Improving Occupancy Prediction for Multiscale Point Cloud Geometry Compression

Zehong Li, Jiahao Zhu, Dandan Ding, *Senior Member, IEEE*, and Zhan Ma, *Senior Member, IEEE*

***Abstract*—Multiscale sparse representation offers significant advantages in point cloud geometry compression, delivering state-of-the-art performance compared to both standardized solutions and other learned approaches. A crucial component of this framework is the cross-scale occupancy prediction, which employs the lower-scale reference representation either from the current frame alone or from both the current and temporal reference frames to establish conditional priors for either static or dynamic coding. However, existing works mainly use local computations, *e.g.*, sparse convolutions and *k*NN attention, to exploit correlations in such a representation; these methods usually fail to adequately capture global coherence. In addition, the fixed configuration of lossless-lossy scales cannot adapt to temporal dynamics, which limits the reconstruction quality of temporal references in dynamic coding. These limitations constrain the generation of more effective priors used for conditional coding. To address these issues, we propose two new techniques. The first is KPA (Key Point-driven Attention), which integrates both local and global characteristics. The second is AdaScale (Adaptive Lossy/Lossless Scale), which decides whether the transitional scale should be in lossless or lossy mode based on temporal displacement, thereby enhancing the reconstruction quality of the temporal reference. Extensive experiments demonstrate that our approach significantly outperforms state-of-the-art methods, including rules-based standard codecs like G-PCC and V-PCC, as well as learning-based approaches like Unicorn and TMAP, across both static/dynamic and lossy/lossless coding scenarios.**

***Index Terms*—Point cloud geometry compression, key point, cross attention, conditional coding.**

I. Introduction

POINT clouds are widely used as a 3D data representation format and are extensively applied in fields such as autonomous driving, virtual and augmented reality, robotic navigation, and cultural heritage preservation [1], [2], [3]. They often consist of millions of unstructured points that capture complex geometric details of objects and scenes, with each point containing geometric coordinates (x, y, z) and optional attributes like RGB color (r, g, b), normal vector, or reflectance. This large volume of data imposes a significant burden on existing network bandwidth and storage resources.

Received 30 April 2025; revised 19 September 2025; accepted 4 December 2025. Date of publication 9 December 2025; date of current version 7 April 2026. This work was supported by the National Natural Science Foundation of China under Grant 62171174. This article was recommended by Associate Editor G. Valenzise. *(Corresponding author: Dandan Ding.)*

Zehong Li, Jiahao Zhu, and Dandan Ding are with the School of Information Science and Technology, Hangzhou Normal University, Hangzhou 311121, China (e-mail: DandanDing@hznu.edu.cn).

Zhan Ma is with the School of Electronic Science and Engineering, Nanjing University, Nanjing, Jiangsu 210093, China (e-mail: mazhan@nju.edu.cn).

This article has supplementary downloadable material available at https://doi.org/10.1109/TCSVT.2025.3642141, provided by the authors.

Digital Object Identifier 10.1109/TCSVT.2025.3642141

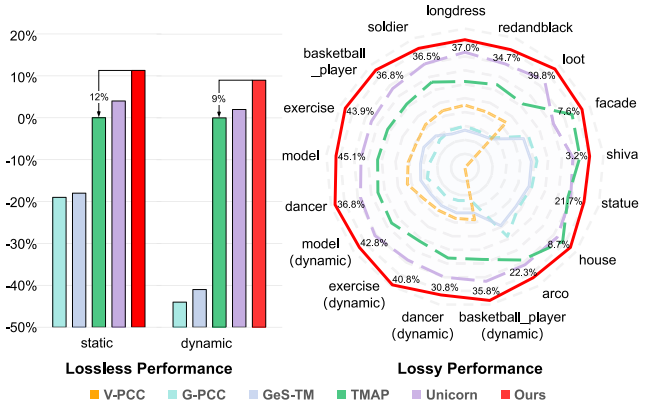


Fig. 1. Compression Gains. The proposed method demonstrates a significant advantage over both rules-based approaches, such as V-PCC [8], G-PCC [9], and GeS-TM [10], as well as learned methods such as TMAP [11] and Unicorn [12], in both lossless and lossy, as well as static and dynamic geometry coding modes. We evaluate the lossless performance using the 8iVFB and Owlii datasets. TMAP, a recent AI-PCC model released by MPEG, serves as the anchor.

To address this challenge, point cloud compression (PCC) has become a crucial research area, focusing on reducing data redundancy while preserving geometry and attribute fidelity [4], [5], [6], [7].

A. Problem Formulation

To date, the international standardization committee Motion Picture Experts Group (MPEG) has released two PCC standards [13]: V-PCC (video-based PCC) [8] and G-PCC (geometry-based PCC) [9]. V-PCC projects a 3D point cloud into 2D images and utilizes existing video codecs (*e.g.*, H.265/HEVC [14] and H.266/VVC [15]) for compression. In contrast, G-PCC directly processes 3D points, maintaining spatial coherence. For the geometry component, G-PCC provides octree, trisoup, and predictive tree schemes. For the attribute component, it offers region-adaptive hierarchical transform (RAHT) and predicting/lifting transform techniques.

With the advancement of artificial intelligence (AI) technologies, recent explorations have typically utilized deep neural networks to develop end-to-end compression frameworks that outperform G-PCC and V-PCC [16], [17], [18], [19]. Recently, standardization committees such as MPEG and JPEG (Joint Photographic Experts Group) have initiated AI-based PCC standardization activities (AI-PCC). Among these initiatives, the *multiscale sparse representation* framework [20], [21], [22], [23] has emerged as a leading approach,

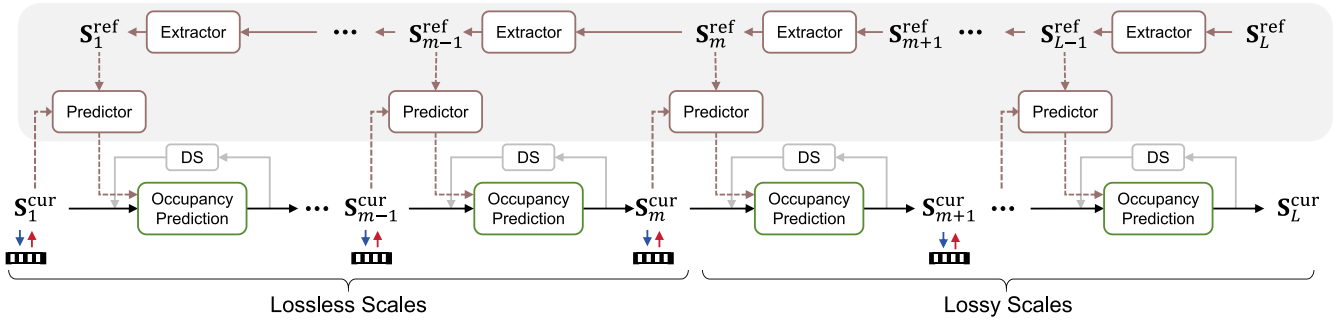


Fig. 2. Multiscale Sparse Representation Framework widely used for neural point cloud compression. The full-resolution point cloud frame $\mathbf{S}_L^{\text{cur}}$ is progressively downscaled (DS) to create $\{\mathbf{S}_l^{\text{cur}}\}_{l=1}^{L}$. Compression starts with the thumbnail point cloud $\mathbf{S}_1^{\text{cur}}$, and each subsequent scale is encoded using its lower-scale reference, which may be the spatial-only reconstruction $\mathbf{S}_{l-1}^{\text{cur}}$ or a fusion of spatial and temporal reconstructions like $\mathbf{S}_{l-1}^{\text{cur}}$ and $\mathbf{S}_{l-1}^{\text{ref}}$. The Occupancy Prediction block uses this representation to generate conditional priors for voxel occupancy prediction in geometry compression. The Extractor and Predictor blocks embed and transfer temporal references for dynamic coding. Typically, scales $l = 1$ to m are processed in lossless mode, while scales $l = m + 1$ to L are handled in lossy mode.

achieving superior coding performance, particularly for geometry compression. For instance, TMAP [11], a preliminary reference software released by the MPEG AI-PCC group, is developed based on the multiscale framework, with > 50% BD-BR (Bjøntegaard Delta Bit Rate [24]) gains over V-PCC in static geometry compression for dense point cloud samples.

Figure 2 illustrates the commonly used multiscale sparse representation framework. In this approach, the current full-resolution point cloud frame $\mathbf{S}_L^{\text{cur}}$ is progressively downscaled to create multiscale representations $\{\mathbf{S}_l^{\text{cur}}\}_{l=1}^{L}$. Compression begins with the thumbnail point cloud $\mathbf{S}_1^{\text{cur}}$ at the first scale using simple fixed-length codes, as seen in existing implementations [11], [12]. Each subsequent scale is encoded based on the lower-scale reference.

For the current scale l, the lower-scale reference might be the spatial-only point cloud reconstruction $\mathbf{S}_{l-1}^{\text{cur}}$ from the preceding scale in static coding, or a fusion of spatial and temporal reconstructions in dynamic coding, denoted as $\mathbf{S}_{l-1}^{\text{cur}}$ and $\mathbf{S}_{l-1}^{\text{ref}}$, respectively. In dynamic coding, the Extractor and Predictor blocks are carefully designed for each scale to embed and transfer the temporal reference to be merged with the spatial reference.

Typically, the initial scales, from $l = 1$ to m, are coded in lossless mode, while the remaining scales, from $l = m + 1$ to L, are coded in lossy mode. This framework establishes a conditional coding chain to leverage cross-scale correlations, with the Occupancy Prediction block playing a crucial role. It inputs the lower-scale reference to generate conditional priors for predicting the current scale's voxel occupancy, thereby enhancing compression efficiency.

Despite this, the potential of occupancy prediction remains largely underexplored in current research. Most existing methods focus on local modeling and embedding, employing techniques like stacked sparse convolutions or k nearest neighbors (kNN)-based attention. These techniques are limited by the size of convolutional kernels or the value of k, confining them to local neighbors. As a result, they fail to capture global correlations, which can negatively affect overall compression performance. Additionally, using a fixed configuration of lossless-lossy scales across all temporal frames, *e.g.*, a constant m, is insufficient for fully exploiting temporal redundancy. For instance, while using more lossless scales, *i.e.*, a larger m, might require additional bits for a specific frame, it provides a better reference for subsequent frames, ultimately leading to improved rate-distortion (R-D) performance.

B. Our Approach

Therefore, this work introduces two new techniques to enhance the occupancy prediction process, generating better conditional priors for geometry compression of dense point clouds in immersive media applications.

First, we introduce Key Point-driven Attention (KPA) to enhance existing local computation. At each scale l of the current frame, the input lower-scale reference tensor with N points from the preceding scale $l - 1$ (such as the lower-scale geometry reconstruction $\mathbf{S}_{l-1}^{\text{cur}}$) is mapped from its original 3D space into a 1D ordered sequence for uniform point sampling to extract n key points, where $N \gg n$. Local feature embedding is applied to these key points through multi-level sparse convolutions with varying strides. Finally, each element in the input tensor is fused with key points via cross-attention optimization to enable global feature embedding. In this way, KPA effectively exploits both local and global characteristics, generating more effective conditional priors for occupancy prediction in subsequent steps.

