GAEM: Graph-driven Attention-based Entropy Model for LiDAR Point Cloud Compression

Mingyue Cui, *Member, IEEE,* Yuyang Zhong, Mingjian Feng, Junhua Long, Yehua Ling,

Jiahao Xu, and Kai Huang, *Member, IEEE*

***Abstract*—High-quality LiDAR point cloud (LPC) coding is essential for efficiently transmitting and storing the vast amounts of data required for accurate 3D environmental representation. The Octree-based entropy coding framework has emerged as the predominant method, however, previous study usually overly relies on large-scale attention-based context prediction to encode Octree nodes, overlooking the inherent correlational properties of this structure. In this paper, we propose a novel Graph-driven Attention-based Entropy Model (GAEM), which adopts partitioned graph attention mechanisms to uncover contextual dependencies among neighboring nodes. Different from the Cartesian coordinate-based coding mode with higher redundancy, GAEM uses the multi-level spherical Octree to organize point clouds, improving the quality of LPC reconstruction. GAEM combines graph convolution for node feature embedding and grouped-graph attention for exploiting dependency among contexts, which preserves performance in low-computation using localized nodes. Besides, to further increase the receptive field, we design a high-resolution cross-attention module introducing sibling nodes. Experimental results show that our method achieves state-of-the-art performance on the LiDAR benchmark SemanticKITTI and MPEG-specified dataset Ford, compared to all baselines. Compared to the benchmark GPCC, our method achieves gains of up to 53.9% and 53.6% on SemanticKITTI and Ford while compared to the sibling-introduced methods, we achieve up to 42.3% and 44.7% savings in encoding/decoding time. In particular, our GAEM allows for extension to downstream tasks (*i.e.,* vehicle detection and semantic segmentation), further demonstrating the practicality of the method.**

***Index Terms*—LiDAR point cloud compression, entropy estimation, Octree, graph-driven networks, 3D vision.**

I. Introduction

LIDAR is becoming a widely adopted sensor in various application areas, such as autonomous driving and robotics, benefiting from its high accuracy and resolution of 3D geometry information [1]–[6]. However, such a large amount of LiDAR point cloud (LPC) data brings challenges for storage and transmission. Taking only the single Velodyne LiDAR of HDL64 as an example, each scan generates 100,000 points, approximately 2.88 million points per second [7]–[10]. The disorder and sparsity of point clouds also increase the difficulty of solving this problem [11], [12].

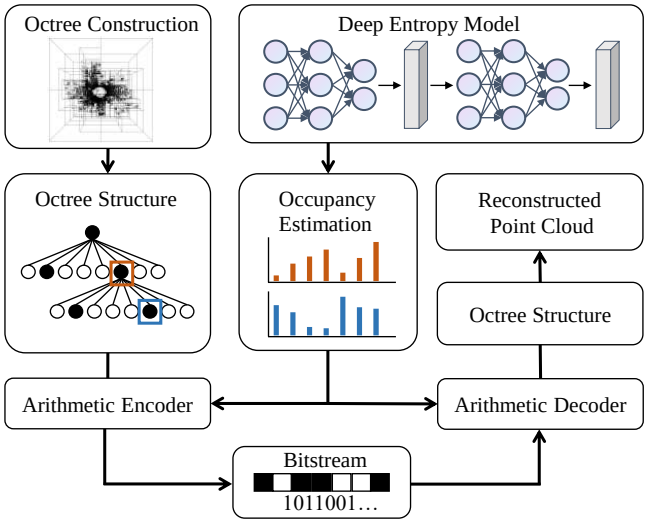


Fig. 1. The structure of octree-based entropy model for LPC compression. The LPC is organized by Octree to the node sequence, which is fed to the deep entropy model for occupancy distribution prediction. With the predicted probabilities, Octree is encoded into the bitstream and decoded from it to reconstruct the point cloud.

In recent years, some works have attempted to study efficient point cloud geometry compression, such as voxel-based, image-based, and tree-based algorithms. The MPEG group develops a standard LPC compression method GPCC [13]–[15], adapting a hand-crafted context-adaptive arithmetic encoder for bit allocation, which can be seen as a prediction for the currently encoding node based on the coded information. OctSqueeze [7] proposes the first entropy encoder based on deep learning by leveraging context formation about conditions on ancestor nodes, gradually becoming the predominant structure [16]–[20], as shown in Fig. 1. Some of these methods [21]–[24] organize point clouds into voxels and then use 3D convolution to predict each voxel's occupancy. However, these methods are not robust to point cloud resolution and only have a limited receptive field of neighborhood information in fixed-size voxels. At the same time, it brings a lot of computations due to introducing voxel encoding, especially for higher resolution. Compared to the above voxel-based

Manuscript received June 7, 2024; This work was supported in part by the National Key Research and Development Program of China under Grant 2023YFB4503703, in part by the Guangdong Basic and Applied Basic Research Foundation under Grant 2025A1515011485, and in part by the National Natural Science Foundation of China under Grant 62232008. (*Corresponding author: Kai Huang.*)

M. Cui, M. Feng, J. Long, J. Xu, and K. Huang are with the School of Computer Science and Engineering, Sun Yat-sen University, Guangzhou 400100, China. M. Cui is also with the key Laboratory of Machine Intelligence and Advanced Computing, Sun Yat-sen University, Guangzhou 400100, China. (email: cuimy5@mail.sysu.edu.cn)

Y. Zhong is with the College of Computing and Data Science, Nanyang Technological University, 639798, Singapore.

Y. Ling is with the Guangxi Transportation Science and Technology Group Co., Ltd, Nanning 530029, China.

methods, other works [17]–[20], [25]–[30] encode the point cloud into an Octree and then encode the occupancy of Octree nodes based on their ancestor nodes. Octree-based methods have attracted great attention due to their robust resolution and a broader receptive field. However, previous studies focus more on large-scale attention-based context prediction, overlooking the inherent correlational properties of Octree nodes. Overall, prior voxel-based and octree-based methods do not fully use much spatial context information, which provides an opportunity for reduction in the bitrate.

In this work, we propose a **G**raph-driven **A**ttention-based **E**ntropy **M**odel called **GAEM**, which is a multi-level spherical Octree-based LPC compression framework. GAEM combines graph convolution for node feature embedding and grouped-graph attention for exploiting dependency among contexts, which preserves performance in low-computation using localized nodes. Specifically, raw point clouds are firstly organized into multi-layer spherical Octree, which are the node contexts with similar structures. Then, we build parent and distance graph adjacency matrices for the contexts based on ancestral nodes and spatial information within contexts and use graph convolution to project nodes, which effectively introduces encoded information and original features of nodes. GAEM further adopts grouped-graph attention mechanisms to uncover contextual dependencies among neighboring nodes, and predicts the node occupancy distribution using the parent context with the encoded sibling contexts. The module divides the context into sub-windows based on edge weights and uses grouped-graph attention to share local dependencies in each sub-window. In addition, for global dependencies, we use a multi-layer perception (MLP) instead of self-attention for feature mapping. These two designs bring faster operations to our model. Finally, to further increase the receptive field, we design a cross-attention to obtain high-resolution information by introducing the encoded sibling context.

We compare our GAEM with state-of-the-art methods [14], [16]–[20], [31] on both LPC datasets (*i.e.,* LiDAR benchmark SemanticKITTI [32] and MPEG-specified dataset Ford [33]) and downstream tasks (*i.e.,* vehicle detection and semantic segmentation). The experiments show that our method outperforms these methods, which are only designed for a specific category of point clouds. To summarize, our contributions are highlighted as follows:

- We propose GAEM, a novel graph-driven attention point cloud entropy model that combines the graph convolution augmentation of node features and an improved grouped-graph attention module achieving accurate prediction for occupancy of multi-layer spherical Octree nodes.
- GAEM introduces grouped-graph self-attention to discover the correlation of sub-window nodes, which prevents traversing the former levels of multi-layer Octree repeatedly, thus speeding up encoding and decoding. Besides, graph adjacency matrices fully introduce encoded information and original features of nodes by using both ancestral nodes and spatial information within contexts.
- To further enlarge the receptive field of the model, we design a cross-attention module to provide higher-resolution information for the node features by introducing the encoded sibling contexts.

II. Related Work

A. Latent Representation-based Point Cloud Compression

Using structures like variational auto-encoders (VAE) to transform point clouds into latent representations is a typical method of point cloud compression. Latent Representation-based methods can be summarized into two categories, voxel-based methods, and point-based methods, depending on the way of organization of the point clouds. Voxel-based methods convert point clouds into 3D voxels. Benefiting from the 3D-voxel structures, such methods apply 3D convolution-based feature extraction on 3D voxels [34]–[42] to transform point cloud data into latent representations. Wang *et al.* [34] partition point cloud data into non-overlapping 3D voxels, which are then fed into stacked 3D convolutions for compact latent feature and hyperprior generation. They then use the hyperprior to improve the conditional probability modeling of latent features. Alexiou *et al.* [36] further explore neural network implementations for separate, or joint compression of geometric and textural information from point cloud contents. Extracting features from voxels achieves good performance. However, such methods are sensitive to density variations and introduce huge computation costs for sparse point clouds.

Different from voxel-based methods, point-based methods do not require reorganizing point clouds and directly extract features on point sets [43]–[45] using backbones like PointNet [46] and PointNet++ [47]. Huang *et al.* [43] compress the raw point cloud data into codewords by extracting features of the raw model with an encoder based on PointNet++ [47]. Then, the codeword is further compressed with sparse coding to achieve a higher compression ratio. To compress point clouds with larger sizes, Wen *et al.* [44] decomposes point cloud data into non-overlapping local patches through adaptive Octree decomposition and clustering. They propose a framework of a point cloud auto-encoder network with a quantization layer for learning compact latent feature representations from each patch. Point-based methods require less computation while preserving acceptable compression performance. However, point-based methods focus on global feature extraction and are hard to extend to large sparse point clouds. Overall, while these methods achieve satisfactory compression ratios, the latent representations often struggle to preserve the fine-grained details of point clouds. Therefore, the quality of reconstructed point clouds needs improvement and more controllability.

