Received 2 February 2024, accepted 30 May 2024, date of publication 7 June 2024, date of current version 17 June 2024.

Digital Object Identifier 10.1109/ACCESS.2024.3411029

RESEARCH ARTICLE

A Density-Aware Point Cloud Geometry Compression Leveraging Cluster-Centric Processing

FARIHA AFSANA[1], (Graduate Student Member, IEEE), MANORANJAN PAUL[1], (Senior Member, IEEE), FARANAK TOHIDI[1], (Member, IEEE), AND PAN GAO[2], (Member, IEEE)
[1]School of Computing, Mathematics and Engineering, Charles Sturt University, Bathurst, NSW 2795, Australia
[2]College of Computer Science and Technology, Nanjing University of Aeronautics and Astronautics, Nanjing 210016, China
Corresponding author: Fariha Afsana (fafsana@csu.edu.au)

ABSTRACT Recent years have encountered a noticeable expansion of point cloud-based 3D applications that compels the necessity of high-efficiency point cloud compression. Preserving the local density of point cloud is crucial for compression, however, it's challenging due to the unordered nature of points in 3D space and has been overlooked by the majority of the existing compression methods. Recently, the farthest point method has demonstrated effectiveness in sampling point clouds across various state-of-the-art methods, its limitation lies in being non-density aware, yielding reconstructions that fall short of the desired quality. Intending to achieve density-aware compression for improved reconstruction, this paper proposes an end-to-end learnable cluster-based compression method for efficient lossy point cloud geometry compression. Our method utilizes an autoencoder architecture that performs point-wise operation for compression and reconstruction. To extract effective latent features, the encoder segments the point clouds into clusters using a different approach than farthest point sampling that is capable of well capturing uneven point densities and employs cluster-wise compression independently. To enhance density retention further, we leveraged the capabilities of an attention mechanism, allowing it to learn complex point-wise dependencies within clusters and effectively capture local density information. At the decoder, decompressed clusters are accumulated to reconstruct point clouds completely. In terms of rate-distortion trade-off, the experimental analysis reveals the superiority of the proposed method over prior arts. Furthermore, our approach adapts well to different datasets, for example, the model learned from the ModelNet40 dataset works well and achieves state-of-the-art performances on ShapeNet datasets. Finally, the qualitative comparison demonstrates that the proposed framework can preserve satisfactory local geometry details of point clouds compared to existing Representative methods.

INDEX TERMS Autoencoder, clustering, deep learning, geometry compression, point cloud compression, point-based learning.

I. INTRODUCTION

In the recent years, the development of 3D acquisition and scanning technologies such as LiDAR, stereo vision, structured light, Time of Flight, and so on, has accelerated due to the diverse industrial demands and technological advancements. These technologies are capable of capturing the realistic representation of 3D objects and scenes with accurate geometric details and attributes using different modalities of 3D data like volumetric grids, point clouds, depth images, and meshes [1], [2]. Among these, point cloud (PC) has emerged as an essential data format to address

The associate editor coordinating the review of this manuscript and approving it for publication was Mohamed Elhoseny.

the growing need for representing real-world objects and environments, allowing dynamic exploration across users' perspectives in hyper-realistic visual applications [3], [4], [5]. Point clouds are collections of sparsely and non-uniformly distributed unstructured 3D points dissipated in the 3D space that can be employed to describe a 3D surface. Each point has Cartesian coordinates (X, Y, Z) to describe the geometry information and associated photometry attributes such as RGB color values, opacity, and normal, to portray the visual appearance [6]. Due to the capability of carrying high-precision fine-grained details of 3D objects and scenes realistically, PC is sought after in many emerging visual applications including autonomous driving, Virtual/Augmented reality (VR/AR), mixed reality (MR), metaverse, immersive telepresence, gaming and robotics, medical imaging, heritage preservation, mining space and so on [7], [8], and [9]. However, densely sampled point clouds (containing up to a million or even billions of 3D points) necessitate enormous volume for storage and bandwidth and, thus, pose challenges for efficient storage and transmission [10]. Therefore, efficient point cloud compression becomes necessary to meet the bandwidth and storage requirements while preserving the quality.

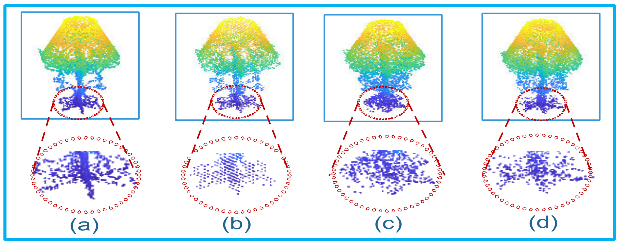


FIGURE 1. Local density observation from reconstructed PC. (a) Ground Truth ('Lamp' point cloud); (b) reconstructed PC using G-PCC [Bpp: 1.80, D1 PSNR: 38.71], points are uniformly distributed; (c) reconstructed PC using PDAE [24] [Bpp:1.82, D1 PSNR: 41.62], points are clustered; (d) reconstructed PC using proposed method [Bpp:1.82, D1 PSNR: 42.59], reconstructed points are capable of capturing local density.

Unlike 2D images, organized in well-structured pixel grids, 3D PCs comprise of geometric appearance having irregularly dissipated points embedded in 3D space. These properties have posed several challenges in PC compression compared to traditional image/video coding [11]. Firstly, the compression model is required to be capable of handling irregular and unordered input points. Additionally, the compression model needs to be generic so that it can cope with both small-scale and large-scale PCs due to variability in data density [12]. On top of that, the reconstructed output PCs should preserve local geometric structures rather than getting cluttered together or being uniformly distributed [13], see Figure 1. In this paper, our main focus is directed to the lossy Point Cloud Geometry (PCG) Compression.

The traditional compression methods commonly use binary space partitioning approaches like Octree [14] or KDtress [15] to address PC compression. However, these methods experience performance issues like generating blocky results at low bitrates [16], [17], [18]. To overcome the challenges of PCG compression, numerous deep learning-based approaches have been devised. Some methods employ voxelization and 3D convolution to optimize compression, however, at the cost of incurring high memory usage [3], [19], [20], [21]. Furthermore, these methods fail to fully exploit irregular and sparse point clouds and lose local density. With the remarkable achievement of point-based approaches like PointNet [22] and PointNet++ [23], operating directly on points has become the mainstream of PC analysis [7]. Embracing this direction, authors in [24] proposed a patch-based architecture for lossy geometry compression. Using patches, this method changed the global reconstruction problem into a local one and achieved state-of-the-art performance. However, while creating patches, this method sometimes fails to adapt to the underlying structure of non-uniformly distributed PCs due to relying on the farthest point sampling (FPS) method. Consequently, variations in point density go unnoticed. Thus, this method fails to well characterize the extracted features and fails to obtain efficient reconstruction.

To this end, in this paper, we propose an end-to-end deep autoencoder framework leveraging cluster-wise processing for high-efficiency PCG compression. Our motivation is to develop a deep learning architecture that is capable of better retaining the local geometry and density of PCs by exploiting local structural integrity across different segments of PCs. Towards this goal, the PCs from the input space are first segmented into clusters according to their geometric properties. Rather than feeding the whole PC into a deep autoencoder directly, we propose a cluster-wise compression inspired by [24]. At the encoder, pairwise interactions and long-range dependencies between cluster head and intra-cluster points are extracted as local features and then aggregated to retrieve global features. At the bottleneck, the latent representation is entropy encoded. Finally, at the decoder, each cluster is reconstructed separately and then, combined based on the information of the associated cluster head.

The main contributions of this paper are threefold.

First, we propose agglomerative clustering-based autoencoder architecture for lossy compression of PCG. The agglomerative clustering method considers local density during merging, resulting in more accurate clusters while due to depending on the farthest point as sampling criteria, the FPS scheme results in imbalanced clustering being unaware of uneven point distribution. Thus, the integration of the agglomerative method in the proposed autoencoder architecture is capable of considering the disparity of point cloud density and well captures the characteristics of points. To compensate for its computational intensity, we opt for a single-layer architecture instead of involving multiple stages in the compression process, thus reducing overall complexity.