On the other hand, the quality of the lower-scale reference to KPA is crucial. An intuitive example is the hierarchical quality configurations used by temporal frames in video coding for decades [25], where enhancing the quality of the temporal reference frame typically results in improved overall compression performance. Inspired by this, we propose the AdaScale (adaptive lossy/lossless scale) to relax the fixed configuration of lossless-lossy scales used in existing approaches [11], [12] for lossy dynamic coding. We define the $(m+1)$-th scale of a frame as the transitional scale. Our method determines whether this transitional scale is encoded in lossy or lossless mode based on temporal displacement, producing reconstructed temporal references with varying quality levels. Consequently, unlike existing approaches that strictly use the immediately preceding

frame as a reference, AdaScale allows the encoding of scale $m+1$ to utilize a high-quality temporal reference, yielding improved coding efficiency.

We then incorporate the proposed KPA and AdaScale methods into the Unicorn codebase to demonstrate its performance and complexity. Experimental studies strictly follow the common test conditions defined by the standardization committees for fair comparisons with existing solutions.

C. Contribution

Our contributions are summarized as follows:

- By integrating KPA and AdaScale into the multiscale sparse representation framework, this work demonstrates significant improvements over rules-based solutions like G-PCC and V-PCC, as well as learned approaches like Unicorn and TMAP.[1]
  - In static coding, it surpasses TMAP by 9.1% to 22.3% in lossless mode and by 12.7% to 40.7% in lossy mode. For dynamic coding, it outperforms TMAP by 8.8% in lossless mode and 37.6% in lossy mode. Note that TMAP requires significantly longer processing time, likely due to redundant modules such as point-wise enhancement.
  - Compared to V-PCC, our method achieves BD-BR gains of about 56% in static lossy mode and 91% in dynamic lossy mode, while maintaining comparable decoding latency.
- The impressive compression gains of this work are attributed to the use of KPA and AdaScale. KPA combines both local and global characteristics of the input reference representation, while AdaScale improves the quality of the reference representation itself. Together, these mechanisms strive to generate better conditional priors for occupancy prediction.
  - KPA samples a limited number of key points to perform cross attention, maintaining global structural coherence while significantly reducing complexity.
  - AdaScale loosens the fixed configuration of lossless-lossy scales, opening new opportunities to optimize the reference frame quality for better performance.

Table I lists frequently used notations throughout this work.

TABLE I
NOTATIONS

Notation	Description
PCC	Point Cloud Compression
AI-PCC	PCC using Artificial Intelligence (*e.g.*, Deep Learning)
V-PCC	Video-based PCC
G-PCC	Geometry-based PCC
KPA	Key Point-driven Attention
AdaScale	Adaptive (Lossy or Lossless) Scale
kNN	k nearest neighbors
BD-BR	Bjøntegaard Delta Bit Rate
MPEG	Moving Picture Experts Group
JPEG	Joint Photographic Experts Group
R-D	Rate-Distortion
DS	Downscale
US	Upscale

II. Related Work

This section focuses on reviewing learning-based methods for point cloud geometry compression. Prior to these advancements, substantial efforts were invested in developing rules-based approaches, ultimately leading to the development of the G-PCC and V-PCC standards through collaborative efforts within MPEG since 2017. These endeavors continue today, with the emergence of GeS-TM [10], specifically designed for solid/dense colored point clouds within the G-PCC framework, and L3C2 [26], dedicated to LiDAR point cloud compression.

[1]Compression gains are shown in comparison to TMAP and V-PCC, which are standardized solutions with the leading performance on dense samples.

In recent years, learning-based explorations have grown exponentially, primarily categorized into point-based, voxel-based, octree-based, and sparse tensor-based models. Point-based approaches typically process raw input directly, without the need for voxelization, while others generally operate on voxelized point clouds.

Point-based Models often apply PointNet [27] or PointNet++ [28] that employ multilayer perceptrons (MLPs) to directly process raw, unstructured 3D points for feature extraction and then generate compact feature vectors for compression [29], [30], [31], [32]. One advantage of these methods lies in avoiding voxelization, as mentioned above. However, they are more suitable for small-scale samples with thousands of points.

Voxel-based Models utilize dense 3D convolutions on voxelized point clouds to exploit neighborhood correlations in 3D space [33], [34], [35]. However, these models demonstrate significant complexity because dense 3D convolutions process all points within the receptive field, regardless of their occupancy status, even when patch-wise computation is employed.

Octree-based Models recursively divide the point cloud to build a hierarchical tree, then compress it starting from the root node down to the leaf nodes [36], [37], [38], [39], [40], [41], [42]. Significant efforts have been made to engineer deep models to enhance the octree context model, aiming for more accurate probability estimation by thoroughly characterizing correlations across parent-child-sibling nodes. Early explorations included OctSqueeze [36] and MuSCLE [43], which utilized MLPs. Subsequently, further improvements were achieved in VoxelContext-Net [44] using dense 3D convolutions, PCC-S [45] using both stacked convolutions and MLPs, and attention-based models like OctAttention [38], EHEM [39], and OctFormer [40].

The octree-based model essentially converts 3D points into 1D node sequences to effectively explore correlations. This approach is particularly suited for LiDAR scans, as they are often extremely sparse, making it challenging for a reasonable-sized 3D convolution to cover sufficient neighbors.

Sparse tensor-based Models are extensively studied lately. PCGCv2 [20] utilized a sparse convolution-based autoencoder to downsample and compress static point clouds. PCGFormer [46] further enhanced PCGCv2 by embedding the self-attention mechanism. SparsePCGC [22] extended PCGCv2 by

implementing the cross-scale conditional coding, achieving superior performance in both lossless and lossy geometry compression for static point clouds.

Beyond compressing static point clouds, numerous studies have explored dynamic point cloud compression using sparse tensors. D-DPCC [47] introduced a motion estimation technique to align the coordinates of the reference frame with those of the current frame and calculate their residual differences for compression. Xia et al. [48] later enhanced this motion estimation by incorporating localized attention using kNN. In contrast, Akhtar et al. [49] utilized target convolutions to aggregate information from the preceding frame to predict the latent representation of the current frame.

Recent efforts focus on developing a comprehensive solution that unifies various modes (*e.g.*, static and dynamic, geometry and attribute, lossy and lossless) [11], [12], [16]. Among these, the multiscale framework shown in Fig. 2 is becoming the mainstream approach. Notable examples include Unicorn [12], a successor to SparsePCGC, and TMAP, a test model released by the MPEG AI-PCC group, which was developed following concepts from both Unicorn and SparsePCGC. More specifically,

- Unicorn provides a versatile solution for both lossless and lossy, as well as static and dynamic PCC, for any input point cloud. By adjusting lossless-lossy scales and incorporating a multistage mechanism within each scale, Unicorn achieves state-of-the-art performance. Specifically, built upon the framework in Fig. 2, Unicorn utilizes three cascaded inception residual networks (IRNs) to implement Occupancy Prediction and Extractor, and leverages a single target convolution layer in Predictor (called Warper in Unicorn) to warp temporal priors from the reference frame to the current frame. As seen, Unicorn mainly uses sparse convolutions in Occupancy Prediction, Extractor, and Predictor to exploit local information, limiting its ability to capture global correlation.
- TMAP is also developed upon the multiscale framework in Fig. 2. Similar to Unicorn, it employs three IRNs to perform Occupancy Prediction and Extractor. The difference is that, while Unicorn initializes all features in these blocks to 1, TMAP initializes them using a pre-defined algorithm that leverages available information such as 3D coordinates, levels, eighth-order exponents, and voxel sizes from the lower scale. For the Predictor block, TMAP first merges the reference and current scales and applies three IRNs to extract motion features. Both scales are then downscaled to a lower resolution, merged as a downscaled sample, and processed by another set of three IRNs to extract additional motion features. Finally, the motion features from both scales are fused and passed through an MLP to generate the temporal prior information. Similarly, TMAP focuses on local regions and largely overlooks the global structure.

Additionally, both Unicorn and TMAP apply the fixed configuration of lossless-lossy scales across all frames, disregarding the impact of reference quality on overall performance, which further restricts their dynamic coding efficiency.

Next, we will elaborate on the proposed KPA and AdaScale techniques to tackle the challenges aforementioned.

III. KPA-Based Conditional Prior Enhancement

Existing occupancy prediction techniques mainly exploit local correlations using sparse convolutions or kNN attention, neglecting the global coherence of a point cloud. Applying global self-attention to the original point cloud could alleviate this issue but incurs an unbearable computational complexity of $O(N^2)$, where N denotes the total points in a scale.

To address this challenge, we introduce key point-driven attention, called KPA, where a set of key points is extracted and processed at each scale to effectively combine local correlations with global dependencies, thereby enhancing the accuracy of occupancy prediction.

A. Key Point-Driven Attention (KPA)

The proposed KPA method comprises three stages: key point identification, multi-level local feature embedding, and cross-attention embedding. Below, we provide a detailed explanation of each stage of KPA.

1) Key Point Identification: To derive a set of key points that thoroughly represent the geometric structure of the point cloud at a given scale, we employ a serialization technique known as a spatial filling curve, which converts unstructured 3D points into an ordered 1D representation. Common spatial filling curves include the Z-order and Hilbert curves [50], [51], [52]. According to the investigation of Chen et al. [53], the Z-order curve is simple and efficient, while the Hilbert curve excels in preserving locality, which is more effective for compression tasks. Therefore, to achieve uniform sampling of the point cloud for a consistent global structure while preserving local correlations, we utilize the Hilbert curve for serialization.

Figure 3a illustrates the serialization process using a projected 2D view. From the resulting 1D point sequence with ordered numbers, key points are uniformly sampled. The number of key points can be controlled by adjusting the sampling rate. Specifically, let $\mathbf{P} = \{p_i\}_{i=1}^{N}$ denote a reconstructed point cloud geometry tensor at a scale $l-1$ with N points and n feature channels, *i.e.*, $\mathbf{P} \in \mathbb{R}^{N\times n}$. Sampling M key points from $\mathbf{P}$ involves dividing N points into $\lfloor \frac{N}{M} \rfloor$ segments and selecting the last point of each segment as the key point, expressed as:

$$\mathbf{P}_{\text{key}} = \{p_{\text{key}}^{j}\}_{j=1}^{M} = \left\{ p_{j\cdot\lfloor \frac{N}{M} \rfloor} \right\}_{j=1}^{M}, \tag{1}$$

where p_{key}^{j} denotes the j-th key point.

