DAPCC: Diverse Attention-Based Entropy Model for Dynamic LiDAR Point Cloud Compression

Mingyue Cui, *Associate Member, IEEE*, Yuyang Zhong, Mingjian Feng, Yehua Ling, Junhua Long, Jinhong Xia, and Kai Huang, *Member, IEEE*

***Abstract*— LiDAR point cloud (LPC) compression is an indispensable component for 3-D vision tasks, especially for dynamic point clouds. However, the existing methods based on traditional spatial–temporal attention (STA) are immature, causing little improvement in interframe feature extraction. In this article, we propose diverse attention-based point cloud compression (DAPCC), an LPC compression entropy model combining aggregation embedding (AggEmb) modules for temporal point matching and STA blocks for dynamic Octree node encoding, which can effectively utilize the change information of dynamic point clouds. Specifically, we first introduce AggEmb to match the Octree sequences from two sweeps to establish temporal correlation. To effectively capture the feature details, we further design local and global combined attention for the spatial–temporal information of point clouds which can focus on the whole context. Finally, we organize a symmetric multilayer perception (MLP) module capable of strengthening vital features. We conduct experiments of static and dynamic compression on both indoor/outdoor point cloud benchmark datasets (i.e., ScanNet, SemanticKITTI, and MPEG common test conditions (CTCs) Category 3 datasets) and downstream applications (i.e., vehicle detection and semantic segmentation). Compared with the previous state-of-the-art methods, our method achieves up to 14.7% bits per point (bpp) and 45% decoding time savings and adapts to the downstream tasks with almost no impact on performance.**

***Index Terms*— 3-D version, autonomous driving, diverse attention, entropy model, LiDAR point cloud (LPC) compression, Octree.**

I. Introduction

LiDAR point cloud (LPC) is undergoing rapid development and becoming popular in robots and autonomous driving vehicles for capturing the 3-D geometry of scenes, benefiting from its high accuracy and resolution [1], [2], [3], [4]. However, the large-volume point cloud brings a serious challenge in storage and transmission. For example, a single Velodyne LiDAR of HDL64 generates over 100 000 points per sweep, and about 2.88 million points are generated per second [5]. Therefore, it is necessary to develop an efficient compression method for point clouds.

Fortunately, there have been many methods [6], [7], [8] of LPC compression in recent years. Quach et al. [9] proposed a data-driven point cloud geometry compression method using learned convolutional transforms and a uniform quantization. Nguyen et al. [10] further extended the 2-D images autoregressive generative model to the 3-D voxel space, and incorporate 3-D data augmentation to exploit the redundancies between points. In addition, Wiesmann et al. [11] proposed an end-to-end compressing network to exploit the occurrence of common structures, which learns a small and compact set of local feature descriptors for compressing and reconstructing point clouds. In general, these methods focus more on the local geometry details and spatial correlations of static point clouds, ignoring the temporal characteristics of interframe point clouds.

Compared with point cloud coding using voxel grids, Octree has more advantages due to its higher coding efficiency [12], [13], [14]. For Octree-based LPC compression, there are mainly two challenges. First, how to efficiently extract strong prior information between spatially neighboring nodes, particularly for the relationship between sibling nodes? Sibling nodes provide low-level local geometry features that are essential for exploiting geometry redundancy. However, unlike ancestor nodes' relationships, it is difficult to obtain the neighbor geometric relationship of sibling nodes by traversing Octree directly. Second, how to effectively reduce interframe redundancy between temporal neighboring nodes (as shown in Fig. 1)? Different from image and video compression that can learn dependencies directly through motion information [15], LPCs are sparse and varying in point numbers, making it difficult to achieve accurate interframe registration [16]. In addition, during the registration process, some outliers would be lost, which significantly reduces the quality of point cloud reconstruction [17].

Recently, OctSqueeze [5] proposes an Octree-based deep entropy model that gathers context information of ancestor nodes to generate occupancy distribution, which achieves higher point cloud coding efficiency. Benefiting from the

Received 11 August 2024; revised 27 November 2024, 26 March 2025, and 15 May 2025; accepted 18 May 2025. Date of publication 23 May 2025; date of current version 16 June 2025. This work was supported in part by Guangxi Key Research and Development Program under Grant GuikeAB24010324; in part by Guangdong Basic and Applied Basic Research Foundation under Grant 2025A1515011485; and in part by the Open Project Program for the Engineering Research Center of Software/Hardware Co-Design Technology and Application, Ministry of Education, East China Normal University, under Grant 67000-42990016. *(Corresponding author: Kai Huang.)*

Mingyue Cui, Mingjian Feng, Junhua Long, Jinhong Xia, and Kai Huang are with the School of Computer Science and Engineering, Sun Yat-sen University, Guangzhou 400100, China (e-mail: cuimy@mail2.sysu.edu.cn; huangk36@mail.sysu.edu.cn).

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

Yehua Ling is with Guangxi Transportation Science and Technology Group Company Ltd., Nanning 530029, China.

Digital Object Identifier 10.1109/TGRS.2025.3573206

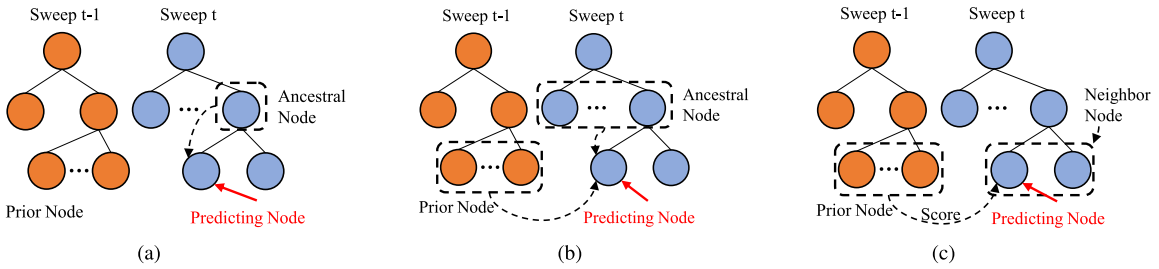


Fig. 1. Comparison of feature extraction between different methods. (a) OctSqueeze [5] processes features from the corresponding ancestral node in the same sweep. (b) STAEM [18] processes features from the large-scale context of prior neighbor nodes and ancestral node sequences by graph network. (c) DAPCC processes features from the prior and current neighbor nodes by adaptive dependency calculation.

above work, many researchers [18], [19], [20], [21], and [22] consider how to learn node context information in entropy model efficiently. However, previous works focus on extracting features from individual nodes through a stack of multilayer perceptions (MLPs). They embed the bounding box coordinates of Octree nodes into the feature space without considering the geometric features present in the joint distribution of coordinates, resulting in the inability to utilize dynamic features. Actually, nodes on similar geometry patterns from consecutive frames tend to have similar distributions, which is crucial for occupancy prediction. This requires an efficient LPC compression method that fully utilizes node spatial information and the context temporal variation.

To address these problems, this article proposes a diverse attention-based point cloud compression called DAPCC, which is a novel dynamic LPC compression model based on diverse improved attention mechanisms. DAPCC employs aggregation embedding (AggEmb) modules to process temporal features, along with spatial–temporal attention (STA) blocks for encoding dynamic Octree nodes. In the AggEmb module, we process temporal features by concatenating current sibling nodes with reference neighbor nodes that are selected by calculating node dependency, which effectively introduces dynamic point information. To capture local features and global dependencies, improved spatial attention (SA) and temporal attention (TA) are utilized to process the aggregated node features, which benefits the prediction of occupancy. In addition, we design a symmetric U-shaped MLP module (U-MLP) with downsample and upsample linear layers for each block to project the attention output and enhance feature representation. Finally, we compare our methods on both indoor ScanNet [23], outdoor SematicKITTI [24], and MPEG CTC Category 3 datasets (dynamic acquisition) [25]. The experimental results show that our method outperforms other state-of-the-art methods [5], [18], [19], [20], [21], [22], [26], [27], [28], [29], which are only designed for a specific category of point clouds. For downstream applications (i.e., vehicle detection and semantic segmentation), our DAPCC also obtains low bitrate while maintaining high performance. Overall, our contributions are summarized as follows.

1) We propose DAPCC, a novel compression model for dynamic LPCs, which introduces an AggEmb module for temporal feature generating and converted STA blocks for dynamic dependency extraction.
2) Our embedding module concentrates on interframe reliance and matches the high-relevance prior neighbors with the current sibling nodes, which can improve the effectiveness of aggregated features.
3) The innovative STA block is designed to separately excavate local and global dependencies, which is beneficial for aggregated context feature sharing.
4) We apply a simple module with symmetric MLPs to project the output of the attention block and enhance the node expression.

II. Related Works

A. Structured Point Cloud Coding

Structured data formats intuitively present the point cloud data and are suitable for convolutional operations. Those methods are mainly divided into image-based methods [30], [31] and voxel-based methods [9], [20], [32]. Houshiar and Nüchter [33] proposed to project points onto three panorama images and use an image coding method to compress them. VPCC [34] divides the point cloud into connected regions called 3-D patches, which are then projected independently into 2-D patches and further compressed with HEVC [35]. Inspired by the above work, Sun et al. [36] divided the frames into intraframes and interframes. They code the intraframes with the technique proposed in [30] and code interframes with a prediction network using convolutional LSTM cells. Wang et al. [37] further proposed R-PCC, a region-based clustering method using farthest point sampling to represent point clouds. Based on voxel, Wang et al. [37] presented a novel static data-driven geometry compression method for point clouds based on learned convolutional transforms and uniform quantization. Furthermore, Kaya et al. [32] proposed a lossless compression method, where the first stage is constructing a bounding volume for the point cloud and the following steps succeed at adding all the remaining points.

B. Unstructured Point Cloud Coding

Point-based method is a direct way of unstructured point cloud organization [38], [39], [40], [41]. PointNet [42] proposes the first deep learning-based framework that learns directly on points with pointwise MLP. PointNet++ [43] further considers local information by proposing two abstraction layers that intelligently aggregate multiscale information according to local point densities. PointConv [44] and

KPConv [45] propose pointwise convolutional operators with points convoluted with neighbors. Furthermore, the attention-based method PCT [46] replaces the encoder layers in the PointNet [42] with self-attention layers. On the basis of the above work, PointFormer [47] learns features effectively by leveraging the superiority of the Transformer models on set-structured data. To tackle different input shapes, PatchDPCC [48] designs a patch grouping module coupled with a deep compression model. This patch grouping module segments point cloud frames into fixed-size patches and groups patches representing the same object, thereby leveraging interframe similarity and handling frames with arbitrary point counts.

