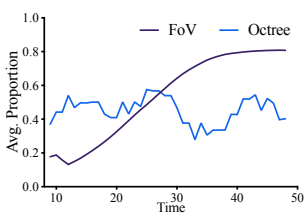

Figure 14: The Avg. proportion of point after FoV or Octree pruning

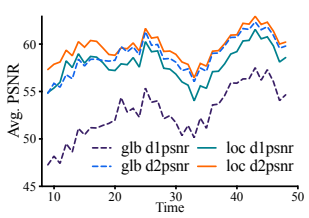

Figure 15: The Avg. value of four types of PSNR over time

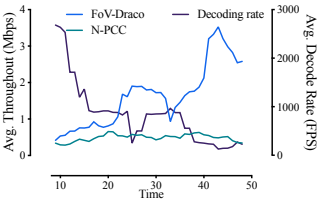

Figure 16: The Avg. throughout and decoding rate of various codecs

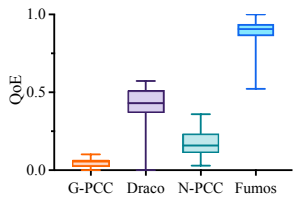

Figure 17: Performance comparison of various codecs using overall QoE

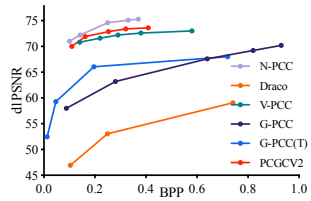

Figure 18: Performance comparison of codec using rate-distortion curves

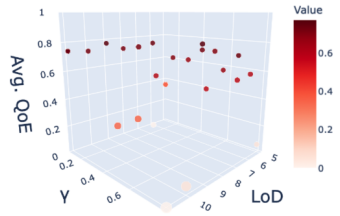

Figure 19: *QoE* under fixed γ and *LoD* without Lyapunov Optimization

Table 2: Performance Metrics of methods across datasets; values in mean (std.) format, bolded if best. `Fumos` denotes our method in a low-bandwidth scenario, while `Fumos`* signifies 10× larger bandwidth scenario (e.g., around 40 Mbps on *Longdress*).

Video	Method	throughout (Mbps) ↓	decode (FPS) ↑	d1psnr (loc)↑	d2psnr (loc)↑
Longdress	G-PCC	2.90 (0.1)	0.28 (0.0)	70.02 (0.1)	74.21 (0.1)
	Draco	19.79 (0.5)	8.94 (0.3)	**77.23 (0.1)**	**82.00 (0.1)**
	`N-PCC`	**1.29 (0.0)**	3.85 (0.4)	75.97 (0.2)	78.02 (0.2)
	`Fumos`	2.13 (0.9)	**1004.03 (109)**	62.14 (9.5)	68.25 (7.2)
	`Fumos`*	2.53 (0.9)	582.23 (112)	70.12 (8.6)	77.21 (8.2)
Loot	G-PCC	2.65 (0.0)	0.29 (0.0)	67.41(0.1)	71.29(0.1)
	Draco	18.77 (0.3)	9.31 (0.3)	**77.20 (0.1)**	**81.95 (0.2)**
	`N-PCC`	**1.21 (0.2)**	3.82 (0.4)	70.64 (7.6)	72.58 (8.8)
	`Fumos`	1.56 (0.6)	**2227.63 (239)**	60.50 (10.0)	71.12 (9.1)
	`Fumos`*	2.43 (2.5)	728 (281)	69.23 (12.2)	82.12 (9.9)
Red&Black	G-PCC	2.78 (0.1)	0.31 (0.0)	67.92 (0.1)	71.23 (0.2)
	Draco	18.09 (0.8)	10.00 (0.6)	**77.44 (0.1)**	**82.22 (0.1)**
	`N-PCC`	1.88 (0.1)	3.71 (1.0)	72.07 (6.3)	75.32 (7.8)
	`Fumos`	**0.52 (0.7)**	**3122.06 (391)**	61.02 (8.5)	71.65 (7.5)
	`Fumos`*	1.70 (0.9)	1449 (358)	68.02 (9.8)	77.65 (8.2)
Soldier	G-PCC	3.90 (0.0)	0.22 (0.0)	65.82 (0.2)	70.10 (0.1)
	Draco	26.20 (0.0)	6.44 (0.2)	**77.07 (0.1)**	**81.90 (0.1)**
	`N-PCC`	1.98 (0.7)	2.27 (0.5)	71.56 (6.6)	74.67 (8.1)
	`Fumos`	**1.75 (0.5)**	**2928.13 (216)**	61.59 (9.3)	71.97 (8.5)
	`Fumos`*	3.36 (1.7)	1297 (220)	71.89 (5.8)	79.97 (8.9)