B. Entropy Model-based Point Cloud Compression

Entropy coding is a typical method of lossless compression, which assigns bits to symbols based on their probabilities, with a higher probability resulting in a lower bit allocation. In point cloud compression, GPCC [13], [14] adopts hand-crafted features for bit allocation and entropy coding, which utilizes the coded information to encode the current node. Nowadays, many works [7], [16]–[18], [27], [31] have combined the learning-based entropy model with the Octree structure, where the only loss comes from the voxelization procedure of Octree

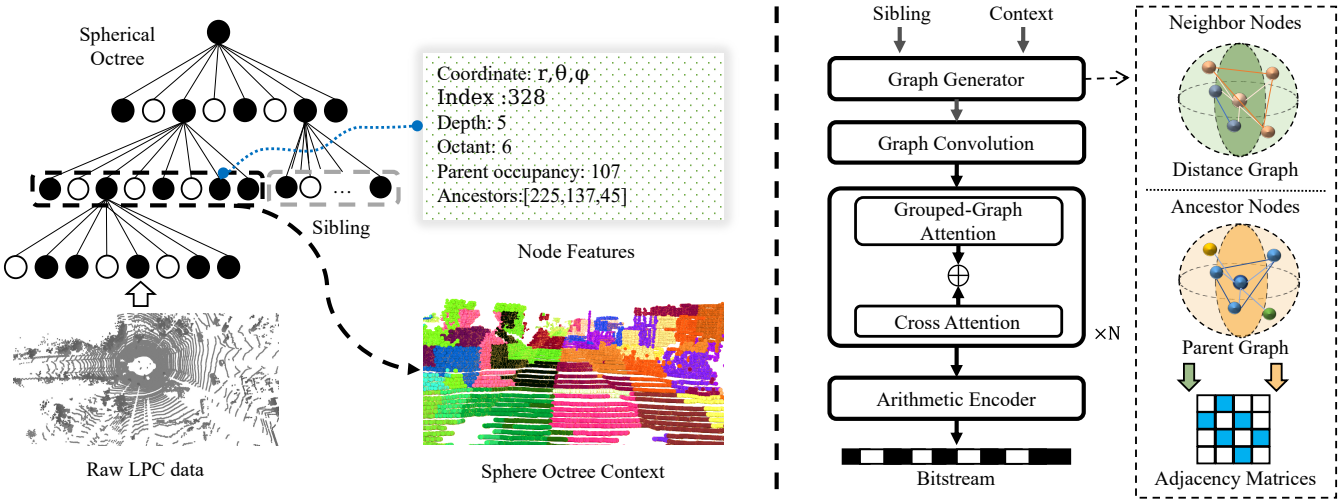


Fig. 2. The framework of GAEM. We first construct the spherical Octree from the raw point cloud by converting the Cartesian coordinates to spherical coordinates. The input features for each octree node include the spherical coordinates of the node center, index, depth, Octant, parent occupancy, and the indexes of the ancestor nodes. Then, we construct the graph for each Octree node context based on ancestors and distances and generate adjacency matrices individually. These matrices are further applied for graph convolution and grouped-graph attention. Finally, a cross-attention module is introduced in the grouped-graph attention block to improve the prediction granularity.

construction, as shown in Fig. 1. From the figure, we can observe that the key of the learning-based entropy model is to effectively extract context features from Octree for accurate prediction of Octree occupancies. OctSqueeze [7] firstly introduces deep neural networks to Octree entropy coding, which encodes the point cloud into an Octree and designs a tree-structured conditional entropy model to predict the probabilities of the Octree occupancies. MuSCLE [25] models the probabilities of the octree symbols by considering both coarse level geometry and previous sweeps' geometric and intensity information, which leverages spatio-temporal relationships across multiple LiDAR sweeps. Song *et al.* [26] propose a layer-wise geometry aggregation framework for lossless compression of LPCs, which adaptively segments the data into ground curves, object Octrees, and noise points, each processed with tailored algorithms for efficient representation. YOGA [48] employs a unified end-to-end learning-based backbone with multiscale sparse convolutions to compress geometry and attributes. It utilizes a two-layer structure where a downscaled thumbnail point cloud is encoded using GPCC at the base layer. Fan *et al.* [27] propose an end-to-end deep framework that hierarchically constructs the octree entropy model in layers, which utilizes a hierarchical latent variable as side information to encapsulate the sibling and ancestor dependence and encodes the latent variable by residual coding. However, these methods mainly focus on lossless point cloud compression that requires high-resolution Octree to organize the point cloud, which increases bitstream length and further causes a limitation to the ratio of bit-to-reconstruction quality.

Recently, VoxelContext-Net [31] introduces voxel features in their proposed method, which extracts the local voxel representation that encodes the spatial neighboring context information for each node in the constructed octree and proposes a voxel context-based deep entropy model to compress the symbols of non-leaf nodes. SparsePCGC [17] further uses sparse convolutions to deal with the unstructured points directly, efficiently aggregates the local neighborhood information, and proposes the resolution scaling to extensively enable multiscale representation to exploit the cross-scale correlation. Instead of high computation convolutions, OctAttention [18] designs a conditional entropy model with a large receptive field that models the sibling contexts, employs an attention mechanism to emphasize the correlated nodes in the context, and introduces a mask operation to speed up the encoding process. However, frequent attention mechanism operation for one node takes expensive time consumption in the decoding process. To solve this, OctFormer [16] eliminates the decoding dependencies between the nodes with non-overlapped context windows and shares the operation result within the sequence. EHEM [49] further proposes a hierarchical attention structure with a linear complexity to the context scale and maintains the global receptive field. Different from previous methods based on Cartesian coordinate, SCP-EHEM [20] proposes a multi-level Octree to mitigate the reconstruction error for distant areas within the spherical-coordinate-based Octree and adopts existing EHEM [49] method to compress the Octree further. These existing methods achieve notable compression performance, but there is still more context information to be discovered in the Octree.

III. Methodology

As shown in Fig. 2, we propose a graph-driven attention-based entropy model for the LPC compression named GAEM, which uses the information from the parent node and the encoded sibling nodes to predict the occupancy distribution of the current node. Specifically, we first organize the raw LPC data into a multi-level spherical coordinate Octree. Then we generate the adjacency matrices of parent and distance graphs

from the ancestor nodes and point distance of the context, followed by embedding the context using graph convolution. In the attention module, we adopt grouped-graph attention for the parent contexts and cross-attention for the parent context and sibling context. Grouped-graph attention addresses the local correlation of contexts by reducing the far node computation and sharing dependencies across the split sub-window of the context, which improves feature representation and reduces computation. Cross-attention increases the perceptual field of the model by discovering the relationship between parent context and sibling context, enriches the information at a higher Octree resolution, and provides fine-grained prediction variations. Finally, through an arithmetic encoder, the predicted node occupancy is losslessly encoded into a bitstream.

A. Multi-Level Spherical Octree Constructing

LPC data is obtained by rotating emitted laser beams to scan the environment from left to right. Theoretically, it is more apt for Spherical Octree representation, which transforms the quadratic Cartesian coordinate structure to linear angle values. In addition, the redundancy of the Spherical Octree is lower than that of the Cartesian Octree due to angle-based division which can represent more raw data under the same tree depth, thus improving the quality of high point cloud reconstruction. In the method, we firstly transform the raw point cloud coordinates from Cartesian coordinates $[x, y, z]$ to spherical coordinates $[\rho, \theta, \phi]$, where the transformation equations are defined as:

$$\rho = \sqrt{x^2 + y^2 + z^2},\ \theta = arctan(\frac{y}{x}),\ \phi = arccos(\frac{z}{\rho}). \quad (1)$$

After this transformation, the range of the spherical coordinate should fall within $[0, \rho_{max}]$, $[0, 2\pi]$, and $[0, \pi]$. Subsequently, we set the quantization step R_s for the spherical bounding box and uniformly scale all three dimensions into the same interval to achieve higher point coverage:

$$\rho' = \frac{\rho}{R_s}, \quad \theta' = \frac{\rho_{max}}{2\pi R_s}\theta, \quad \phi' = \frac{\rho_{max}}{\pi R_s}\phi. \quad (2)$$

We use scaled coordinates $[\rho', \theta', \phi']$ for Octree constructing, and R_s is fixed for all point cloud frames during the process, which is exponentially related to the max Octree depth.

The process of constructing a Spherical Octree closely resembles that of building a Cartesian Octree. However, we introduce a more efficient Octree representation by automatically adjusting the tree depth according to the local point density. Based on the radial range of points, we partition the entire point cloud into several parts, assigning extra depths to outer points for maintaining a similar level of precision among Octree nodes across the whole structure. Each non-empty Octree node produces the input sequence through a breadth-first traversal. When processing subsequent nodes, we introduce the ancestor nodes with the following attributes: 1) the depth within the Octree; 2) ancestral node occupancy in the range of $[0, 255]$; 3) octant in the range of $[0, 7]$; 4) spherical coordinates regularized to the range $[0, 1]$. Considering the spatial correlation between neighbor contexts, our model retains the coordinates and occupancy of previous context.

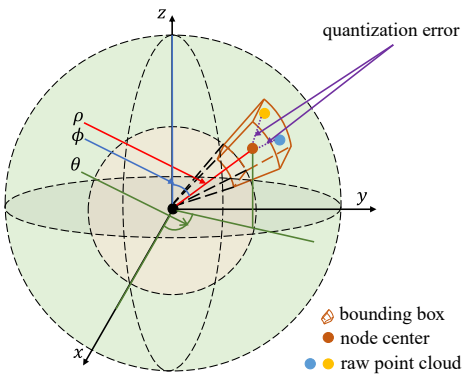


Fig. 3. Spatial partition and quantization error of Spherical Octree. The size of the quantization step in the Spherical Octree data structure solely depends on the radius of the spherical region associated with each point.

When predicting occupancies for the current features, these attributes are used as additional inputs in cross attention.

The encoding and decoding for Spherical Octree is the same as Cartesian Octree. During encoding, our proposed entropy model along with the arithmetic encoder converts the Octree to bitstream by predicting the occupancy distribution Q of the entire spherical sequence for a point cloud frame, which can be formulated as:

$$Q = \prod_i q_i(v_i|\mathbf{f_i}, G_p, G_d; \omega) \quad (3)$$

where v_i is the occupancy of node o_i in the sequence, q_i is the distribution of v_i. $\mathbf{f_i}$ indicates the local context features of node o_i, including $[f_0, ..., f_n]$, where $n+1$ is the window size. G_p and G_d are the parent and distance graphs generated according to the Octree respectively. ω is the weight of the entropy model. Considering that the loss function of our method is defined by the sequence distribution Q and its true distribution $\hat{Q}$, we use the node sequence distribution V instead of individual node distribution v_i for modeling. In the decoding stage, we initiate a sequential bitstream decoding from the center of the spherical coordinate. We only need to reverse the scaling of the Octree node centers to obtain the reconstructed spherical coordinates. This losslessly restores the entire Octree structure, after which all spherical centers of leaf nodes are transformed into Cartesian coordinates to recover the entire LPC. Since the encoding and decoding process is serial, we use both the parent node features and the encoded or decoded sibling node features for occupancy prediction.