Another typical unstructured point cloud organization is the Octree-based method, which offers higher coding efficiency through hierarchical representation [49], [50], [51]. GPCC [26] proposes using a hand-crafted context-adaptive arithmetic encoder for bit allocation, which predicts the currently encoding node based on the coded information. Garcia et al. [49] further developed a super-resolution technique that utilizes Octree to generate potential contexts that can be arithmetically encoded. MuSCLE [19] exploits spatial–temporal correlations within the data by taking both coarse-level information at the current frame and relevant neighboring node information from the previous frame. Based on the GPCC, the GPCC++ [52] employs k-nearest neighbor (kNN) techniques to enhance the geometric and attribute characteristics of data, which incorporates GeoNet and AttNet to refine the geometric and color information, respectively. Similarly, the YOGA [53] uses multiscale sparse convolution to compress geometric and attribute features of point cloud data and encodes the downsampled point cloud thumbnail with GPCC at the foundational layer. Inspired by the Trisoup mode of GPCC [26], Pointsoup [54] proposes a point model-based lightweight geometry codec model, using the aligned downsampling module with the attention-based aggregation for local point cloud surface characterization, and the upsampling blocks combining dilated entropy model for further feature distribution estimation. Duan et al. [55] used a density-adaptive compression method according to the LPC characteristic, which can adjust the sample and compress resolution by the range of point data. COT-PCC [56] introduces a generative adversarial network for modeling the rate distribution in compression, which applies a density-sensitive encoder for local distribution learning, with the edge convolution and point transformer for point feature sampling. However, these methods mainly aim to achieve lightweight low-decoding-latency point cloud compression, which is not suitable for high-quality LPC reconstruction.

Recently, OctSqueeze [5] proposed the first Octree-based deep entropy model for point cloud compression. It uses stacks of MLPs to extract tree node features and predicts the distribution of a node symbol, given the features of the ancestor nodes. The VoxelContext-Net model [20] further incorporates voxel context into a tree-based deep learning framework and utilizes 3-D convolution on the generated local voxel context, which captures the spatial information of the surrounding area within the Octree structure. SibContext [57] uses a voxel-based geometry-aware module to fit quadratic surfaces and enhance the use of geometric priors in entropy coding. SparsePCGC [27] further uses sparse convolutions to deal with the unstructured points directly, and proposes the resolution scaling strategy to exploit the cross-scale correlation based on multiscale representation. OctAttention [21] proposes utilizing a conditional entropy model with a large receptive field to capture ancestor and sibling nodes, exploiting strong dependencies among neighboring nodes. In our previous work, we propose the Octree-based deep entropy model called OctFormer [22], which shares the results within multihead self-attention (MSA) operation among sibling nodes and achieves efficient encoding and decoding. Based on the above work, EHEM [28] proposed using grouped attention to extract features from the large-scale context of ancestor and sibling nodes to reduce the compression redundancy, and serial coding to accelerate the decoding process. Based on EHEM, SCP [58] further introduces a model-agnostic spherical-coordinate-based point cloud compression method and a multilevel Octree to mitigate reconstruction errors in distant voxels. Furthermore, STAEM [18] adopts a graph-based feature extraction model to exploit the geometric features of the point cloud and employs attention models to emphasize dependent nodes in the large-scale spatial–temporal context. DuOctree [29] explores a two-Octree structure for LPC representation and a cross-attention entropy model to discover the hierarchical spatial dependencies. Compared with the above methods, our proposed DAPCC has the following benefits.

1) Compared with OctSqueeze [5], MuSCLE [19], and VoxelContext-Net [20], we introduce attention mechanics to capture essential connections between the Octree nodes, which eliminates the need for computationally intensive 3-D convolutional operations.
2) Compared to SparsePCGC [27], OctAttention [21], OctFormer [22], EHEM [28], and DuOctree [29], we propose a novel AggEmb module to match interframe nodes by relevance, which achieves better compression performance for dynamic point clouds.
3) Compared with STAEM [18], we design the revised SA and TA to focus on local and global dependencies of aggregated features, which do not rely on increasing the layers of embedding and STA to achieve bitrate savings, as shown in Fig. 1.

III. Methodology

As shown in Fig. 2, we propose a dynamic LPC compression model with diverse attention mechanisms called DAPCC. We construct Octree from point cloud data and use prior neighbor nodes as reference information to help predict the occupancy distributions of the current sequence. For each encoding block, an AggEmb module is used for node matching, and an STA block is used for dynamic information learning and intrafeature enhancement. The aggregation module searches reference neighbors and uses a dependency calculation to select prior nodes for matching, which introduces temporal information for occupancy prediction. To utilize the dynamic correlation of aggregated features, SA is designed to focus on local dependencies between two

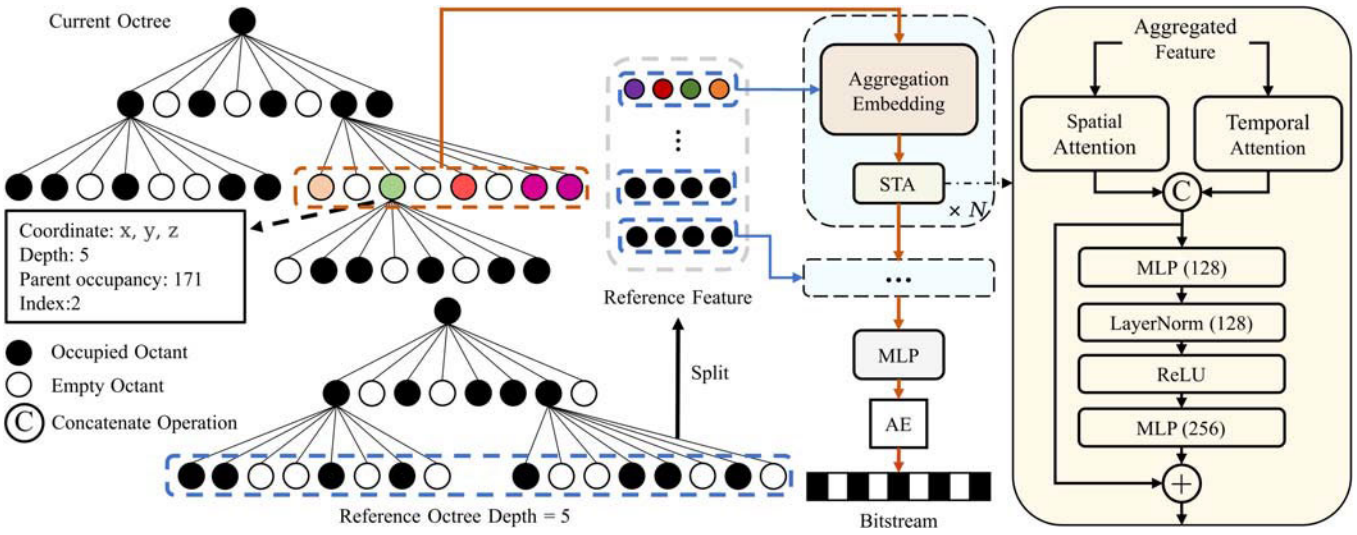


Fig. 2. Overview of DAPCC. Given the reference and current Octree sequences, we feed them into the AggEmb module to introduce matched prior nodes. Then, we serve the aggregated context as input for the STA block where features are extracted by temporal and SA to discover node dependencies. In each STA block, a UM is employed to improve crucial features. Finally, the output occupancy distribution is passed for lossless arithmetic encoding and becomes bitstream.

sequences while TA is used to perceive global temporal variance. In addition, we add a symmetric MLP module after the attention mechanism to capture feature dependencies. By arithmetic encoder, we encode predicted occupancy into the bitstream losslessly and make parallel decoding to reconstruct the point cloud.

A. Octree Constructing

With the purpose of making high-compress efficiency for the sparse point cloud data, we organize raw point clouds by transforming them into Octree. The Octree from a single point cloud frame is generated by dividing the space into eight equal cubes until the maximum depth or an empty cube is reached. The occupancy symbol ranges from 0 to 255, denoting the occupancy status of the children nodes with its 8-bit binary code. As shown in Fig. 3, we can observe that Octree nodes from close regions usually have similar children node occupancy, which reduces the difficulty of predicting node distribution and makes more efficient compression.

In the Octree construction procedure, the coordinate of the cube center is typically used to represent each node, introducing a quantization error e. Given the normalized point cloud $P = \{p_1, p_2, \ldots, p_i, \ldots\}$, we quantify it into the leaf-nodes' center points $\hat{P}$

$$L = \max(P) - \min(P) \tag{1}$$

$$\hat{P} = \frac{L}{2^D}\left\lfloor \frac{P \cdot 2^D}{L} \right\rfloor + \frac{L}{2^{D+1}} \tag{2}$$

where L is the length of the bounding box and D is the maximum depth level in the Octree structure, $\lfloor \cdot \rfloor$ represents the floor operation. It should be noted we need to calculate the bounding box length along the x-, y-, and z-axes, respectively. L can represent the length of the bounding box along the x-, y-, or z-axes and be calculated with corresponding max and min values on point cloud data P at the x-, y-, or z-axes. The quantization error e between per point can be further described as follows:

$$e = \max_i \left\| p_i, \hat{p}_i \right\|_\infty \leq \frac{L}{2^{D+1}} \tag{3}$$

where p_i is the coordinate from point cloud P and $\hat{p}_i$ is the coordinate of cube center.

By breath-first Octree traversal, we serialize nonoverlapped sequences from the Octree. The features f_a of the node n_a and the context $\mathbf{c}$ is defined as follows:

$$\begin{aligned} f_a &= \left[\text{coord}_a, d_a, \text{in}_a, p_a\right] \\ \mathbf{c} &= \left[f_a, f_{a+1}, \ldots, f_{a+w-1}\right] \end{aligned} \tag{4}$$

where coord_a, d_a, in_a, and p_a individually indicate bounding box coordinates, Octree depth, index, and parent occupancy of n_a, and w indicates context window size.