Second, We combine an attention mechanism to more effectively leverage local density alongside a PointNet-based architecture as attention proves effective in capturing complex dependencies between points. While point-based processing is effective in certain scenarios, it cannot adequately retain the geometry details of points, leading to deficient reconstructions. The integration of position-embedding-based attention mechanism with point-based processing mutually complement each other, resulting in a more effective reconstruction that exploits local density to a greater extent.

Finally, we conduct comprehensive experimental analyses that reveal that the proposed method achieves better performance than state-of-the-art methods while keeping tolerable loss and ensuring better visual reconstruction. Ablation studies have verified the significant contribution of the clustering method and other modules to the final performance.

II. RELATED WORK

Numerous works have been carried out during the last decade to efficiently compress 3D point clouds. The existing PGC compression methods can be summarized as non-learning-based traditional approaches and deep learning-based approaches. The learning-based PGC methods are mainly focused on voxel-based, point-based, and attention-assisted solutions. Compared to traditional PCG compression algorithms, deep learning-based architectures have gained remarkable performance allowing more amenable structures for efficient neural processing.

A. TRADITIONAL NON-LEARNING-BASED METHODS

In the past few years, several tree representation-based approaches have been proposed for compressing point cloud data [17], [25], [26], [27], [28], [29], [30], [31], [32]. The octree-based PC compression [17] is the most straightforward traditional method to compress PCs. Octree [14] is capable of storing 3D data efficiently and can be used to represent maps that are commonly used in robotics [33], [34]. The method in [17] recursively partitions 3D coordinates in octants by utilizing an octree structure. By adopting a tree-based architecture, this method captures varying levels of details including spatial coherence based on the characteristics of PCs. At the lowest level of the octree hierarchy, each octant is considered as a leaf node and then the tree is then encoded. The octree decomposition along with other nested partitions is used in PC compression standard, MPEG G-PCC (Geometry-based Point Cloud Compression), also known as octree geometry codec [28], [35]. Zhu et al. in [4] proposed region-wise processing for PCG compression to exploit the inter-region redundancy. After region-wise clustering, this method used octree-based G-PCC to encode reference regions. Using spanning trees, other approaches focused on iterative prediction of neighboring points to compress point clouds [31], [32]. However, in real-world scenarios, tree structures can not exploit the full potential of 3D objects. Moreover, at low-bit rate scenarios, these algorithms struggle to perform well as the number of reconstructed points diminishes significantly with the decrease of the tree depth. Thus, the hand-crafted techniques can not be well-optimized while using large-scale data.

B. LEARNING-BASED METHODS

1) VOXEL-BASED DEEP ARCHITECTURES

The voxel-based point cloud compression methods represent PCs into 3D volumetric grids (voxels) and then employ 3D CNN-based autoencoders on the volumetric representation [19], [21], [36]. The authors modeled the 3D occupancy reconstruction as a classification problem via the binary cross entropy (BCE) loss optimization. Among these methods, the approach in [21] achieved leading compression efficiency at the cost of excessive complexity. Due to the sparseness of PC data, many 3D voxel spaces remain unoccupied resulting in redundant convolutional computation. Some methods integrate both voxel-based schemes and octree-based methods with deep learning to gain better performance, such as [37]. VoxelContext-Net [37] proposed voxel and octree-based hybrid architecture to learn the context in the previous octree depth. However, this method fails to extract features from sibling nodes [38]. Wang et al. in [3] proposed a learning-based approach that utilizes progressive re-sampling autoencoder to compress point cloud. In this method, features were extracted from the down-sampled PC to exploit the sparsity nature of the PC. Finally, reconstruction was conducted hierarchically from sparsely sampled points. Since this method was designed for dense voxelized point clouds, this might encounter a performance drop on the non-voxelized PCs.

2) POINT-BASED METHODS

Apart from the aforementioned volumetric models, point-based approaches directly process raw point clouds instead of data transformations beforehand. PointNet [22] was a groundbreaking point-based approach that takes raw points directly as input and employs a symmetric function using stacked shared MLP layers and max pooling to learn features from input. Yan et al. [39] proposed an autoencoder-based geometry compression codec that is point-based and used PointNet at the encoder and MLPs at the decoder. However, PointNet fails to obtain local features efficiently. To address the issue, PointNet++ [23] was developed on top of PointNet which performs multiple repetitive actions of sampling, grouping, and local PointNet operations to extract fine details of PCs. Huang and Liu in [40] proposed a hierarchical autoencoder resembling PointNet++ tailored specifically to operate on sparse point clouds at low resolution. Drawing inspiration from the success of the point-based methods, authors in [24] proposed a patch-based deep autoencoder for lossy geometry compression of PCs and improved the patch-based approach in [12] by including octree coding for sampling points. However, these point-based approaches sometimes fail to fully exploit the correlation across unevenly distributed points.

C. ATTENTION-ASSISTED METHODS

The application of self-attention based transformer modeling in large-scale pre-training has shown efficiency in powerfully modeling global contexts [41] and the success has inspired 3D point cloud research [42], [43]. Attention is an essential module in transformer architecture that is capable of learning feature information self-adaptively. In this mechanism, aggregated information from input data is assigned elevated weights so that the significance of features can be prioritized and the subsequent stages of computation can utilize the emphasis on salient features. The authors in [44] have introduced an effective feature learning backbone for 3D PC by proposing a transformer-based encoding framework. Li et al. [45] and wang et al. [43] attempted to perform a point cloud completion task by employing a self-attention based deep neural network allowing well-exploited inter-point correlations. Some other approaches employed autoencoder-based transformer architecture to conduct PCG compression [46], [47]. Wang et al. in [48] proposed a separable self-attention mechanism to enhance feature extraction by concurrently modeling local and global features. Despite achieving impressive outcomes, these methods experience excessive complexity in both the time and space domains due to integrating multiple consecutive stages in processing. In this paper, we propose a single-layer cluster-based autoencoder network, that well captures the local geometry of point clouds by utilizing point-based feature extraction and the representation learning capability of the self-attention mechanism. Due to adopting a single-layer network, our simple architecture maintains an acceptable level of complexity while preserving reconstruction quality.

III. PROPOSED METHOD

The proposed method relies on an end-to-end learnable autoencoder architecture to achieve efficient point cloud coding. The structure comprises three sequential steps: (i) generating clusters to exploit local properties of PCs efficiently; (ii) cluster-wise feature refinement and (iii) PC reconstruction by accumulating all the clusters. Figure 2 depicts the schematic representation of the proposed framework. Figure 3 and Figure 4 portray details of the corresponding basic parts in Figure 2. In brief, the architecture directly takes raw PCs as input and segments them into multiple clusters. The encoder then refines concise and effective features, concluding into a latent representation. The latent feature undergoes processing through quantizer and entropy coder and the final output is then directed to the decoder. Finally, the decoder maps the encoded representation back to reconstruct clusters and combines all clusters to reconstruct PCs.

A. GROUPING AND CLUSTERING

In this phase, the input point cloud, $P = \{x_i\}$ ($|P| = \eta$) with coordinates $x_i \in \mathbb{R}^3$, is divided into C number of clusters of the same resolution which will be compressed individually. To this aim, initially points in P are segmented into C number of groups by adopting an agglomerative clustering approach. Agglomerative clustering with complete linkage is a method that preserves the local density of points by considering the maximum dissimilarity (distance) between clusters during the merging process. It tends to group points that are close to each other and have similar local densities, effectively capturing the density structure within the dataset. Thus, utilizing this capability, initial data groups are successively merged to form C groups based on a complete linkage distance metric. Then, for each group in $\mathbb{C}$, ($\mathbb{C}$ is the set of resulting groups) a point that has the minimum average distance, $D_\mu(x_i)$, to all other points within the group is elected as group leader. A point, x_i is elected as group leader if it satisfies the following equation (1),

$$D_\mu(x_i) = min_{i=1}^{n} \left\{ \frac{1}{n-1} \sum_{j=1, j\neq i}^{n} ||x_i - x_j||_2 \right\} \tag{1}$$

Here, n is the total number of points in a group, and $||x_i - x_j||_2$) is the Euclidean distance between two points. These group leader points will be considered as cluster heads, $c_h = \{c_i | i = 1, 2, \cdots C\}$. Now, clusters will be generated around the c_h determined by the K-Nearest Neighbors (KNN) method. The clustering process ensures that the raw PC is divided into C clusters each having K points. The coordinates of the c_h will serve as the auxiliary information of the clusters and will be encoded losslessly so that these can be used as the central points at the decoder to reconstruct the PC. Finally, a set of relative coordinates of K points concerning the c_h; $x_i^{(j)} = x_i^{(j)} - x_i^{\prime(j)}$ for $i = 1, 2, \cdots K$ and $j = 1, 2, \cdots C$, where $x_i^\prime$ is the coordinate of c_h; is obtained to be used as input in the network model.