B. Quantization Error

We note that node serialization is lossless because occupancy information is all exactly preserved and the only lossy procedure is due to quantization during Octree construction. In the Cartesian octree construction procedure, the Octree from a single point cloud frame is generated by dividing the space into 8 equal cubes until the maximum depth or an empty cube is reached. The coordinate of the cube center is typically used to represent each node, introducing a quantization error R_c, which can also represent the length of the bounding box along the x, y, or z axis at the maximum depth. Given the

original point P_c, we quantify it into the nodes center point $\hat{P}_c = [x_0, y_0, z_0]$. For a straight comparison with Spherical Octree, we define the quantization error ϵ_c between per point as:

$$\epsilon_c = \|P_c - \hat{P}_c\|_\infty \tag{4}$$

where P_c and $\hat{P}_c$ represent the real position and reconstructed position, and ϵ_c is the quantization error. For the situation when real coordinates of P_c are on the vertex of the bounding box, the quantization error reaches its maximum as $\frac{R_c}{2}$.

The quantization error of the Spherical Octree can be calculated by converting it to Cartesian coordinates. As shown in Fig. 3, the quantization step size is only related to the radius ρ of the point in Spherical Octree. According to Eq. 2, the range of the bounding box at the maximum depth is $[R_s, \frac{2\pi R_s}{\rho_{max}}, \frac{\pi R_s}{\rho_{max}}]$. We assume that the central position of a node is $\hat{P}_s = [\rho_0, \theta_0, \phi_0]$, then the coordinates of the farthest points in its bounding box should be $P_s = [\rho_0 + \frac{R_s}{2}, \theta_0 + \frac{\pi R_s}{\rho_{max}}, \phi_0 + \frac{\pi R_s}{2\rho_{max}}]$. Let's set $\hat{P_{s \rightarrow c}} = [\hat{x_s}, \hat{y_s}, \hat{z_s}]$, $P_{s \rightarrow c} = [x_s, y_s, z_s]$ are the Cartesian coordinates that converted from the spherical coordinates $\hat{P}_s$ and P_s:

$$\hat{P_{s \rightarrow c}} = [\rho_0 \sin\phi_0 \cos\theta_0, \rho_0 \sin\phi_0 \sin\theta_0, \rho_0 \cos\phi_0] \tag{5}$$

Given that $\frac{R_s}{2}$, $\frac{\pi R_s}{\rho_{max}}$, and $\frac{\pi R_s}{2\rho_{max}}$ are sufficiently infinitesimal, the first-order Taylor expansion is utilized, with higher-order terms being omitted:

$$\begin{aligned} x_s &\approx \rho_0 \sin\phi_0 \cos\theta_0 + \rho_0 \frac{\pi R_s}{\rho_{max}} \cos\phi_0 \cos\theta_0 \\ y_s &\approx \rho_0 \sin\phi_0 \sin\theta_0 + \rho_0 \frac{\pi R_s}{2\rho_{max}} \cos\phi_0 \sin\theta_0 \\ z_s &\approx \rho_0 \cos\phi_0 - \rho_0 \frac{\pi R_s}{\rho_{max}} \sin\phi_0 \end{aligned} \tag{6}$$

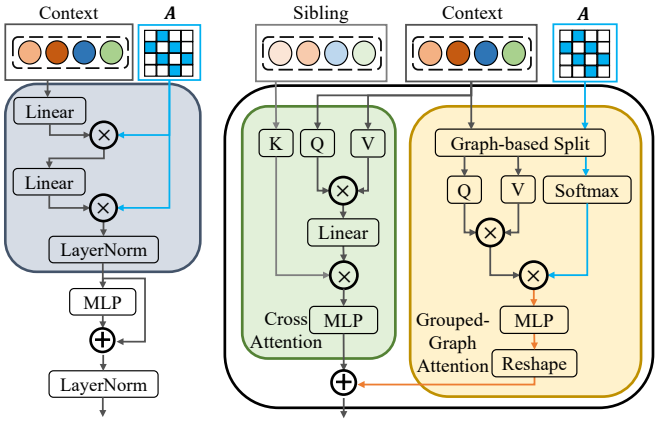
Then we calculate the max quantization error ϵ_s:

$$\begin{aligned} \epsilon_s &= \|P_{s \rightarrow c} - \hat{P_{s \rightarrow c}}\|_\infty \\ &= \max(|x_s - \hat{x_s}|, |y_s - \hat{y_s}|, |z_s - \hat{z_s}|) \\ &\leq \rho_0 \frac{\pi R_s}{\rho_{max}} \end{aligned} \tag{7}$$

From Eq. 7, the quantization error will increase in distant space, which also demonstrates the importance of using extra Octree depths for large radius LPC representation.

C. Octree-based Graph Generation & Convolution

In our method, we design two ways to generate Octree feature graphs separately using ancestral and sibling contexts: parent and distance graphs. For the parent graph, we calculate the similarity of their ancestors within the context nodes. Specifically, a unit edge weight is assigned to context nodes that share the same ancestors. To introduce more context information from other Octree levels and enrich the modeling of the structural relationships among the Octree context, we traverse several levels of ancestor nodes and add different depths' parent graphs. For a pair of nodes o_1 and o_2 in the same context, the edge weights of nodes $Edge_p(o_1, v_2)$ in parent graph can be formulated as:



(a) Graph convolution (b) Grouped-Graph attention

Fig. 4. Our purposed graph convolution module and grouped-graph attention block. ***A*** represents the adjendency matrix. **Sibling** and **Context** individually represents the previously and currently context. The graph convolution module consists of two cascaded matrices multiplication structures, and the attention module consists of grouped-graph attention in parallel with cross attention.

$$\begin{aligned} Edge_p(o_1, o_2) &= \sum_{m=1}^{N} U_m \\ U_m &= \begin{cases} 1, if\ \ Anc_{o_1}^m = Anc_{o_2}^m \\ 0, if\ \ Anc_{o_1}^m \neq Anc_{o_2}^m \end{cases} \end{aligned} \tag{8}$$

where the ancestor nodes in the Spherical Octree of o_1 and o_2 are represented as $Anc_{o_1}^m$ and $Anc_{o_2}^m$, and the N is the times of searching. The parent graph contains the relationship between contexts and ancestral nodes.

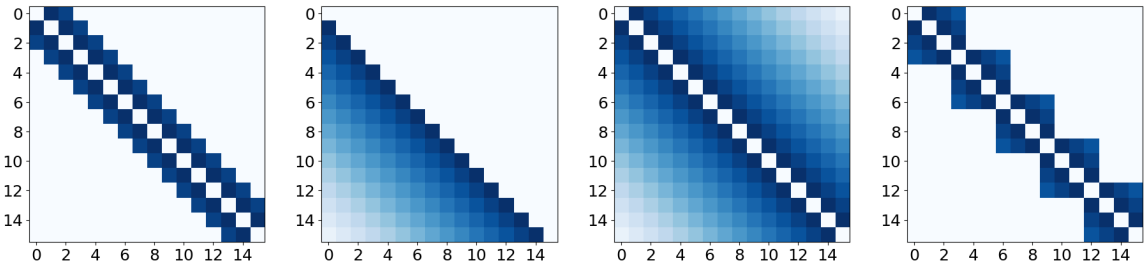
For the distance graph, we compute the geometric center distance of the context. In general, the neighbor point clouds in space are more likely to characterize the same object and thus exhibit higher dependence in occupancy prediction. Through the distance graph, we capture the spatial dependencies between the parent nodes, thus sharing node information in the window. The distance D for two nodes o_1 and o_2 in spherical coordinates can be formulated as:

$$\begin{aligned} D(o_1, o_2) &= \sqrt{\Delta x^2 + \Delta y^2 + \Delta z^2} \\ \Delta x = |x_{o_1} - x_{o_2}|, \Delta y &= |y_{o_1} - y_{o_2}|, \Delta z = |z_{o_1} - z_{o_2}| \end{aligned} \tag{9}$$

We use $[x_{o_1}, y_{o_1}, z_{o_1}]$ and $[x_{o_2}, y_{o_2}, z_{o_2}]$ to represent the Cartesian coordinates of the centers of nodes o_1 and o_2. Through trigonometric relationships, the elements in $D(o_1, o_2)$ can be further calculated using spherical coordinates of the node centers:

$$\begin{aligned} h_{o_1} &= \rho'_{o_1} \sin\phi'_{o_1} \\ h_{o_2} &= \rho'_{o_2} \sin\phi'_{o_2} \\ \Delta x^2 + \Delta y^2 &= h_{o_1}^2 + h_{o_2}^2 - 2h_{o_1}h_{o_2}cos(\theta'_{o_1} - \theta'_{o_2}) \\ \Delta z^2 &= \left|h_{o_1}tan(\phi'_{o_1}) - h_{o_2}tan(\phi'_{o_2})\right|^2 \end{aligned} \tag{10}$$

where $[\rho'_{o_1}, \theta'_{o_1}, \phi'_{o_1}]$ and $[\rho'_{o_2}, \theta'_{o_2}, \phi'_{o_2}]$ are the scaled coordinates of nodes o_1 and o_2, as shown in Eq. 2. We apply a normalization to adjust the weight to $[0, 1]$ through the



(a) GAT [50] (b) OctAttention [18] (c) EHEM [19] (d) Ours

Fig. 5. Illustration of the dependencies discovered through different attention mechanisms, in which the x-axes and y-axes represent the different positions of the node pairs, and the colors are deeper for stronger dependencies.

minimum D_{min} and maximum D_{max}. For closer nodes, their edge weights in the distance graph are larger:

$$Edge_d(o_1, o_2) = 1 - \frac{D(o_1, o_2) - D_{min}}{D_{max} - D_{min}} \tag{11}$$

We generate the adjacency matrices for all the input contexts using both parent and distance graph methods, which enables the graph convolution module to learn node dependencies constructed by different features.

Considering that the adjacency matrices are symmetric, the convolution mechanism is beneficial to model more semantic details. We use the graph convolution module for context embedding instead of positional encoding [51] to better exploit the edge weights of graph adjacency matrices. As Fig. 4(a) shows, the graph convolution module is composed of two cascaded matrix multiplication structures and a residual connection. We project the context with learnable parameters and multiply it with an adjacency matrix in graph convolution blocks. A multi-layer perception (MLP) is applied with the residual connection after convolution to project output features. Both the adjacency matrices of the parent and distance graph methods are fed into different graph convolution modules, and the output results are concatenated. With the graph convolution module, we effectively utilize the original node features and graph connectivity to be the contexts with rich semantic information and feed them into the graph attention module.

