EFFICIENT POINT CLOUD GEOMETRY COMPRESSION THROUGH NEIGHBORHOOD POINT TRANSFORMER

Anonymous authors
Paper under double-blind review

ABSTRACT

Although convolutional representation of multiscale sparse tensor demonstrated its superior efficiency to compress the Point Cloud Geometry (PCG) through exploiting cross-scale and same-scale correlations, its capacity was yet bounded. This is because 1) the fixed receptive field of the convolution cannot best characterize sparsely and irregularly distributed points; and 2) pretrained convolutions with fixed weights are insufficient to capture dynamic information conditioned on the input. This work proposes the Neighborhood Point transFormer (NPFormer) to replace the existing solutions by taking advantage of both convolution and attention mechanism to best exploit correlations under the multiscale representation framework for better geometry occupancy probability estimation. With this aim, a Neighborhood Point Attention layer (NPA) is devised and stacked with Sparse Convolution layers (SConvs) to form the NPFormer. In NPA, for each point, it uses its k Nearest Neighbors (kNN) to construct an adaptive local neighborhood; and then leverages the self-attention to dynamically aggregate information within this neighborhood. Compared with the anchor using standardized G-PCC, our method provides averaged 17% BD-Rate gains and 14% bitrate reduction for respective lossy and lossless modes when compressing the LiDAR point clouds (e.g. SemanticKITTI, Ford). There are also 20%-40% lossy BD-Rate improvement and 37%-53% lossless bitrate reduction for the compression of object point clouds (e.g. MVUB, MPEG 8i). Compared with the state-of-the-art solution using attention optimized octree codec, our approach requires much less decoding runtime with about 640$\times$ speedup on average, while still presenting better compression efficiency.

1 INTRODUCTION

Efficient compression of point clouds is increasingly demanded fast market adoption of virtual reality, augmented reality, and autonomous machinery technologies in various scenarios that require content caching and networking for service provisioning (Schwarz et al., 2019). However, it is challenging because it is hard to exploit inter-correlations among unstructured 3D points. To tackle it, numerous 3D representation models like uniform voxel (Wang et al., 2021c; Guarda et al., 2021; Quach et al., 2020), octree (Meagher, 1982), multiscale sparse tensor (Wang et al., 2021b;a), etc, are developed to explicitly specify neighborhood relations upon which rule- or learning-based approaches (Cao et al., 2021; Quach et al., 2022) are utilized to exploit inter-dependency among points for compression.

Among them, the convolutional representation of multiscale sparse tensor, noted as *SparsePCGC* (Wang et al., 2021a), has demonstrated encouraging performance on the compression of Point Cloud Geometry (PCG). This is mainly because of the utilization of multiscale representation to effectively exploit cross-scale and same-scale correlations for accurate geometric occupancy probability approximation, where 3D Sparse Convolution (SConv) is extensively used for neighborhood information characterization and embedding. However, the compression performance of *SparsePCGC* is still bounded because SConvs that come with the fixed receptive field and fixed weights (after training) cannot deal with dynamic content efficiently.

As inspired by the studies in (Park & Kim, 2022; Pan et al., 2022; Guo et al., 2022), we suggest the Neighborhood Point transFormer (NPFormer) to tackle the aforementioned issues. As in Fig. 1(a), we follow (Wang et al., 2021a) to use multiscale sparse tensors to process the collection of Positively Occupied Voxels (POVs) from one scale to another, where dyadic downscaling at all axes (e.g., x-,

y- and z-axis) in cartesian coordinate system is used to generate multiscale representations. Such dyadic downscaling is the same as the depth adaptation used in (Huang et al., 2020; Que et al., 2021; Fu et al., 2022) to squeeze and build parent-child octree structure for correlation exploration.

This work devises a novel Multistage Occupancy Probability Approximation (MOPA) structure to exploit correlations for occupancy probability estimation, where the MOPA accepts the sparse tensor of POVs from the preceding scale (e.g., $(i-2)$-th), and outputs the probability of each element of upscaled sparse tenor at the current scale (e.g., $(i-1)$-th) for encoding or decoding.

For each POV, we first apply an NPFormer for local information aggregation and embedding. The POVs with aggregated neighborhood features are then upscaled to generate eight child nodes (octants) in parallel. Then, the stacked NPFormer and Classifier modules stagewisely process the octants to produce occupancy probability for compression.