B. FEATURE REFINEMENT

This step comprises two components: feature extraction and enhancement. The feature extraction module takes $(\eta, 3)$ points as input in the form of clusters, each having $(K, 3)$ points, and extracts features in two simultaneous steps: (i) through set abstraction layer; (ii) through self-attention layer.

In the network model, a set abstraction layer is first used to learn the local patterns of each cluster at different levels of detail. The concept of the set abstraction layer is derived from PointNet++ [23] where it is formulated with three layers: sampling, grouping, and PointNet layer. As per the approach outlined in [24], the proposed set abstraction layer comprises solely grouping and neural network components to extract features. Each cluster, $C_i \in \mathbb{C}$ is abstracted by point-wise KNN, followed by multi-layer perceptrons (MLPs) to capture local details of the points. From $(C, K, 3)$ input, this step gives $F^s = (C, K, D)$ dimensional feature matrix. However, due to the lack of emphasis on dependencies among points, this module often fails to capture fine-grained local geometry. Thus, to create an enriched feature representation, a self-attention layer is also used in the feature extraction module. The self-attention mechanism is a key module in the transformer [41] that captures long-range and complex dependencies through weighted interactions between points. To employ this mechanism, at first the distribution of input $(C, K, 3)$ is captured by adding local position embedding

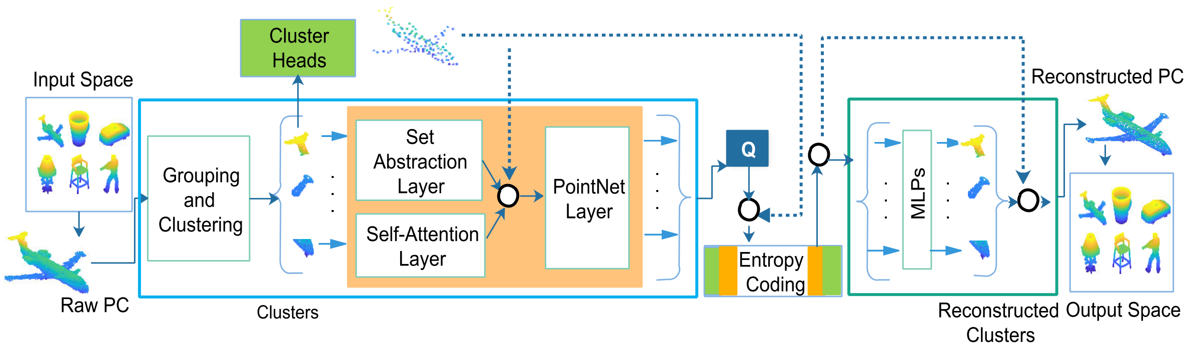


FIGURE 2. **Overview of the proposed PCG compression Framework. We propose to adopt a cluster-centric deep autoencoder architecture to learn local features and complex dependencies among points. The raw point clouds are divided into clusters and the encoder processes each cluster to extract a compact representation from them, while the decoder reconstructs the clusters and assembles them into output point clouds.**

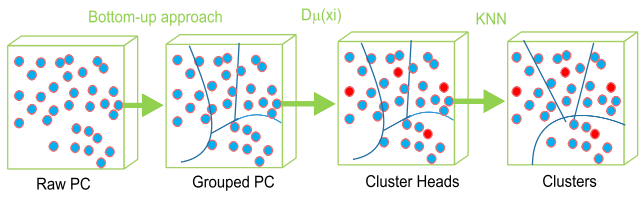


FIGURE 3. **Conceptual diagram of clustering process.**

that helps to attain spatial relationship. For each $x_i^{(j)} \in \mathbb{C}$, the direction (3D) and relative distance (scalar) of $x_i^{(j)}$ are calculated as below (equation (2)):

$$\left(\frac{x_i^{(j)} - x_i^{\prime(j)}}{||x_i^{(j)} - x_i^{\prime(j)}||_2}, ||x_i^{(j)} - x_i^{\prime(j)}||_2 \right); \quad \text{where, } x_i^{\prime(j)} \in c_h \text{ and } x_i^{(j)} \in \mathbb{C} \tag{2}$$

Consequently, the 4D vector representing point distribution is mapped to higher dimensional space using MLPs and generates a feature map, $\mathcal{F}^p = \{f_i\}$ as output. Given, input feature map, $\mathcal{F}^p \in \mathbb{R}^{Kx4}$, the embeddings are transformed in Query(α), Key(β) and Value (θ) using linear projections or MLPs. After that, the attention score map is obtained from the dot product of α and β, followed by its normalization through the softmax function. Finally, the weighted sum of the value vectors yields the output, $\mathcal{F}^A$, of the self-attention mechanism. Finally, this step gives $\mathcal{F}^A = (C, K, D')$ dimensional feature matrix. Thus, for the input, $\mathcal{F}^p\{f_i\}_i \in \mathbb{R}^{Kx4}$, the output feature map, $\mathcal{F}^A \in \mathbb{R}^{CxD'}$ can be obtained as follows [49] (equation (3)):

$$\mathcal{F}^A = \sum_{f_j \in \mathcal{F}^p} \rho(\alpha(f_j)\beta(f_i)^T + \delta)\theta(f_j) \tag{3}$$

Here, f_i denotes a feature vector under a set of feature vectors, $\mathcal{F}^p$, ρ denotes a normalization function such as *softmax* and δ stands for position encoding function.

As the conclusive step, two distinct feature matrices $\mathcal{F}^s$ and $\mathcal{F}^A$ are fused into the PointNet [22] module. This module enhances features using shared MLPs to each feature independently and captures local patterns. Finally, a summarized global feature vector $(C, 1, d)$ is obtained by aggregating the point-level features.

At the bottleneck, we have sampled clusters and per-point d- dimensional features. To enforce a more compact representation of the latent space, a uniform scalar quantization process is used, inspired by [24] and [50]. In the training phase, $(1, d)$ dimensional hidden features are augmented with uniform noise spanning between -0.5 and 0.5 to each element under each cluster. The uniform noise approximation makes the quantization process differentiable, allowing backpropagation during stochastic gradient descent optimization. In the testing phase, the hidden layer features undergo rounding operation. Quantized features are further compressed by an entropy encoder. In the training process, an arithmetic encoder is integrated to jointly optimize the feature entropies [50], [51]. This process is accompanied by a loss function details of which will be outlined later in III-D.

C. PC RECONSTRUCTION

At the decoder, the encoded features of each cluster are individually decoded. These decoded features are then added back to the cluster's head points to generate reconstructed clusters. To this aim, several fully connected MLPs are employed. At the final stage, the output from the MLPs is reshaped in the $\hat{K}x3$ point geometry matrix that represents one predicted cluster, where $\hat{K}$ is the number of points in the predicted cluster. Finally, the union of all the reconstructed clusters generates the final reconstructed PC, $\hat{P}$. It is worth mentioning that, in the compression process, the clusters are formed fulfilling the criteria, $C \times K = \gamma\eta$ $(\gamma > 1)$ that ensures maximum points to be captured in the whole process. Again, to maintain an equal number of points in reconstruction as the input, we set $\hat{K} = \frac{K}{\gamma}$ at the decoder.