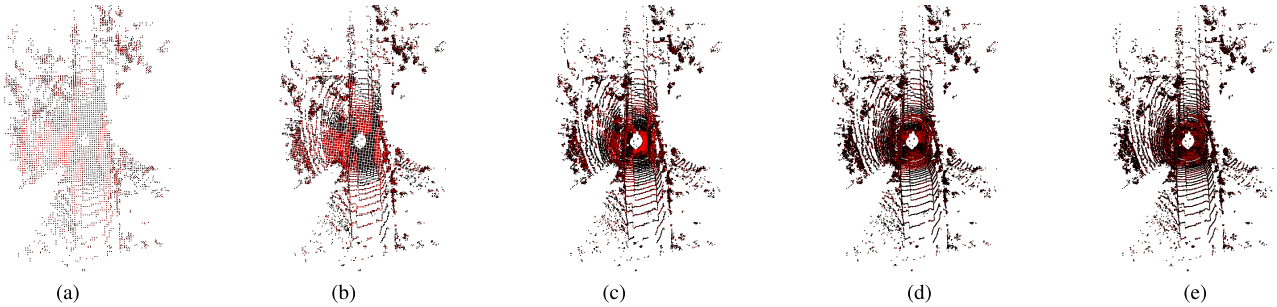
The objective of our model is to reduce the cross entropy between estimated distribution $\hat{Q}$ and ground truth Q, expressed as $\mathbb{E}_{n\sim Q}[-\log_2 \hat{Q}(n)]$. Assuming conditional independence, the joint distribution $\hat{Q}(n)$ is factorized as follows:

$$\hat{Q}(n) = \prod_a \hat{q}_a\left(n_a | f_a, \ldots, f_{a+w-1}, f'_a, \ldots, f'_{a+w-1}; W\right) \tag{5}$$

where n_j is the occupancy of node, f_a and f'_a indicate embedding current features and reference features, and W indicates the weights of entropy model.

B. Aggregation Embedding

To extract the correlation between the reference and current frames effectively, we propose the AggEmb module to calculate the dependency score between prior neighbors and the predicting context. The whole module is shown in Fig. 4. We first extract all the nonempty nodes under the same depth from the encoded Octree as the reference context. Since the context length tends to be longer at greater depths, we divide it into multiple subcontexts with shorter context lengths.



(a) (b) (c) (d) (e)

Fig. 3. Visualization of Octree occupancies at different depths. We take the center of the Octree leaf nodes to generate point clouds. Each point's color is determined by the node's occupancy value, with similar colors being similar occupancies. As the depth increases, the point cloud becomes denser. (a) Depth = 8. (b) Depth = 9. (c) Depth = 10. (d) Depth = 11. (e) Depth = 12.

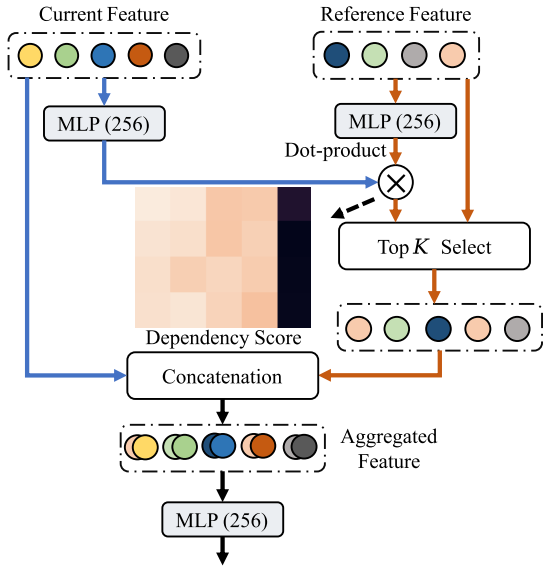


Fig. 4. Our proposed AggEmb module. The heat map reflects the correlation of each reference and the current node pair.

To ensure that each subcontext can be selected as a reference, we maintain the number of subcontexts equal to the number of embedding modules (i.e., N in Fig. 2) within the network during the division process. In each module, a nonrepeat subcontext is matched with the current context to ensure that all reference nodes are traversed. Another point to note is that the context features generated by breadth-first search show a segmented distribution. This results in significant differences between subcontexts and increases the error in relevance calculation between each subcontext and the current context. To solve this problem, we randomly shuffle the nodes in the reference context before the division operation, ensuring a more even and normalized distribution of nodes within each subcontext.

Both the reference subcontext $c'_j = [f'_j, f'_{j+1}, \ldots, f'_{j+n-1}]$ and current context $c_i = [f_i, f_{i+1}, \ldots, f_{i+m-1}]$ are then organized through a linear layer to extract the hidden features $h'_j = \delta(c'_j; \rho)$ and $h_i = \delta(c_i; \rho)$, where $\delta(.; \rho)$ denotes the MLP with learnable parameter ρ. By performing the dot-product operation between the subcontext features and the encoding window features, we can obtain the dependency score S between each pair of nodes in the two contexts

$$S = \begin{bmatrix} s_{ij} & s_{i(j+1)} & \cdots & s_{i(j+n-1)} \\ s_{(i+1)j} & \cdots & \cdots & s_{(i+1)(j+n-1)} \\ \cdots & \cdots & \cdots & \cdots \\ s_{(i+m-1)j} & \cdots & \cdots & s_{(i+m-1)(j+n-1)} \end{bmatrix}. \quad (6)$$

The score indicates the correlation for the node pair. According to the index sorted by the score, we use the top-k algorithm for each row of S to extract the highly correlated nodes corresponding to each predicting node from the subcontext and aggregate them to the current context window. The nodes with low scores are considered to have low relevance with the current node and are discarded. Suppose the hidden feature dimensions of the contexts h_i and h'_j are both d, it will increase to $d + d \times k$ after aggregation because each current node is concatenated with the highest k dependent prior neighbors. To keep the input and output dimensions consistent, we occupy an MLP with the parameter dimension $(d + d \times k, d)$ to process the aggregated window features.

C. Spatial–Temporal Attention

The vanilla self-attention mechanism applies linear projections to generate query, key, and value tokens [21], [59], [60], [61]. However, this method is not well-suited for breadth-first traversal Octree sequences, which typically exhibit more local correlations and fewer long-distance dependencies. Therefore, the STA module is designed to discover the distribution occupancy of Octree nodes in the spatial and temporal dimensions, respectively, utilizing context features that have accomplished the aggregation of interframe nodes. We combine SA to focus on the geometric and topological properties and TA to enhance the modeling of dynamic node features. As shown in Fig. 5, we improve the STA module by incorporating more efficient network layers to extract aggregated context features better. Specifically, we replace the traditional MLP with convolutional layers and average pooling operations in various attention modules to generate attention vectors. This modification improves the extraction of spatial and temporal dependencies and enhances overall model performance.

For each STA block, we separate it into an SA module and a TA module. As shown in Fig. 5(a), in the SA module, we first apply a convolutional layer to extract hidden feature dependencies from the input sequence. This step enhances the representation of Octree node dependencies within the

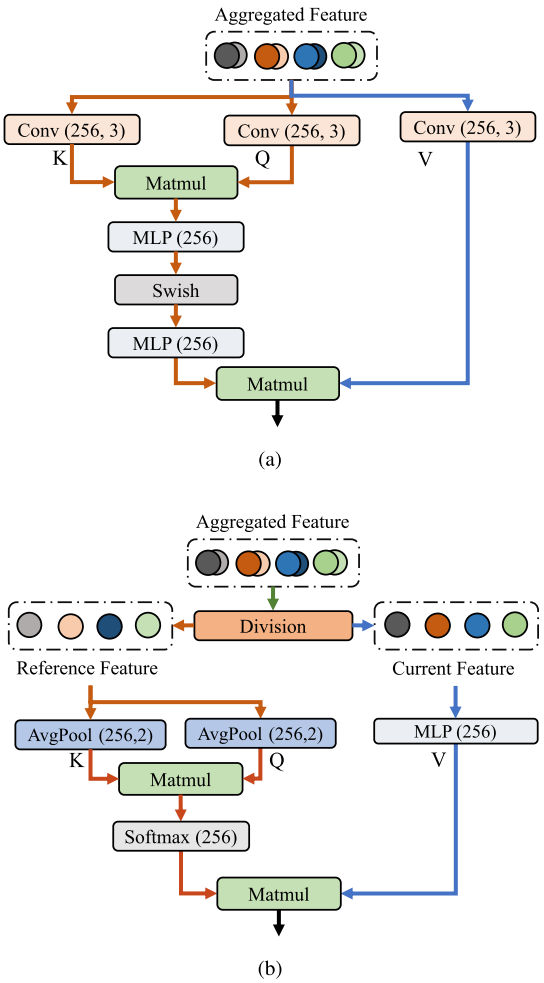


Fig. 5. Network architecture of two attention modules. The division operation in the TA module denotes the redivision of the aggregated features in the channel dimension, which is based on not changing the feature order of two contexts when they are concatenated in the AggEmb module. (a) SA module. (b) TA module.

local context. Considering that the kernel calculation of the convolutional layer can cause invalid weights on the edges of attention, after making a dot-product operation for the query and key, we use two MLPs to adjust the weights and enhance the robustness of the module. Finally, the result of SA is obtained by combining the values and MLP outputs. The spatial module $\mathrm{SA}(\mathbf{c_a})$ can be defined as follows:

$$\begin{aligned} \mathbf{Q}, \mathbf{K}, \mathbf{V} &= \mathrm{Conv}(\mathbf{c_a}) \\ \mathbf{Attn} &= \mathrm{MLP}(\mathbf{Q} \otimes \mathbf{K}) \\ \mathbf{Attn}' &= \mathrm{Swish}(\mathbf{Attn}) \\ \mathbf{Attn}'' &= \mathrm{MLP}(\mathbf{Attn}') \\ \mathbf{c}_{\mathrm{out}} &= \mathbf{Attn}'' \otimes \mathbf{V} \end{aligned} \tag{7}$$

where $\mathbf{Q}$, $\mathbf{K}$, and $\mathbf{V}$ are the query, key, and value tokens in attention mechanism, respectively, Conv and MLP indicate convolutional layer and MLP, respectively, $\mathbf{c_a}$ indicates aggregated context, Swish is the swish activation [62], and $\mathbf{c}_{\mathrm{out}}$ indicates output context.