2) Local Feature Embedding via Multi-Level Convolution: The obtained key points carry the global structure of a point cloud scale but lack local details. To address this, we apply sparse convolution to each key point to aggregate information from nearby neighbors. Specifically, for each key point p_{key}^{j} in $\mathbf{P}_{\text{key}}$, we use sparse convolution to gather surrounding information:

$$f_j = \text{TConv}_{\text{str}_i}(p_{\text{key}}^{j}), \tag{2}$$

where f_j is the feature vector extracted by the target convolution with stride str_i at the key point p_{key}^{j}. The stride parameter

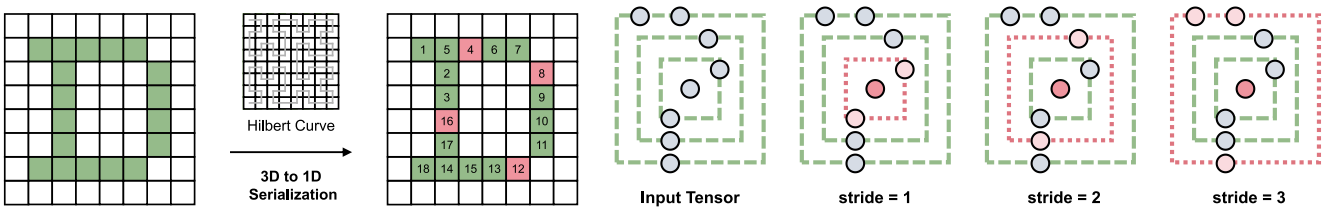


Fig. 3. (a) Key Point Identification with a projected 2D view. Uniform sampling of key points from an ordered 1D point sequence following 3D-to-1D Hilbert Curve mapping is illustrated. Valid points are shown in green boxes, while sampled key points are highlighted in red boxes; (b) Local Feature Embedding using multi-level sparse convolutions at various strides, with an illustration from a 2D view.

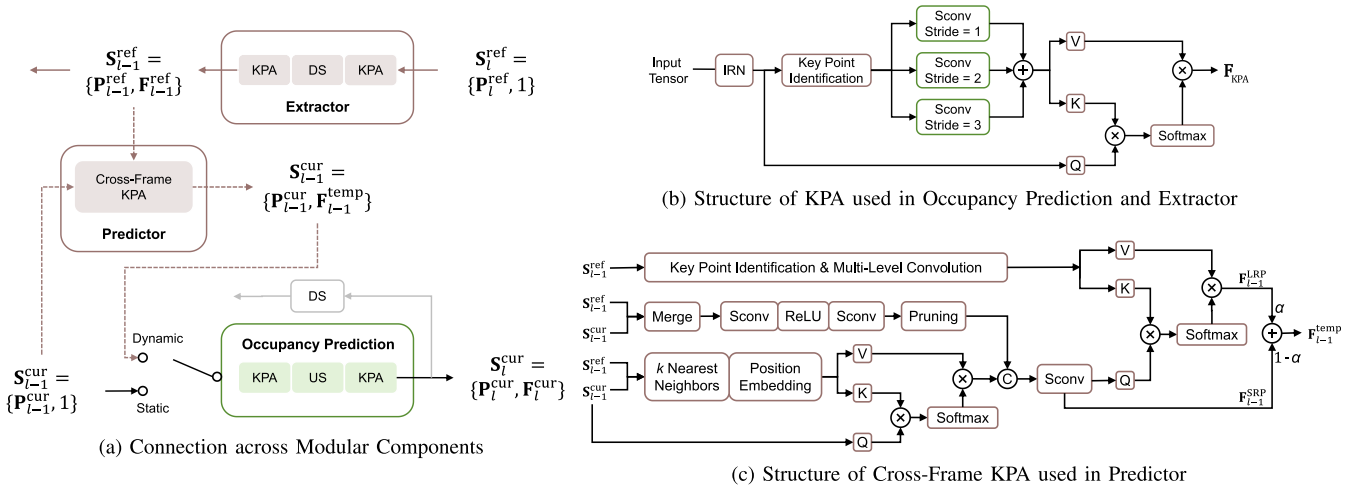


Fig. 4. Modular Component Details. (a) Connections among Occupancy Prediction, Extractor, and Predictor. (b) Structure of KPA used in Occupancy Prediction and Extractor. (c) Structure of Cross-Frame KPA used in Predictor.

str_i is adjusted to different values to capture information from varying spatial distances.

Subsequently, feature vectors from different strides are summed to characterize the local neighborhood of p^j_{key}, *i.e.*,

$$F_j = \sum_{j=1}^{W} f_j, \tag{3}$$

with $W = 3$ in this work, meaning we set str_i to 1, 2, and 3 separately to balance computational complexity and coding performance, as illustrated in Fig. 3b. Finally, the local feature vector associated with key points is $\mathbf{F}_{\text{local}} = \{F_j\}_{j=1}^{M}$.

3) Global Feature Embedding via Cross Attention: To further exploit global correlations, we propose using cross-attention computations between each point in the scale $l-1$ and its key points. Specifically, we transform the key point features $\mathbf{F}_{\text{local}} \in \mathbb{R}^{M\times n}$ into $\mathbf{K}_{\text{key}}$ and $\mathbf{V}_{\text{key}}$ representations, while converting the original input tensor $\mathbf{P} \in \mathbb{R}^{N\times n}$ into $\mathbf{Q}$. The output feature $\mathbf{F}_{\text{KPA}} \in \mathbb{R}^{N\times n}$ used to formulate conditional priors for subsequent occupancy prediction is then computed as follows:

$$\mathbf{F}_{\text{KPA}} = \text{Softmax}\left(\frac{\mathbf{Q}\mathbf{K}^{\text{T}}_{\text{key}}}{\sqrt{d_e}}\right)\mathbf{V}_{\text{key}}, \tag{4}$$

where d_e represents the feature channel dimension.

This cross-attention mechanism between the key point and each point in the input tensor allows for effectively capturing both short-range and long-range spatial dependencies, thereby improving the accuracy of occupancy prediction. Additionally, the computational complexity is reduced from $O(N^2)$ to $O(NM)$. For instance, if $N \gg M$, the computational complexity is reduced by orders of magnitude.

Figure 4b illustrates the structure of KPA. The IRN layer, adopted from PCGCv2 [20], is utilized at the beginning to process the input tensor with local neighborhood embedding efficiently.

B. Using KPA in Unicorn

Next, we integrate the proposed KPA into the multiscale point cloud geometry compressor, Unicorn, and describe its implementations under both static and dynamic coding modes.

1) Spatial Aggregation: In static coding mode, only spatially lower-scale reconstruction is used for occupancy prediction of the current scale l. The input tensor $\mathbf{S}^{\text{cur}}_{l-1}$ comprises the point coordinates and associated features initialized to 1, *i.e.*, $\mathbf{S}^{\text{cur}}_{l-1} = \{\mathbf{P}^{\text{cur}}_{l-1}, \mathbf{1}\}$, which is directly fed into the Occupancy Prediction block shown in Fig. 4a to generate $\mathbf{S}^{\text{cur}}_{l}$ for subsequent computations, regardless of lossless or lossy configurations, *i.e.*,

$$\mathbf{S}^{\text{cur}}_{l} = \{\mathbf{P}^{\text{cur}}_{l}, \mathbf{F}^{\text{cur}}_{l}\} = \mathbf{KPA}\left(\mathbf{US}\left(\mathbf{KPA}\left(\mathbf{S}^{\text{cur}}_{l-1}\right)\right)\right), \tag{5}$$

with US denoting the upscale operation. Here, Occupancy Prediction employs two KPA modules shown in Fig. 4b.

2) Temporal Aggregation: In dynamic coding mode, previously reconstructed temporal references can be used to enhance occupancy prediction, as depicted in Fig. 2. By default, Unicorn employs a straightforward mechanism to achieve this, consisting of the Extractor block, which embeds local aggregation within the temporal reference frame, and the Predictor block, which transfers this local aggregation with the spatially lower-scale reconstruction of the current frame[2] for subsequent occupancy prediction. As a result, Unicorn fails to handle large motions.

This work extends the design of KPA to fuse temporal references more effectively. One aspect involves using two KPA modules in Extractor (as illustrated in Fig. 4a) to generate the temporal reference at scale $l-1$, *i.e.*, $\mathbf{S}_{l-1}^{\text{ref}}$ through

$$\mathbf{S}_{l-1}^{\text{ref}} = \mathbf{KPA}\left(\mathbf{DS}\left(\mathbf{KPA}(\mathbf{S}_{l}^{\text{ref}})\right)\right), \tag{6}$$

with $\mathbf{S}_{l}^{\text{ref}} = \{\mathbf{P}_{l}^{\text{ref}}, \mathbf{1}\}$.

Further, the Predictor block employs Cross-Frame KPA (Fig. 4c) to transfer the derived $\mathbf{S}_{l-1}^{\text{ref}}$ to the same-scale spatial reference $\mathbf{S}_{l-1}^{\text{cur}}$. The Predictor block has two branches.

- One branch uses sparse convolutions and kNN attention to transfer the local neighbors in the temporal reference $\mathbf{S}_{l-1}^{\text{ref}}$ to the current scale $\mathbf{S}_{l-1}^{\text{cur}}$ to form the short-range prediction $\mathbf{F}_{l-1}^{\text{SRP}}$, mainly comprising two parallel lines:
  - On one hand, we combine $\mathbf{S}_{l-1}^{\text{ref}}$ and $\mathbf{S}_{l-1}^{\text{cur}}$ into a single set, merging any duplicate points by linking their high-dimensional features to the same spatial coordinates. Next, we apply sparse convolutions to aggregate information from local neighbors. Finally, the set is pruned based on the coordinate $\mathbf{P}_{l-1}^{\text{cur}}$ of $\mathbf{S}_{l-1}^{\text{cur}}$, resulting in $\mathbf{S}_{l-1}^{\text{merge}} = \{\mathbf{P}_{l-1}^{\text{cur}}, \mathbf{F}_{l-1}^{\text{merge}}\}$.
  - On the other hand, for each point in $\mathbf{S}_{l-1}^{\text{cur}}$, we identify its k nearest neighbors from $\mathbf{S}_{l-1}^{\text{ref}}$ to form the local neighborhood. Local attention is then applied to capture correlations, resulting in $\mathbf{S}_{l-1}^{\text{targetKNN}} = \{\mathbf{P}_{l-1}^{\text{cur}}, \mathbf{F}_{l-1}^{\text{targetKNN}}\}$.
  - The local aggregations from the two distinct processes above are integrated to produce the short-range prediction, *i.e.*, $\mathbf{S}_{l-1}^{\text{SRP}} = \{\mathbf{P}_{l-1}^{\text{cur}}, \mathbf{F}_{l-1}^{\text{SRP}}\} = \mathbf{concat}\left(\mathbf{S}_{l-1}^{\text{merge}}, \mathbf{S}_{l-1}^{\text{targetKNN}}\right)$.
- The other branch applies key point identification & multi-level convolution on $\mathbf{S}_{l-1}^{\text{ref}}$ to form key point sets, and then these key points are processed with $\mathbf{S}_{l-1}^{\text{SRP}}$ via the cross attention, producing the long-range prediction $\mathbf{S}_{l-1}^{\text{LRP}} = \{\mathbf{P}_{l-1}^{\text{cur}}, \mathbf{F}_{l-1}^{\text{LRP}}\}$.

Finally, $\mathbf{F}_{l-1}^{\text{LRP}}$ and $\mathbf{F}_{l-1}^{\text{SRP}}$ are linearly weighted with a learnable factor a to create the feature-space temporal reference $\mathbf{F}_{l-1}^{\text{temp}}$. This reference is then attached to $\mathbf{P}_{l-1}^{\text{cur}}$, resulting in updated $\mathbf{S}_{l-1}^{\text{cur}} = \{\mathbf{P}_{l-1}^{\text{cur}}, \mathbf{F}_{l-1}^{\text{temp}}\}$. $\mathbf{S}_{l-1}^{\text{cur}}$ is then used as input tensor to the Occupancy Prediction block depicted in Eq. (5) for dynamic coding.