D. RATE-DISTORTION OPTIMIZATION

Rate-distortion trade-off has been broadly used in data compression to attain a balance between the compression rate and distortion loss introduced by the compression process [3], [21]. It aims to minimize the necessary bit rate for representation while controlling the distortion within an acceptable threshold. For a better trade-off, we employ the standard rate distortion loss function (equation (4)):

$$\mathcal{L} = \mathcal{L}_{CD} + \lambda R \tag{4}$$

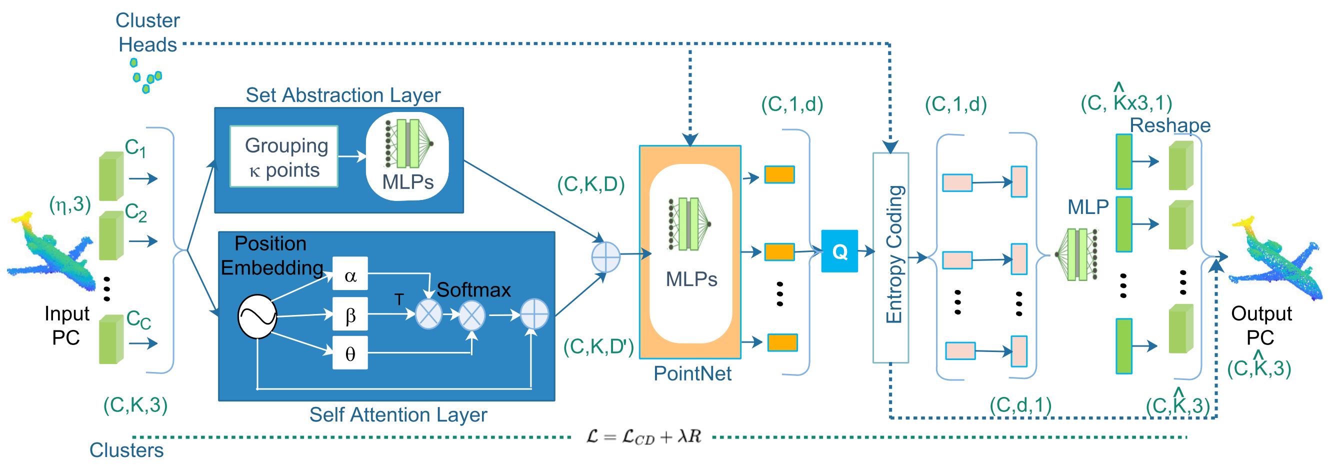


FIGURE 4. Detailed architecture of the proposed method. The model takes a PC of dimension (η, 3) as input and gives output PC of dimension ($C \times \hat{K}$, 3). Here, C is the number of clusters, K and $\hat{K}$ are the number of points in the encoded and decoded clusters, respectively, Q implies quantization, MLP stands for multi-layer perceptron. The Chamfer distance (CD) is used to quantify the alignment between input and output PC.

Here, R is the bitrate estimated by the entropy engine; λ is a Lagrange multiplier that controls the trade-off between rate and distortion; $\mathcal{L}_{CD}$ is the loss function which we use to measure the distortion between input and reconstructed results. Since the proposed compression method is conducted through cluster-wise processing, our rate-distortion optimization is also approximated in a group-wise manner. Given that, the reconstructed cluster is expected to be close enough to the input cluster, while preserving the local density as much as possible, we use the symmetric point-to-point Chamfer Distance to measure $\mathcal{L}$ in (4). For an input cluster, C_i, and associated reconstructed cluster, $\hat{C}_i$, the Chamfer loss, $\mathcal{L}_{CD}$ measures the mean distance of one point to its nearest neighbor between C_i and $\hat{C}_i$; which is shown below (equation (5)):

$$\mathcal{L}_{CD} = \frac{1}{c_b}\sum_{i=1}^{c_b}\mathcal{L}'_{CD}(C_i, \hat{C}_i)$$

$$\text{Where: } \mathcal{L}'_{CD}(C_i, \hat{C}_i) = \overline{d^2}(C_i, \hat{C}_i) + \overline{d^2}(\hat{C}_i, C_i)$$

$$\overline{d^2}(C_m, C_n) = \frac{1}{|C_m|}\sum_{x \in C_m} min_{y \in C_n}\|x - y\|_2^2 \qquad (5)$$

Here, c_b is the number of clusters in a batch during training; $\mathcal{L}'_{CD}$ represents the loss function for one cluster; x and y represent point from the cluster, C_i, and predicted cluster, $\hat{C}_i$, respectively.

To calculate rate consumption, R, we use the probability distribution of hidden layer features. The expression for rate loss, R, estimation is given as follows ((6)):

$$R = \frac{1}{c_b}\sum_{i=1}^{c_b}(-q(z_i|C_i) \cdot log_2 P_{z_i}(z_i)) \qquad (6)$$

where, z_i is the hidden representation after adding uniform noise for cluster C_i; $-q(z_i|C_i)$ is the actual marginal distribution of z_i [24] and $P_{z_i}(z_i)$ is the entropy model of z_i.

IV. EXPERIMENTAL EVALUATION

This section details the performance analysis of the proposed cluster-based PC compression method. The proposed method is evaluated by comparing its performance to state-of-the-art methods. Finally, ablation studies are provided as a justification to the design choices.

A. EXPERIMENT SETUP

1) DATASETS

To assess the effectiveness of the proposed autoencoder network, we choose to conduct experiments on ModelNet40 [52] and ShapeNet [53] datasets. The ModelNet40 contains 12311 CAD-generated objects from 40 categories. The ShapeNet dataset also consists of 3D objects of which we have used 2874 shapes for testing purposes. We followed all the same training/testing split and data pre-processing as suggested in [24]. Table 1 outlines the details of dataset splits.

TABLE 1. Dataset overview.

Dataset	Training/Testing split	No. of Shapes	Points
ModelNet40	Training	9843	8192
	Testing	2468	8192
ShapeNet	Testing	2874	2048

We have trained our proposed network using ModelNet40 training set and performed tests on the ModelNet40 test set. To further assess the robustness of the proposed method, we tested the trained model using the ShapeNet test dataset. It is worth mentioning that, the object models in ModelNet40 exhibit more complexity compared to Shapenet, irrespective of the size, orientation, and position of the PCs [24]. As a part of data preparation, 8192 points from ModelNet40 and 2048 points from Shapenet are sampled uniformly on each shape. Finally, all point coordinates are zoomed to [0, 63] to make a fair comparison with related methods.

2) BASELINES

We have compared our method with the learning-based method: patch-based deep autoencoder (PDAE) [24] and non-learning method: MPEG static point cloud codec TMC13 [17] which we will term as G-PCC here. Note that, the learning-based PDAE has been retrained on the same dataset as our method. We have used TMC13 [35] with all default settings enabled following the Common Test Conditions (CTC). Our dataset cannot be used directly in G-PCC as PCs are normalized. Therefore, each PC is scaled before applying G-PCC.

3) EVALUATION METRICS

The effectiveness of a compression algorithm depends on the trade-off between the compression ratio and reconstruction error. A compression algorithm is considered efficient if it achieves a good compression ratio while keeping the distortion low. However, due to the permutation-invariant nature of PCs, it is complicated to compare two PCs directly. To address this issue, we have adopted the symmetric point-to-point Chamfer distance (CD), a permutation invariant metric, to compare the reconstruction error between two point clouds. The compression rate is evaluated using bits per point (Bpp). We have used the point-to-point (D1) and point-to-plane (D2) symmetric PSNR [54] to test the reconstruction quality and BD-Rate measurement [55]. The D1 distortion is used to calculate the PSNR between the reconstructed point and its closest corresponding point in the reference PC, and the D2 distortion is used to calculate the PSNR between the reconstructed point and the surface plane in the reference PC.

4) IMPLEMENTATION DETAILS

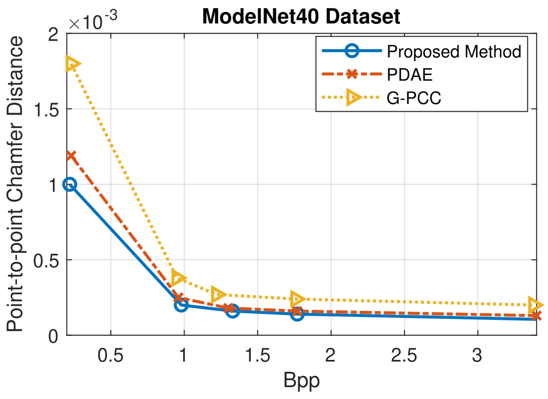
The proposed model is implemented on Python 3.9.6 and Pytorch 1.10.0. Training took place on an RTX 3080 GPU and with an AMD Ryzen 95950x 16−core processor CPU. The Adam optimizer [56] is used with an initial learning rate of 0.0005 and a learning decay rate of 0.1. The model is trained for 50 epochs with a batch size of 16. We varied the bottleneck size to 8 or 16. As the compression ratio differs with the cluster size, we varied the value of C to get different quality reconstructions. The total loss from equation (4) was optimized with $\lambda = 10^{-6}$. As in [24], for cluster division, we set $\gamma = 2$, i.e., $C \times K = 2\eta$ to cover the whole PC as much as possible. The number of input points is kept $\eta = 8192$ to speed up the training process. Only the x,y, and z-coordinates are considered as input features. On the decoder side, a fully connected neural network of size 256 × 512 × 1024 is used.