As shown in Fig. 5(b), we further propose the TA module to model the dependencies of nodes with high similarity between two frames and adaptively match the occupancy of encoded nodes to the current context. Since AggEmb does not change the distribution order of features in the hidden layer dimension when extracting node relevance, we first separate reference node context and current node context from aggregated features by a division operation. To maintain low computational complexity, the reference node context is organized with the average pooling operation to generate the smaller scale query and key, while the current node context is operated with an MLP to generate the value token. Then, we apply the softmax function to get the weights of the dot-product value. The temporal module $\mathrm{TA}(\mathbf{c_t}, \mathbf{c_r})$ can be defined as follows:

$$\begin{aligned} \mathbf{c_t}, \mathbf{c_r} &= \mathrm{Dv}(\mathbf{c_a}) \\ \mathbf{Q}, \mathbf{K} &= \mathrm{AvgPool}(\mathbf{c_r}) \\ \mathbf{V} &= \mathrm{MLP}(\mathbf{c_t}) \\ \mathbf{Attn} &= \mathrm{Softmax}(\mathbf{Q} \otimes \mathbf{K}) \\ \mathbf{c}_{\mathrm{out}} &= \mathbf{Attn} \otimes \mathbf{V} \end{aligned} \tag{8}$$

where Dv and AvgPool indicate the division and average pooling operation, respectively. Supposing the original current and reference window size is w, the output size of the average pooling layer is w_0 and the feature dimensions are d. According to the attention mechanism, the computational complexity decreases from $O(w^2 * d)$ to $O(w_0^2 * d)$, contributing to a faster encoding and decoding process.

D. UM for Feature Enhancing

The larger parameters can increase the model generalization and improve occupancy prediction; however, they also introduce more redundant parameters for the network, making it difficult for training, thereby limiting the model's performance. Therefore, we add a symmetric U-shaped module after the encoding block, which uses MLP layers to reduce parameter redundancy. To reduce the number of channels, two consecutive linear layers are used to organize attention output. The first MLP reduces the dimension of the features and removes redundant parameters, while the second MLP increases the dimension to the original level.

Combining SA, TA , and the U-MLP module (UM), the entire STA block is defined as follows:

$$\begin{aligned} \mathbf{c}_{\mathbf{a}}' &= \mathrm{MLP}([\mathrm{SA}(\mathbf{c_a}), \mathrm{TA}(\mathbf{c_a})]) \\ \mathbf{c}_{\mathbf{a}}'' &= \mathrm{ReLU}\big(\mathrm{LN}\big(\mathbf{c}_{\mathbf{a}}' + \mathbf{c_a}\big)\big) \\ \mathbf{c}_{\mathbf{a}}''' &= \mathrm{MLP}\big(\mathbf{c}_{\mathbf{a}}''\big) + \mathbf{c}_{\mathbf{a}}'' \end{aligned} \tag{9}$$

where $\mathbf{c}_{\mathbf{a}}'$ indicates the linear output of the concatenation of spatial attention SA and temporal attention TA, and $\mathbf{c}_{\mathbf{a}}''$ represents the output after layer normalization [63] LN and ReLU activation for the addition of $\mathbf{c}_{\mathbf{a}}'$ and original aggregated features $\mathbf{c_a}$. $\mathbf{c}_{\mathbf{a}}'''$ is the final output.

E. Network Learning

For our entropy model, the loss function is defined in the following manner:

$$\ell = -\sum_i \log \hat{q}_a\big(n_a | f_a, \ldots, f_{a+w-1}, f_a', \ldots, f_{a+w-1}'; W\big) \tag{10}$$

where the estimated distribution of occupancy for the ath Octree node's occupancy n_a is represented by $\hat{q}_a(n_a|f_a, \ldots, f_{a+w-1}, f'_a, \ldots, f'_{a+w-1}; W)$.

IV. Experiments

In this section, we evaluate the performance of our proposed method on both indoor and outdoor point cloud benchmark datasets (i.e., ScanNet, SemanticKITTI, and MPEG CTC Category 3 datasets) and downstream applications (i.e., vehicle detection and semantic segmentation).

A. Experimental Setup

We evaluate our method on indoor ScanNet [23] dataset, outdoor SemanticKITTI [24] and MPEG CTC Category 3 datasets (dynamic acquisition) [25], respectively. For ScanNet, it contains more than 1500 noncontinuous frames captured from different scenes or objects, where each frame is independent. We use the same training/testing splits as ScanNet's benchmark tasks, which include 1045 as the training set, 156 as the validation set, and 312 as the testing set. We sample 50 000 points from each scan the same as [20]. For SemanticKITTI, it provides 22 sequences of continuous frames captured under the same road conditions over time which shows temporal movements, adding up to 43 504 scans, and each scan contains over 120 000 points. We adopt the official data splitting methodology, which uses sequences 00–10 for training and 11–21 for testing. Category 3 (dynamic acquisition) [25] is a dataset class proposed by MPEG, which contains a series of 3-D point cloud sequences with dynamic acquisition scenarios for compression tests. It includes the Ford, qnxadas series, Innoviz series, citytunnel_q1mm, overpass_q1mm, and tollbooth_q1mm datasets, consisting of over 3351 continuous frames in total. In particular, Ford [64] is the most widely used dataset in LPC compression scenarios [18], [20], [21], [27], [28], which contains more than 1500 scans per sequence. We strictly adhere to the partitioning guidelines recommended by MPEG standardization [25], in which Sequence 01 is designated for model training, while Sequences 02 and 03 are set aside for comprehensive performance evaluation. For other Category 3 datasets, we apply the Ford-trained model to evaluate.

1) Baselines: For the indoor scenario, we compare with GPCC [26] (Octree mode and Trisoup mode), OctSqueeze [5], VoxelContext-Net [20], SparsePCGC [27], OctAttention [21], and OctFormer [22]. For outdoor datasets, we compare with GPCC [26], MuSCLE [19], VoxelContext-Net [20], SparsePCGC [27], OctAttention [21], OctFormer [22], EHEM [28], STAEM [18], and DuOctree [29]. In addition, we also compare the compression performance of two different coding modes of GPCC (intra- and inter-frame) on the Category 3 datasets. The GPCC operates in Octree mode by default.

For standardization purposes, we adhere to the common test condition (CTC) [25] for conducting our experiments with the GPCC [26] from the MPEG standard (TMC13 v27.0[1]). Our method is compared with others in terms of lossy compression for geometry, corresponding to the C2 test condition [25]. In addition, to make precise distortion alignment with other methods, we introduce an extra voxelization step to fine-tune the distortion of GPCC. Note that this adjustment only slightly shifts the data points on the rate–distortion (RD) curve, while the overall shape of the curve remains consistent with before, which has virtually no impact on the performance of GPCC. Specifically, for the ScanNet dataset, we voxelize the data with the size of 0.001 and adjust the "positionQuantizationScale" to 0.7, 0.5, and 0.3, to achieve various bitrates. For the SemanticKITTI dataset, we voxelize the data with a size of 0.00003 and subsequently apply GPCC. The "positionQuantizationScale" parameter is varied among values such as 0.1, 0.06, and 0.03 to achieve a range of bitrates. Due to the absence of prior knowledge corresponding to the point cloud acquisition, we deactivate GPCC's angular coding mode and set related parameters to default values for both ScanNet and SemanticKITTI datasets. For example, "angularEnabled," "zCompensationEnabled," and "numLaser" are set to 0, while "lasersTheta," "lasersZ," and "lasersNumPhiPerTurn" are set to null. For Category 3 datasets, we adjust the GPCC parameter of the "positionQuantizationScale" to 0.125, 0.03125, and 0.015625, to achieve various bitrates. We activate the angular coding mode, using prior knowledge from the point cloud acquisition phase. Related parameters are set as the configurations provided in the GPCC repository.

2) Implementation Details: Our entropy model has three attention blocks, for each AggEmb module we sample four dependent prior nodes. The channel dimension is set to 256 for attention layers except for the UMs. To evaluate our model at different bitrates, we follow the quantization settings in [28] for SemanticKITTI, setting the quantization step to $(400/2^{D-1})$ to build the Octree with depth D. We use the maximum depth of 13 with sequence size 1024 for training and set the Octree depth from 9 to 13 and the context length from 128 to 1024 in testing. For Category 3 datasets, we also follow the settings in [18] and [28], which sets the quantization step R_s to $(2^{18}/2^D)$ to build the Octree from the normalized point cloud. We train our model using a maximum depth of 15 and test with a depth from 11 to 15. For ScanNet, we follow the settings in [22], which sets the quantization step to $(1/2^{D-1})$ to build the Octree from the normalized point cloud. We train our model using a maximum depth of 10 and test with a depth from 6 to 10. It should be noted that our model directly uses the current Octree nodes as reference inputs when training and testing in static mode. This allows for continued utilization of the above modules without altering the input propagation path. 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 (40-GB 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 ScanNet, two days on Ford, and three days on SemanticKITTI dataset.