Efficiency. Table 2 indicates that `N-PCC` can decode at a rate over ten times higher in FPS compared to G-PCC, albeit being two times slower than Draco. These results pertain to each codec decoding an entire frame. `N-PCC` only processes $1-\gamma$ of the frame in `Fumos`, implying a potentially faster decoding rate than indicated above.

7.2.3 Ablation Study

Effectiveness of Lyapunov Optimization. We evaluate the system performance without the Lyapunov optimization strategy by fixing *LoD* and γ. Fig. 19 presents the normalized overall *QoE* value across varying fixed hyperparameters, with the normalized max *QoE* value obtained by our method normalized to 1. The findings suggest that a static hyperparameter setting, unresponsive to network conditions, diminishes the overall *QoE*.

Effectiveness of Components in the System. We evaluate each component's impact by excluding it and assessing the performance thereafter. Table 3 reveals that utilizing the downsampler Octree or FoV filter leads to a more than 7.4% and 1.6% decline in overall visual quality (local PSNR), respectively. However, it significantly improves the decoding rate by more than 40 and 20 times, respectively, implying the critical importance of these components.

Table 3: Ablation on density learning, `N-PCC`, Octree and FoV filtering.

Ablation	thourghout (Mbps) ↓	decode (FPS) ↑	d1psnr (loc) ↑	d2psnr (loc)↑
`Fumos`	2.13 (2.9)	1004.03 (109)	67.27 (7.8)	69.20 (7.0)
W/O Den. Loss	2.32 (2.1)	1034.31 (127)	65.13 (7.4)	68.52 (6.8)
W/O `N-PCC`	10.43 (3.2)	24.12 (12.2)	61.09 (6.2)	67.26 (4.2)
W/O FoV	8.80 (2.3)	25.09 (8.3)	68.34 (7.2)	76.93 (8.9)
W/O Octree	6.43 (3.2)	54.62 (42.9)	72.47 (7.0)	77.06 (3.9)

7.2.4 Discussion on Continuous Playback

Existing systems stall playback until all tiles in the actual FoV arrive. Our extensive evaluations indicate that `N-PCC` could greatly reduce the bandwidth consumption and FoV-adaptive Codec could enable instantaneous downloading and decoding at low prediction window. The design of `Fumos` could greatly decimate the motion and network-stalls, thereby enabling PCV streaming with continuous playback.

8 Conclusion Remarks and Limitation

This paper proposes `Fumos`, a novel system that preserves interactive experience by avoiding playback stalls while maintaining high perceptual quality and high compression rate. Our system combines the benefits of neural, FoV-adaptive, and octree-based compression, opening up a promising direction for current point cloud video transmission and a new avenue to co-design the application and transport layers for better-quality point cloud video. Our progressive refining framework could enable low bitrate streaming and greatly alleviate the frequent stalls, but does not eliminate them entirely. The main limitation of `Fumos` is the requirement for consumer GPUs or other similar accelerators to run neural networks, which are increasingly accessible, powerful, and commonly found in most devices. Otherwise, the decoding speed would become the bottleneck.