D. Graph Attention

After the graph convolution module, the entropy model learned a shallow representation of the Octree from the context and graph features. We propose a graph attention module for deeper node feature learning further on, which contains cross attention for sibling context dependencies extraction and grouped-graph attention for context's self-dependency discovery. As shown in Fig. 4(b), we empirically divide the context into a series of sub-windows based on the threshold of graph's edge weights. For low-dependency node pairs with edge weights less than the threshold, they would be assigned to another sub-window and use the graph for attention matrix masking. In this way, we can find the proper sub-window size for the grouped-graph attention. The attention only computes the dependencies in each sub-window. Specifically, we crop the graph to match the length of the sub-windows, making the graph able to perform the matrix product. For the window-to-window dependency, we directly use MLP to project the entire reshaped context after attention. The proposed attention aims at Octree context features and has the following main advantages:

Low computational complexity. As shown in Fig. 5, we visualize the dependency relationships among attention mechanisms for different methods. The deeper colors mean a stronger dependency on node pairs. For example, in Fig. 5(a), the color of the square at position (1,2) is deeper than that at (1,6), which means that compared to the relationship between the 2nd and 7th nodes, the 2nd and 3rd nodes show stronger dependency. The vanilla graph self-attention (GAT) only computes the neighboring nodes and requires a computation complexity of $(n * p * d)$, where n is the context window size, p is the number of neighbors, and d is the hidden dimension. It utilizes the graph weights and reduces computation, but the node features that can be discovered are limited. Meanwhile, as shown in Fig. 5(b) and Fig. 5(c), other self-attention-based methods require a computation complexity of $O(n^2 * d)$ in computing node dependencies whether the non-casual features are discarded or retained. This attention mechanism consumes high computation with the node increasing. Compared to these methods, our grouped-graph attention (Fig. 5(d)) reduces discovering nodes with far connections but performs more in the local window. Our mechanism requires a computation complexity of $O(\frac{n}{k} * k^2 * d)$, which is reduced by a factor of $\frac{n}{k}$ compared to the traditional attention, where k is the size of sub-window. Due to the more centralized distribution of node dependencies in the Octree context, good predictions can also be achieved using a small k, as we will present in our experiments.

Convolution operation provided. It has been demonstrated that the use of a convolution module removes the translation invariance of Octree nodes [7], [16]. However, the convolution module introduces additional computation that affects the speed of Octree coding and decoding. The parameter of the grouped-graph attention learns about the local features and serves a similar function to the convolution operation. Compared to traditional self-attention, this operation discards the result sharing in the context window but is more accurate for neighboring nodes.

To extract features from the deeper levels, we apply a cross-

attention for sibling context and parent context. As shown in Fig. 4(b), the current context is projected through two linear layers to generate the query and value tokens with the same feature size, while the sibling context is squeezed on the feature dimension using a linear layer to generate key tokens. The generated attention weights are mapped to the original dimensions through another linear layer. Key token squeezing amplifies feature variability, which enables the attention to discover the main correlation between contexts.

E. Network Learning

The optimization objective of the Octree entropy model is to reduce the total bitrate when encoding Octree occupancy into the bitstream. The bitrate can be expressed as the cross-entropy between the predictive distribution Q and the true distribution $\hat{Q}$. The loss function can be formulated as:

$$\ell = -\sum_{v} \log q_i(v_i|\mathbf{f_i}, G_p, G_d; \omega) \tag{12}$$

where the estimated occupancy distribution for Octree node o_i is represented by $q_i(v_i|\mathbf{f_i}, G_p, G_d; \omega)$.

IV. Experiments

In this section, we conduct extensive experiments to demonstrate the superior performance of our proposed method on the LiDAR benchmark SemanticKITTI and MPEG-specified dataset Ford. The experimental results show that our method has comparable performance with the raw LPC and outperforms other baseline methods.

A. Datasets

- **SemanticKITTI** [32] is a public open-source LiDAR dataset for autonomous driving. SemanticKITTI provides 22 sequences, 43,504 scans in total, and each scan contains over 120,000 individual points. We adopt the official data splitting methodology, which uses sequences 00 to 10 for training and 11 to 21 for testing.
- **Ford** [33] is another LiDAR dataset suggested by MPEG Common Test Conditions (CTC) [52]. Ford contains more than 1,500 scans of static point cloud scenarios. We use the same train/test splits as Ford's benchmark tasks, including 1,045 as the training set, 156 as the validation set, and 312 as the test set. We sample 50,000 points from each scan the same as [31].

B. Implementation Details

Our entropy model has 4 attention blocks, in which the channel dimension is set to 256. In the graph generation module, we search $N = 3$ former layers of ancestors for the parent graph. To evaluate our model at different bitrates, we follow the quantization settings in [19] for SemanticKITTI, setting the quantization step R_s to $\frac{400}{2^{D-1}}$ to build the Octree with depth D. We use the maximum depth of 14 with sequence size 1024 for training and set the Octree depth from 9 to 14 and the context length from 32 to 1024 in testing. For Ford, we also follow the settings in [19], which sets the quantization step R_s to $\frac{2^{18}}{2^D}$ to build the Octree from the normalized point cloud. We train our model using a maximum depth of 16 and test with a depth from 11 to 16. Our model is implemented in PyTorch and trained/tested on a machine equipped with an Xeon Gold 6134 CPU and a single NVIDIA Tesla A100 GPU (40GB Memory). For the training procedure, we use the Adam optimizer with a learning rate of 1e-4 and a decay rate of 1e-2 for our model. The model training takes approximately one day on Ford and three days on SemanticKITTI datasets.

C. Baseline Setup

We verify the effectiveness of our method on the 3D point cloud datasets Ford and SemanticKITTI. For both two datasets, we compare with GPCC [14] with MPEG standard (TMC13 v27.0) and other learning-based methods including voxel-based Voxel-ContextNet [31] and SparsePCGC [17], Cartesian Octree-based OctAttention [18], OctFormer [16], and EHEM [19], and Spherical Octree-based SCP-EHEM [20]. All the above methods are only designed for a specific category of LPC.

We follow the common test condition (CTC) [52] to proceed with the experiments, which provides a standardized benchmark for evaluating performance. All configurations except for positionQuantizationScale are set as default. We set different positionQuantizationScales ranging from 0 to 1 to adjust the compression ratios and qualities. The experiments are conducted on all intra modes, for each frame is individually encoded and decoded, without relying on other frames. We compare our method with others on lossy compression for geometry, which corresponds to the C2 test condition in CTC. Since the source codes of some methods are not publicly available, we keep our training/testing setting consistent with these methods and directly use the results presented in their respective papers.

D. Evaluation Metrics

For evaluation, we use bits per point (bpp) [16], [18] for the compression ratio metric, which describes the data volume after compression:

$$bpp = \frac{1}{n_1} \sum_{a=1}^{n_2} s_a \tag{13}$$

where n_1 is the total number of raw points in all sequences, n_2 is the total number of LiDAR sweeps from sequences, and s_a denotes the size of the bitstream for each sweep. As for assessing the quality of point cloud reconstruction, we use point-to-point PSNR (D1 PSNR), and point-to-plane PSNR (D2 PSNR) proposed by the MPEG standards [13]. We use the official metric calculating tool *pc_error* provided by MPEG's GPCC, and set the PSNR peak value *r* = 59.7 following [17]. Besides, we also use chamfer distance (CD) [53], [54] to further evaluate the reconstruction quality:

$$\begin{aligned} \mathrm{CD}(\mathbf{P}, \overline{\mathbf{P}}) &= \max\left(\mathrm{CD}'(\mathbf{P}, \overline{\mathbf{P}}), \mathrm{CD}'(\overline{\mathbf{P}}, \mathbf{P})\right) \\ \mathrm{CD}'(\mathbf{P}, \overline{\mathbf{P}}) &= \frac{1}{|\mathbf{P}|} \sum_{i} \min_{j} \left\|\mathbf{p}_i - \overline{\mathbf{p}}_j\right\|_2 \end{aligned} \tag{14}$$

TABLE I
BD-RATE GAINS OVER BASELINE GPCC MEASURED USING PSNR (D1, D2), AND CHAMFER DISTANCE (CD) FOR OURS AND OTHER METHODS.

Method	Reference	Ford			SemanticKITTI		
		D1 PSNR	D2 PSNR	CD	D1 PSNR	D2 PSNR	CD
VoxelContext-Net [31]†	CVPR'21	-28.3%	-27.4%	-28.9%	-16.6%	-21.1%	-15.7%
SparsePCGC [17]†	TPAMI'22	-33.1%	-32.3%	-34.9%	-15.7%	-10.8%	-6.5%
OctAttention [18]‡	AAAI'22	-38.7%	-38.1%	-38.6%	-22.4%	-26.6%	-24.6%
OctFormer [16]†	AAAI'23	-39.5%	-38.9%	-39.6%	-23.5%	-27.7%	-26.3%
EHEM [19]‡	CVPR'23	-40.3%	-39.5%	-37.7%	-29.8%	-33.4%	-26.4%
SCP-EHEM [20]‡	AAAI'24	-51.1%	-54.2%	-49.6%	-40.6%	-40.3%	-37.7%
Ours‡	-	**-53.6%**	**-56.5%**	**-52.0%**	**-53.9%**	**-54.1%**	**-49.4%**

Classification of introduced features: † parent-introduced that only depends on the parent context, ‡ sibling-introduced that depends on both parent and sibling contexts.