[2]It is important to note that Extractor and Predictor operate scale by scale, aligning with the multiscale steps when processing the current point cloud frame.

IV. AdaScale-Based Temporal Reference Enhancement

As for dynamic coding, existing solutions like Unicorn and TMAP apply a fixed configuration of lossless-lossy scales, such as lossless mode from 1 to m scales and lossy mode from $m+1$ to L scales. This largely overlooks the content dynamics across temporal frames. Recalling that a temporal reference with better reconstruction quality would lead to the improvement of overall rate-distortion efficiency [25], we therefore propose to implement a similar strategy in this work. In general, the process involves addressing two subproblems: first, determining how to enhance the quality of the reconstructed point cloud geometry; and second, selecting the high-quality temporal frame for reference.

A. Transitional Scale Adaptation

As m governs the lossless-lossy scales to adjust the quality of compressed reconstruction, we propose AdaScale to determine the mode of the $(m+1)$-th scale, which is called the transitional scale. By default, the $(m+1)$-th scale of each frame is encoded in lossy mode. This approach allows the frame to be compressed in either lossless or lossy mode. When set to lossless, it retains more geometry details, resulting in a high-quality frame (HF) that serves as a superior reference for subsequent frames. Conversely, in lossy mode, it adheres to the same quantization level as other lossy scales.

To this end, we devise a motion-aware solution: if a frame exhibits significant motion relative to its preceding frame, it is designated as a high-quality frame (HF), with the transitional scale $m+1$ encoded in lossless mode. Otherwise, the transitional scale is encoded in lossy mode. The motion is quantified by the key point displacement D between the current frame and its previous HF frame:

$$D = \frac{1}{H}\sum_{i=1}^{H} \|p_{key\_cur}^{i} - p_{key\_hf}^{i}\|_2, \tag{7}$$

where $p_{key\_cur}^{i}$ represents the key points in the current frame, $p_{key\_hf}^{i}$ represents the key points in the previous HF, and H is the number of key points sampled.

A decision is then made based on the value of D:

$$\begin{cases} \text{if } D > T, & \text{lossless mode} \\ \text{otherwise,} & \text{lossy mode} \end{cases}, \tag{8}$$

where T is a predefined threshold, set to 90 in this work after extensive experiments.

B. Reference Selection

By adjusting the lossy and lossless modes of the transitional scale $m+1$, we can achieve reference frames with varying qualities. The next step involves selecting the most appropriate reference for each scale to optimize performance while maximizing the use of HF reconstructions in lossy mode whenever possible. As illustrated in Fig. 5, when coding the current frame, for the lossless scale, the corresponding scale in the immediately preceding frame is used as the reference to reduce bitrate. For the lossy scale, the corresponding scale in

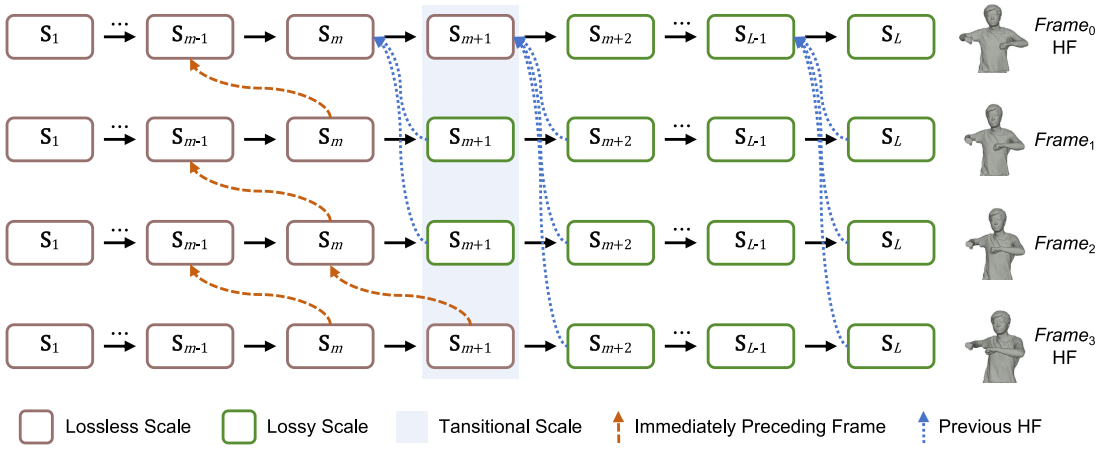


Fig. 5. AdaScale. The transitional scale $m+1$ of a point cloud frame is adjusted to either lossy or lossless modes based on frame displacement. If a frame exhibits significant motion, the $(m+1)$-th scale is set to lossless mode, creating a high-quality frame (HF) that serves as a superior reference for the next frames. Otherwise, the $(m+1)$-th scale is set to lossy mode. Reference selection enables lossless scales to directly reference their temporally corresponding scale in the immediately previous frame, while lossy scales reference their corresponding scale in the most recent HF.

the previous HF frame, which offers higher quality, is chosen to enhance reconstruction fidelity.

For example, in Fig. 5, $Frame_0$ is identified as an HF and its transitional scale $\mathbf{S}_{m+1}$ is losslessly coded (indicated by the brown rectangle). When coding $Frame_2$, our predefined rules in Eqs. (7) and (8) determine that $Frame_2$ is not an HF, and thus its transitional scale $\mathbf{S}_{m+1}$ is lossily coded (indicated by the green rectangle). Then, for lossless scales (except for the first one) in $Frame_2$, the reference is the corresponding scale in the immediately previous frame. For example, $\mathbf{S}_m$ in $Frame_2$ references $\mathbf{S}_{m-1}$ in $Frame_1$ for compression. For all lossy scales in $Frame_2$, *i.e.*, from $\mathbf{S}_{m+1}$ to $\mathbf{S}_L$, the reference is the corresponding scale in the most recent HF, *e.g.*, $\mathbf{S}_{m+1}$ in $Frame_2$ references $\mathbf{S}_m$ in $Frame_0$ for compression.

V. Experiments and Analysis

A. Experimental Settings

1) Datasets: We performed comprehensive experiments on point cloud datasets used by MPEG for PCC standardization. These datasets encompass both static and dynamic samples with varied content and geometry properties. We follow the guidelines commonly accepted by MPEG for preparing training and testing samples [54], [55]. Figure 6 illustrates some of the samples from our training and testing datasets.

Regarding the training datasets for static coding, we utilize 10-bit and 11-bit samples from the RWTT dataset [56]. This dataset originally includes 568 textured models created using various 3D reconstruction pipelines. In accordance with the common test conditions (CTC) [54] set by the MPEG AI-PCC group, only models with permissive licenses are selected for training. For 12-bit point clouds, we follow the CTC to use four samples for training: *Frog*, *Head*, *ULB Unicorn*, and *Egyptian mask*.

For the testing dataset in static coding, to ensure fair comparisons with existing G-PCC and Unicorn models, we utilize the datasets recommended by the G-PCC CTC [55]. Specifically, we employ a total of 13 samples: four 10-bit

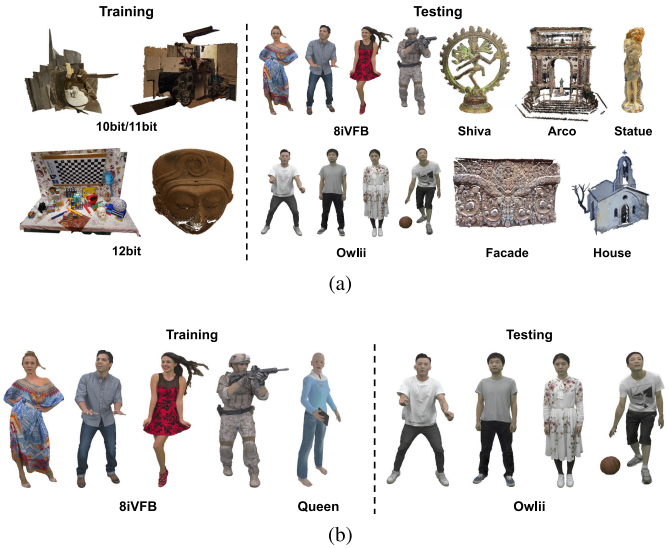


Fig. 6. Visualization of training and testing samples in (a) static coding and (b) dynamic coding.

samples from the 8i Voxelized Full Body (8iVFB) dataset [57], four 11-bit samples from the Owlii Dynamic Human Mesh dataset [58], and five 12-bit samples, including *facade*, *house*, *shiva*, *statue*, and *arco*.

Regarding the training dataset for dynamic coding, we employ four sequences from the 8iVFB dataset along with the *queen* sequence, strictly following the method recommended in the CTC of MPEG AI-PCC. The Owlii dataset [58] is utilized as the dynamic testing dataset. Both the training and testing sequences are quantized to 10-bit precision. To ensure a fair comparison with existing methods, we evaluate the first 100 frames of each sequence.

2) Settings: Our experiments are conducted on a platform equipped with an Intel Xeon 6230R CPU, 128GB of DDR4 memory, and an Nvidia GTX 3090 GPU. The initial learning rate is set to 1e-4 and is halved every epoch until it reaches

TABLE II
COMPRESSION PERFORMANCE COMPARISON ON STATIC LOSSLESS CODING

	Point Cloud	Bits Per Point (bpp)					Compression Gain (CR Gain)			
		G-PCC	GeS-TM	TMAP	Unicorn	Ours	vs. G-PCC	vs. GeS-TM	vs. TMAP	vs. Unicorn
vox 10	longdress	0.740	0.750	0.610	0.594	0.552	-25.39%	-26.38%	-9.51%	-7.07%
	loot	0.702	0.709	0.579	0.564	0.526	-25.07%	-25.81%	-9.15%	-6.74%
	redandblack	0.823	0.831	0.679	0.651	0.617	-25.03%	-25.75%	-9.13%	-6.11%
	soldier	0.737	0.743	0.601	0.588	0.550	-25.37%	-26.00%	-8.49%	-6.46%
	Average	**0.750**	**0.758**	**0.617**	**0.599**	**0.561**	**-25.22%**	**-25.99%**	**-9.07%**	**-6.35%**
vox 11	basketball_player	0.646	0.615	0.506	0.489	0.438	-32.29%	-28.78%	-13.44%	-10.49%
	dancer	0.629	0.607	0.503	0.484	0.435	-30.84%	-28.34%	-13.52%	-10.12%
	exercise	0.601	0.590	0.492	0.468	0.420	-30.12%	-28.81%	-14.63%	-10.26%
	model	0.576	0.570	0.506	0.476	0.435	-24.48%	-23.86%	-14.03%	-8.61%
	Average	**0.613**	**0.596**	**0.502**	**0.479**	**0.432**	**-29.43%**	**-27.45%**	**-13.91%**	**-9.87%**
vox 12	facade	5.907	6.560	7.493	6.094	5.218	-11.66%	-20.43%	-30.35%	-14.38%
	house	4.736	5.140	5.707	4.788	4.677	-1.25%	-9.00%	-18.03%	-2.32%
	shiva	9.157	9.600	10.910	9.083	8.835	-3.51%	-7.95%	-18.90%	-2.74%
	statue	9.062	9.718	11.107	9.052	8.591	-5.19%	-11.61%	-22.67%	-5.08%
	arco	9.934	10.021	12.498	9.958	9.818	-1.17%	-2.02%	-21.40%	-1.41%
	Average	**7.759**	**8.208**	**9.543**	**7.795**	**7.427**	**-4.55%**	**-10.22%**	**-22.27%**	**-5.18%**

1e-5. On average, model convergence takes approximately 40 hours on our platform. Similar to Unicorn, our training process is limited to two consecutive spatial scales, and the trained models are applied across all scales.