Acknowledgments

The work was supported in part by the Basic Research Project No. HZQB-KCZYZ-2021067 of Hetao Shenzhen-HK S&T Cooperation Zone, the Shenzhen Science and Technology Program (Grant No. RCBS20221008093120047 and No. JCYJ20230807114204010), the Shenzhen Outstanding Talents Training Fund 202002, the Guangdong Research Projects No. 2017ZT07X152 and No. 2019CX01X104, the Young Elite Scientists Sponsorship Program of CAST (Grant No. 2022QNRC001), the Guangdong Provincial Key Laboratory of Future Networks of Intelligence (Grant No. 2022B1212010001), and the Shenzhen Key Laboratory of Big Data and Artificial Intelligence (Grant No. ZDSYS201707251409055).

REFERENCES

[1] Yiling Xu, Ke Zhang, Lanyi He, Zhiqian Jiang, and Wenjie Zhu. Introduction to point cloud compression. *ZTE Communications*, 16(3):8, 2018. 1, 2

[2] Yili Jin, Kaiyuan Hu, Junhua Liu, Fangxin Wang, and Xue Liu. From capture to display: A survey on volumetric video. *arXiv preprint arXiv:2309.05658*, 2023. 1

[3] Jie Li, Cong Zhang, Zhi Liu, Richang Hong, and Han Hu. Optimal volumetric video streaming with hybrid saliency based tiling. *IEEE Transactions on Multimedia*, 2022. 1, 2, 3

[4] Mohammad Hosseini and Christian Timmerer. Dynamic adaptive point cloud streaming. In *Proceedings of the 23rd Packet Video Workshop*, pages 25–30, 2018. 1

[5] Zhi Liu, Qiyue Li, Xianfu Chen, Celimuge Wu, Susumu Ishihara, Jie Li, and Yusheng Ji. Point cloud video streaming: Challenges and solutions. *IEEE Network*, 35(5):202–209, 2021. 1

[6] Kyle MacMillan, Tarun Mangla, James Saxon, and Nick Feamster. Measuring the performance and network utilization of popular video conferencing applications. In *Proceedings of the 21st ACM Internet Measurement Conference*, pages 229–244, 2021. 1, 3

[7] Vibhaalakshmi Sivaraman, Pantea Karimi, Vedantha Venkatapathy, Mehrdad Khani, Sadjad Fouladi, Mohammad Alizadeh, Frédo Durand, and Vivienne Sze. Gemino: Practical and robust neural compression for video conferencing. *arXiv preprint arXiv:2209.10507*, 2022. 1

[8] Bo Han, Yu Liu, and Feng Qian. Vivo: Visibility-aware mobile volumetric video streaming. In *Proceedings of the 26th annual international conference on mobile computing and networking*, pages 1–13, 2020. 1, 2, 3, 6, 8

[9] Anlan Zhang, Chendong Wang, Bo Han, and Feng Qian. Yuzu:neural-enhanced volumetric video streaming. In *19th USENIX Symposium on Networked Systems Design and Implementation (NSDI 22)*, pages 137–154, 2022. 1, 2, 3, 8

[10] Junhua Liu, Boxiang Zhu, Fangxin Wang, Yili Jin, Wenyi Zhang, Zihan Xu, and Shuguang Cui. Cav3: Cache-assisted viewport adaptive volumetric video streaming. In *2023 IEEE Conference Virtual Reality and 3D User Interfaces (VR)*, pages 173–183. IEEE, 2023. 1, 3, 6

[11] Yongjie Guan, Xueyu Hou, Nan Wu, Bo Han, and Tao Han. Metastream: Live volumetric content capture, creation, delivery, and rendering in real time. In *Proceedings of the 29th Annual International Conference on Mobile Computing and Networking*, 2023. 1

[12] Draco 3d data compression. `https://google.github.io/draco`, 2023. 1, 8

[13] 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:e13, 2020. 1, 2

[14] Yili Jin, Junhua Liu, and Fangxin Wang. Ebublio: Edge assisted multi-user 360-degree video streaming. In *2022 IEEE Conference on Virtual Reality and 3D User Interfaces Abstracts and Workshops (VRW)*, pages 600–601. IEEE, 2022. 1

[15] Lovish Chopra, Sarthak Chakraborty, Abhijit Mondal, and Sandip Chakraborty. Parima: Viewport adaptive 360-degree video streaming. In *Proceedings of the Web Conference 2021*, pages 2379–2391, 2021. 1