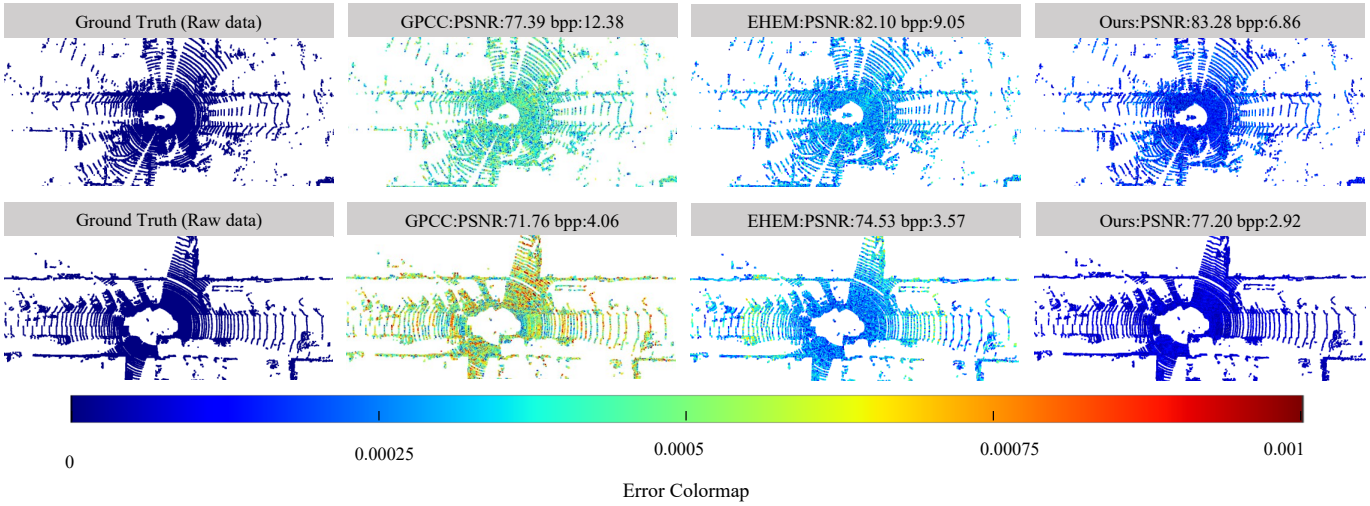


Fig. 6. The visualization of compression results of GPCC, EHEM, and ours on Ford (top) and SemanticKITTI (bottom) under different bpps. Considering that Octree-based baselines have the same reconstruction qualities, we only visualize the quantitative results of GPCC, EHEM, and ours.

where $\mathbf{p}_i$ in point cloud $\mathbf{P}$ is the closest point to the $\overline{\mathbf{p}}_j$ in $\overline{\mathbf{P}}$. Using the compression ratio and reconstruction metrics, we calculate the Bjøntegaard Delta Rate Gains (BD-Rate Gains) [55] as the quantitative results and use the GPCC for anchor, which intuitively describes rate-distortion performance.

E. Experiment Results

1) Comparison with State-of-the-Art Methods: The quantitative experimental results are presented in Table. I. We calculate the BD-Rate gains over GPCC using D1 PSNR, D2 PSNR, and CD of our and other methods. Overall, our method outperforms other methods as expected. For Ford, our method achieves up to 53.6%, 56.5%, and 52.0% gains for D1 PSNR, D2 PSNR, and CD. For SemanticKITTI, our method achieves up to 53.9%, 54.1%, and 49.4% gains for D1 PSNR, D2 PSNR, and CD. Specifically, for Ford, we obtain the average bpp=6.83 for depth-16 Octree, making 47.5%, 46.3%, 44.6%, 36.2%, 36.1%, and 28.0% reduction in bitrates compared with GPCC (bpp=13.02), VoxelContextNet (bpp=12.73), SparsePCGC (bpp=12.32), OctAttention (bpp=10.71), OctFormer (bpp=10.69), and EHEM (bpp=9.49). For SemanticKITTI, our method obtains the average bpp=3.88 for depth-14 Spherical Octree and achieves 37.9%, 35.0%, 32.3%, 31.7%, 9.1%, and 7.0% reduction in bitrates compared with GPCC (bpp=6.25), Voxel-ContextNet (bpp=5.97), OctAttention (bpp=5.73), OctFormer (bpp=5.68), SparsePCGC (bpp=4.27), and EHEM (bpp=4.17) that has lower reconstruction qualities. Compared to state-of-the-art method SCP-EHEM [20] (bpp=7.39 and 4.16), our method still achieves up to 6.9% and 6.7% bitrate saving on two datasets separately. The main reason is that we introduce context information through the graph objects based on the original node features, which effectively improves the accuracy in capturing context dependencies. The experimental results show that our method achieves higher BD-Rate gains on both datasets compared with other methods, which demonstrates the advantages of our method.

We also visualize the qualitative results, as shown in Fig. 6. The bluer areas on the color represent smaller quantization errors, while the redder areas represent larger errors. The figure clearly illustrates that the errors between the ground-truth point clouds and our reconstructed point clouds are smaller compared to GPCC and EHEM [19]. Our method makes lower bpp than both GPCC and EHEM in higher PSNR, which demonstrates the effectiveness of our method in achieving superior compression performance while maintaining high-quality reconstruction of the point clouds. Considering that

TABLE II
ENCODING/DECODING TIME OF A D DEPTH OCTREE ON SEMANTICKITTI.

Method	Reference	Encoding / Decoding Time (s) ↓				Params ↓
		D=10	D=12	D=14	D=16	
GPCC [14]	MPEG'24	0.24 / 0.07	0.40 / 0.21	0.69 / 0.50	0.82 / 0.59	-
Voxel-ContextNet [31]†	CVPR'21	25.5 / 24.9	29.3 / 29.4	75.2 / 74.6	154 / 153	-
SparsePCGC [17]†	TPAMI'22	0.19 / 0.13	0.33 / 0.21	0.88 / 0.56	1.44 / 0.97	5.74M
OctAttention [18]‡	AAAI'22	0.13 / 94.5	0.15 / 166	0.39 / 415	0.77 / 828	4.23M
OctFormer [16]†	AAAI'23	0.27 / 0.36	0.32 / 0.47	0.79 / 1.01	1.55 / 2.16	4.28M
EHEM [19]‡	CVPR'23	0.32 / 0.38	0.60 / 0.57	1.48 / 1.57	2.92 / 3.13	13.01M
SCP-EHEM [20]‡	AAAI'24	0.55 / 0.52	1.16 / 1.11	2.34 / 2.41	3.92 / 4.09	13.01M
Ours‡	-	0.24 / 0.25	0.52 / 0.57	1.19 / 1.31	2.26 / 2.47	4.21M

Classification of introduced features: † parent-introduced that only depends on the parent context, ‡ sibling-introduced that depends on both parent and sibling contexts.

TABLE III
GAEM PERFORMANCE WITH DIFFERENT MODULES. GGAT, GCN, AND CA REPRESENT THE GROUPED-GRAPH ATTENTION, GRAPH CONVOLUTION, AND CROSS ATTENTION, RESPECTIVELY. THE DEFAULT SETTINGS ARE MARKED IN GRAY.

GGAT	GCN	CA	bpp ↓					BD-Rate Gains
			D=10	D=11	D=12	D=13	D=14	
✓			0.259	0.882	1.744	2.677	4.003	-52.3%
✓	✓		0.247	0.843	1.692	2.593	3.927	-53.3%
✓	✓	✓	**0.235**	**0.829**	**1.635**	**2.554**	**3.878**	**-53.9%**

SCP-EHEM [20] has a similar reconstructed LPC quality to ours, we do not include it here.

2) Runtime Analysis: As shown in Table.II, we provide a comparison with the baselines in terms of encoding and decoding time for Octrees in different depths and the settings are the same as [19]. We can observe that the sibling-introduced methods (using both the parent and sibling contexts) need more encoding and decoding time than the parent-introduced methods (only using the parent context). The reason is that when introducing sibling context, it is necessary to first predict sibling nodes, which means more time is required. While the parent-introduced methods can directly predict the occupancy of all nodes from the same layer in parallel. Nevertheless, our method still achieves the optimal coding and decoding time among all sibling-introduced methods. This is because our GAEM uses grouped-graph self-attention by dividing the context into multiple sub-windows and sharing attention dependencies only in the sub-windows, which effectively reduces the computation. Besides, GAEM serially predicts the node distribution of a multi-level Octree through non-overlapping context windows, which avoids repetitive operations on the form level and reduces the overall coding and decoding time. At the same time, GAEM only needs to restore the sibling context feature from the bitstream using arithmetic decoding in one go without extra computation for the sibling dependency, which further reduces the decoding time.

F. Ablation Study

To further investigate the impact of different combinations of modules and hyperparameters on the performance of the model, we perform the following ablation studies. Considering that altering hyperparameters of the entropy model does not affect compression quality, we only use bpp as the metric.

TABLE IV
GAEM PERFORMANCE WITH DIFFERENT MODULES ON SEMANTICKITTI.

Embedding Module	Attention Module	bpp↓ / BD-Rate Gains	
		No CrossAttn[1]	With CrossAttn
PE[2] [51]	MSA[3]	5.133 / -39.0%	5.002 / -40.5%
	MSA (no mask)	4.783 / -43.1%	4.719 / -43.9%
	GAT[4]	5.088 / -39.5%	4.956 / -41.1%
	GGAT[5]	4.203 / -50.0%	4.121 / -51.0%
OctPEG[6] [16]	MSA	4.892 / -41.8%	4.828 / -42.6%
	MSA (no mask)	4.503 / -46.4%	4.446 / -47.1%
	GAT	4.865 / -42.1%	4.797 / -43.0%
	GGAT	4.075 / -51.5%	4.003 / -52.3%
GraphConv[7]	MSA	4.811 / -42.8%	4.726 / -43.8%
	MSA (no mask)	4.417 / -47.5%	4.348 / -48.3%
	GAT	4.734 / -43.7%	4.689 / -44.2%
	GGAT	3.927 / -53.3%	**3.878 / -53.9%**

[1] cross attention, [2] position encoding, [3] multi-head self-attention, [4] graph attention, [5] grouped-graph attention, [6] Octree positional encoding generator, [7] graph convolution.

1) Different Module Analysis: To evaluate the effectiveness of our proposed grouped-graph attention (GGAT), graph convolution (GCN), and cross attention module (CA) in GAEM, we conduct an ablation study as shown in Table. III. In general, all the modules improve compression performance as expected, and comparison shows that our method obtains maximum BD-Rate gains of 53.9% on average. On one hand, it can be indicated that GCN improves performance, which can make node data augmentation more efficient compared to a normal embedding layer. Besides, using the parent and distance graph adjacency matrices on the entire context, nodes are better characterized by their structural and positional features. On the other hand, CA increases the receptive field of the model by introducing high-resolution sibling context nodes, which further improves the prediction.