The NPFormer is generally comprised of convolutional layers that use SConv, and Neighborhood Point Attention (NPA) layers that apply the self-attention to local k Nearest Neighbors (kNN). Note that instantaneous kNN is formulated for each POV. Together with the self-attention computation, the NPA layers can best capture the content dynamics, which makes the information embedding more robust and efficient (Lu et al., 2022).

Extensive experiments have reported the superior efficiency of the proposed method for compressing various large-scale point clouds. Having the anchor using standard compliant *G-PCC* (Geometry based Point Cloud Compression) (WG7, 2021), for LiDAR PCGs using SemanticKITTI and Ford, our method shows $>$17% BD-Rate (Bjøntegaard Delta Rate) gain (Bjøntegaard, 2001) for lossy compression, and $>$ 14% compression ratio improvement in lossless modes, which can be translated to about 10 absolute percentage points improvement over the respective gains obtained by the *SparsePCGC* (Wang et al., 2021a). There is also 20%-40% lossy BD-Rate improvement and 37%-53% bitrate reduction for compressing object PCGs from MVUB and 8i datasets.

In the meantime, compared with the state-of-the-art (SOTA) *OctAttention* (Fu et al., 2022) using attention optimized octree codec, our method not only provides compression improvement, particularly at high bitrates that are more preferred by high-precision tasks in autonomous driving (e.g., $\approx$7% BD-Rate gains from 6 to 16 bpp), but also enormously reduces the decoding complexity to several orders of magnitude, e.g., less than 6 seconds of the proposed method vs. $\approx$1 hour of the *OctAttention* when decoding SemanticKITTI sequences (about 640$\times$ speedup on average in Table 3).

Contributions of this work include: 1) We propose an NPFormer that takes advantage of both convolution and self-attention for better information aggregation; 2) The proposed NPA dynamically characterizes and embeds information among k nearest neighbors, and significantly reduces the complexity, promising attractive prospects for practical applications; 3) Together with the multiscale sparse tensor representation, the proposed method can better exploit cross-scale and same-scale correlations with the state-of-the-art efficiency on the compression of various PCGs.

2 Related Work

Explicit 3D Representation Models of PCG. The uniform voxel model (Wang et al., 2021c; Guarda et al., 2021; Quach et al., 2020) is the most straightforward way for PCG representation, where 3D convolutions are often utilized. The octree model is a lightweight and effective approach through the use of parent-child tree decomposition, with which the compression efficiency is improved by carefully exploiting parent-child dependency. Notable coding solutions using octree include standardized *G-PCC* that applies rules-based contexts, and *OctSqueeze* (Huang et al., 2020), *MuSCLE* (Biswas et al., 2020), *VoxelContextNet* (Que et al., 2021), *OctAttention* (Fu et al., 2022), etc that utilizes neural networks like CNN (Convolutional Neural Network), MLP (Multi-Layer Perceptron), and Transformers to improve the context modeling. Recently, a promising multiscale sparse tensor representation using stacked sparse convolutions has emerged (Wang et al., 2021b;a) with great flexibility to exploit cross-scale and same-scale correlations for better compression.

Sparse Convolution. Upon the sparse tensor, it is naturally to apply the SConv as specified in (Choy et al., 2019):

$$f_u^{out} = \sum\nolimits_{i \in \mathbb{N}^3(u, C^{in})} W_i f_{u+i}^{in} \quad \text{for} \quad u \in C^{out}, \tag{1}$$