[16] Ruwen Schnabel and Reinhard Klein. Octree-based point-cloud compression. *PBG@ SIGGRAPH*, 3, 2006. 2, 3

[17] Xihua Sheng, Li Li, Dong Liu, and Zhiwei Xiong. Attribute artifacts removal for geometry-based point cloud compression. *IEEE Transactions on Image Processing*, 31:3399–3413, 2022. 2

[18] Jounsup Park, Philip A Chou, and Jenq-Neng 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. 2

[19] Gary J Sullivan, Jens-Rainer Ohm, Woo-Jin Han, and Thomas Wiegand. Overview of the high efficiency video coding (hevc) standard. *IEEE Transactions on circuits and systems for video technology*, 22(12):1649–1668, 2012. 2

[20] Borko Furht, Joshua Greenberg, and Raymond Westwater. *Motion estimation algorithms for video compression*, volume 379. Springer Science & Business Media, 2012. 2

[21] Linyao Gao, Tingyu Fan, Jianqiang Wan, Yiling Xu, Jun Sun, and Zhan Ma. Point cloud geometry compression via neural graph sampling. In *2021 IEEE International Conference on Image Processing (ICIP)*, pages 3373–3377. IEEE, 2021. 2

[22] Tianxin Huang and Yong Liu. 3d point cloud geometry compression on deep learning. In *Proceedings of the 27th ACM international conference on multimedia*, pages 890–898, 2019. 2

[23] Jiahao Pang, Muhammad Asad Lodhi, and Dong Tian. Grasp-net: Geometric residual analysis and synthesis for point cloud compression. In *Proceedings of the 1st International Workshop on Advances in Point Cloud Compression, Processing and Analysis*, pages 11–19, 2022. 2

[24] Lila Huang, Shenlong Wang, Kelvin Wong, Jerry Liu, and Raquel Urtasun. Octsqueeze: Octree-structured entropy model for lidar compression. In *Proceedings of the IEEE/CVF conference on computer vision and pattern recognition*, pages 1313–1323, 2020. 2

[25] Ehab Ghabashneh, Chandan Bothra, Ramesh Govindan, Antonio Ortega, and Sanjay Rao. Dragonfly: Higher perceptual quality for continuous 360 video playback. In *Proceedings of the ACM SIGCOMM 2023 Conference*, pages 516–532, 2023. 2

[26] Yihua Cheng, Anton Arapin, Ziyi Zhang, Qizheng Zhang, Hanchen Li, Nick Feamster, and Junchen Jiang. Grace: Loss-resilient real-time video communication using data-scalable autoencoder. *arXiv preprint arXiv:2210.16639*, 2022. 2

[27] Johannes Ballé, Valero Laparra, and Eero P Simoncelli. End-to-end optimized image compression. In *5th International Conference on Learning Representations, ICLR*, 2017. 2, 5

[28] Johannes Ballé, David Minnen, Saurabh Singh, Sung Jin Hwang, and Nick Johnston. Variational image compression with a scale hyperprior. *arXiv preprint arXiv:1802.01436*, 2018. 2, 5

[29] Kevin Spiteri, Rahul Urgaonkar, and Ramesh K Sitaraman. Bola: Near-optimal bitrate adaptation for online videos. *IEEE/ACM transactions on networking*, 28(4):1698–1711, 2020. 2

[30] Ziyu Ying, Shulin Zhao, Sandeepa Bhuyan, Cyan Subhra Mishra, Mahmut T Kandemir, and Chita R Das. Pushing point cloud compression to the edge. In *2022 55th IEEE/ACM International Symposium on Microarchitecture (MICRO)*, pages 282–299. IEEE, 2022. 2

[31] Ricardo L de Queiroz, Diogo C Garcia, Philip A Chou, and Dinei A Florencio. Distance-based probability model for octree coding. *IEEE Signal Processing Letters*, 25(6):739–742, 2018. 2

[32] Diogo C Garcia and Ricardo L de Queiroz. Context-based octree coding for point-cloud video. In *2017 IEEE International Conference on Image Processing (ICIP)*, pages 1412–1416. IEEE, 2017. 2