3) Evaluation Metrics: For evaluation, we use bits per point (bpp) [21], [22] for the compression ratio metric, which

[1]https://git.mpeg.expert/MPEG/3dgh/g-pcc/software/tm/mpeg-pcc-tmc13

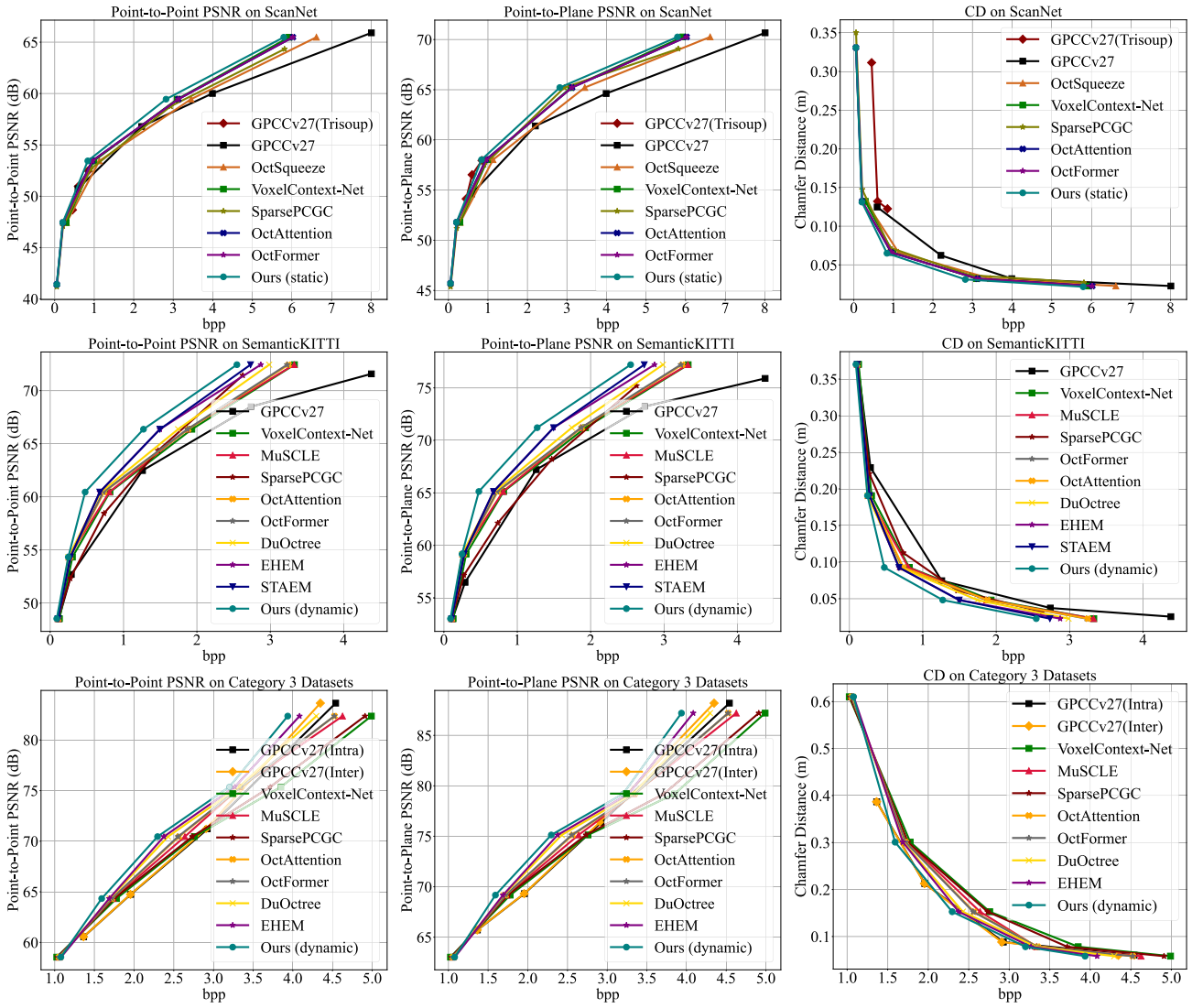


Fig. 6. Quantitative results of different methods on ScanNet, SemanticKITTI, and Category 3 datasets. In addition, for a fair comparison, we only compare with STAEM on the SemanticKITTI dataset, in which data in their paper are used.

describes the data volume after compression

$$\text{bpp} = \frac{1}{n_1}\sum_{a=1}^{n_2} s_a \tag{11}$$

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 [34]. We use the official metric calculating tool *pc_error* provided by MPEG's GPCC, and set the PSNR peak value following [27], [28]. Specifically, we set r to 1 for ScanNet, 59.7 for SemanticKITTI, and 30 000 for Category 3 datasets. In addition, we also use chamfer distance (CD) [8], [65] to further evaluate the reconstruction quality

$$\begin{aligned}\text{CD}\left(P,\overline{P}\right) &= \max\left(\text{CD}'\left(P,\overline{P}\right),\text{CD}'\left(\overline{P},P\right)\right)\\ \text{CD}'\left(P,\overline{P}\right) &= \frac{1}{|P|}\sum_i \min_j \left\|p_i-\overline{p}_j\right\|_2\end{aligned} \tag{12}$$

where $\mathbf{p}_i$ in point cloud P is the closest point to the $\overline{p}_j$ in $\overline{P}$.

B. Experimental Results

The quantitative experimental results are presented in Fig. 6. Overall, our method outperforms other methods in both indoor and outdoor scenarios, as expected. Specifically, for the indoor dataset ScanNet, we obtain the average bpp = 5.79 for depth-10 Octree, making a 27.7% and 13.5% reduction in bitrates compared with GPCC (Octree) (bpp = 8.01) and OctSqueeze (bpp = 6.69). In static point cloud compression, OctFormer [22] performs better than other methods due to its transformer blocks. Nevertheless, compared to OctFormer (bpp = 6.02), we achieve up to 3.8% advantage, which indicates that our embedding module is capable of introducing global node features of intraframe for the context window. Another observation is that, in the ScanNet dataset, GPCC achieves better compression results in low bpp using the trisoup mode compared to the Octree mode, due to its exploitation of the continuous point distribution within nodes partitioned by the pruned Octree across larger spaces.

For outdoor scenarios SemanticKITTI, our method gets the average bpp = 1.27 for depth-12 Octree and achieves 42.7%, 34.2%, 34.9%, 32.8%, 32.0%, 31.4%, 27.0%, and

TABLE I
BD-RATE GAINS OVER THE GPCC (OCTREE) MEASURED USING BOTH D1 AND D2 PSNR FOR OUR AND OTHER METHODS ON SCANNET

Method	Reference	ScanNet		
		PSNR D1	PSNR D2	CD
OctSqueeze [5]	CVPR'20	-3.09%	-12.37%	-39.96%
VoxelContext-Net [20]	CVPR'21	-13.69%	-21.98%	-46.24%
SparsePCGC [27]	TPAMI'22	-7.02%	-23.38%	-43.26%
OctAttention [21]	AAAI'22	-14.81%	-23.60%	-51.01%
OctFormer [22]	AAAI'23	-14.45%	-23.20%	-50.40%
Ours (static)	-	**-22.86%**	**-31.05%**	**-56.82%**

16.9% reduction in bitrates compared with GPCC (Octree) (bpp = 2.21), VoxelContext-Net (bpp = 1.93), SparsePCGC (bpp = 1.95), MuSCLE (bpp = 1.89), OctAttention (bpp = 1.87), OctFormer (bpp = 1.85), DuOctree (bpp = 1.74), and EHEM (bpp = 1.53). For Category 3 datasets, our method achieves an average bpp of 3.19 for a depth-14 Octree and results in bitrate reductions of 8.1%, 6.5%, 17.1%, 14.2%, 5.3%, 4.8%, 4.5%, 3.9%, and 2.5% compared with GPCC (intra, bpp = 3.47), GPCC (inter, bpp = 3.41), VoxelContext-Net (bpp = 3.85), SparsePCGC (bpp = 3.72), MuSCLE (bpp = 3.37), OctAttention (bpp = 3.35), OctFormer (bpp = 3.34), DuOctree (bpp = 3.32), and EHEM (bpp = 3.27). Benefiting from its feature extraction model, STAEM [18] performs better than other methods in dynamic compression. However, compared to STAEM (bpp = 1.49 on SemanticKITTI), our method still achieves up to 14.7% bitrate savings. This is attributed to our aggregation module, which traverses all interframe reference nodes and concatenates them with STA blocks, thereby introducing more cross-contextual interactions and multiscale correlations. We can also see that in the Category 3 datasets, the intermode GPCC demonstrates superior compression performance compared to the intramode because the intermode leverages predictive geometry to model the correlation of point clouds across the temporal dimension.

In addition, we calculate the BD-Rate gains over GPCC using D1 PSNR, D2 PSNR, and CD for our method and others, as shown in Tables I–III. It can be observed that our method achieves advanced BD-rate gains in all the datasets, which can be attributed to our designed aggregation module that compensates for the lack of spatial correlation by introducing dynamic node relevance. Specifically, for the ScanNet dataset, our method achieves gains of up to 22.9%, 31.1%, and 56.8% for D1 PSNR, D2 PSNR, and CD, respectively. For the SemanticKITTI dataset, our method yields improvements of up to 40.1%, 43.2%, and 37.1% for D1 PSNR, D2 PSNR, and CD metrics. For the Category 3 datasets (taking the widely-used Ford dataset as an example), our method also achieves respective uplifts of up to 18.8%, 15.9%, and 8.1% for D1 PSNR, D2 PSNR, and CD, respectively. These experimental results demonstrate that our method consistently achieves higher BD-Rate gains on all the datasets compared with other methods, highlighting its advantages. A further observation is that the BD-Rate gains in the SemanticKITTI dataset are usually higher than those in other datasets. The potential reason is that compared with other datasets, the point distribution is more concentrated in SemanticKITTI, leading to stronger correlations among context nodes, which makes the model easier to predict the point distributions. We also visualize the qualitative results, as shown in Fig. 7. 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 with the baseline GPCC. We can observe that our method makes lower bpp than GPCC in higher PSNR, which demonstrates the effectiveness of our method in achieving superior compression performance while maintaining high-quality reconstruction of the point clouds.

C. Ablation Study

1) Different Module Analyses: To evaluate the effectiveness of our proposed AggEmb module, SA, TA, and UM in DAPCC, we conduct an ablation study as shown in Table IV. In general, all the modules improve compression performance as expected, and comparison shows that DAPCC reduces 8.1% bitrate on average compared with only applying SA in different depths and modes. In static mode, our bpp is decreased by 3.6% with a 13-layer Octree, while a nine-layer Octree decreased by 11.8%. In dynamic mode, our bpp is decreased by 9.2% with a 13-layer Octree, while a nine-layer tree decreased by 18.3%. This indicates that dynamic feature aggregation has a significant effect on occupancy prediction by introducing correlated spatial–temporal information. The TA module also serves a similar function by capturing global context features.