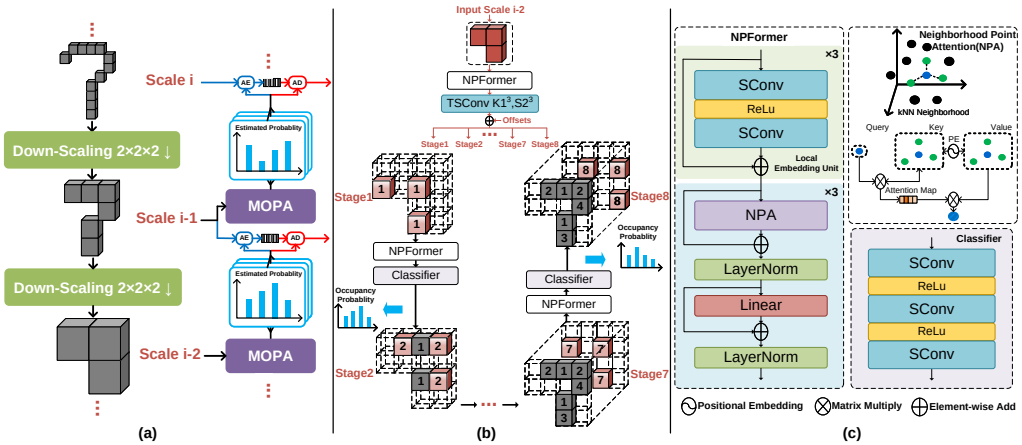

Figure 1: **Framework.** (a) Multiscale Sparse Tensor Representation exploits cross-scale and same-scale correlations using MOPA (Multistage Occupancy Probability Approximation). Estimated probability is used to perform the entropy coding and decoding of geometric occupancy. (b) The architecture of MOPA, each MOPA inputs the sparse tensor of Positively Occupied Voxels (POVs) from the preceding scale, e.g., $(i-2)$-th, to produce occupancy probabilities for encoding/decoding at the current scale, e.g., $(i-1)$-th, where the NPFormer first aggregates neighborhood information at $(i-1)$-th scale for each POV, then TSConv upscales each POV (w/ aggregated neighborhood features) to eight child nodes (octants) in parallel and then the stacked NPFormer and Classifier stagewisely output occupancy probability and determine the occupancy status of each octant. (c) The architecture of NPFormer and Classifier modules. AE and AD are for Arithmetic Encoding and Decoding. SConv is Sparse Convolution, and TSConv is Transposed Sparse Convolution.

having C^{in} and C^{out} as the coordinate sets for input and output POVs. f_u^{in} and f_u^{out} are input and output feature vectors at coordinate u. $\mathbb{N}^3(u, C^{in}) = \{i | u + i \in C^{in}, i \in \mathbb{N}^3\}$ defines a 3D convolutional kernel with a predefined size like $3 \times 3 \times 3$ or $5 \times 5 \times 5$, covering a set of locations centered at u with offset i in C^{in}. W_i denotes the corresponding kernel weight at offset i centered at u. As seen, the neighborhood coverage for information aggregation is basically constrained by the size of convolutional kernel; and as in (1), convolutional weights W_is are fixed after training, which is incapable of effectively characterizing the dynamics of input unseen in training.

Self-attention & Transformer. Recently, attention mechanism and Transformer are migrated quickly to process point clouds, showing encouraging results for the different tasks including segmentation, classification, compression, etc (Guo et al., 2021; Zhao et al., 2021; Engel et al., 2021; Fu et al., 2022). However, existing solutions often demand huge computational costs and memory consumption that grows quadratically with the sequence length of underlying tokens. This complexity issue is of particular importance for practical applications. This work proposes the Neighborhood Point transFormer (NPFormer) taking advantage of both sparse convolution and Neighborhood Point Attention (NPA), where NPA not only pursues the local processing for lightweight complexity but also leverages self-attention weighting for dynamic information embedding.

3 METHOD

Figure 1(a) illustrates the compression framework using multiscale sparse tensor with the proposed MOPA to accurately estimate the occupancy probability for encoding and decoding. As aforementioned, dyadic downscaling is enforced to derive multiscale sparse tensors where scale-wise sparse tensors can be easily mapped to depth-wise octree-structured representation as well. Different from those octree coding approaches (Huang et al., 2020; Fu et al., 2022) that exploit correlations following the parent-child tree structure, we leverage cross-scale and same-scale correlations through the use of MOPA for compression.

3.1 MOPA - MULTISTAGE OCCUPANCY PROBABILITY APPROXIMATION

Figure 1(b) details the MOPA used in this work. Assuming the input sparse tensor at the $(i-2)$-th scale as the $\mathbb{S}_{i-2}$, the proposed MOPA processes all POVs in it to generate the occupancy probability of each element of its upscaled sparse tensor $\tilde{\mathbb{S}}_{i-1}$ at the $(i-1)$-th scale. Note that $\mathbb{S}_{i-2} \subset \tilde{\mathbb{S}}_{i-1}$ because the elements in $\tilde{\mathbb{S}}_i$ include both POVs and non-POVs. Whether this element is a POV or non-POV is fully known in encoder for encoding, while the status of POV or non-POV in decoder is determined by parsing the bitstream, through the use of occupancy probability respectively. The arithmetic coder is guided by estimated occupancy probablity output by MOPA. The closer the estimated probability is to the ground truth, the less the compression bitrate according to Shannon's theory (Shannon, 1948).