To train the lossless models, we minimize the Binary Cross-Entropy (BCE) loss, defined as follows:

$$\mathcal{L}_{\text{BCE}} = -\frac{1}{N}\sum_{i=1}^{N}\left(\hat{p}_i \log(p_i) + (1-\hat{p}_i)\log(1-p_i)\right), \quad (9)$$

where p_i is the ground-truth occupancy status (1 for an occupied point and 0 for a non-occupied point), and $\hat{p}_i$ is the predicted probability. N is the total points in this scale.

As for the feature-enhanced lossy coding mode, which incurs auxiliary bitrate, we also introduce a cross-entropy (CE) loss:

$$\mathcal{L}_{\text{CE}} = -\frac{1}{N}\sum_{j=1}^{\hat{N}} \log_2(p_{f_j}), \quad (10)$$

where p_{f_j} is the probability of features aggregated by stacking two KPA modules. $\hat{N}$ denotes the number of points used to generate the feature. Therefore, our total loss is formulated as $\mathcal{L}_{\text{CE}} + \lambda \cdot \mathcal{L}_{\text{BCE}}$, where λ controls the rate-distortion trade-off. Similar loss functions are used in Unicorn and TMAP.

3) Evaluation Metrics: The compressed bitrate is measured in bits per point (bpp), and distortion is assessed using Peak Signal-to-Noise Ratio (PSNR) for point-to-point error (D1) and point-to-plane error (D2). In lossless coding, the Compression Ratio (CR) gain is evaluated through relative bpp reduction. For lossy coding, the Bjøntegaard Delta Rate (BD-BR) [24] is used to assess coding efficiency between two compression methods.

We compare our approach with the latest reference models of MPEG standards, including V-PCC TMC2v25 (utilizing HM-16.20 by default),[3] G-PCC TMC13v23,[4] and GeS-TMv8.[5] Additionally, we compare with two leading learned methods, Unicorn[6] and TMAPv0.[7] For fair comparisons, we use the same training dataset for Unicorn, TMAP, and our method. All methods are tested using the same testing datasets on the same experimental platform.

[3]https://github.com/MPEGGroup/mpeg-pcc-tmc2
[4]https://github.com/MPEGGroup/mpeg-pcc-tmc13
[5]https://git.mpeg.expert/MPEG/3dgh/g-pcc/software/tm/mpeg-pcc-ges-tm
[6]https://github.com/NJUVISION/Unicorn
[7]https://git.mpeg.expert/MPEG/3dgh/ai-gc/software/mpeg-pcc-tmap

B. Compression Performance

1) Static Coding: Table II presents the compression performance of various methods in lossless mode. Our method consistently outperforms others across all bit depths. For example, compared to the latest G-PCC (TMC13v23), it achieves average CR gains of -25.22%, -29.43%, and -4.55% on voxel 10, 11, and 12-bit point clouds, respectively. Furthermore, when compared to Unicorn, the state-of-the-art approach, our method achieves gains ranging from 5.18% to 9.87%, highlighting its effectiveness.

Table III reports the BD-BR gains over compared solutions in lossy mode. Clearly, our method uniformly outperforms others. Compared to TMAP, our method attains 12.69% (23.12%), 37.00% (33.16%), and 40.69% (36.84%) BD-BR reduction in D1 (D2) measurements on voxel 10, 11, and 12-bit point clouds, respectively. Similarly, our method receives 11.73% (10.32%) to 19.47% (19.41%) BD-BR gains over Unicorn. Notably, for the 12-bit point cloud *house*, our method yields only a 0.70% BD-BR improvement in D2 over Unicorn. This is likely because *house* contains as many as 4,848,745 points, requiring it to be sliced into 16 chunks for coding on our experimental platform, whereas other 12-bit samples having far fewer points are typically split into only 2 to 4 chunks. Recall that KPA is designed to capture long-range and global structural dependencies. When a point cloud is partitioned into many small chunks, the available global context is substantially reduced, thereby limiting the effectiveness of KPA. Corresponding R-D curves in Fig. 7 further validate the consistent performance improvements across a wide bitrate range, confirming the effectiveness of our proposed method.

2) Dynamic Coding: We also compare the lossless compression of different methods in Table IV. In line with the lossless results for static coding, our method outperforms

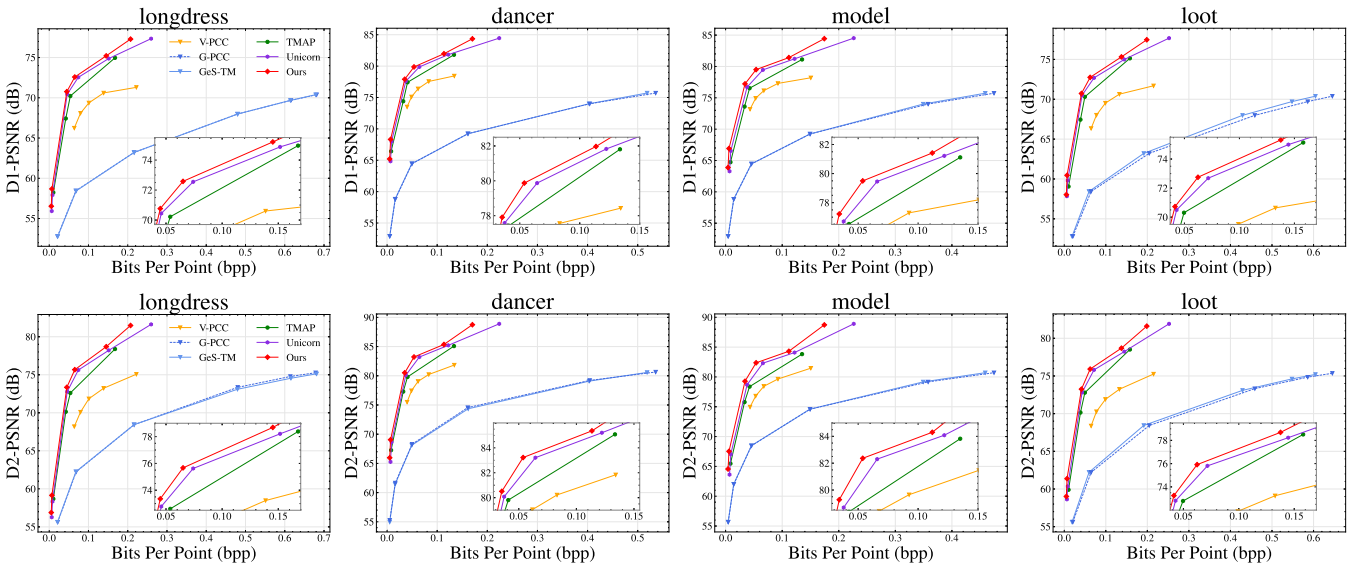


Fig. 7. R-D Curves for static lossy coding. We specifically zoom in on part of the curves for better illustration.

TABLE III
BD-BR GAINS AGAINST VARIOUS ANCHORS FOR STATIC LOSSY CODING

	Point Cloud	vs. V-PCC		vs. G-PCC		vs. GeS-TM		vs. TMAP		vs. Unicorn	
		D1	D2	D1	D2	D1	D2	D1	D2	D1	D2
vox 10	longdress	-66.84%	-66.89%	-93.11%	-87.73%	-92.88%	-87.67%	-37.00%	-33.50%	-16.11%	-17.06%
	loot	-69.40%	-68.56%	-94.30%	-89.73%	-93.89%	-87.65%	-39.83%	-36.36%	-16.89%	-19.37%
	redandblack	-73.05%	-72.58%	-93.09%	-87.40%	-91.51%	-84.54%	-34.73%	-29.37%	-13.24%	-12.82%
	soldier	-73.18%	-73.14%	-93.63%	-88.95%	-95.54%	-92.31%	-36.45%	-33.42%	-11.03%	-12.82%
	Average	**-70.62%**	**-70.29%**	**-93.53%**	**-88.45%**	**-93.45%**	**-88.04%**	**-37.00%**	**-33.16%**	**-14.32%**	**-15.52%**
vox 11	basketball_player	-67.50%	-65.12%	-95.60%	-91.69%	-95.54%	-91.61%	-36.87%	-32.52%	-17.22%	-17.63%
	dancer	-68.13%	-65.54%	-94.98%	-90.23%	-95.04%	-90.52%	-36.87%	-32.52%	-17.88%	-17.70%
	exercise	-63.82%	-62.07%	-95.29%	-90.80%	-95.23%	-90.71%	-43.91%	-40.09%	-17.11%	-17.28%
	model	-60.88%	-59.39%	-93.82%	-87.66%	-93.66%	-87.56%	-45.11%	-42.23%	-25.68%	-25.03%
	Average	**-65.08%**	**-63.03%**	**-94.92%**	**-90.10%**	**-94.87%**	**-90.10%**	**-40.69%**	**-36.84%**	**-19.47%**	**-19.41%**
vox 12	facade	-18.55%	-52.18%	-56.47%	-65.09%	-56.76%	-65.37%	-7.55%	-36.22%	-23.10%	-16.10%
	house	-14.68%	-45.55%	-63.80%	-69.58%	-64.78%	-74.32%	-8.71%	-30.00%	-11.80%	-0.70%
	shiva	-47.56%	-75.78%	-43.30%	-45.58%	-43.91%	-46.11%	-3.21%	-4.70%	-11.80%	-17.40%
	statue	-58.28%	-63.39%	-37.50%	-48.75%	-38.89%	-46.67%	-21.67%	-21.78%	-4.70%	-4.70%
	arco	-71.62%	-81.69%	-24.78%	-25.57%	-24.16%	-25.28%	-22.30%	-22.90%	-7.26%	-12.68%
	Average	**-42.14%**	**-63.72%**	**-45.17%**	**-50.92%**	**-45.70%**	**-51.55%**	**-12.69%**	**-23.12%**	**-11.73%**	**-10.32%**

TABLE IV
COMPRESSION PERFORMANCE COMPARISON ON DYNAMIC LOSSLESS CODING

	Point Cloud	Bits Per Point (bpp)					Compression Gain (CR Gain)			
		G-PCC	GeS-TM	TMAP	Unicorn	Ours	vs. G-PCC	vs. GeS-TM	vs. TMAP	vs. Unicorn
dynamic	basketball_player	0.593	0.589	0.401	0.401	0.366	-38.28%	-37.86%	-8.73%	-8.73%
	dancer	0.607	0.598	0.425	0.424	0.393	-35.25%	-34.28%	-7.53%	-7.31%
	exercise	0.586	0.572	0.401	0.387	0.363	-38.09%	-36.54%	-9.48%	-6.20%
	model	0.583	0.563	0.421	0.404	0.382	-34.48%	-32.15%	-9.26%	-5.45%
	Average	**0.592**	**0.581**	**0.412**	**0.404**	**0.376**	**-36.53%**	**-35.21%**	**-8.75%**	**-6.92%**

others across all testing sequences. For example, compared to the latest GeS-TMv8, our method achieves 35.21% gains on average on four testing sequences.