2) Runtime Analysis: We compare our model with other methods on SemanticKITTI in terms of encoding and decoding time for Octree with different depths. Similar to EHEM and Light EHEM [28], we set the Octree sequence size to 8192. The runtime results are shown in Table V. From the table, we can see that the voxel-based SparsePCGC [27] achieves the advanced encoding and decoding time above all the methods, because it uses the sparse voxels for multiscale representation of LPC data, which can prevent the introduction of more neighboring nodes and save time. Despite this, our method still achieves satisfactory results, outperforming most Octree-based methods. That is because our single-Octree-based DAPCC uses the linear AggEmb module for dynamic node feature construction and two lightweight attentions for spatial–temporal node dependencies discovery instead of a large deep entropy model to predict occupancy. In addition, light EHEM [28] proposes reducing the network depth and hidden vector dimension, which effectively decreases parameters and improves the inference speed of EHEM. Nevertheless, our method is still slightly superior to Light EHEM, especially in low-depth Octrees. That is because our method only introduces the sibling nodes of the same layer to construct the context window and does not need to use ancestral nodes in the current frame for encoding or decoding, which means that contexts from the same Octree depth can be processed in parallel during encoding and decoding. This also explains why our method has similar encoding and decoding time. Due to the lack of time results in the manuscript STAEM [18] and the difficulty in reproducing them, we do not include this method here.

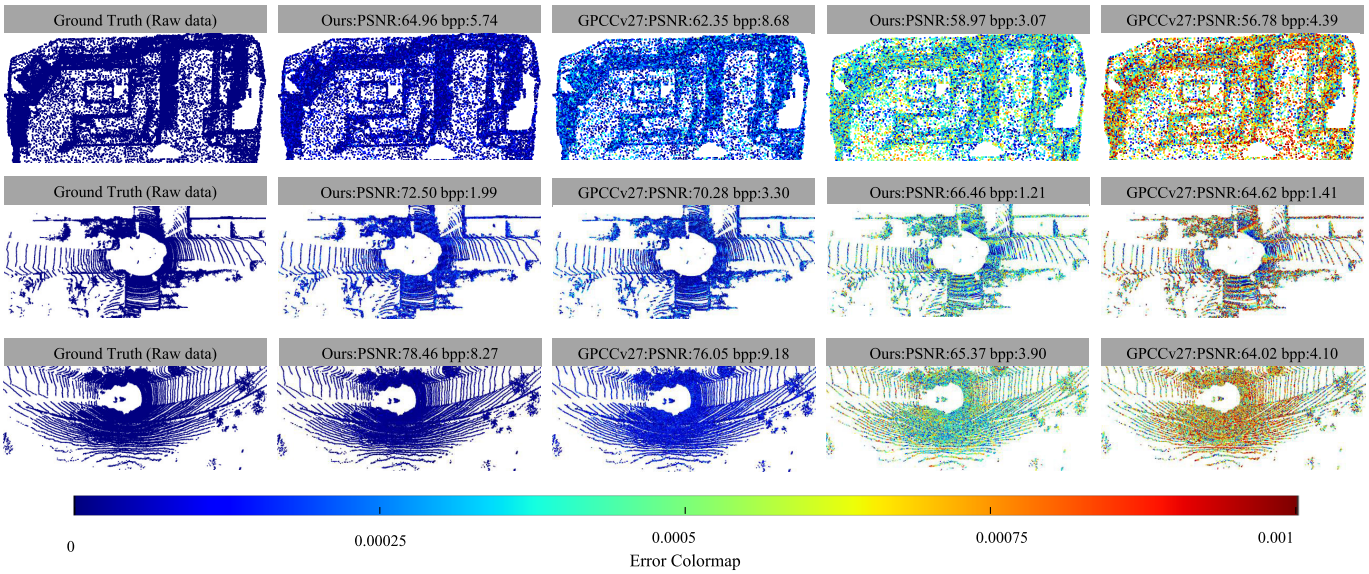


Fig. 7. Visualization of compression results of our method and GPCC on (top) indoor ScanNet, (middle) outdoor SemanticKITTI, and (bottom) Ford under different bpps. Note that we only visualize the quantitative results of ours and GPCC, considering that Octree-based methods have the same reconstruction qualities under the same Octree depth.

TABLE II

BD-RATE GAINS OVER THE GPCC (OCTREE) MEASURED USING BOTH D1 AND D2 PSNR AND CD FOR OUR AND OTHER METHODS ON SEMANTICKITTI AND FORD DATASETS

Method	Reference	SemanticKITTI			Ford		
		PSNR D1	PSNR D2	CD	PSNR D1	PSNR D2	CD
MuSCLE [19]	NeurIPS'20	-20.07%	-24.33%	-25.67%	-5.98%	-3.06%	8.48%
VoxelContext-Net [20]	CVPR'21	-13.66%	-18.05%	-15.50%	8.23%	12.36%	20.55%
SparsePCGC [27]	TPAMI'22	-7.50%	-4.19%	-10.61%	4.04%	8.03%	16.88%
OctAttention [21]	AAAI'22	-20.43%	-24.66%	-24.59%	-7.90%	-4.64%	5.53%
OctFormer [22]	AAAI'23	-21.74%	-25.92%	-25.67%	-8.29%	-5.00%	4.99%
EHEM [28]	CVPR'23	-27.58%	-31.23%	-26.12%	-14.84%	-11.88%	-3.04%
STAEM [18]	ICME'23	-27.82%	-31.40%	-26.26%	-	-	-
DuOctree [29]	ICRA'24	-26.06%	-29.95%	-29.87%	-12.08%	-8.96%	-0.32%
Ours	-	**-40.12%**	**-43.16%**	**-37.06%**	**-18.76%**	**-15.90%**	**-8.14%**

TABLE III

BD-RATE GAINS OVER THE GPCC (OCTREE) MEASURED USING D1 PSNR FOR OUR AND OTHER METHODS ON OTHER MPEG CTC CATEGORY 3 DATASETS, OF WHICH THREE DATASETS ARE NOT INCLUDED DUE TO THEIR CURRENT MAINTENANCE

Method	Datasets						
	InnQC1	InnQC2	InnQC3	qnx_approach	qnx_bend	qnx_exit	qnx_join
MuSCLE [19]	-4.63%	-4.19%	-4.89%	-1.29%	-2.12%	-1.68%	-2.47%
VoxelContext-Net [20]	8.65%	7.95%	8.55%	2.33%	3.52%	2.83%	4.93%
SparsePCGC [27]	4.79%	4.23%	4.72%	1.41%	2.08%	1.52%	2.49%
OctAttention [21]	-6.42%	-5.97%	-6.71%	-1.79%	-2.64%	-2.15%	-3.56%
OctFormer [22]	-6.66%	-6.21%	-6.69%	-2.15%	-2.73%	-2.38%	-3.71%
EHEM [28]	-12.90%	-12.09%	-13.22%	-3.60%	-5.29%	-4.22%	-7.31%
DuOctree [29]	-10.24%	-9.48%	-10.30%	-3.21%	-4.24%	-3.38%	-5.85%
Ours	**-16.76%**	**-15.45%**	**-16.58%**	**-4.77%**	**-6.87%**	**-5.42%**	**-9.17%**

TABLE IV

PERFORMANCE OF DAPCC WITH DIFFERENT MODULES ON SEMANTICKITTI, IN WHICH SA, AGGEMB, TA, AND UM REPRESENT THE SPATIAL ATTENTION, AGGREGATION EMBEDDING, TEMPORAL ATTENTION, AND U-MLP MODULE, RESPECTIVELY

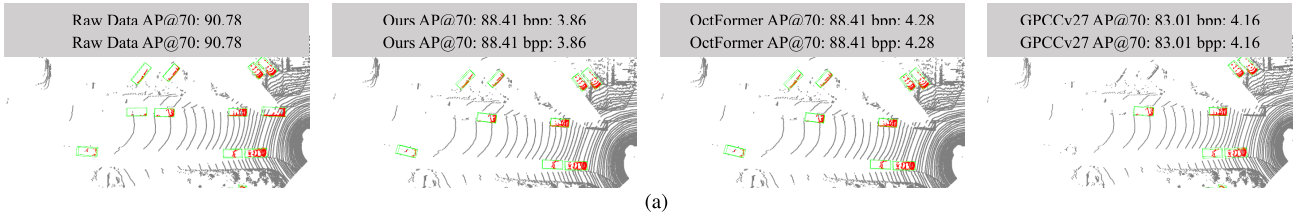
SA	AggEmb	TA	UM	bpp ↓									
				Static					Dynamic				
				D=9	D=10	D=11	D=12	D=13	D=9	D=10	D=11	D=12	D=13
✓				0.109	0.278	0.527	1.384	2.785	0.111	0.280	0.534	1.396	2.799
✓	✓			0.102	0.267	0.519	1.375	2.758	0.101	0.264	0.510	1.328	2.666
✓	✓	✓		0.098	0.259	0.510	1.368	2.727	0.095	0.259	0.490	1.293	2.618
✓	✓	✓	✓	0.093	0.251	0.505	1.367	2.684	**0.090**	**0.247**	**0.476**	**1.272**	**2.542**

3) Context Window Size Analysis: As shown in Table VI, we perform ablation study on context size. From the table, our method achieves up to 0.4431 bpp savings between the 1024 context size and 32 when setting the Octree depth to 13, which shows that the model with a larger context size obtains better performance. The sequence with a larger context size

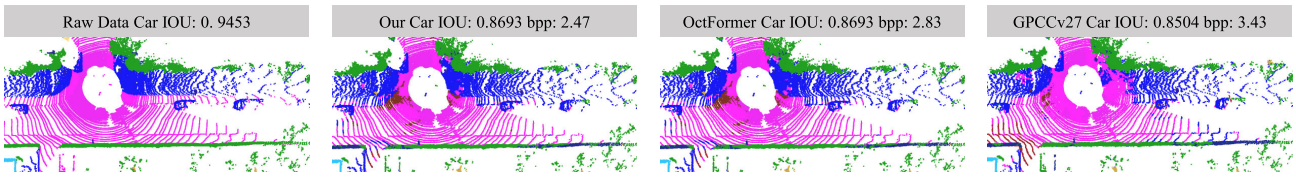
TABLE V

ENCODING/DECODING TIMES OF A D DEPTH OCTREE ON SEMANTICKITTI, IN WHICH LIGHT EHEM IS A LIGHTWEIGHT IMPLEMENTATION WITH THE SAME STRUCTURE AS EHEM [28]

Method	Reference	Encoding/Decoding Time (s) ↓				Params ↓
		D=10	D=12	D=14	D=16	
GPCCv27 [26]	MPEG'24	0.14/0.07	0.25/0.12	0.66/0.32	1.07/0.54	-
MuSCLE [19]	NeurIPS'20	5.47/14.2	12.9/43.1	23.8/182	28.6/234	0.34M
VoxelContext-Net [20]	CVPR'21	10.4/10.2	17.5/17.3	55.2/54.6	102.7/100.8	-
SparsePCGC [27]	TPAMI'22	0.12/0.11	0.21/0.14	0.52/0.33	0.89/0.51	5.74M
OctAttention [21]	AAAI'22	0.03/20.6	0.08/83.1	0.32/321	0.66/708	4.23M
OctFormer [22]	AAAI'23	0.07/0.10	0.18/0.26	0.73/1.02	1.49/2.09	4.28M
EHEM [28]	CVPR'23	0.18/0.21	0.40/0.43	1.21/1.39	2.53/3.01	13.01M
Light EHEM [28]	CVPR'23	0.12/0.15	0.29/0.33	0.79/0.92	1.63/1.94	6.34M
DuOctree [29]	ICRA'24	0.58/0.60	1.87/1.92	4.91/5.02	9.50/9.78	66.10M
Ours	-	0.09/0.09	0.22/0.22	0.72/0.72	1.84/1.84	4.57M



(a)



(b)

Fig. 8. Visualization of vehicle detection and semantic segmentation task under different bpps. (a) Performance on vehicle detection for different methods. (b) Performance on semantic segmentation for different methods.