The MOPA includes three major steps. As for the processing from the $(i-2)$-th to the $(i-1)$-th scale,

- First, it stacks the NPFormer to aggregate neighborhood information at the $(i-2)$-th scale. The detail of NPFormer will be discussed later.
- Then, each POV with aggregated neighborhood features in $\mathbb{S}_{i-2}$ is upscaled using transposed SConv, e.g., "TSConv K1^3, S2^3", with convolutional kernel at a size of $1\times1\times1$ and upscaling stride of 2 at three axis dimensions, and added corresponding offset to generate eight child nodes (red cubic with number mark); Such operation can be executed in parallel for all POVs from the $(i-2)$ scale.
- Third, cascading NPFormer and Classifier modules are devised to stagewisely estimate the occupancy probability for each child node, where the probability approximation of succeeding child node (e.g., node "2" red cubic) also includes the occupancy status of proceeding child node (e.g., node "1" as POV in dark grey cubic). And it is also worth it to point out that this stage-wise operation is made concurrently for same-group child nodes at the $(i-1)$-th scale that are labeled with the same number but upscaled from different POVs from the $(i-2)$-th scale.

3.2 NPFORMER - NEIGHBORHOOD POINT TRANSFORMER

One advantage of the Transformer is that it can model long-range dependencies, but it incurs unbearable complexity at the same time. Recent studies (Park & Kim, 2022; Liu et al., 2021; Lu et al., 2022) have shown that 1) the introduction of inductive bias is beneficial for model training and convergence; 2) self-attention and convolution are complementary to each other to capture the full-spectrum information of the underlying signal; 3) The characteristic of self-attention that dynamically constructs weights conditioned on the input helps to build content-adaptive compression.

As seen in Fig. 1(c), the NPFormer is mainly comprised of the Local Embedding Unit, the NPA layer, the Layer Normalization layer and the Linear layer.

Local Embedding Unit aims to model neighborhood point contexts from low-level to high-level semantic primitives through stacked convolutional layers for subsequent NPA. It consists of two SConv layers and a residual connection (He et al., 2016). Simple ReLU is applied for activation.

Neighborhood Point Attention (NPA). The overall architecture of NPA is shown in Fig. 2. Assuming the input of a NPA layer is a sparse tensor consisting of d_{in}-dimensional features $F_{in} \in \mathbb{R}^{N\times d_{in}}$ and 3-dimensional coordinates $C_{in} \in \mathbb{R}^{N\times 3}$. We then conduct the kNN search for each element in input sparse tensor, yielding dynamic kNN tensor $\left\{C_{\text{kNN}} \in \mathbb{R}^{N\times k\times 3}, F_{\text{kNN}} \in \mathbb{R}^{N\times k\times d_{in}}\right\}$.

Unconstrained displacement of points allows us to accurately and flexibly represent 3D space. We propose to concatenate the relative position with corresponding features, e.g.,

$$F_e = concat(F_{\text{kNN}}, C_{\text{kNN}} - C_{in}), \;\; F_e \in \mathbb{R}^{N\times k\times (d_{in}+3)}, \tag{2}$$

with which we can best retain the spatial coherency for information characterization in local neighborhood.

As in Fig. 2, let Q, K_{kNN} and V_{kNN} be the $query$, key, and $value$ vectors respectively. $Q \in \mathbb{R}^{N\times d_e}$ is computed through a linear transformation of F_{in}; $K_{\text{kNN}} \in \mathbb{R}^{N\times k\times d_e}$ and $V_{\text{kNN}} \in \mathbb{R}^{N\times k\times d_e}$ are

computed by two separate linear transformations of F_e:

$$Q = F_{in} \cdot W_Q, \;\; K_{\text{kNN}} = F_e \cdot W_K, \;\; V_{\text{kNN}} = F_e \cdot W_V, \tag{3}$$

having linear transformations $W_Q \in \mathbb{R}^{d_{in} \times d_e}$ and $W_K, W_V \in \mathbb{R}^{(d_{in}+3) \times d_e}$. Here, d_{in} is the dimension of input feature F_{in} and d_e is the dimension of Q, K_{kNN} and V_{kNN}. Then the NPA can be calculated using:

$$\text{NPA}(Q, K_{\text{kNN}}, V_{\text{kNN}}) = softmax(\frac{QK_{\text{kNN}}^T}{\sqrt{d_e}})V_{\text{kNN}}. \tag{4}$$