As shown in Table IV, we also investigate the performance of different combinations of embedding module, attention module, and the cross attention. From the table, we can see that our method combining GraphConv, GGAT, and CrossAttn exhibits the best compression performance compared with other module combinations, achieving a maximum BD-Rate gain of 53.9%, as expected. Specifically, on one hand, GraphConv module outperforms other embedding modules, due to its layered design enabling the modeling of higher-order neighborhoods' local connectivity and topological features. On the other hand, GGAT module also obtains better performance

TABLE V
GAEM PERFORMANCE WITH DIFFERENT OCTREE TYPES ON SEMANTICKITTI.

depth D	Octree Type							
	Cartesian Octree				Spherical Octree			
	bpp↓	D1 PSNR↑	D2 PSNR↑	CD (cm)↓	bpp↓	D1 PSNR↑	D2 PSNR↑	CD (cm)↓
D=10	0.243	54.28	59.17	19.08	0.235	56.05	60.98	17.82
D=11	0.854	60.19	65.14	9.27	0.829	62.45	67.29	7.24
D=12	1.677	66.03	71.21	4.82	1.636	68.36	73.48	3.57
D=13	2.683	72.37	77.20	2.28	2.554	74.43	78.93	1.91
D=14	4.025	77.84	84.45	1.53	**3.878**	**80.01**	**85.11**	**1.27**
BD-Rate Gains	-	-32.9%	-37.9%	-25.2%	-	**-53.9%**	**-54.1%**	**-49.4%**

TABLE VI
GAEM PERFORMANCE WITH DIFFERENT SUB-WINDOW SIZES. k REPRESENTS THE SUB-WINDOW SIZE AND THE DEFAULT SETTINGS ARE MARKED IN GRAY.

Sub-Window Size	bpp					BD-Rate Gains	Params↓
	D=10	D=11	D=12	D=13	D=14		
k=16	0.235	0.829	**1.635**	2.554	**3.878**	**-53.9%**	**4.21M**
k=32	0.237	**0.828**	1.636	2.557	3.879	-53.8%	6.83M
k=64	**0.234**	0.829	1.638	**2.553**	3.881	-53.8%	12.55M

TABLE VII
GAEM PERFORMANCE WITH THE INCREASE OF SPHERICAL OCTREE CONTEXT SIZES w. THE DEFAULT SETTINGS ARE MARKED IN GRAY.

Size w	bpp ↓					BD-Rate Gains
	D=10	D=11	D=12	D=13	D=14	
32	0.263	0.864	1.693	2.638	3.982	-52.6%
64	0.258	0.859	1.686	2.624	3.959	-52.9%
128	0.255	0.852	1.673	2.597	3.933	-53.2%
256	0.248	0.846	1.661	2.583	3.915	-53.4%
512	0.241	0.837	1.649	2.568	3.892	-53.7%
1024	**0.235**	**0.829**	**1.636**	**2.554**	**3.879**	**-53.9%**

compared with other attention mechanisms (MSA, MSA with no mask, and GAT), owing to its dynamic allocation of weights to nodes, enabling it to effectively leverage structural relationships and positional features within the Octree context. Additionally, we can observe that combining with the cross attention achieves lower bpps, benefiting from its ability to further broaden the model's receptive field by incorporating high-resolution sibling context nodes. In general, the experimental results show that our proposed method shows better compression performance, compared with the combination of PE and MSA, without the cross attention component.

2) Octree Structure Analysis: As shown in Table. V, we evaluate the impact of different Octree structures on performance. From the table, we can see that the Spherical Octree achieves better BD-Rate gains than the Cartesian Octree at all depths, as expected. One potential reason is the Spherical Octree divides the point cloud based on sphere coordinates, resulting in more regular node distributions, which reduces the complexity of predicting node occupancies for the model. Besides, compared to Cartesian Octree which abruptly divides point clouds based on Cartesian coordinates, the Spherical Octree divided by angle and radius can decrease the point offsets, thereby improving the quality of point cloud reconstruction. The experiment results demonstrate the effectiveness of Spherical Octree coding for LPC compression.

3) Sub-Window Size Analysis: We conduct an ablation study for the models with a series of sub-window sizes to verify the performance of our proposed grouped-graph attention in different group window sizes. As shown in Table. VI, we can observe that decreasing the sub-window size k has almost no effect on the model performance, while effectively reducing the number of parameters for lower computation. A reasonable explanation is that the currently set window size already exceeds the high dependency field of vast nodes in the Octree context, allowing the model to maintain optimal prediction results, while for a larger node range, the MLP can efficiently explore their dependencies. The experiment results demonstrate that the grouped-graph attention effectively addresses the local node dependencies and decreases the computation.

TABLE VIII
GAEM PERFORMANCE ON SEMANTICKITTI WITH DIFFERENT LEARNING RATES lr AND BATCH SIZES bz IN TRAINING PROCESS. THE DEFAULT SETTINGS ARE MARKED IN GRAY.

Hyperparameters					
lr	bpp↓	Gains	bz	bpp↓	Gains
1×10^{-4}	4.004	-52.4%	4	4.976	-40.8%
3×10^{-4}	**3.891**	**-53.7%**	8	4.421	-47.4%
5×10^{-4}	4.116	-51.1%	12	4.069	-51.6%
1×10^{-3}	4.646	-44.7%	16	**3.884**	**-53.8%**
5×10^{-3}	5.003	-40.5%	32	3.892	-53.7%
1×10^{-2}	5.148	-38.8%	64	3.894	-53.7%

4) Context Window Size Analysis: As shown in Table. VII, we perform ablation study on context size. Our method gets better compression results as the window size increases. The reason is that with the increase in context size, the sequence contains more parent and sibling node information, which is beneficial for generating graphs with richer connections. For example, our method achieves up to 0.103 bpp savings between the 1024 context size and 32 when setting the Octree depth to 14, which shows that the model with a larger context size can obtain better performance. However, even at the smallest context window size, our method is still better than other methods, which demonstrates our improved feature extraction for local nodes without relying on large-scale context discovery.

5) Sensitivity Analysis: As shown in Table. VIII, we also analyze sensitivities for learning rate and training batch size. On one hand, we can see that the learning rate yields superior performance at lower values but quickly leads to a decline as it increases. The potential reason is that the occupancy distributions of nodes are not uniform, with many dense points tending to exhibit the same Octree node occupancy. When the learning rate is overly high, it makes model underfitness,

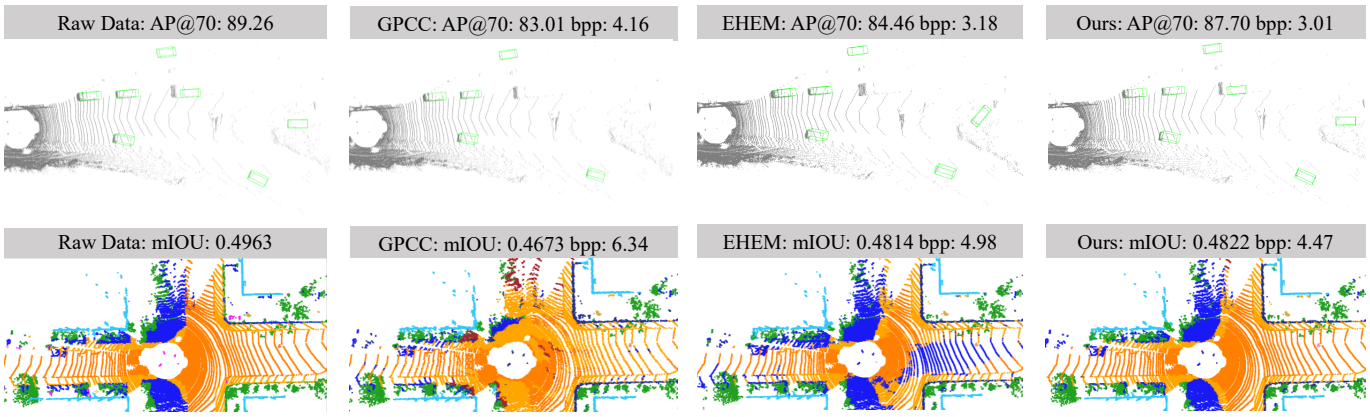


Fig. 7. The visualization of vehicle detection (top) and semantic segmentation (bottom) task under different bpps. For object detection, the green bounding box identifies the predicted position. For semantic segmentation, different colors indicate different point cloud classes, with the higher segmentation accuracy would show a more similar color distribution to the ground truth.

TABLE IX
THE RESULTS OF VEHICLE DETECTION UNDER DIFFERENT BPPS OF DIFFERENT METHODS.

Method	bpp	Detection AP		
		Easy	Mod	Hard
Raw Data	-	89.26	78.71	73.73
GPCC	1.36	79.26	69.37	65.81
	2.88	82.73	74.17	68.43
	4.16	83.01	74.55	72.37
EHEM	1.08	79.40	70.60	67.33
	2.07	82.64	71.53	68.27
	3.18	84.46	74.80	71.54
Ours	**0.98**	80.68	71.54	68.23
	1.93	84.06	74.52	69.24
	3.01	**87.70**	**69.24**	**73.01**

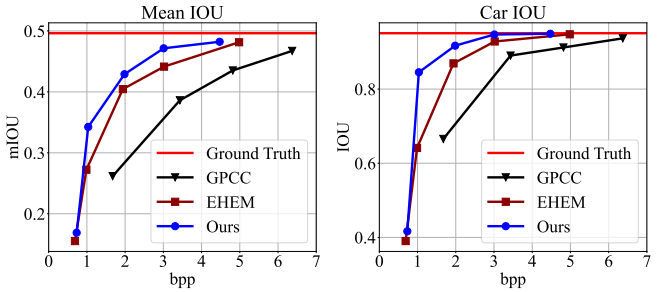


Fig. 8. Quality performance of semantic segmentation within different point cloud compression methods on SemanticKITTI dataset.

indicating that it may be inadequate for sparse region feature learning. On the other hand, as batch size increases, the performance faces a sharp improvement firstly and stabilizes thereafter. The reason is that an overly small batch size causes training instability, as the direction of gradient descent can be affected by the dataset. The experimental results demonstrate the high sensitivity of both the learning rate and batch size to the model.

V. APPLICATIONS

To further verify the practicality of the method, we employ GEAM on two downstream applications (*i.e.* vehicle detection and semantic segmentation). We compare our method with the baseline GPCC [14], and the state-of-the-art Cartesian Octree methods EHEM [19]. Considering that SCP-EHEM [20] has a similar reconstructed LPC quality to ours, we do not include it here.

A. Vehicle Detection

Our method is implemented and assessed on downstream tasks in vehicle detection. Part-A^2 net [56] is selected as our benchmark for evaluating performance. We measure the average precision (AP) of our method's vehicle detection capabilities and set a stringent threshold of 0.7 for the mean intersection-over-union (mIOU), which is a standard in the evaluation of object detection algorithms. Experimental results are displayed in Fig. 7 and detailed in Table. IX. These presentations are structured to categorize the detection performance based on the difficulty levels of the tasks, which are classified as easy, moderate, and hard, following the KITTI benchmark. Compared with other methods, our method achieves higher AP while maintaining a relatively lower bpp. For instance, when our bpp is 3.01 lower than the EHEM, the AP metrics exceed it by 3.24%, 1.49%, and 1.47% for easy, moderate, and hard modes. Additionally, our method has similar detection results to the raw point cloud data.