The results for dynamic coding in lossy mode are shown in Table V. Corresponding R-D curves are illustrated in Fig. 8. We observe that our method rivals V-PCC with gains of 91.04% (89.21%) in D1 (D2) measurements. The gains against the learning-based methods TMAP and Unicorn are also remarkable, reaching as high as 37.55% (30.28%) and 24.04% (20.23%), respectively.

C. Computational Complexity

The encoding and decoding times of all methods are presented in Fig. 9. It's important to note that these runtimes

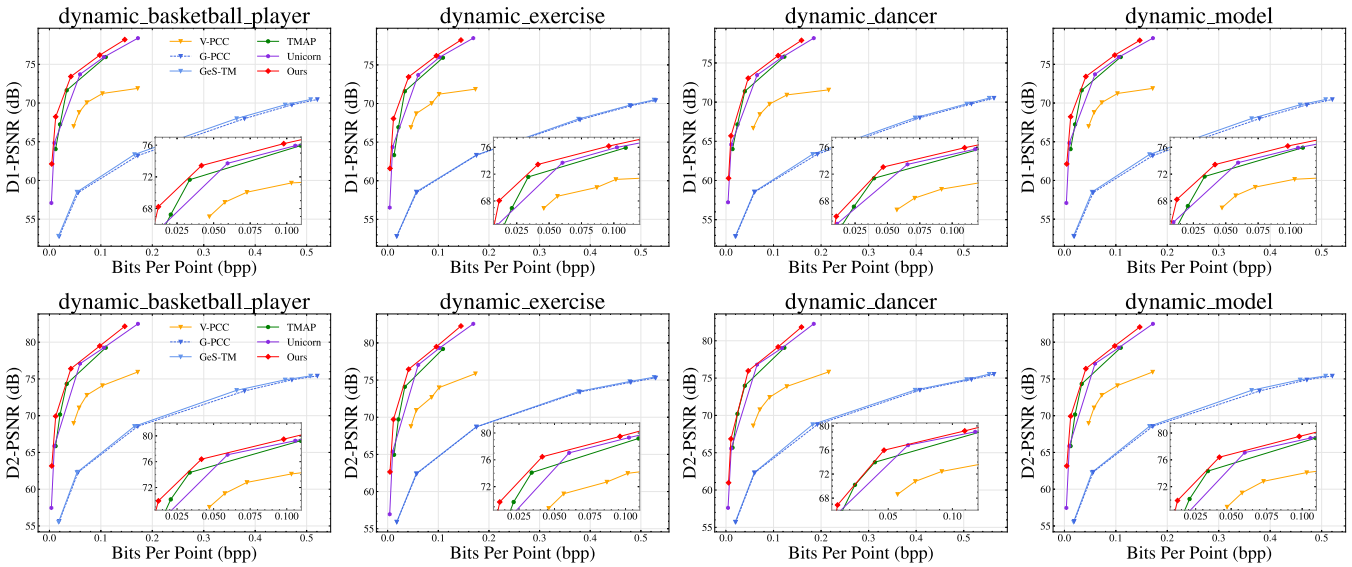


Fig. 8. R-D Curves for dynamic lossy coding. We specifically zoom in on part of the curves for better illustration.

TABLE V
BD-BR GAINS AGAINST VARIOUS ANCHORS FOR DYNAMIC LOSSY CODING

	Point Cloud	vs. V-PCC		vs. G-PCC		vs. GeS-TM		vs. TMAP		vs. Unicorn	
		D1	D2	D1	D2	D1	D2	D1	D2	D1	D2
dynamic	basketball_player	-90.26%	-87.96%	-97.01%	-94.24%	-96.73%	-94.20%	-35.76%	-25.85%	-25.82%	-17.95%
	dancer	-91.44%	-89.06%	-96.55%	-96.12%	-95.79%	-91.87%	-30.76%	-24.75%	-18.59%	-16.12%
	exercise	-90.83%	-89.17%	-97.32%	-94.51%	-96.87%	-93.96%	-40.85%	-37.01%	-29.71%	-27.58%
	model	-91.41%	-90.65%	-95.47%	-90.78%	-95.10%	-89.83%	-42.83%	-33.49%	-22.02%	-19.27%
	Average	**-91.04%**	**-89.21%**	**-96.59%**	**-93.91%**	**-96.12%**	**-92.47%**	**-37.55%**	**-30.28%**	**-24.04%**	**-20.23%**

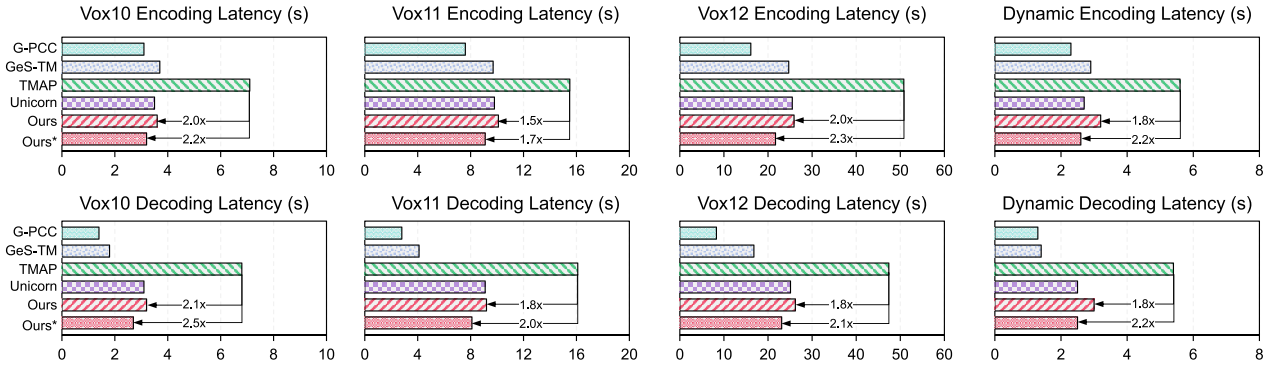


Fig. 9. Runtime Comparison. Average encoding and decoding time (seconds/frame) of different methods. Ours* is a lightweight version of our model.

are provided for intuitive comparison only, as the methods are implemented in different programming languages and executed on different hardware. For example, V-PCC, G-PCC, and GeS-TM are implemented in C/C++ and run on the CPU of our platform, whereas Unicorn, TMAP, and our method are mainly implemented in Python and run on the GPU. As seen, our approach is significantly faster than TMAP, and comparable to the Unicorn (even though we use the attention mechanism). A lightweight version of our model further accelerates the processing speed using a much smaller model with negligible performance loss (see Table X).

Moreover, we collect the model size and running memory of all methods in Table VI. For lossless coding, our model size is comparable to that of Unicorn for 10 and 11-bit point cloud samples, and it is smaller than TMAP's; however, for 12-bit samples, the model sizes of both our method and Unicorn are larger than TMAP's due to the use of a larger kernel size (set as 5). For lossy coding, our model size is larger because of the use of additional KPAs in lossy scales. In contrast, TMAP uses smaller kernel sizes and a point-based processing unit, resulting in a smaller model size.

Meanwhile, we compare our model with TMAP and Unicorn regarding the maximum running memory. As reported in Table VI, the running memory of our model is comparable to that of Unicorn but slightly lower than that of TMAP. Specifically, although our model employs a larger model size

TABLE VI
MODEL SIZE (MB) AND RUNNING MEMORY (GB) COMPARISON WITH TMAP AND UNICORN

Point Cloud	Lossless Model Size			Lossy Model Size			Running Memory		
	TMAP	Unicorn	Ours	TMAP	Unicorn	Ours	TMAP	Unicorn	Ours
vox10/vox11	8.5	4.9	5.6	8.5	11.6	16.1	7.9	7.4	7.6
vox12	7.1	21.8	22.2	7.1	47.2	52.4	9.2	8.2	9.2
dynamic	13.3	8.7	8.3	13.3	22.8	24.4	5.4	4.5	4.5

TABLE VII
IMPACT OF THE NUMBER OF KEY POINTS IN KPA

Point Cloud	Proportion of key points				
	0.2%	0.5%	1%	2%	4%
longdress	0.563	0.560	0.557	0.552	0.555
loot	0.534	0.532	0.531	0.526	0.529
redandblack	0.622	0.621	0.619	0.617	0.618
soldier	0.558	0.556	0.554	0.550	0.552
Average	**0.569**	**0.567**	**0.565**	**0.561**	**0.563**

TABLE VIII
IMPACT OF KTA POSITION: APPLYING KTA FOR EVERY STAGE VS. APPLYING KTA BEFORE ALL STAGES

Point Cloud	KTA before every stage			KTA before all stages		
	bpp	Enc (s)	Dec (s)	bpp	Enc (s)	Dec (s)
longdress	0.539	5.3	5.4	0.552	3.3	3.2
loot	0.513	5.3	5.0	0.526	3.3	2.8
redandblack	0.609	5.2	4.9	0.617	3.1	2.8
soldier	0.542	6.1	5.7	0.550	4.5	3.9
Average	**0.551**	**5.5**	**5.3**	**0.561**	**3.6**	**3.2**

in 12-bit samples, its memory usage remains comparable to that of TMAP.

D. Ablation Study

We perform a series of ablation studies to further understand the capacity of the proposed algorithm.

1) The Number of Key Points: We adjust the number of key points to examine its impact. Results are shown in Table VII. Generally, increasing the number of key points enhances the ability to capture more detailed information. For example, increasing the key point proportion from 0.2% to 2% in a scale leads to a bitrate reduction in lossless mode (from 0.569 bpp to 0.561 bpp, about 1.4% CR gains), indicating improved compression efficiency. However, further increasing the proportion to 4% does not result in additional gains and even causes a slight loss (from 0.561 bpp to 0.563 bpp). This is because the correlated points have already been sufficiently captured; incorporating additional points merely introduces redundancy. Based on these observations, we choose to use 2% key points per scale for KPA processing.

2) KPA Position: Currently, we apply KTA on the spatially lower-scale tensor, *e.g.*, scale $l-1$, before it is upscaled dyadically to generate priors for subsequent multistage occupancy prediction (*e.g.*, the eight stages used in TMAP and Unicorn). Clearly, KTA can also be implemented at each stage for more precise prediction, although this would result in a larger model size and longer processing time.

TABLE IX
IMPACT OF KPA AND ADASCALE ON BD-BR GAINS OF DYNAMIC LOSSY CODING

KPA	AdaScale	D1	D2
✗	✗	–	–
✗	✓	-12.39%	-10.78%
✓	✗	-11.84%	-13.16%
✓	✓	-24.04%	-20.23%

We conduct this ablation study for static coding in lossless mode using samples from the 8iVFB dataset, with results shown in Table VIII. It is observed that applying KTA before each of the eight stages achieves a lower bpp (0.551), leading to approximately 1.78% CR gains. However, this improvement comes at the cost of significantly increased encoding and decoding times—over 35% higher compared to using a single KTA before all eight stages. Similarly, the model size increases from 5.6 MB to 7.2 MB due to the substantial rise in the number of points after upscaling.