The dot product of Q and K_{kNN} is divided by $\sqrt{d_e}$ and then passed through the $softmax$ function to derive the Attention Map, then the dot product of Attention Map and V_{kNN} is denoted as F_o. The output of NPA comprises of F_o and C_{in}.

Multihead NPA. To best capture correlations from different representation subspaces, we extend the aforementioned single-head NPA to multihead NPA. We use several different linear projections upon the $query$, key, and $value$ vectors, and then feed them into multiple NPA modules in parallel. The outputs of parallel NPAs are concatenated together and passed through a linear transformation. We generally adapt the number of attention heads (#ah) in Multihead NPA, e.g., #ah at $1, 2 \ldots 4$, and enforce the product of #ah and associated channels per head #cph fixed to 32 to have the fixed total dimensions for lightweight processing. As reported in Table 4, the result shows that having #ah = 4 and #cph = 8 is a balanced option.

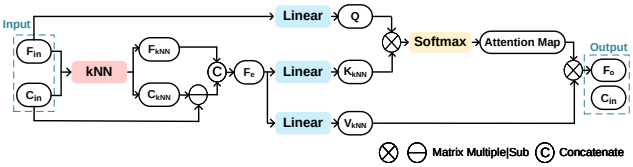

Figure 2: Neighborhood Point Attention Architecture.

Complexity Analysis. Having NPA computation on the local neighborhood with k elements avoids the quadratic increase of computation and memory cost with respect to the number of points contrast with methods using Global Self-Attention. Here, for simplicity, we use the single-head NPA as an example for discussion. Multihead NPA can be easily extended. Assuming the total amount of points is N, the number of neighbors in the local neighborhood is k, and the dimensional size of the feature is C. Note that we often have $k << N$ and $C << N$ in practical settings (e.g., $k = 16$, $C = 32$ in our examples for LiDAR PCGs but N is typically hundreds or tens of thousand). Thus, the computation complexity of NPA and Global Self-Attention can be approximated as:

$$\Omega(\text{GlobalSelfAttention}) = 3NC^2 + 2N^2C + N^2 \propto \mathcal{O}(N^2), \tag{5}$$

$$\Omega(\text{NPA}) = NC^2 + 2NkC^2 + 8NkC + Nk \propto \mathcal{O}(N). \tag{6}$$

As we can see, our NPA greatly reduces the complexity for attention computation, making our solution attractive in practices.

4 Results

4.1 Dataset

Object Point Cloud. Following the *OctAttention* (Fu et al., 2022), we use Soldier10 and Longdress10 sequences in MPEG 8i (d'Eon et al., May 2016), Andrew10, David10, and Sarah10 sequences in MVUB (Charles et al., May 2016) for training. Other point clouds in MPEG 8i (d'Eon et al., May 2016) including Thaidancer, Boxer, Redandblack, and Loot are selected for testing. In addition to 9-bit or 10-bit point clouds mentioned above, we also select some 12-bit precision samples to further evidence the efficiency of our method, including Head12 and Frog12 in *G-PCC* CTC (WG7, 2021) (Common Test Conditions) for training, and Boxer12 and Solider12 for testing.

LiDAR Point Cloud. We select SemanticKITTI (Behley et al., 2019) and Ford sequences (mpe) provided by the MPEG committee as typical LiDAR data for evaluation. SemanticKITTI is a publicly-accessible large-scale LiDAR point clouds dataset which includes 43,552 raw scans (frames) with 4.5 billions points collected from a Velodyne HDL-64E sensor. They are quantized to 1mm unit

precision, Then we use sequences from #00 to #10 (23,201 scans in total) for training, and the remaining sequences from #11 to #21 (20,351 scans) for testing, following the official training/testing split as suggested. Ford is dynamically acquired for MPEG point cloud compression standardization. It contains 3 sequences, e.g., Ford01, Ford02, and Ford03, each of which includes 1500 frames at 1mm precision (or 18 bits per geometry component). We take all of them as our testing sequence.