B. Semantic Segmentation

In addition to vehicle detection, we also conduct verification on 3D semantic segmentation, with RandLA-Net [57] as our benchmark. Mean intersection-over-union (mIOU) is used as the metric to measure segmentation accuracy, which is a standard measure for evaluating the accuracy of segmentation algorithms. The mIOU provides a comprehensive view of the segmentation quality by considering both the precision and recall of the segmentation process. Fig. 7 and Fig. 8 show the segmentation results. The experimental data reveal that our method achieves highly accurate segmentation performance, with results close to raw point cloud data. This is particularly evident at a bpp value of around 4.47, where our method demonstrates its capability to maintain high fidelity in the

segmentation process. Overall, our method excels in both vehicle detection and semantic segmentation, proving effective for downstream tasks.

In conclusion, our method stands out as a highly competent and versatile solution for vehicle detection and semantic segmentation, with its proven effectiveness translating into a multitude of downstream applications.

VI. Conclusion

In this paper, we present **GAEM**, a novel **G**raph-driven **A**ttention **E**ntropy **M**odel for LPC compression, combining the graph convolution augmentation of node features and an improved graph attention module achieving accurate prediction for occupancy of multi-layer Spherical Octree nodes. GAEM introduces grouped-graph self-attention to discover the correlation of sub-window nodes, which prevents traversing the former levels of multi-layer Octree repeatedly, thus speeding up encoding and decoding. Besides, to further enlarge the receptive field of the model, we design a cross-attention module to provide higher-resolution information for the node features by introducing the encoded sibling contexts. Experimental results show that our method outperforms all state-of-the-art baselines on LiDAR benchmark SemanticKITTI and MPEG-specified dataset Ford and downstream tasks vehicle detection and semantic segmentation.

References

[1] Z. Yuan, X. Song, L. Bai, Z. Wang, and W. Ouyang, "Temporal-channel transformer for 3d lidar-based video object detection for autonomous driving," *IEEE Transactions on Circuits and Systems for Video Technology*, vol. 32, no. 4, pp. 2068–2078, 2022.

[2] Y. Yu, W. Zhang, G. Li, and F. Yang, "A regularized projection-based geometry compression scheme for lidar point cloud," *IEEE Transactions on Circuits and Systems for Video Technology*, vol. 33, no. 3, pp. 1427–1437, 2023.

[3] J. Li, C. Luo, and X. Yang, "Pillarnext: Rethinking network designs for 3d object detection in lidar point clouds," in *2023 IEEE/CVF Conference on Computer Vision and Pattern Recognition (CVPR)*, pp. 17567–17576, 2023.

[4] A. Boulch, C. Sautier, B. Michele, G. Puy, and R. Marlet, "ALSO: automotive lidar self-supervision by occupancy estimation," in *2023 IEEE/CVF Conference on Computer Vision and Pattern Recognition (CVPR)*, pp. 13455–13465, IEEE, 2023.

[5] W. Dong, T. Yang, J. Qu, T. Zhang, S. Xiao, and Y. Li, "Joint contextual representation model-informed interpretable network with dictionary aligning for hyperspectral and lidar classification," *IEEE Transactions on Circuits and Systems for Video Technology*, vol. 33, no. 11, pp. 6804–6818, 2023.

[6] X. Chang, H. Pan, W. Sun, and H. Gao, "A multi-phase camera-lidar fusion network for 3d semantic segmentation with weak supervision," *IEEE Transactions on Circuits and Systems for Video Technology*, vol. 33, no. 8, pp. 3737–3746, 2023.

[7] L. Huang, S. Wang, K. Wong, J. Liu, and R. Urtasun, "Octsqueeze: Octree-structured entropy model for lidar compression," in *Proceedings of the IEEE/CVF conference on computer vision and pattern recognition*, pp. 1313–1323, 2020.

[8] L. de Paula Veronese, F. Auat-Cheein, F. Mutz, T. Oliveira-Santos, J. E. Guivant, E. de Aguiar, C. Badue, and A. F. De Souza, "Evaluating the limits of a lidar for an autonomous driving localization," *IEEE Transactions on Intelligent Transportation Systems*, vol. 22, no. 3, pp. 1449–1458, 2021.

[9] W. Huang, H. Liang, L. Lin, Z. Wang, S. Wang, B. Yu, and R. Niu, "A fast point cloud ground segmentation approach based on coarse-to-fine markov random field," *IEEE Transactions on Intelligent Transportation Systems*, vol. 23, no. 7, pp. 7841–7854, 2022.

[10] H. Liu, H. Yuan, Q. Liu, J. Hou, H. Zeng, and S. Kwong, "A hybrid compression framework for color attributes of static 3d point clouds," *IEEE Transactions on Circuits and Systems for Video Technology*, vol. 32, no. 3, pp. 1564–1577, 2022.

[11] Z. Luo, Z. Zeng, W. Tang, J. Wan, Z. Xie, and Y. Xu, "Dense dual-branch cross attention network for semantic segmentation of large-scale point clouds," *IEEE Transactions on Geoscience and Remote Sensing*, vol. 62, pp. 1–16, 2024.

[12] J. He, J. Deng, T. Zhang, Z. Zhang, and Y. Zhang, "Hierarchical shape-consistent transformer for unsupervised point cloud shape correspondence," *IEEE Transactions on Image Processing*, vol. 32, pp. 2734–2748, 2023.

[13] S. Schwarz, M. Preda, V. Baroncini, M. Budagavi, P. Cesar, P. A. Chou, R. A. Cohen, M. Krivokuća, S. Lasserre, Z. Li, *et al.*, "Emerging mpeg standards for point cloud compression," *IEEE Journal on Emerging and Selected Topics in Circuits and Systems*, vol. 9, no. 1, pp. 133–148, 2018.

[14] Mammou, "Mpeg 3d graphics coding. g-pcc test model v27.," in *Output document N18189. ISO/IEC MPEG (JTC 1/SC 29/WG 11)*, 145th MPEG meeting, 2024.

[15] 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 Transactions on Broadcasting*, vol. 66, no. 3, pp. 701–717, 2020.

[16] M. Cui, J. Long, M. Feng, B. Li, and H. Kai, "Octformer: Efficient octree-based transformer for point cloud compression with local enhancement," in *Proceedings of the AAAI Conference on Artificial Intelligence*, vol. 37, pp. 470–478, 2023.

[17] J. Wang, D. Ding, Z. Li, X. Feng, C. Cao, and Z. Ma, "Sparse tensor-based multiscale representation for point cloud geometry compression," *IEEE Transactions on Pattern Analysis and Machine Intelligence*, 2022.

[18] C. Fu, G. Li, R. Song, W. Gao, and S. Liu, "Octattention: Octree-based large-scale contexts model for point cloud compression," in *Proceedings of the AAAI Conference on Artificial Intelligence*, 2022.

[19] R. Song, C. Fu, S. Liu, and G. Li, "Efficient hierarchical entropy model for learned point cloud compression," in *Proceedings of the IEEE/CVF Conference on Computer Vision and Pattern Recognition*, pp. 14368–14377, 2023.

[20] A. Luo, L. Song, K. Nonaka, K. Unno, H. Sun, M. Goto, and J. Katto, "Scp: Spherical-coordinate-based learned point cloud compression," in *Proceedings of the AAAI Conference on Artificial Intelligence*, vol. 38, pp. 3954–3962, 2024.

[21] M. Quach, G. Valenzise, and F. Dufaux, "Learning convolutional transforms for lossy point cloud geometry compression," in *2019 IEEE International Conference on Image Processing (ICIP)*, pp. 4320–4324, 2019.

[22] J. Wang, H. Zhu, H. Liu, and Z. Ma, "Lossy point cloud geometry compression via end-to-end learning," *IEEE Transactions on Circuits and Systems for Video Technology*, vol. 31, no. 12, pp. 4909–4923, 2021.

[23] D. T. Nguyen, M. Quach, G. Valenzise, and P. Duhamel, "Learning-based lossless compression of 3d point cloud geometry," in *ICASSP 2021 - 2021 IEEE International Conference on Acoustics, Speech and Signal Processing (ICASSP)*, pp. 4220–4224, 2021.

[24] Z. Wei, B. Niu, H. Xiao, and Y. He, "Isolated points prediction via deep neural network on point cloud lossless geometry compression," *IEEE Transactions on Circuits and Systems for Video Technology*, vol. 33, no. 1, pp. 407–420, 2023.

[25] S. Biswas, J. Liu, K. Wong, S. Wang, and R. Urtasun, "Muscle: Multi sweep compression of lidar using deep entropy models," in *Advances in Neural Information Processing Systems 33: Annual Conference on Neural Information Processing Systems 2020, NeurIPS 2020, December 6-12, 2020, virtual* (H. Larochelle, M. Ranzato, R. Hadsell, M. Balcan, and H. Lin, eds.), 2020.

[26] F. Song, Y. Shao, W. Gao, H. Wang, and T. Li, "Layer-wise geometry aggregation framework for lossless lidar point cloud compression," *IEEE Transactions on Circuits and Systems for Video Technology*, vol. 31, no. 12, pp. 4603–4616, 2021.

[27] T. Fan, L. Gao, Y. Xu, D. Wang, and Z. Li, "Multiscale latent-guided entropy model for lidar point cloud compression," *IEEE Transactions on Circuits and Systems for Video Technology*, vol. 33, no. 12, pp. 7857–7869, 2023.

[28] Z. Wang, S. Wan, and L. Wei, "Local geometry-based intra prediction for octree-structured geometry coding of point clouds," *IEEE Transactions on Circuits and Systems for Video Technology*, vol. 33, no. 2, pp. 886–896, 2023.