[33] Diogo C Garcia, Tiago A Fonseca, Renan U Ferreira, and Ricardo L de Queiroz. Geometry coding for dynamic voxelized point clouds using octrees and multiple contexts. *IEEE Transactions on Image Processing*, 29:313–322, 2019. 2

[34] Yiqun Xu, Wei Hu, Shanshe Wang, Xinfeng Zhang, Shiqi Wang, Siwei Ma, Zongming Guo, and Wen Gao. Predictive generalized graph fourier transform for attribute compression of dynamic point clouds. *IEEE Transactions on Circuits and Systems for Video Technology*, 31(5):1968–1982, 2020. 2

[35] Rufael Mekuria, Kees Blom, and Pablo Cesar. Design, implementation, and evaluation of a point cloud codec for tele-immersive video. *IEEE Transactions on Circuits and Systems for Video Technology*, 27(4):828–842, 2016. 2

[36] Sebastian Schwarz, Marius Preda, Vittorio Baroncini, Madhukar Budagavi, Pablo Cesar, Philip A Chou, Robert A Cohen, Maja Krivokuća, Sébastien Lasserre, Zhu Li, et al. Emerging mpeg standards for point cloud compression. *IEEE Journal on Emerging and Selected Topics in Circuits and Systems*, 9(1):133–148, 2018. 2, 8

[37] Jianqiang Wang, Hao Zhu, Haojie Liu, and Zhan Ma. Lossy point cloud geometry compression via end-to-end learning. *IEEE Transactions on Circuits and Systems for Video Technology*, 31(12):4909–4923, 2021. 2

[38] Jianqiang Wang, Dandan Ding, Zhu Li, and Zhan Ma. Multiscale point cloud geometry compression. In *2021 Data Compression Conference (DCC)*, pages 73–82. IEEE, 2021. 2, 4, 8

[39] André FR Guarda, Nuno MM Rodrigues, and Fernando Pereira. Adaptive deep learning-based point cloud geometry coding. *IEEE Journal of Selected Topics in Signal Processing*, 15(2):415–430, 2020. 2

[40] Chunyang Fu, Ge Li, Rui Song, Wei Gao, and Shan Liu. Octattention: Octree-based large-scale contexts model for point cloud compression. In *Proceedings of the AAAI Conference on Artificial Intelligence*, volume 36, pages 625–633, 2022. 2

[41] Zizheng Que, Guo Lu, and Dong Xu. Voxelcontext-net: An octree based framework for point cloud compression. In *Proceedings of the IEEE/CVF Conference on Computer Vision and Pattern Recognition*, pages 6042–6051, 2021. 2

[42] Charles Ruizhongtai Qi, Li Yi, Hao Su, and Leonidas J Guibas. Pointnet++: Deep hierarchical feature learning on point sets in a metric space. *Advances in neural information processing systems*, 30, 2017. 2

[43] Christopher Choy, JunYoung Gwak, and Silvio Savarese. 4d spatio-temporal convnets: Minkowski convolutional neural networks. In *Proceedings of the IEEE/CVF conference on computer vision and pattern recognition*, pages 3075–3084, 2019. 2, 3

[44] Cha Zhang, Dinei Florencio, and Charles Loop. Point cloud attribute compression with graph transform. In *2014 IEEE International Conference on Image Processing (ICIP)*, pages 2066–2070. IEEE, 2014. 2

[45] Michael Zink, Ramesh Sitaraman, and Klara Nahrstedt. Scalable 360 video stream delivery: Challenges, solutions, and opportunities. *Proceedings of the IEEE*, 107(4):639–650, 2019. 2

[46] Chengjun Guo, Ying Cui, and Zhi Liu. Optimal multicast of tiled 360 vr video in ofdma systems. *IEEE Communications Letters*, 22(12):2563–2566, 2018. 2

[47] Jounsup Park, Philip A Chou, and Jenq-Neng 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. 2

[48] Jeroen van der Hooft, Tim Wauters, Filip De Turck, Christian Timmerer, and Hermann Hellwagner. Towards 6dof http adaptive streaming through point cloud compression. In *Proceedings of the 27th ACM International Conference on Multimedia*, pages 2405–2413, 2019. 2