TABLE VI

PERFORMANCE OF DAPCC WITH DIFFERENT CONTEXT SIZES w ON SEMANTICKITTI

Size w	bpp ↓				
	D=9	D=10	D=11	D=12	D=13
32	0.1202	0.3063	0.6533	1.5023	2.9853
64	0.1091	0.2743	0.5839	1.4424	2.8887
128	0.1033	0.2629	0.5630	1.3874	2.6979
256	0.0983	0.2628	0.5354	1.3225	2.6114
512	0.0944	0.2537	0.5122	1.2842	2.5731
1024	**0.0895**	**0.2465**	**0.4762**	**1.2716**	**2.5422**

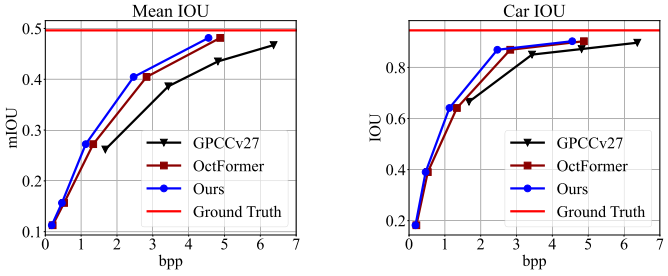


Fig. 9. Quality performance of semantic segmentation within different point cloud compression methods on SemanticKITTI dataset.

contains more spatial–temporal features, which is beneficial to sample more reference nodes from embedding with the increase of context size.

D. Downstream Applications

1) Vehicle Detection: The 3-D vehicle detection [66] involves recognizing and locating objects within a 3-D scene, playing a crucial role in robotics and autonomous driving. We employ the Part-A^2 net [67] for evaluation, reporting average precision (AP) for 3-D vehicle detection with a mean intersection-over-union (mIOU) threshold set to 0.7. We visualize the results on 3-D vehicle detection under different settings in Fig. 8(a) and report the detailed results in Table VII. Easy, moderate, and hard represent different detection difficulty levels in the dataset, aligning with the official KITTI object detection benchmark definition. We can observe that compared with other methods, our method achieves higher AP while maintaining a relatively lower bpp, which achieves at most 10% bitrate saving. In addition, our method obtains similar detection results to the raw point cloud data.

2) Semantic Segmentation: The 3-D semantic segmentation [68], [69] involves assigning class labels to individual points, serving as a critical task for scene understanding in applications like autonomous driving perception. In our experiments, we use mIOU as the metric for segmentation, with RandLA-Net [40] serving as the baseline method. Figs. 8(b) and 9 show the segmentation results. The figure shows that our method achieves accurate segmentation performance, close to the raw data, especially when the bpp

TABLE VII
RESULTS OF VEHICLE DETECTION UNDER DIFFERENT BPPS OF DIFFERENT METHODS

Method		Raw Data	GPCCv27			OctFormer			Ours		
bpp ↓		-	1.36	2.88	4.16	1.02	2.17	4.28	0.92	2.02	3.86
AP@70 ↑	Easy	90.78	79.26	82.73	83.01	80.68	84.46	88.41	80.68	84.46	88.41
	Mod	78.60	69.37	74.17	74.55	71.54	74.80	77.98	71.54	74.80	77.98
	Hard	73.73	65.81	68.43	72.37	68.23	71.25	73.60	68.23	71.25	73.60

is around 4.5. In summary, our method outperforms others in both vehicle detection and semantic segmentation tasks, demonstrating its effectiveness in downstream applications.

V. CONCLUSION

We propose DAPCC, an LPC compression model based on diverse attention mechanisms, which compresses dynamic point clouds efficiently. Specifically, we construct Octree sequences with nonoverlapped context and apply dependency calculation in AggEmb modules for dynamic node matching and temporal feature introduction. We further design converted spatial and TA to capture sequence dependencies and dynamic point motion. In addition, we utilize a UM to adjust feature expressions and improve occupancy prediction. The experimental results show that our method outperforms all state-of-the-art baselines on both indoor and outdoor datasets and downstream tasks, which verifies the effectiveness of our proposed DAPCC. We believe that our DAPCC can provide a new perspective on dynamic point cloud compression.

REFERENCES

[1] X. Chu and S. Zhao, "Adaptive guided convolution generated with spatial relationships for point clouds analysis," *IEEE Trans. Geosci. Remote Sens.*, vol. 62, 2024, Art. no. 5700412.

[2] W. Peng, Y. Wang, H. Zhang, Y. Cao, J. Zhao, and Y. Jiang, "Deep correspondence matching based robust point cloud registration of profiled parts," *IEEE Trans. Ind. Informat.*, vol. 20, no. 2, pp. 2129–2143, Feb. 2024.

[3] Y. Xie, J. Zhu, S. Li, N. Hu, and P. Shi, "HECPG: Hyperbolic embedding and confident patch-guided network for point cloud matching," *IEEE Trans. Geosci. Remote Sens.*, vol. 62, 2024, Art. no. 5702212.

[4] R. She et al., "PointDifformer: Robust point cloud registration with neural diffusion and transformer," *IEEE Trans. Geosci. Remote Sens.*, vol. 62, 2024, Art. no. 5701015.

[5] 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. (CVPR)*, Jun. 2020, pp. 1313–1323.

[6] X. Zou, K. Li, and C. Chen, "Multilevel attention based U-shape graph neural network for point clouds learning," *IEEE Trans. Ind. Informat.*, vol. 18, no. 1, pp. 448–456, Jan. 2022.

[7] P. A. Chou, M. Koroteev, and M. Krivokuca, "A volumetric approach to point cloud compression—Part I: Attribute compression," *IEEE Trans. Image Process.*, vol. 29, pp. 2203–2216, 2020.

[8] T. Huang and Y. Liu, "3D point cloud geometry compression on deep learning," in *Proc. 27th ACM Int. Conf. Multimedia*, Oct. 2019, pp. 890–898.

[9] 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.

[10] 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.

[11] L. Wiesmann, A. Milioto, X. Chen, C. Stachniss, and J. Behley, "Deep compression for dense point cloud maps," *IEEE Robot. Autom. Lett.*, vol. 6, no. 2, pp. 2060–2067, Apr. 2021.

[12] Y. Huang, J. Peng, C.-C. J. Kuo, and M. Gopi, "Octree-based progressive geometry coding of point clouds," in *Proc. PBG SIGGRAPH*, Jul. 2006, pp. 103–110.

[13] Z. Liu, H. Tang, Y. Lin, and S. Han, "Point-voxel CNN for efficient 3D deep learning," in *Proc. Adv. Neural Inf. Process. Syst.*, H. Wallach, H. Larochelle, A. Beygelzimer, F. d'Alché-Buc, E. Fox, and R. Garnett, Eds., Red Hook, NY, USA: Curran Associates, Jun. 2019, pp. 1–11.

[14] Z. Wang, S. Wan, and L. Wei, "Local geometry-based intra prediction for octree-structured geometry coding of point clouds," *IEEE Trans. Circuits Syst. Video Technol.*, vol. 33, no. 2, pp. 886–896, Feb. 2023.

[15] G. Lu, W. Ouyang, D. Xu, X. Zhang, C. Cai, and Z. Gao, "DVC: An end-to-end deep video compression framework," in *Proc. IEEE/CVF Conf. Comput. Vis. Pattern Recognit. (CVPR)*, Jun. 2019, pp. 10998–11007.

[16] J. Kammerl, N. Blodow, R. B. Rusu, S. Gedikli, M. Beetz, and E. Steinbach, "Real-time compression of point cloud streams," in *Proc. IEEE Int. Conf. Robot. Autom.*, May 2012, pp. 778–785.

[17] S. Liu et al., "Deep semantic graph matching for large-scale outdoor point cloud registration," *IEEE Trans. Geosci. Remote Sens.*, vol. 62, 2024, Art. no. 5701412.

[18] R. Song, C. Fu, S. Liu, and G. Li, "Large-scale spatio-temporal attention based entropy model for point cloud compression," in *Proc. IEEE Int. Conf. Multimedia Expo (ICME)*, Jul. 2023, pp. 2003–2008.

[19] 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.*, Dec. 2020, pp. 22170–22181.

[20] 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.

[21] 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.*, Aug. 2022, pp. 625–633.

[22] 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.*, vol. 37, Jun. 2023, pp. 470–478.

[23] A. Dai, A. X. Chang, M. Savva, M. Halber, T. Funkhouser, and M. Nießner, "ScanNet: Richly-annotated 3D reconstructions of indoor scenes," in *Proc. IEEE Conf. Comput. Vis. Pattern Recognit. (CVPR)*, Jul. 2017, pp. 5828–5839.