B. PERFORMANCE COMPARISON

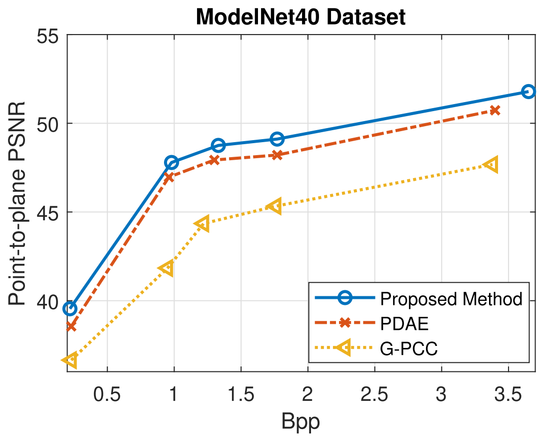
1) QUANTITATIVE COMPARISON

To verify the compression performance of the proposed method, in the first experiment, we compared our compression results to the baselines on the rate-distortion curve.

Figure 5 represents the point-to-point Chamfer distance (CD) and point-to-plane PSNR (D2 PSNR) of all methods



(a) CD vs Bpp



(b) D2 PSNR vs Bpp

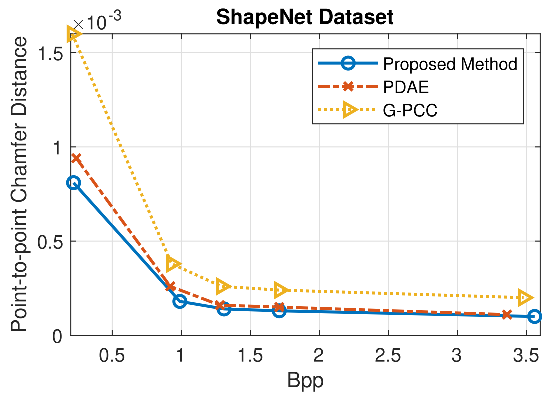
FIGURE 5. Compression results on the ModelNet40 dataset. We use PDAE [24] and MPEG G-PCC as baselines [17]. The proposed method consistently outperforms the previous methods across the full range of bitrates.

TABLE 2. BD-rate reduction and BD-PSNR gains against PDAE and G-PCC compression methods on ModelNet40 datasets.

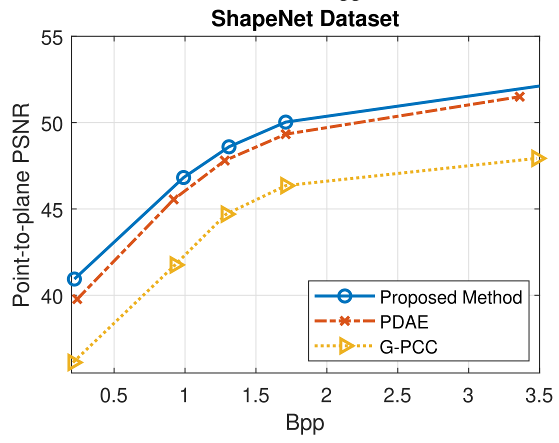
	D1		D2	
	BD-BR (%)	BD-PSNR (dB)	BD-BR (%)	BD-PSNR (dB)
Proposed method vs. PDAE	-26.54	0.78	-23.73	1.2
Proposed method vs. G-PCC	-50.16	5.04	-56.52	5.08

against Bpp. It is seen that our method persistently yields better performance in terms of CD and D2 PSNR across the full spectrum of Bpp on the ModelNet40 dataset. The proposed method achieves over 1.6 times lower distortion on average compared to the baselines. It is to be noted that, the rate-distortion curves are derived from the average of individual coding results of all example PCs in the test set.

Table 2 shows the BD-Rate reduction and BD-PSNR gains of the proposed method against baselines. It is seen that, for D1, the proposed method achieves 26.54% BD-Rate reduction and 0.78 *dB* BD-PSNR gain, and for D2, 23.73% BD-Rate reduction and 1.2 *dB* BD-PSNR gain compared to PDAE. When comparing the Proposed method with G-PCC,



(a) CD vs Bpp



(b) D2 PSNR vs Bpp

FIGURE 6. Compression results on the ShapeNet dataset using a model trained with ModelNet40 training set. The proposed method still outperforms the previous methods across the full range of bitrates.

D1 exhibits a significant BD-rate reduction of 50.16% along with a substantial BD-PSNR increase of 5.04 *dB*. For D2, the BD-rate reduction is 56.52% and 5.08 *dB* the BD-PSNR increases.

2) GENERALIZATION CAPABILITY

In an alternate dataset, when implemented, learning-based methods often degrade performance. We assert that the proposed model generalizes well to other datasets due to learning local geometries well. To validate this, we have tested our trained model (with ModelNet40 training set) on the ShapeNet dataset. The rate-distortion curves in terms of CD and D2 PSNR against Bpp in Figure 6 show that our method still achieves better performance and excellent robustness. The associated BD-Rate reduction and BD-PSNR improvements are also presented in Table 3.

3) QUALITATIVE COMPARISON

A visualization of the reconstruction quality of the proposed network with the baselines is displayed in Table 4. It can be observed that, at approximately equivalent bit rate, the proposed method produces better visual and objective results compared to the method in PDAE and G-PCC. For instance, for the 'Flowerpot' point cloud (1st row, 2nd column), PDAE fails to hold the local geometry of the root of the flowers. The G-PCC creates a similar visual object to ground truth, but if zoomed in, it can be seen that it assigns points in uniform distance rather than maintaining geometry details. The same holds true for the 'Car' point cloud (1st row, 1st column) where the geometry details near the four wheels are well preserved in the proposed method compared to that of PDAE and G-PCC. Overall, it is obvious that the proposed method achieves comparatively better geometry at low Bpp and preserves quality.

TABLE 3. BD-rate reduction and BD-PSNR gains against PDAE and G-PCC compression methods on Shapenet datasets.

	D1		D2	
	BD-BR (%)	BD-PSNR (dB)	BD-BR (%)	BD-PSNR (dB)
Proposed method vs. PDAE	-7.13	1.17	-16.96	0.82
Proposed method vs. G-PCC	-42.11	4.74	-54.03	4.70

4) COMPLEXITY ANALYSIS

To quantitively assess the computational complexity, we compare the average encoding and decoding time of the proposed method and PDAE method on ModelNet40 dataset. We also record the memory footprint of the two methods. As shown in Table 5, the average runtime of the proposed method has experienced a slight increase compared to that of the PDAE method. Regarding the size, the proposed model costs 6.6 *MB* on average, which is marginally larger than the PDAE method. While the proposed method exhibits a slightly higher level of runtime and memory usage, considering the achieved rate-distortion trade-off the rise is acceptable.

C. ABLATION STUDY

We further extend our studies by analyzing different aspects of the proposed method to enhance the depth of the understanding. All the studies are conducted on the ModelNet40 dataset. We build a baseline model consisting of a clustering module, a PointNet module, and fully connected MLPs at the decoder.

D. EFFECTIVENESS OF CLUSTERING METHOD

To understand the contribution of the grouping and clustering module, we examine the performance of the proposed method using the proposed clustering scheme as it is followed by its replacement with the widely used farthest point sampling (fps) method. Table 6 shows the impact of the clustering module for low-bit rate and higher-bit rate scenarios. It is observed that the proposed method outperforms the method with fps in both cases. At approximately similar bitrate, the proposed method achieves higher PSNR gain for both D1 and D2.

Figure 7 represents the cluster head distribution of the proposed method using the clustering scheme (green

TABLE 4. Visual comparison of the compression results of the proposed method and the baselines for closest Bpp on few example point clouds from ModelNet40 and ShapeNet test set. The proposed method produces better visual quality in terms of holding local geometry and lowest Bpp.