[49] Jie Li, Cong Zhang, Zhi Liu, Wei Sun, and Qiyue Li. Joint communication and computational resource allocation for qoe-driven point cloud video streaming. In *2020 IEEE International Conference on Communications (ICC)*, pages 1–6. IEEE, 2020. 2

[50] Zeqi Lai, Y Charlie Hu, Yong Cui, Linhui Sun, Ningwei Dai, and Hung-Sheng Lee. Furion: Engineering high-quality immersive virtual reality on today's mobile devices. *IEEE Transactions on Mobile Computing*, 19(7):1586–1602, 2019. 2

[51] B Graham. Sparse 3d convolutional neural networks. bmvc.(2015). 3

[52] Mengye Ren, Andrei Pokrovsky, Bin Yang, and Raquel Urtasun. Sbnet: Sparse blocks network for fast inference. In *Proceedings of the IEEE Conference on Computer Vision and Pattern Recognition*, pages 8711–8720, 2018. 3

[53] Haotian Tang, Zhijian Liu, Xiuyu Li, Yujun Lin, and Song Han. Torchsparse: Efficient point cloud inference engine. *Proceedings of Machine Learning and Systems*, 4:302–315, 2022. 3

[54] Yu Liu, Bo Han, Feng Qian, Arvind Narayanan, and Zhi-Li Zhang. Vues: practical mobile volumetric video streaming through multiview transcoding. In *Proceedings of the 28th Annual International Conference on Mobile Computing And Networking*, pages 514–527, 2022. 3

[55] Kyungjin Lee, Juheon Yi, Youngki Lee, Sunghyun Choi, and Young Min Kim. Groot: a real-time streaming system of high-fidelity volumetric videos. In *Proceedings of the 26th Annual International Conference on Mobile Computing and Networking*, pages 1–14, 2020. 3, 8

[56] Eugene d'Eon, Bob Harrison, Taos Myers, and Philip A Chou. 8i voxelized full bodies-a voxelized point cloud dataset. *ISO/IEC JTC1/SC29 Joint WG11/WG1 (MPEG/JPEG) input document WG11M40059/WG1M74006*, 7(8):11, 2017. 3, 7

[57] Christian Szegedy, Sergey Ioffe, Vincent Vanhoucke, and Alexander Alemi. Inception-v4, inception-resnet and the impact of residual connections on learning. In *Proceedings of the AAAI conference on artificial intelligence*, volume 31, 2017. 5

[58] Alain Hore and Djemel Ziou. Image quality metrics: Psnr vs. ssim. In *2010 20th international conference on pattern recognition*, pages 2366–2369. IEEE, 2010. 6

[59] Geometry based point cloud compression (g-pcc) test model. https://github.com/MPEGGroup/mpeg-pcc-tmc13, 2021. 6

[60] Shishir Subramanyam, Irene Viola, Alan Hanjalic, and Pablo Cesar. User centered adaptive streaming of dynamic point clouds with low complexity tiling. In *Proceedings of the 28th ACM international conference on multimedia*, pages 3669–3677, 2020. 7

[61] Lifan Mei, Runchen Hu, Houwei Cao, Yong Liu, Zifan Han, Feng Li, and Jin Li. Realtime mobile bandwidth prediction using lstm neural network and bayesian fusion. *Computer Networks*, 182:107515, 2020. 8

[62] Diederik Kingma and Jimmy Ba. Adam: A method for stochastic optimization. In *International Conference on Learning Representations (ICLR)*, San Diega, CA, USA, 2015. 8

[63] Yi Xu, Yao Lu, and Ziyu Wen. Owlii dynamic human mesh sequence dataset. In *ISO/IEC JTC1/SC29/WG11 m41658, 120th MPEG Meeting*, volume 1, page 8, 2017. 8

[64] Rufael Mekuria and Pablo Cesar. Mp3dg-pcc, open source software framework for implementation and evaluation of point cloud compression. In *Proceedings of the 24th ACM international conference on Multimedia*, pages 1222–1226. ACM, 2016. 8