[24] J. Behley et al., "SemanticKITTI: A dataset for semantic scene understanding of LiDAR sequences," in *Proc. IEEE/CVF Conf. Comput. Vis. Pattern Recognit. (CVPR)*, Jun. 2019, pp. 9297–9307.

[25] *Common Test Conditions for G-PCC*, ISO/IEC JTC1/SC29/WG Standard 7 N800, WG7, 2024.

[26] *Mpeg 3D Graphics Coding. G-PCC Test Model V27.*, ISO/IEC MPEG(JTC1/SC29/WG11) Standard N18189, Mammou, 2024.

[27] 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.

[28] 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)*, Jun. 2023, pp. 14368–14377.

[29] M. Cui, M. Feng, J. Long, D. Hu, S. Zhao, and K. Huang, "A du-octree based cross-attention model for LiDAR geometry compression," in *Proc. IEEE Int. Conf. Robot. Autom. (ICRA)*, May 2024, pp. 3796–3802.

[30] X. Sun, H. Ma, Y. Sun, and M. Liu, "A novel point cloud compression algorithm based on clustering," *IEEE Robot. Autom. Lett.*, vol. 4, no. 2, pp. 2132–2139, Apr. 2019.

[31] Y. Feng, S. Liu, and Y. Zhu, "Real-time spatio-temporal LiDAR point cloud compression," in *Proc. IEEE/RSJ Int. Conf. Intell. Robots Syst. (IROS)*, Oct. 2020, pp. 10766–10773.

[32] E. C. Kaya, S. Schwarz, and I. Tabus, "Refining the bounding volumes for lossless compression of voxelized point clouds geometry," in *Proc. IEEE Int. Conf. Image Process. (ICIP)*, Sep. 2021, pp. 3408–3412.

[33] H. Houshiar and A. Nuchter, "3D point cloud compression using conventional image compression for efficient data transmission," in *Proc. 25th Int. Conf. Inf., Commun. Autom. Technol. (ICAT)*, Oct. 2015, pp. 1–8.

[34] S. Schwarz et al., "Emerging MPEG standards for point cloud compression," *IEEE J. Emerg. Sel. Topics Circuits Syst.*, vol. 9, no. 1, pp. 133–148, Mar. 2019.

[35] 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.

[36] X. Sun, S. Wang, M. Wang, Z. Wang, and M. Liu, "A novel coding architecture for LiDAR point cloud sequence," *IEEE Robot. Autom. Lett.*, vol. 5, no. 4, pp. 5637–5644, Oct. 2020.

[37] S. Wang, J. Jiao, P. Cai, and L. Wang, "R-PCC: A baseline for range image-based point cloud compression," in *Proc. Int. Conf. Robot. Autom. (ICRA)*, May 2022, pp. 10055–10061.

[38] Y. Lin et al., "FPConv: Learning local flattening for point convolution," in *Proc. IEEE/CVF Conf. Comput. Vis. Pattern Recognit. (CVPR)*, Jun. 2020, pp. 4292–4301.

[39] M. Xu, R. Ding, H. Zhao, and X. Qi, "PAConv: Position adaptive convolution with dynamic kernel assembling on point clouds," in *Proc. IEEE/CVF Conf. Comput. Vis. Pattern Recognit. (CVPR)*, Jun. 2021, pp. 3172–3181.

[40] Q. Hu et al., "RandLA-Net: Efficient semantic segmentation of large-scale point clouds," in *Proc. IEEE/CVF Conf. Comput. Vis. Pattern Recognit. (CVPR)*, Jun. 2020, pp. 11108–11117.

[41] M. Wang, R. Huang, H. Dong, D. Lin, Y. Song, and W. Xie, "MsLPCC: A multimodal-driven scalable framework for deep LiDAR point cloud compression," in *Proc. AAAI Conf. Artif. Intell.*, vol. 38, Mar. 2024, pp. 5526–5534.

[42] R. Q. Charles, H. Su, M. Kaichun, and L. J. Guibas, "PointNet: Deep learning on point sets for 3D classification and segmentation," in *Proc. IEEE Conf. Comput. Vis. Pattern Recognit. (CVPR)*, Jul. 2017, pp. 77–85.

[43] C. R. Qi, L. Yi, H. Su, and L. J. Guibas, "PointNet++: Deep hierarchical feature learning on point sets in a metric space," in *Proc. 31st Int. Conf. Neural Inf. Process. Syst.* Red Hook, CA, USA: Curran Associates, 2017, pp. 5105–5114.

[44] W. Wu, Z. Qi, and L. Fuxin, "PointConv: Deep convolutional networks on 3D point clouds," in *Proc. IEEE/CVF Conf. Comput. Vis. Pattern Recognit. (CVPR)*, Jun. 2019, pp. 9613–9622.

[45] H. Thomas, C. R. Qi, J. Deschaud, B. Marcotegui, F. Goulette, and L. Guibas, "KPConv: Flexible and deformable convolution for point clouds," in *Proc. IEEE/CVF Int. Conf. Comput. Vis. (ICCV)*, Oct. 2019, pp. 6410–6419.

[46] M.-H. Guo, J.-X. Cai, Z.-N. Liu, T.-J. Mu, R. R. Martin, and S.-M. Hu, "PCT: Point cloud transformer," *Comput. Vis. Media*, vol. 7, no. 2, pp. 187–199, Jun. 2021.

[47] X. Pan, Z. Xia, S. Song, L. E. Li, and G. Huang, "3D object detection with pointformer," in *Proc. IEEE/CVF Conf. Comput. Vis. Pattern Recognit. (CVPR)*, Jun. 2021, pp. 7459–7468.

[48] Z. Pan, M. Xiao, X. Han, D. Yu, G. Zhang, and Y. Liu, "PatchD-PCC: A patchwise deep compression framework for dynamic point clouds," in *Proc. AAAI Conf. Artif. Intell.*, vol. 38, Mar. 2024, pp. 4406–4414.

[49] D. C. Garcia, T. A. Fonseca, R. U. Ferreira, and R. L. de Queiroz, "Geometry coding for dynamic voxelized point clouds using octrees and multiple contexts," *IEEE Trans. Image Process.*, vol. 29, pp. 313–322, 2020.

[50] 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)*, Jul. 2020, pp. 1–6.

[51] A. Dricot, F. Pereira, and J. Ascenso, "Rate-distortion driven adaptive partitioning for octree-based point cloud geometry coding," in *Proc. 25th IEEE Int. Conf. Image Process. (ICIP)*, Oct. 2018, pp. 2969–2973.

[52] J. Zhang, T. Chen, D. Ding, and Z. Ma, "G-PCC++: Enhanced geometry-based point cloud compression," in *Proc. 31st ACM Int. Conf. Multimedia*, Jun. 2023, pp. 1352–1363.

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

[54] K. You, K. Liu, L. Yu, P. Gao, and D. Ding, "Pointsoup: High-performance and extremely low-decoding-latency learned geometry codec for large-scale point cloud scenes," in *Proc. 33rd Int. Joint Conf. Artif. Intell.*, Aug. 2024, pp. 5380–5388.

[55] C. Duan, C. Liao, S. Wang, Q. Zhang, C. Zhang, and C. Zhu, "LiDAR point cloud compression improvement through sampling operations," in *Proc. IEEE Int. Symp. Broadband Multimedia Syst. Broadcast. (BMSB)*, Jun. 2024, pp. 1–5.

[56] Z. Li, W. Wang, Z. Wang, and N. Lei, "Point cloud compression via constrained optimal transport," in *Proc. IEEE Int. Conf. Multimedia Expo (ICME)*, Jul. 2024, pp. 1–6.

[57] Z. Chen, Z. Qian, S. Wang, and Q. Chen, "Point cloud compression with sibling context and surface priors," in *Proc. Eur. Conf. Comput. Vis.*, S. Avidan, G. Brostow, M. Cissé, G. M. Farinella, and T. Hassner, Eds., Cham, Switzerland: Springer, 2022, pp. 744–759.

[58] A. Luo et al., "SCP: Spherical-coordinate-based learned point cloud compression," in *Proc. AAAI Conf. Artif. Intell.*, vols. 4–38, 2024, pp. 3954–3962.

[59] A. Vaswani et al., "Attention is all you need," in *Proc. Adv. Neural Inf. Process. Syst.*, vol. 30, Jun. 2017, pp. 5998–6008.

[60] S. Xie et al., "ResiDual: Transformer with dual residual connections," 2023, *arXiv:2304.14802*.

[61] Y. Guo, Y. Tao, Y. Chong, S. Pan, and M. Liu, "Edge-guided hyperspectral image compression with interactive dual attention," *IEEE Trans. Geosci. Remote Sens.*, vol. 61, 2023, Art. no. 5500817.

[62] P. Ramachandran, B. Zoph, and Q. V. Le, "Searching for activation functions," 2017, *arXiv:1710.05941*.

[63] J. L. Ba, J. R. Kiros, and G. E. Hinton, "Layer normalization," 2016, *arXiv:1607.06450*.

[64] G. Pandey, J. R. McBride, and R. M. Eustice, "Ford campus vision and LiDAR data set," *Int. J. Robot. Res.*, vol. 30, no. 13, pp. 1543–1552, Nov. 2011.

[65] H. Fan, H. Su, and L. Guibas, "A point set generation network for 3D object reconstruction from a single image," in *Proc. IEEE Conf. Comput. Vis. Pattern Recognit. (CVPR)*, Jul. 2017, pp. 2463–2471.

[66] C. Yu, J. Lei, B. Peng, H. Shen, and Q. Huang, "SIEV-Net: A structure-information enhanced voxel network for 3D object detection from LiDAR point clouds," *IEEE Trans. Geosci. Remote Sens.*, vol. 60, 2022, Art. no. 5703711.

[67] 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 Trans. Pattern Anal. Mach. Intell.*, vol. 43, no. 8, pp. 2647–2664, Aug. 2020.

[68] C. Liu et al., "Context-aware network for semantic segmentation toward large-scale point clouds in urban environments," *IEEE Trans. Geosci. Remote Sens.*, vol. 60, 2022, Art. no. 5703915.

[69] A. J. Afifi et al., "Tinto: Multisensor benchmark for 3D hyperspectral point cloud segmentation in the geosciences," *IEEE Trans. Geosci. Remote Sens.*, vol. 62, 2024, Art. no. 5501015.