4.2 EXPERIMENTS

Baselines include both rules- and learning-based PCG coding solutions for comparative studies:

- *G-PCC* (tmc) uses the latest reference model TMC13v14 with rules-based octree codec to generate anchor following the common test conditions (CTC) defined in (WG7, 2021).
- *SparsePCGC* (Wang et al., 2021a) represents the solution that applies stacked SConvs only to process multiscale sparse tensors;
- *OctAttention* (Fu et al., 2022), *VoxelContextNet* (Que et al., 2021) & *OctSqueeze* (Huang et al., 2020) are learned octree codecs.

Training. As aforementioned, our MOPA works across the adjacent scales. Having the native resolution of training samples, e.g., scaling factor $S = 1$, we downscale them to lower scales to form a set of multiscale PCGs, e.g., (1/2, 1/4), (1/8, 1/16) etc, for model training.

For Object PCGs, the model is trained with S from 1 to $\frac{1}{8}$ and the number of neighbors k is set to 64. For LiDAR PCGs, concerning about its wider geometry precision range, we train two models (1 to $\frac{1}{8}$ and $\frac{1}{16}$ to $\frac{1}{128}$) for LiDAR PCGs to adapt different distributions cross scales, and corresponding k is set to 16.

Learned models are trained with binary cross entropy loss by (7) in a supervised end-to-end means, e.g.,

$$L_{\text{BCE}} = \frac{1}{N}\sum\nolimits_i -(x_i \log(p_i) + (1 - x_i)\log(1 - p_i)), \tag{7}$$

where x_i is the voxel label that is either truly occupied (1) or empty (0), and p_i is the probability of voxel-being-occupied, which is activated by the $sigmoid$ function.

Testing. We strictly follow the test conditions in other approaches or directly quote results from their publications. For example, when comparing with the anchor like *G-PCC* or *SparsePCGC* (Wang et al., 2021a), we scale input LiDAR PCGs with S from 1 to $\frac{1}{512}$ to obtain proper bitrates according to CTC (WG7, 2021).

When comparing with prevalent learned octree coding approaches, e.g., *OctAttention* (Fu et al., 2022), *VoxelContextNet* (Que et al., 2021), and *OctSqueeze* (Huang et al., 2020), testing LiDAR PCGs P are quantized from 8 to 12 bits per geometry component through the use of equation 8 defined in (Fu et al., 2022), e.g.,

$$P_Q = round((P - offset)/q_s), \quad q_s = 2/(2^D - 1), \quad offset = \min(P_x, P_y, P_z), \tag{8}$$

P is the normalized PCGs in $[-1, 1]$, D is the max depth of octree from 8 to 12. P_x, P_y and P_z represent coordinate level in a given PCG.

Evaluation Metrics. We closely follow the G-PCC CTC (WG7, 2021) to measure the bit rate using bpp (bit per input point), and quantitatively measure the distortion using PSNR (Peak Signal-to-Noise Ratio) of both point to point error (D1) and point to plane error (D2). Note that compression ratio gain measured by bpp is for lossless mode while the BD-Rate (Bjontegaard, 2001) evaluates the lossy compression.

When comparing with *G-PCC* and *SparsePCGC* (Wang et al., 2021a), the PSNR peak values are set according to CTC (WG7, 2021); When comparing with the *OctAttention* (Fu et al., 2022), *VoxelContextNet* (Que et al., 2021), and *OctSqueeze* (Huang et al., 2020), we normalize the points to the range of $(-1, 1)$ and set PSNR peak value to 1 following their rules.

For complexity measurement, we report the encoding and decoding runtime for reference. Tests are examined on a computer with an Intel Xeon 6226R CPU and an Nvidia GeForce RTX 3090 GPU.

Table 1: Compression Performance Evaluation Using Object PCGs for both Lossless and Lossy Modes. Anchor is standardized *G-PCC* (TMC13v14). *SparsePCGC* & *OctAttention* are also provided.