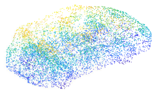
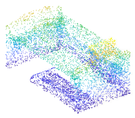
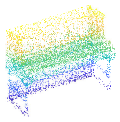
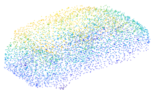
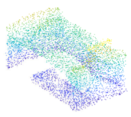
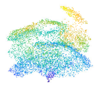
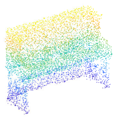
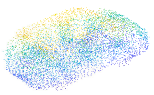
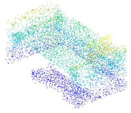
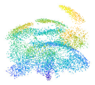
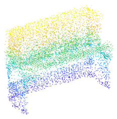
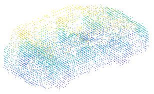
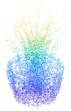
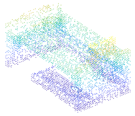
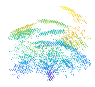
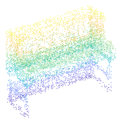
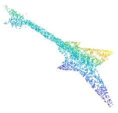
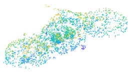
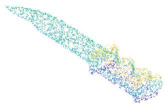
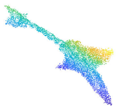
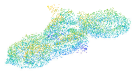
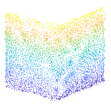
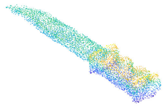
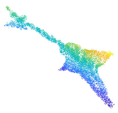
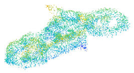
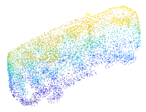
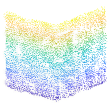
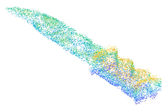
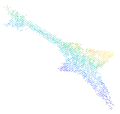
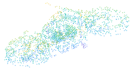
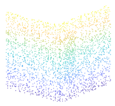
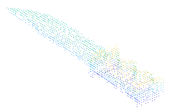
	ModelNet40 dataset example				
GT					
PDAE					
Bpp	1.82	2.19	1.91	1.80	1.94
PSNR	40.17	42.47	40.88	43.69	40.98
Ours					
Bpp	1.80	2.02	1.90	1.75	1.94
PDAE	40.83	43.90	41.41	44.50	41.17
G-PCC					
Bpp	1.80	2.20	1.98	1.80	
PSNR	37.21	37.04	37.39	37.21	37.38
	ShapeNet dataset example				
GT					
PDAE					
Bpp	1.30	1.27	1.27	1.27	1.29
PSNR	41.44	40.6	42.57	40.59	41.19
Ours					
Bpp	1.32	1.29	1.30	1.28	1.31
PDAE	42.47	41.68	43.55	41.62	41.62
G-PCC					
Bpp	1.32	1.26	1.45	1.30	1.27
PSNR	38.34	38.22	38.23	38.59	37.92

TABLE 5. Average runtime and model size in ModelNet40 test set.

Method	Enc. Time (s)	Dec. Time (s)	Total time (s)	Model Size (MB)
PDAE	64.62	16.8	81.42	6.56
Proposed method	72.6	19.8	92.4	6.60

TABLE 6. The effectiveness of the grouping method. The proposed method performs well when the bottom-up clustering approach is applied instead of the farthest-point sampling method while grouping points.

Grouping Method	Bpp	D1 PSNR	D2 PSNR
Using FPS	0.90	40.11	46.817
Proposed Clustering Method	**0.88**	**40.46**	**47.07**
Using FPS	3.65	43.67	51.33
Proposed Clustering Method	**3.65**	**43.91**	**51.78**

TABLE 7. The effectiveness of each component in the proposed method.

Components	Absent components	D1		D2	
		BD-BR (%)	BD-PSNR (dB)	BD-BR (%)	BD-PSNR (dB)
Baseline + Self-Attention Layer	Set Abstraction Layer	-29.99	0.46	-24.76	0.55
Baseline + Set Abstraction Layer	Self-Attention Layer	-25.18	0.88	-23.15	1.31

dot points) and fps scheme (red dot points). For better comprehension, we observe the extended part of the image (marked with a green box) where it is found that the proposed method assigns two cluster heads (two green dots) to group that portion while the fps method assigns three cluster heads (three red dots) which is redundant. Since the fps method selects cluster centers based on the points farthest from each other in the dataset, it fails to group unevenly distributed points properly. In the figure (Figure 7a), the marked area contains fewer data points which could be grouped using less number of clusters and assigned the other points where data is more dense. This work is successfully conducted by the proposed method, thus, it is capable of defining local groups more efficiently compared to the fps method.

E. EFFECTIVENESS OF EACH COMPONENT

To comprehend the individual impact of each module on the entire network, we conduct experiments on the baseline (prepared for the ablation studies) by adding the absent modules- the set abstraction layer and the self-attention layer alternatively. The results are presented in Table 7 where BD-BR reduction and BD-PSNR gain of the proposed method against the considered two networks ('Baseline + Self-Attention Layer' and 'Baseline + Set Abstraction Layer') are presented. For the "Baseline + Self-Attention Layer" configuration, the BD-BR values indicate, the proposed method achieves a bitrate reduction

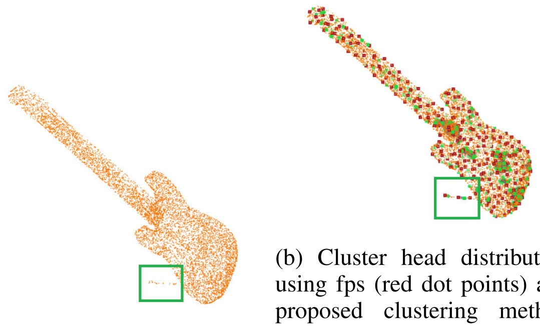

(a) Ground Truth (b) Cluster head distribution using fps (red dot points) and proposed clustering method (green dot points).

FIGURE 7. Visual comparison of the effectiveness of the proposed method.

of −29.99% (D1) and −24.76%($D2$), with corresponding BD-PSNR improvements of 0.46 dB (D1) and 0.55 dB (D2) over selected configuration. In contrast, for the "Baseline + Set Abstraction Layer" configuration, the proposed method achieves a bitrate reduction of −25.18% (D1) and −23.15% (D2), with BD-PSNR improvements of 0.88 dB (D1) and 1.31 dB (D2) against the selected configuration. These results illustrate that each module is contributing to the proposed network and they complement each other to attain favorable outcomes comprehensively.

V. CONCLUSION

In this work, we propose a cluster-based framework for lossy point cloud geometry compression via an autoencoder architecture. Leveraging a cluster-based approach, the proposed method excels in efficiently exploring local structures. Initiated with cluster division, the entire network is constructed through point-wise operations, complemented by a self-attention mechanism, and ultimately fused into a PointNet module, resulting in comprehensive and effective representation. Finally, The decoding phase, driven by MLPs, is designed to determine the final coordinates, aiding in the reconstruction process. The experimental results reveal that the proposed method not only achieves the best rate-distortion trade-off compared to prior methods but also demonstrates better recovery of local density. Qualitatively, the proposed method also achieves better reconstruction quality. Further ablation studies have been conducted, providing additional insights into the efficacy of our approach.

REFERENCES

[1] A. Xiao, J. Huang, D. Guan, X. Zhang, S. Lu, and L. Shao, "Unsupervised point cloud representation learning with deep neural networks: A survey," *IEEE Trans. Pattern Anal. Mach. Intell.*, vol. 45, no. 9, pp. 11321–11339, Sep. 2023, doi: 10.1109/TPAMI.2023.3262786.

[2] C. Cao, M. Preda, V. Zakharchenko, E. S. Jang, and T. Zaharia, "Compression of sparse and dense dynamic point clouds—Methods and standards," *Proc. IEEE*, vol. 109, no. 9, pp. 1537–1558, Sep. 2021, doi: 10.1109/JPROC.2021.3085957.

[3] J. Wang, D. Ding, Z. Li, and Z. Ma, "Multiscale point cloud geometry compression," 2020, *arXiv:2011.03799*.

[4] W. Zhu, Y. Xu, D. Ding, Z. Ma, and M. Nilsson, "Lossy point cloud geometry compression via region-wise processing," *IEEE Trans. Circuits Syst. Video Technol.*, vol. 31, no. 12, pp. 4575–4589, Dec. 2021, doi: 10.1109/TCSVT.2021.3101852.