[29] C. Sun, H. Yuan, X. Mao, X. Lu, and R. Hamzaoui, "Enhancing octree-based context models for point cloud geometry compression with attention-based child node number prediction," *IEEE Signal Processing Letters*, vol. 31, pp. 1835–1839, 2024.
[30] C. Sun, H. Yuan, S. Li, X. Lu, and R. Hamzaoui, "Enhancing context models for point cloud geometry compression with context feature residuals and multi-loss," *IEEE Journal on Emerging and Selected Topics in Circuits and Systems*, vol. 14, no. 2, pp. 224–234, 2024.
[31] Z. Que, G. Lu, and D. Xu, "Voxelcontext-net: An octree based framework for point cloud compression," in *Proceedings of the IEEE/CVF Conference on Computer Vision and Pattern Recognition*, pp. 6042–6051, 2021.
[32] J. Behley, M. Garbade, A. Milioto, J. Quenzel, S. Behnke, C. Stachniss, and J. Gall, "Semantickitti: A dataset for semantic scene understanding of lidar sequences," in *Proceedings of the IEEE/CVF International Conference on Computer Vision*, pp. 9297–9307, 2019.
[33] G. Pandey, J. R. McBride, and R. M. Eustice, "Ford campus vision and lidar data set," *The International Journal of Robotics Research*, vol. 30, no. 13, pp. 1543–1552, 2011.
[34] J. Wang, H. Zhu, Z. Ma, T. Chen, H. Liu, and Q. Shen, "Learned point cloud geometry compression," *arXiv preprint arXiv:1909.12037*, 2019.
[35] A. F. Guarda, N. M. Rodrigues, and F. Pereira, "Point cloud coding: Adopting a deep learning-based approach," in *2019 Picture Coding Symposium (PCS)*, pp. 1–5, IEEE, 2019.
[36] E. Alexiou, K. Tung, and T. Ebrahimi, "Towards neural network approaches for point cloud compression," in *Applications of digital image processing XLIII*, vol. 11510, pp. 18–37, SPIE, 2020.
[37] M. Quach, G. Valenzise, and F. Dufaux, "Improved deep point cloud geometry compression," in *2020 IEEE 22nd International Workshop on Multimedia Signal Processing (MMSP)*, pp. 1–6, 2020.
[38] J. Wang, H. Zhu, H. Liu, and Z. Ma, "Lossy point cloud geometry compression via end-to-end learning," *IEEE Transactions on Circuits and Systems for Video Technology*, pp. 1–1, 2021.
[39] A. F. R. Guarda, N. M. M. Rodrigues, and F. Pereira, "Adaptive deep learning-based point cloud geometry coding," *IEEE Journal of Selected Topics in Signal Processing*, vol. 15, no. 2, pp. 415–430, 2021.
[40] P. Yu, D. Zuo, Y. Huang, R. Huang, H. Wang, Y. Guo, and F. Liang, "Sparse representation based deep residual geometry compression network for large-scale point clouds," in *2023 IEEE International Conference on Multimedia and Expo (ICME)*, pp. 2555–2560, 2023.
[41] Z. Liang, J. Liu, M. Dasari, and F. Wang, "Fumos: Neural compression and progressive refinement for continuous point cloud video streaming," *IEEE Transactions on Visualization and Computer Graphics*, vol. 30, no. 5, pp. 2849–2859, 2024.
[42] Z. Luo, W. Jia, and S. Perry, "Transformer-based geometric point cloud compression with local neighbor aggregation," in *2023 International Conference on Digital Image Computing: Techniques and Applications (DICTA)*, pp. 223–228, 2023.
[43] T. Huang and Y. Liu, "3d point cloud geometry compression on deep learning," in *Proceedings of the 27th ACM international conference on multimedia*, pp. 890–898, 2019.
[44] 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 *2020 IEEE International Conference on Multimedia and Expo (ICME)*, pp. 1–6, IEEE, 2020.
[45] L. Gao, T. Fan, J. Wan, Y. Xu, J. Sun, and Z. Ma, "Point cloud geometry compression via neural graph sampling," in *2021 IEEE International Conference on Image Processing (ICIP)*, pp. 3373–3377, IEEE, 2021.
[46] C. R. Qi, H. Su, K. Mo, and L. J. Guibas, "Pointnet: Deep learning on point sets for 3d classification and segmentation," in *Proceedings of the IEEE conference on computer vision and pattern recognition*, pp. 652–660, 2017.
[47] C. R. Qi, L. Yi, H. Su, and L. J. Guibas, "Pointnet++: Deep hierarchical feature learning on point sets in a metric space," *Advances in neural information processing systems*, vol. 30, 2017.
[48] J. Zhang, T. Chen, D. Ding, and Z. Ma, "Yoga: Yet another geometry-based point cloud compressor," in *Proceedings of the 31st ACM International Conference on Multimedia*, pp. 9070–9081, 2023.
[49] R. Song, C. Fu, S. Liu, and G. Li, "Efficient hierarchical entropy model for learned point cloud compression," in *Proceedings of the IEEE/CVF Conference on Computer Vision and Pattern Recognition*, pp. 14368–14377, 2023.
[50] P. Velickovic, G. Cucurull, A. Casanova, A. Romero, P. Lio', and Y. Bengio, "Graph attention networks," *ArXiv*, vol. abs/1710.10903, 2017.
[51] A. Dosovitskiy, L. Beyer, A. Kolesnikov, D. Weissenborn, X. Zhai, T. Unterthiner, M. Dehghani, M. Minderer, G. Heigold, S. Gelly, *et al.*, "An image is worth 16x16 words: Transformers for image recognition at scale," *arXiv preprint arXiv:2010.11929*, 2020.
[52] WG7, *Common test conditions for g-pcc*. ISO/IEC JTC1/SC29/WG11 N00106, 2021.
[53] H. Fan, H. Su, and L. Guibas, "A point set generation network for 3d object reconstruction from a single image," in *2017 IEEE Conference on Computer Vision and Pattern Recognition (CVPR)*, pp. 2463–2471, 2017.
[54] T. Huang and Y. Liu, "3d point cloud geometry compression on deep learning," *Proceedings of the 27th ACM International Conference on Multimedia*, 2019.
[55] G. Bjøntegaard, "Calculation of average psnr differences between rd-curves," in *ITU-T SG 16/Q6, 13th VCEG Meeting*, April 2001.
[56] S. Shi, Z. Wang, J. Shi, X. Wang, and H. Li, "From points to parts: 3d object detection from point cloud with part-aware and part-aggregation network," *IEEE transactions on pattern analysis and machine intelligence*, vol. 43, no. 8, pp. 2647–2664, 2020.
[57] Q. Hu, B. Yang, L. Xie, S. Rosa, Y. Guo, Z. Wang, N. Trigoni, and A. Markham, "Randla-net: Efficient semantic segmentation of large-scale point clouds," in *Proceedings of the IEEE/CVF Conference on Computer Vision and Pattern Recognition*, pp. 11108–11117, 2020.

Mingyue Cui received a B.Sc. degree in Software Engineering from Chongqing Normal University in 2014, and received an M.Sc. degree in Software Engineering, and a Ph.D. degree in Computer Science from Sun Yat-sen University, Guangzhou, Guangdong, China, in 2017, and 2022, respectively. He was a Visiting Student at the Technical University of Munich, Germany (November 2021–November 2022). He is currently working as a joint Postdoctoral Fellow with the School of Computer Science and Engineering at Sun Yat-Sen University, China. His research interests are in the area of 3D vision and entropy coding.

Yuyang Zhong graduated with the B.E. degree in intelligent science and technology, Sun Yat-sen University, Shenzhen, China, in 2024. Presently, he is pursuing an M.Sc. degree in artificial intelligence at the College of Computing and Data Science, Nanyang Technological University, Singapore.

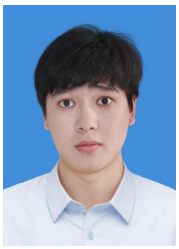
His areas of research interest encompass 3D computer vision, deep learning, and entropy encoding.

Mingjian Feng received the B.Sc degree from Sun Yat-sen University, Guangzhou, China, in 2022. He is currently working toward an M.Sc. degree at Sun Yat-sen University, majoring in computer science. He joined the Robotics and Intelligence Computing Lab and is currently a student under the instruction of Kai Huang.

His research interests include 3D computer vision, autonomous driving, and point cloud coding.

Junhua Long received the B.Sc. degree in computer science and technology from Chongqing University, Chongqing, China, in 2021. He is currently pursuing an M.Sc. degree in computer science and technology from the School of Computer Science and Engineering, Sun Yat-sen University, Guangzhou, China, under the instruction of Kai Huang.

His research interests include 3D computer vision, deep learning, and point cloud compression.



Yehua Ling joined Guangxi Transportation and Science Group Co. LTD as an engineer in 2022. He received his PhD degree in Computer Science and Technology in 2022, from Sun Yat-sen University, Guangzhou, China. He received his B.E. degree in Automation in 2011, from Guilin University of Technology, Guilin, China, and his M.S. degree in Control Theory and Control Engineering in 2014, from Chang'an University, Xian, China. His current research interest includes intelligent robot, and hardware accelerator system design.

Jiahao Xu received the B.Eng. degree in intelligence science and technology from Sun Yat-sen University, Shenzhen, China, in 2024. He is currently pursuing an M.Eng degree in electronic and information engineering from the School of Intelligent Systems Engineering, Sun Yat-sen University, Shenzhen, China, under the instruction of Jiangshan Cheng. His research interests include adversarial training, computer vision, and point cloud compression.

Kai Huang (Member, IEEE) received the B.Sc. degree from Fudan University, Shanghai, China, in 1999, the M.Sc. degree from Leiden University, Leiden, The Netherlands, in 2005, and the Ph.D. degree from ETH Zurich, Zurich, Switzerland, in 2010. He joined Sun Yat-sen University as a Professor in 2015. He was appointed as the Director of the School of Computer Science and Engineering, Institute of Unmanned Systems, in 2016. He was a Senior Researcher with the Computer Science Department, Technical University of Munich, Munich, Germany, from 2012 to 2015, and the Research Group Leader of Fortiss GmbH, Munich, in 2011. His research interests include techniques for the analysis, design, and optimization of embedded/systems/cyber-physical systems (CPS), particularly in the automotive, medical, and robot domains.

Dr. Huang has served as a member of the Technical Committee on Cybernetics for Cyber-Physical Systems of the IEEE Systems, Man, and Cybernetics (SMC) Society, Embedded Systems for China Computer Federation, Vehicle Control, and Intelligence for Chinese Association Automation, and Robotic Intelligence for Chinese Association of Automation. He received the Youth Overseas High-Level Talent Introduction Plan 2014 and was granted the Chinese Government Award for Outstanding Self-Financed Students Abroad 2010. He was a recipient of Best (Student) Paper/Candidates Awards IEEE Intelligent Vehicles Symposium (IV) 2018, Human-Friendly Robotics (HFR) 2018, the IEEE International Conference on Robotics and Biomimetics (ROBIO) 2017, the National Conference on Embedded System Technology (ESTC) 2017, Embedded Systems for Real-Time Multimedia (ESTIMedia) 2013, the IEEE International Conference on Embedded Computer Systems: Architectures, Modeling and Simulation (IC-SAMOS) 2009, ESTIMdedia 2009, and the General Chairs' Recognition Award for Interactive Papers in the IEEE Conference on Decision and Control (CDC) 2009. He also serves as the Deputy Secretary for the Technical Committee on Intelligent Robotics of the China Computer Federation.