Object PCGs	Losslesss							Lossy			
	G-PCC	SparsePCGC		OctAttention		Ours		SparsePCGC		Ours	
	Bpp	Bpp	Gain over G-PCC	Bpp	Gain over G-PCC	Bpp	Gain over G-PCC	D1-BD-Rate	D2-BD-Rate	D1-BD-Rate	D2-BD-Rate
Boxer12	3.58	2.15	-39.94%	-	-	**1.48**	**-58.66%**	-15.71%	-16.60%	**-19.15%**	**-19.99%**
Solider12	3.86	2.62	-32.12%	-	-	**2.00**	**-48.19%**	-16.07%	-17.07%	**-19.79%**	**-20.70%**
Thaidancer10	0.99	0.65	-34.34%	0.65	-34.34%	**0.62**	**-37.37%**	-24.04%	-25.25%	**-28.86%**	**-30.06%**
Boxer10	0.94	0.60	-36.17%	0.59	-37.23%	**0.58**	**-38.30%**	-27.01%	-27.88%	**-32.86%**	**-33.77%**
Loot10	0.97	0.63	-35.05%	0.63	-35.06%	**0.61**	**-37.11%**	-24.90%	-26.02%	**-28.92%**	**-30.06%**
Redandblack10	1.10	0.72	-34.55%	0.74	-32.73%	**0.70**	**-36.36%**	-29.98%	-30.31%	**-33.63%**	**-34.16%**
Thaidancer9	0.99	0.64	-35.35%	0.64	-35.35%	**0.60**	**-39.39%**	-36.37%	-36.39%	**-39.44%**	**-39.45%**
Boxer9	0.96	0.60	-37.50%	0.59	-38.54%	**0.56**	**-41.67%**	-38.20%	-38.24%	**-42.48%**	**-42.51%**
Ave. (12-bit)	3.72	2.38	-36.03%	-	-	**1.74**	**-53.22%**	-15.89%	-16.83%	**-19.47%**	**-20.34%**
Ave. (10-bit)	1.00	0.65	-35.00%	0.65	-35.00%	**0.63**	**-37.00%**	-26.48%	-27.36%	**-31.07%**	**-32.01%**
Ave. (9-bit)	0.98	0.62	-36.73%	0.62	-36.73%	**0.58**	**-40.82%**	-37.28%	-37.31%	**-40.96%**	**-40.98%**

Table 2: Compression Performance Evaluation Using LiDAR PCGs for both Lossless and Lossy Scenarios. Anchor is standardized *G-PCC* using TMC13v14, and *SparsePCGC* is also provided.

LiDAR PCGs	Lossless					Lossy			
	G-PCC	SparsePCGC		Ours		SparsePCGC		Ours	
	Bpp	Bpp	Gain over G-PCC	Bpp	Gain over G-PCC	D1-BD-Rate	D2-BD-Rate	D1-BD-Rate	D2-BD-Rate
KITTI_vox2cm	7.62	7.06	-7.35%	**6.18**	**-18.90%**	-10.22%	-8.40%	**-18.01%**	**-17.96%**
Ford_vox2cm	9.95	9.69	-2.61%	**8.53**	**-14.27%**	-10.18%	-9.18%	**-18.53%**	**-18.53%**
KITTI_vox1mm	20.17	19.73	-2.18%	**16.62**	**-17.60%**	-6.80%	-6.24%	**-18.14%**	**-18.11%**
Ford_vox1mm	22.31	22.27	-0.18%	**19.86**	**-10.98%**	-6.12%	-5.51%	**-16.08%**	**-16.07%**
Average	15.01	14.69	-2.13%	**12.80**	**-14.72%**	-8.33%	-7.33%	**-17.69%**	**-17.67%**

Quantitative Efficiency. The results of lossless and lossy compression for Object PCGs are detailed in Table 1. It shows that our method achieves superior performance compared with previous methods including the *G-PCC*, *SparsePCGC* (Wang et al., 2021a), and *OctAttention* (Fu et al., 2022). Our method achieves 20%-40% BD-Rate improvement and 37%-53% bitrate reduction over *G-PCC* on average when using 12-, 10- and 9-bit Object PCGs.

As shown in Table 2, for LiDAR PCGs, the proposed method offers more than 14% and 17% gains (on average) over the *G-PCC* anchor for respective lossless and lossy scenarios across a variety of test sequences at different precisions (1mm and 2cm). Performance improvements for lossy mode are also confirmed by sample rate-distortion (R-D) plots in Fig. 3 where we can observe consistent improvements across a wide range of bitrates.