[5] P. Gao, S. Luo, and M. Paul, ‘‘Rate-distortion modeling for bit rate constrained point cloud compression,’’ *IEEE Trans. Circuits Syst. Video Technol.*, vol. 33, no. 5, pp. 2424–2438, May 2023, doi: 10.1109/TCSVT.2022.3223898.

[6] R. Hooda, W. D. Pan, and T. M. Syed, ‘‘A survey on 3D point cloud compression using machine learning approaches,’’ in *Proc. SoutheastCon*, Mar. 2022, pp. 522–529, doi: 10.1109/SoutheastCon48659.2022.9763998.

[7] B. Fei, W. Yang, W. Chen, Z. Li, Y. Li, T. Ma, X. Hu, and L. Ma, ‘‘Comprehensive review of deep learning-based 3D point cloud completion processing and analysis,’’ 2022, *arXiv:2203.03311*.

[8] F. Tohidi, M. Paul, and A. Ulhaq, ‘‘Efficient dynamic point cloud coding using slice-wise segmentation,’’ 2022, *arXiv:2208.08061*.

[9] F. Tohidi, P. Manoranjan, and A. Ulhaq, ‘‘Dynamic point cloud compression with cross-sectional approach,’’ in *Image and Video Technology* (Lecture Notes in Computer Science), H. Wang, W. Lin, P. Manoranjan, G. Xiao, K. L. Chan, X. Wang, G. Ping, and H. Jiang, Eds. Cham, Switzerland: Springer, 2023, pp. 61–74, doi: 10.1007/978-3-031-26431-3_6.

[10] X. Sheng, L. Li, D. Liu, Z. Xiong, Z. Li, and F. Wu, ‘‘Deep-PCAC: An end-to-end deep lossy compression framework for point cloud attributes,’’ *IEEE Trans. Multimedia*, vol. 24, pp. 2617–2632, 2022, doi: 10.1109/TMM.2021.3086711.

[11] Z. Chen, Z. Qian, S. Wang, and Q. Chen, ‘‘Point cloud compression with sibling context and surface priors,’’ 2022, *arXiv:2205.00760*.

[12] K. You, P. Gao, and Q. Li, ‘‘IPDAE: Improved patch-based deep autoencoder for lossy point cloud geometry compression,’’ 2022, *arXiv:2208.02519*.

[13] 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, doi: 10.1109/LRA.2021.3059633.

[14] D. Meagher, ‘‘Octree encoding: A new technique for the representation, manipulation and display of arbitrary 3-D objects by computer,’’ Image Process. Lab., Rensselaer Polytech. Inst., Troy, NY, USA, Tech. Rep. IPL-TR-80-111, 1980.

[15] J. L. Bentley, ‘‘Multidimensional binary search trees used for associative searching,’’ *Commun. ACM*, vol. 18, no. 9, pp. 509–517, Sep. 1975, doi: 10.1145/361002.361007.

[16] D. Minnen, J. Ballé, and G. Toderici, ‘‘Joint autoregressive and hierarchical priors for learned image compression,’’ 2018, *arXiv:1809.02736*.

[17] R. Schnabel and R. Klein, ‘‘Octree-based point-cloud compression,’’ in *Proc. Symp. Point-Based Graph.*, 2006, p. 10, doi: 10.2312/SPBG/SPBG06/111-120.

[18] 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)*, London, U.K., Jul. 2020, pp. 1–6, doi: 10.1109/ICME46284.2020.9102866.

[19] M. Quach, G. Valenzise, and F. Dufaux, ‘‘Learning convolutional transforms for lossy point cloud geometry compression,’’ in *Proc. IEEE Int. Conf. Image Process. (ICIP)*, Taiwan, Sep. 2019, pp. 4320–4324, doi: 10.1109/ICIP.2019.8803413.

[20] M. Quach, G. Valenzise, and F. Dufaux, ‘‘Improved deep point cloud geometry compression,’’ in *Proc. IEEE 22nd Int. Workshop Multimedia Signal Process. (MMSP)*, Tampere, Finland, Sep. 2020, pp. 1–6, doi: 10.1109/MMSP48831.2020.9287077.

[21] J. Wang, H. Zhu, H. Liu, and Z. Ma, ‘‘Lossy point cloud geometry compression via end-to-end learning,’’ *IEEE Trans. Circuits Syst. Video Technol.*, vol. 31, no. 12, pp. 4909–4923, Dec. 2021, doi: 10.1109/TCSVT.2021.3051377.

[22] C. R. Qi, H. Su, K. Mo, and L. J. Guibas, ‘‘PointNet: Deep learning on point sets for 3D classification and segmentation,’’ 2016, *arXiv:1612.00593*.

[23] C. R. Qi, L. Yi, H. Su, and L. J. Guibas, ‘‘PointNet++: Deep hierarchical feature learning on point sets in a metric space,’’ 2017, *arXiv:1706.02413*.

[24] K. You and P. Gao, ‘‘Patch-based deep autoencoder for point cloud geometry compression,’’ 2021, *arXiv:2110.09109*.

[25] Google. (2017). *Draco 3D Graphics Compression*. [Online]. Available: https://github.com/google/draco

[26] C. L. Jackins and S. L. Tanimoto, ‘‘Oct-trees and their use in representing three-dimensional objects,’’ *Comput. Graph. Image Process.*, vol. 14, no. 3, pp. 249–270, Nov. 1980, doi: 10.1016/0146-664x(80)90055-6.

[27] R. B. Rusu and S. Cousins, ‘‘3D is here: Point cloud library (PCL),’’ in *Proc. IEEE Int. Conf. Robot. Autom.*, Shanghai, China, May 2011, pp. 1–4, doi: 10.1109/ICRA.2011.5980567.

[28] S. Schwarz, M. Preda, V. Baroncini, M. Budagavi, P. Cesar, P. A. Chou, R. A. Cohen, M. Krivokuca, S. Lasserre, Z. Li, J. Llach, K. Mammou, R. Mekuria, O. Nakagami, E. Siahaan, A. Tabatabai, A. M. Tourapis, and V. Zakharchenko, ‘‘Emerging MPEG standards for point cloud compression,’’ *IEEE J. Emerg. Sel. Topics Circuits Syst.*, vol. 9, no. 1, pp. 133–148, Mar. 2019, doi: 10.1109/JETCAS.2018.2885981.

[29] G. Sandri, R. L. de Queiroz, and P. A. Chou, ‘‘Comments on ‘compression of 3D point clouds using a region-adaptive hierarchical transform,’’’ 2018, *arXiv:1805.09146*.

[30] R. L. de Queiroz and P. A. Chou, ‘‘Transform coding for point clouds using a Gaussian process model,’’ *IEEE Trans. Image Process.*, vol. 26, no. 7, pp. 3507–3517, Jul. 2017, doi: 10.1109/TIP.2017.2699922.

[31] S. Gumhold, Z. Kami, M. Isenburg, and H.-P. Seidel, ‘‘Predictive point-cloud compression,’’ in *Proc. ACM SIGGRAPH Sketches (SIGGRAPH)*, Los Angeles, CA, USA, 2005, p. 137, doi: 10.1145/1187112.1187277.

[32] B. Merry, P. Marais, and J. Gain, ‘‘Compression of dense and regular point clouds,’’ in *Proc. 4th Int. Conf. Comput. Graph., Virtual Reality, Visualisation Interact. Afr.*, New York, NY, USA, Jan. 2006, pp. 15–20, doi: 10.1145/1108590.1108593.

[33] J. Elseberg, D. Borrmann, and A. Nüchter, ‘‘One billion points in the cloud—An octree for efficient processing of 3D laser scans,’’ *ISPRS J. Photogramm. Remote Sens.*, vol. 76, pp. 76–88, Feb. 2013, doi: 10.1016/j.isprsjprs.2012.10.004.

[34] A. Hornung, K. M. Wurm, M. Bennewitz, C. Stachniss, and W. Burgard, ‘‘OctoMap: An efficient probabilistic 3D mapping framework based on octrees,’’ *Auton. Robots*, vol. 34, no. 3, pp. 189–206, Apr. 2013, doi: 10.1007/s10514-012-9321-0.

[35] K. Mammou. (2019). *PCC Test Model Category 13 V14. ISO/IEC JTC 1 (2019)*. [Online]. Available: https://github.com/MPEGGroup/mpeg-pcc-tmc13

[36] A. F. R. Guarda, N. M. M. Rodrigues, and F. Pereira, ‘‘Point cloud coding: Adopting a deep learning-based approach,’’ in *Proc. Picture Coding Symp. (PCS)*, Ningbo, China, Nov. 2019, pp. 1–5, doi: 10.1109/PCS48520.2019.8954537.