3) Serialization Method: In our implementation, we use the Hilbert curve in KPA for serialization. We also investigate the use of other serialization methods, such as Z-order and raster scan. It is shown in Table XI that the Hilbert curve performs the best, which is consistent with the conclusion in [53].

4) KPA vs. AdaScale: In the lossy mode of dynamic coding, both KPA and AdaScale improve occupancy prediction. Here, we examine their individual contributions.

The experimental results are presented in Table IX, with the corresponding R-D curves shown in Fig. 10. Disabling both techniques reduces the method to Unicorn. Using Unicorn as the baseline, incorporating AdaScale achieves an additional 12.39% (10.78%) BD-BR improvement in the D1 (D2) metrics, while using KPA achieves improvements of 11.84% (13.16%). These results confirm the effectiveness of both methods. When both are combined, our approach surpasses Unicorn by 24.04% (20.23%), demonstrating that KPA and AdaScale complement each other.

The R-D curves in Fig. 10 illustrate that our KPA is effective across a wide range of bitrates. On the other hand, the proposed AdaScale is particularly advantageous in low-bitrate scenarios, where the D1 (D2) PSNR is significantly improved at the same bitrate. At high bitrates, the gains become marginal, as the decoded frames are already of high quality, diminishing the advantage of AdaScale.

5) Lightweight Version: We develop a lightweight version by reducing the number of channels in our full model from 32 to 16. As shown in Table X, this lightweight version, termed Ours*, incurs only a 2.31% to 3.34% CR loss compared

TABLE X
COMPLEXITY AND PERFORMANCE OF OUR LIGHTWEIGHT VERSION (OURS*)

Point Cloud	Bits Per Point (bpp)			CR Gain			Model Complexity (Light vs. Full)			
	TMAP	Unicorn	Ours*	vs. TMAP	vs. Unicorn	vs. Full	Model Size	Enc (s)	Dec (s)	Memory (GB)
vox10	0.617	0.599	0.575	-6.81%	-4.01%	+2.43%	1.5 (-73.2%)	3.2 (-11.1%)	2.7 (-15.6%)	5.7 (-8.1%)
vox11	0.502	0.479	0.442	-11.96%	-7.72%	+2.31%	1.5 (-73.2%)	9.1 (-9.9%)	8.1 (-12.0%)	7.1 (-6.6%)
vox12	9.543	7.795	7.653	-19.81%	-1.83%	+2.95%	5.6 (-74.8%)	21.7 (-16.2%)	23.1 (-11.8%)	8.0 (-13.4%)
dynamic	0.412	0.404	0.389	-5.58%	-3.71%	+3.34%	2.2 (-73.5%)	2.7 (-18.1%)	2.5 (-16.7%)	4.3 (-4.5%)

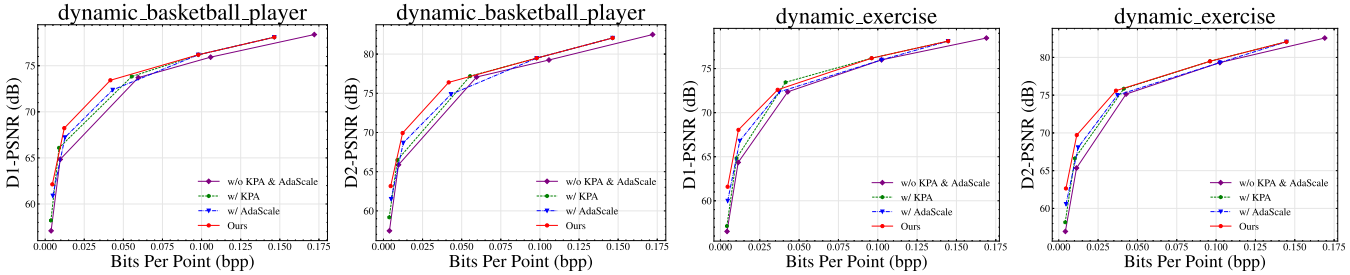


Fig. 10. R-D Curves illustrated by examining the contributions of KPA and AdaScale in the lossy mode of dynamic coding.

TABLE XI
IMPACT OF SERIALIZATION METHODS IN KPA

Point Cloud	Bits Per Point (bpp)		
	Morton	Raster Scan	Hilbert
longdress	0.555	0.558	0.552
loot	0.529	0.531	0.526
redandblack	0.621	0.622	0.617
soldier	0.553	0.555	0.550
Average	**0.565**	**0.567**	**0.561**

to the full model in lossless coding mode, while achieving over a 73% reduction in model size. Still, this lightweight model significantly outperforms existing methods with a much smaller model size. For example, in static coding, it achieves 6.81% and 11.96% CR gains over TMAP on 10 and 11-bit samples, respectively, with a model size of just 1.5 MB compared to TMAP's 8.5 MB (18% of TMAP). In dynamic coding, it delivers 5.58% CR gains with a model size of only 2.2 MB, versus TMAP's 13.3 MB (17% of TMAP).

VI. CONCLUSION

This work enhances occupancy prediction for multiscale point cloud geometry compression by introducing the Key Point-driven Attention (KPA) and Adaptive (Lossy/Lossless) Scale (AdaScale) mechanisms. KPA aggregates local and global characteristics to better utilize the input lower-scale reference for conditional modeling, while AdaScale aims to improve the quality of this reference. Together, they generate better conditional priors for more accurate occupancy prediction, thereby boosting compression efficiency. Experimental results on various datasets demonstrate that our method significantly outperforms previous works, including rules-based and learning-based solutions, achieving state-of-the-art compression performance without requiring additional resources.

Currently, the proposed AdaScale is applied heuristically and exclusively at the transitional scale for dynamic coding. An interesting area of exploration is developing a reinforcement learning approach to guide flexible AdaScale decisions in both static and dynamic coding scenarios. Extending the proposed techniques to sparse LiDAR scans is another potential direction to explore.

REFERENCES

[1] Y. Guo, H. Wang, Q. Hu, H. Liu, L. Liu, and M. Bennamoun, "Deep learning for 3D point clouds: A survey," *IEEE Trans. Pattern Anal. Mach. Intell.*, vol. 43, no. 12, pp. 4338–4364, Dec. 2021.

[2] Q. Zhang, J. Hou, Y. Qian, Y. Zeng, J. Zhang, and Y. He, "Flattening-net: Deep regular 2D representation for 3D point cloud analysis," *IEEE Trans. Pattern Anal. Mach. Intell.*, vol. 45, no. 8, pp. 9726–9742, Aug. 2023.

[3] S. Ren, J. Hou, X. Chen, H. Xiong, and W. Wang, "DDM: A metric for comparing 3D shapes using directional distance fields," *IEEE Trans. Pattern Anal. Mach. Intell.*, vol. 47, no. 8, pp. 6631–6646, Aug. 2025.

[4] M. Quach, J. Pang, D. Tian, G. Valenzise, and F. Dufaux, "Survey on deep learning-based point cloud compression," *Frontiers Signal Process.*, vol. 2, Feb. 2022, Art. no. 846972.

[5] H. Liu, H. Yuan, Q. Liu, J. Hou, and J. Liu, "A comprehensive study and comparison of core technologies for MPEG 3-D point cloud compression," *IEEE Trans. Broadcast.*, vol. 66, no. 3, pp. 701–717, Sep. 2020.

[6] C. Cao, M. Preda, and T. B. Zaharia, "3D point cloud compression: A survey," in *Proc. Int. Conf. 3D Web Technol.*, 2019, pp. 1–9.

[7] C. Cao, M. Preda, V. Zakharchenko, E. S. Jang, and T. Zaharia, "Compression of sparse and dense dynamic point clouds—Methods and standards," *Proc. IEEE*, vol. 109, no. 9, pp. 1537–1558, Sep. 2021.

[8] Information Technology-Coded Representation of Immersive Media—Part 5: Visual Volumetric Video-based Coding (V3C) and Video-based Point Cloud Compression (V-PCC), Standard ISO/IEC 23090-5, 2023, p. 5.

[9] Information Technology-Coded Representation of Immersive Media Part 9: Geometry-based Point Cloud Compression, Standard ISO/IEC 23090-9, 2023, p. 9.

[10] *Test Model for Geometry-Based Solid Point Cloud-GeSTM V8.0*, document N01030, 2025.

[11] *TMAP V0 for AI-based Point Cloud Coding, document N01060, WG 07 MPEG 3D Graphics Coding and Haptics Coding*, 2024.

[12] J. Wang, R. Xue, J. Li, D. Ding, Y. Lin, and Z. Ma, "A versatile point cloud compressor using universal multiscale conditional coding—Part I: Geometry," *IEEE Trans. Pattern Anal. Mach. Intell.*, vol. 47, no. 1, pp. 269–287, Jan. 2024.

[13] D. Graziosi, O. Nakagami, S. Kuma, A. Zaghetto, T. Suzuki, and A. Tabatabai, "An overview of ongoing point cloud compression standardization activities: Video-based (V-PCC) and geometry-based (G-PCC)," *APSIPA Trans. Signal Inf. Process.*, vol. 9, no. 1, p. 13, 2020.

[14] G. J. Sullivan, J.-R. Ohm, W.-J. Han, and T. Wiegand, "Overview of the high efficiency video coding (HEVC) standard," *IEEE Trans. Circuits Syst. Video Technol.*, vol. 22, no. 12, pp. 1649–1668, Dec. 2012.

[15] B. Bross et al., "Overview of the versatile video coding (VVC) standard and its applications," *IEEE Trans. Circuits Syst. Video Technol.*, vol. 31, no. 10, pp. 3736–3764, Oct. 2021.

[16] J. Zhang, T. Chen, D. Ding, and Z. Ma, "YOGA: Yet another geometry-based point cloud compressor," in *Proc. 31st ACM Int. Conf. Multimedia*, 2023, pp. 9070–9081.

[17] W. Zhu, Y. Xu, D. Ding, Z. Ma, and M. Nilsson, "Lossy point cloud geometry compression via region-wise processing," *IEEE Trans. Circuits Syst. Video Technol.*, vol. 31, no. 12, pp. 4575–4589, Dec. 2021.

[18] Y. Zeng, J. Hou, Q. Zhang, S. Ren, and W. Wang, "Dynamic 3D point cloud sequences as 2D videos," *IEEE Trans. Pattern Anal. Mach. Intell.*, vol. 46, no. 12, pp. 9371–9386, Dec. 2024.

[19] J. Wang, R. Xue, J. Li, D. Ding, Y. Lin, and Z. Ma, "A versatile point cloud compressor using universal multiscale conditional coding—Part II: Attribute," *IEEE Trans. Pattern Anal. Mach. Intell.*, vol. 47, no. 1, pp. 252–268, Jan. 2024.

[20] J. Wang, D. Ding, Z. Li, and Z. Ma, "Multiscale point cloud geometry compression," in *Proc. Data Compress. Conf. (DCC)*, Jun. 2021, pp. 73–82.

[21] L. Xie, W. Gao, and H. Zheng, "End-to-end point cloud geometry compression and analysis with sparse tensor," in *Proc. 1st Int. Workshop Adv. Point Cloud Compress.*, 2022, pp. 27–32.

[22] J. Wang, D. Ding, Z. Li, X. Feng, C. Cao, and Z. Ma, "Sparse tensor-based multiscale representation for point cloud geometry compression," *IEEE Trans. Pattern Anal. Mach. Intell.*, vol. 45, no. 7, pp. 9055–9071, Jul. 2022.

[23] K. Wang and W. Gao, "UniPCGC: Towards practical point cloud geometry compression via an efficient unified approach," in *Proc. AAAI Conf. Artif. Intell.*, vol. 39, 2025, pp. 12721–12729.

[24] G. Bjøntegaard, *Calculation of Average PSNR Differences Between RD-Curves*, Standard ITU-T SG 16/Q6, Apr. 2001.

[25] H. Schwarz, D. Marpe, and T. Wiegand, "Analysis of hierarchical B pictures and MCTF," in *Proc. IEEE Int. Conf. Multimedia Expo*, Jul. 2006, pp. 1929–1932.

[26] *WD 2.0 for the Conformance and Reference Software of L3C2, Standard ISO/IEC JTC 1/SC 29/WG 11, MPEG-I WG 07 MPEG 3D Graphics Coding and Haptics Coding*, Jan. 2025.

[27] R. Charles, H. Su, K. Mo, and L. Guibas, "PointNet: Deep learning on point sets for 3D classification and segmentation," in *Proc. IEEE Conf. Comput. Vis. Pattern Recognit.*, Jun. 2017, pp. 77–85.

[28] C. R. Qi, L. Yi, H. Su, and L. Guibas, "PointNet+$: Deep hierarchical feature learning on point sets in a metric space," in *Proc. Adv. Neural Inf. Process. Syst.*, 2017, pp. 5105–5114.

[29] J. Zhang, G. Liu, D. Ding, and Z. Ma, "Transformer and upsampling-based point cloud compression," in *Proc. 1st Int. Workshop Adv. Point Cloud Compress.*, 2022, pp. 33–39.

[30] X. Wen, X. Wang, J. Hou, L. Ma, Y. Zhou, and J. Jiang, "Lossy geometry compression of 3D point cloud data via an adaptive octree-guided network," in *Proc. IEEE Int. Conf. Multimedia Expo (ICME)*, Jun. 2020, pp. 1–6.

[31] K. You, P. Gao, and Q. Li, "IPDAE: Improved patch-based deep autoencoder for lossy point cloud geometry compression," in *Proc. 1st Int. Workshop Adv. Point Cloud Compress., Process. Anal.*, Oct. 2022, pp. 1–10.

[32] L. Gao, T. Fan, J. Wan, Y. Xu, J. Sun, and Z. Ma, "Point cloud geometry compression via neural graph sampling," in *Proc. IEEE Int. Conf. Image Process. (ICIP)*, Aug. 2021, pp. 3373–3377.

[33] M. Quach, G. Valenzise, and F. Dufaux, "Learning convolutional transforms for lossy point cloud geometry compression," in *Proc. IEEE Int. Conf. Image Process. (ICIP)*, Sep. 2019, pp. 4320–4324.

[34] A. F. R. Guarda, N. M. M. Rodrigues, and F. Pereira, "Adaptive deep learning-based point cloud geometry coding," *IEEE J. Sel. Topics Signal Process.*, vol. 15, no. 2, pp. 415–430, Feb. 2021.

[35] J. Wang, H. Zhu, H. Liu, and Z. Ma, "Lossy point cloud geometry compression via end-to-end learning," *IEEE Trans. Circuits Syst. Video Technol.*, vol. 31, no. 12, pp. 4909–4923,Dec. 2021.

[36] L. Huang, S. Wang, K. Wong, J. Liu, and R. Urtasun, "OctSqueeze: Octree-structured entropy model for LiDAR compression," in *Proc. IEEE/CVF Conf. Comput. Vis. Pattern Recognit.*, Jun. 2020, pp. 1310–1320.

[37] D. T. Nguyen, M. Quach, G. Valenzise, and P. Duhamel, "Lossless coding of point cloud geometry using a deep generative model," *IEEE Trans. Circuits Syst. Video Technol.*, vol. 31, no. 12, pp. 4617–4629, Dec. 2021.

[38] C. Fu, G. Li, R. Song, W. Gao, and S. Liu, "Octattention: Octree-based large-scale contexts model for point cloud compression," in *Proc. AAAI Conf. Artif. Intell.*, 2022, pp. 625–633.

[39] R. Song, C. Fu, S. Liu, and G. Li, "Efficient hierarchical entropy model for learned point cloud compression," in *Proc. IEEE/CVF Conf. Comput. Vis. Pattern Recognit. (CVPR)*, Vancouver, BC, Canada, Jun. 2023, pp. 14368–14377.

[40] M. Cui, J. Long, M. Feng, B. Li, and H. Kai, "OctFormer: Efficient octree-based transformer for point cloud compression with local enhancement," in *Proc. AAAI Conf. Artif. Intell.*, Jun. 2023, vol. 37, no. 1, pp. 470–478.

[41] D. T. Nguyen and A. Kaup, "Lossless point cloud geometry and attribute compression using a learned conditional probability model," *IEEE Trans. Circuits Syst. Video Technol.*, vol. 33, no. 8, pp. 4337–4348, Aug. 2023.

[42] Y. Jin, Z. Zhu, T. Xu, Y. Lin, and Y. Wang, "ECM-OPCC: Efficient context model for octree-based point cloud compression," in *Proc. IEEE Int. Conf. Acoust., Speech Signal Process. (ICASSP)*, Apr. 2024, pp. 7985–7989.

[43] S. Biswas, J. Liu, K. Wong, S. Wang, and R. Urtasun, "MuSCLE: Multi sweep compression of LiDAR using deep entropy models," in *Proc. Adv. Neural Inf. Process. Syst.*, vol. 33, 2020, pp. 22170–22181.

[44] Z. Que, G. Lu, and D. Xu, "VoxelContext-Net: An octree based framework for point cloud compression," in *Proc. IEEE/CVF Conf. Comput. Vis. Pattern Recognit. (CVPR)*, Jun. 2021, pp. 6042–6051.

[45] Z. Chen, Z. Qian, S. Wang, and Q. Chen, "Point cloud compression with sibling context and surface priors," in *Proc. Eur. Conf. Comput. Vis.*, Tel Aviv, Israel, 2022, pp. 744–759.

[46] G. Liu, J. Wang, D. Ding, and Z. Ma, "PCGFormer: Lossy point cloud geometry compression via local self-attention," in *Proc. IEEE Int. Conf. Vis. Commun. Image Process. (VCIP)*, Jan. 2023, pp. 1–5.

[47] T. Fan, L. Gao, Y. Xu, Z. Li, and D. Wang, "D-DPCC: Deep dynamic point cloud compression via 3D motion prediction," in *Proc. Thirty-First Int. Joint Conf. Artif. Intell.*, Jul. 2022, pp. 898–904.

[48] S. J. Xia, T. Fan, Y. Xu, J.-N. Hwang, and Z. Li, "Learning dynamic point cloud compression via hierarchical inter-frame block matching," in *Proc. ACM Multimedia*, 2023, pp. 7993–8003.

[49] A. Akhtar, Z. Li, and G. Van der Auwera, "Inter-frame compression for dynamic point cloud geometry coding," *IEEE Trans. Image Process.*, vol. 33, pp. 584–594, 2024.

[50] D. Hilbert, "Über die stetige abbildung einer linie auf ein Flächenstück," in *Dritter Band: Analysis. Grundlagen Der Mathematik. Physik Verschiedenes: Nebst Einer Lebensgeschichte*. Berlin, Germany: Springer, 1935, pp. 1–2, doi: 10.1007/978-3-662-38452-7_1.

[51] G. M. Morton, *A Computer Oriented Geodetic Data Base a New Technique File Sequencing*. Ottawa, Canada: International Business Machines, 1966.

[52] X. Wu et al., "Point transformer v3: Simpler, faster, stronger," in *Proc. IEEE/CVF Conf. Comput. Vis. Pattern Recognit. (CVPR)*, Jun. 2024, pp. 4840–4851.

[53] J. Chen, L. Yu, and W. Wang, "Hilbert space filling curve based scan-order for point cloud attribute compression," *IEEE Trans. Image Process.*, vol. 31, pp. 4609–4621, 2022.

[54] WG 07 MPEG 3D Graphics Coding and Haptics Coding, *CTC on AI-Based Point Cloud Coding*, document N01122, 2025.

[55] WG 07 MPEG 3D Graphics Coding and Haptics Coding, *Common Test Conditions for G-PCC*, document N00944, 2024.

[56] A. Maggiordomo, F. Ponchio, P. Cignoni, and M. Tarini, "Real-world textured things: A repository of textured models generated with modern photo-reconstruction tools," *Comput. Aided Geometric Design*, vol. 83, Nov. 2020, Art. no. 101943.

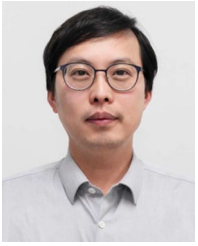
[57] E. D'Eon, B. Harrison, T. Myers, and P. A. Chou, *8i Voxelized Full Bodies—A Voxelized Point Cloud Dataset*, Standard WG11M40059/WG1M74006, 2017.

[58] X. Yi, L. Yao, and W. Ziyu, *Owlii Dynamic Human Mesh Sequence Dataset*, document m41658, Oct. 2017.

Zehong Li received the B.S. degree in computer science and technology from Hangzhou Normal University, Hangzhou, China, in 2024, where he is currently pursuing the M.S. degree in electronic information. His research interests include point cloud compression and processing.

Jiahao Zhu received the B.S. degree in electronic information engineering from Zhejiang University of Science and Technology, Hangzhou, China, in 2023. He is currently pursuing the M.S. degree in electronic information with Hangzhou Normal University. His research interests include point cloud compression and reconstruction.

Dandan Ding (Senior Member, IEEE) received the B.Eng. degree (Hons.) in communication engineering and the Ph.D. degree from Zhejiang University, China, in 2006 and June 2011, respectively. From 2007 to 2008, she was an Exchange Student at the Microelectronic Systems Laboratory (GR-LSM), EPFL, Switzerland. She was a Post-Doctoral Researcher from July 2011 to June 2013 and then a Research Associate with Zhejiang University till 2015. Since 2016, she has been with the School of Information Science and Technology, Hangzhou Normal University, as a tenured Faculty Member. Her research interests include artificial intelligence-based point cloud processing and image/video coding.



Zhan Ma (Senior Member, IEEE) received the B.S. and M.S. degrees from the Huazhong University of Science and Technology, Wuhan, China, in 2004 and 2006, respectively, and the Ph.D. degree from New York University, New York, in 2011. From 2011 to 2014, he was with Samsung Research America, Dallas, TX, USA, and Futurewei Technologies, Inc., Santa Clara, CA, USA. He is currently a Full Professor with the School of Electronic Science and Engineering, Nanjing University, Jiangsu, China. His research interests include learned image/video coding and computational imaging. He was a co-recipient of multiple awards, including the 2019 IEEE Broadcast Technology Society Best Paper Award, the 2020 IEEE MMSP Grand Challenge Best Image Coding Solution, and the 2023 IEEE WACV Best Algorithms Paper Award.