[37] 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)*, Nashville, TN, USA, Jun. 2021, pp. 6038–6047, doi: 10.1109/CVPR46437.2021.00598.

[38] C. Fu, G. Li, R. Song, W. Gao, and S. Liu, ‘‘OctAttention: Octree-based large-scale contexts model for point cloud compression,’’ 2022, *arXiv:2202.06028*.

[39] W. Yan, Y. Shao, S. Liu, T. H. Li, Z. Li, and G. Li, ‘‘Deep autoencoder-based lossy geometry compression for point clouds,’’ 2019, *arXiv:1905.03691*.

[40] T. Huang and Y. Liu, ‘‘3D point cloud geometry compression on deep learning,’’ in *Proc. 27th ACM Int. Conf. Multimedia*, Nice, France, Oct. 2019, pp. 890–898, doi: 10.1145/3343031.3351061.

[41] A. Vaswani, N. Shazeer, N. Parmar, J. Uszkoreit, L. Jones, A. N. Gomez, L. Kaiser, and I. Polosukhin, ‘‘Attention is all you need,’’ 2017, *arXiv:1706.03762*.

[42] Y. Sun, Y. Wang, Z. Liu, J. E. Siegel, and S. E. Sarma, ‘‘PointGrow: Autoregressively learned point cloud generation with self-attention,’’ 2018, *arXiv:1810.05591*.

[43] J. Wang, Y. Cui, D. Guo, J. Li, Q. Liu, and C. Shen, ‘‘PointAttN: You only need attention for point cloud completion,’’ 2022, *arXiv:2203.08485*.

[44] X. Pan, Z. Xia, S. Song, L. Erran Li, and G. Huang, ‘‘3D object detection with pointformer,’’ 2020, *arXiv:2012.11409*.

[45] G. Li, Y. Chen, M. Cheng, C. Wang, and J. Li, ‘‘N-DPC: Dense 3D point cloud completion based on improved multi-stage network,’’ in *Proc. 9th Int. Conf. Comput. Pattern Recognit.*, Xiamen, China, Oct. 2020, pp. 274–279, doi: 10.1145/3436369.3437421.

[46] J. Zhang, G. Liu, D. Ding, and Z. Ma, ‘‘Transformer and upsampling-based point cloud compression,’’ in *Proc. 1st Int. Workshop Adv. Point Cloud Compress., Process. Anal.*, Lisboa, Portugal, Oct. 2022, pp. 33–39, doi: 10.1145/3552457.3555731.

[47] Z. Liang and F. Liang, ‘‘TransPCC: Towards deep point cloud compression via transformers,’’ in *Proc. Int. Conf. Multimedia Retr.*, Newark, NJ, USA, Jun. 2022, pp. 1–5, doi: 10.1145/3512527.3531423.

[48] F. Wang, X. Wang, D. Lv, L. Zhou, and G. Shi, ‘‘Separable self-attention mechanism for point cloud local and global feature modeling,’’ *IEEE Access*, vol. 10, pp. 129823–129831, 2022, doi: 10.1109/ACCESS.2022.3228044.

[49] H. Zhao, L. Jiang, J. Jia, P. Torr, and V. Koltun, ‘‘Point transformer,’’ 2021, *arXiv:2012.09164*.

[50] J. Ballé, V. Laparra, and E. P. Simoncelli, "End-to-end optimized image compression," 2016, *arXiv:1611.01704*.
[51] J. Ballé, D. Minnen, S. Singh, S. Jin Hwang, and N. Johnston, "Variational image compression with a scale hyperprior," 2018, *arXiv:1802.01436*.
[52] Z. Wu, S. Song, A. Khosla, F. Yu, L. Zhang, X. Tang, and J. Xiao, "3D ShapeNets: A deep representation for volumetric shapes," in *Proc. IEEE Conf. Comput. Vis. Pattern Recognit. (CVPR)*, Jun. 2015, pp. 1912–1920, doi: 10.1109/CVPR.2015.7298801.
[53] A. X. Chang, T. Funkhouser, L. Guibas, P. Hanrahan, Q. Huang, Z. Li, S. Savarese, M. Savva, S. Song, H. Su, J. Xiao, L. Yi, and F. Yu, "ShapeNet: An information-rich 3D model repository," 2015, *arXiv:1512.03012*.
[54] D. Tian, H. Ochimizu, C. Feng, R. Cohen, and A. Vetro, "Geometric distortion metrics for point cloud compression," in *Proc. IEEE Int. Conf. Image Process. (ICIP)*, Sep. 2017, pp. 3460–3464, doi: 10.1109/ICIP.2017.8296925.
[55] G. Bjontegaard, *Calculation of Average PSNR Differences between RD-Curves*, document ITU SG16 VCEG-M33, 2001. [Online]. Available: https://cir.nii.ac.jp/crid/1571980074917801984
[56] D. P. Kingma and J. Ba, "Adam: A method for stochastic optimization," 2014, *arXiv:1412.6980*.

FARIHA AFSANA (Graduate Student Member, IEEE) received the B.Sc. (Hons.) and M.Sc. degrees in information technology from Jahangirnagar University, Dhaka, Bangladesh, in 2015 and 2016, respectively. She is currently pursuing the Ph.D. degree with Charles Sturt University, Bathurst, NSW, Australia. Her current research interests include point cloud compression, video coding, image processing, e-health, data mining, the Internet of Things, nanotechnology, and wireless sensor networks.

MANORANJAN PAUL (Senior Member, IEEE) received the Ph.D. degree from Monash University, Australia, in 2005. He was a Postdoctoral Research Fellow with the University of New South Wales, Monash University, and Nanyang Technological University. He is currently a Full Professor, the Director of the Computer Vision Laboratory, and the Leader of the Machine Vision and Digital Health (MaViDH) Research Group, Charles Sturt University, Australia. He has published around 200 peer-reviewed publications, including 72 journals. He was an invited keynote speaker in IEEE DICTA-17 and 13, CWCN-17, WoWMoM-14, and ICCIT-10. He has supervised 15 Ph.D. students to completion. His major research interests include video coding, image processing, digital health, wine technology, machine learning, EEG signal processing, eye tracking, and computer vision. He received more than $3.6 million in competitive external grants, including Australian Research Council (ARC) Discovery Grants and Australia-China Grant. He was awarded the ICT Researcher of the Year 2017 by Australian Computer Society. He was the General Chair of PSIVT-19 and the Program Chair of PSIVT-17 and DICTA-18. He is an Associate Editor of three top-ranked journals, including IEEE Transactions on Multimedia, IEEE Transactions on Circuits and Systems for Video Technology, and *EURASIP Journal on Advances in Signal Processing*.

FARANAK TOHIDI (Member, IEEE) received the B.Sc. degree in computer engineering from the University of Isfahan, in 1997, the M.Sc. degree in computer science (information security) from Universiti Teknologi Malaysia (UTM), in 2013, and the first Ph.D. degree in computer science from Yazd University, Iran. She is currently pursuing the second Ph.D. degree with Charles Sturt University, Bathurst, NSW, Australia. Her master's thesis was nominated to be patented. She was a Visiting Scholar with Charles Sturt University, Australia, in 2020, during the concluding year of the Ph.D. studies, and then she joined Charles Sturt University as a Researcher. She has been employed with Tehran University, for over 16 years, as a Computer Engineer, an IT Senior Executive, and the Director of IT. Her research interests include image and video processing, 3D point cloud compression, information security, and image authentication.

PAN GAO (Member, IEEE) received the Ph.D. degree in electronic engineering from the University of Southern Queensland (USQ), Toowoomba, Australia, in 2017. Since 2016, he has been with the College of Computer Science and Technology, Nanjing University of Aeronautics and Astronautics, Nanjing, China, where he is currently an Associate Professor. From 2018 to 2019, he was a Postdoctoral Research Fellow with the School of Computer Science and Statistics, Trinity College Dublin, Dublin, Ireland, working on the V-SENSE project. He has authored or coauthored more than 50 publications in scientific journals and international conferences. His research interests include point clouds, video compression, computer vision, and artificial intelligence.