Having the same *G-PCC* anchor, our method offers >10 absolute percentage points increasement for both lossy and lossless compression of LiDAR PCGs in comparison to the *SparsePCGC* (see Table 2 and Fig. 3). Given that our method uses the same multiscale sparse tensor as the *SparsePCGC*, the performance improvement reports the superior efficiency of NPFormer by resolving the limitations of stacked SConvs used in (Wang et al., 2021a) for better occupancy probability approximation. As for Object PCGs, lossy BD-Rate improvement and lossless bitrate reduction (except losslessly-coded 12-bit samples) are about 2 to 5 absolute percentage points when comparing the proposed method to *SparsePCGC*. This is because denser point distribution in Object PCGs can be already modelled very well using SConvs to some extent. One exception observed for losslessly-compressed 12-bit samples, 17 absolute percentage points reduction is reported which might come from the sparser distribution of content that can be well characterized using NPFormer.

Comparisons are also conducted with other octree coding approaches as shown in Fig. 4. In the relatively lower bitrate range used by *OctAttention*, *VoxelContextNet*, and *OctSqueeze*, our method shows almost the same R-D behavior of *OctAttention* with overlapped R-D curves in Fig. 4. When

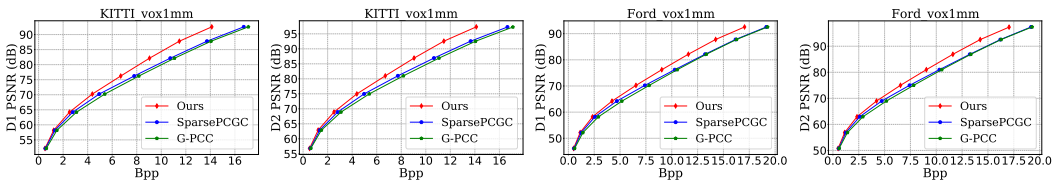

Figure 3: R-D comparison on SemanticKITTI and Ford samples across a wide range of bitrates.

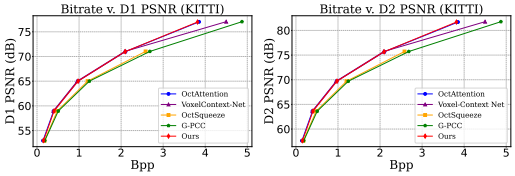

Figure 4: R-D comparison at lower bitrates

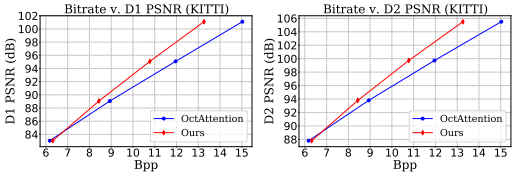

Figure 5: R-D comparison at higher bitrates

we extend the bitrates to higher bitrate range, our method shows gains over the SOTA *OctAttention*, e.g., $\approx 7\%$ BD-Rate improvement on average in Fig. 5

Note that we produce the high birate results of *OctAttention* by re-adapting its model to enable the max depth of octree from 13 to 16. Unfortunately, we could not provide R-D performance at higher bitrates for *VoxelContextNet* and *OctSqueeze* because of the lack of their source codes for high bitrate model training. However, since the *OctAttention* offers the SOTA efficiency, it is indeed a convincing representative of learned octree coding approaches.

Complexity. Comparisons of complexity are performed among representative methods including the *G-PCC*, *OctAttention*, *SparsePCGC*, and our method. We first compare the encoding and decoding time respectively at different bitrates with results shown in Table 3.

Our method shows comparable complexity for encoding and decoding, e.g., less than 6 seconds on average. Although there is still a gap compared with the traditional method of *G-PCC*, it is about 640$\times$ speedup of decoding when compared with the *OctAttention*, and about 2$\times$ speedup of *SparsePCGC*.

Model size measures the space complexity. The consumption for *OctAttention*, *SparePCGC*, and ours are 16.13, 3.95 and 8.96 Mbytes respectively. As seen, our method requires about a half of model size of the *OctAttention*, while it doubles the size as compared to that of the *SparsePCGC*.

Qualitative Visualization. Here, we offer additional qualitative visualizations of error map in Fig. 6 to further demonstrate the efficiency of our method. As we can clearly observe, our method provides the least reconstruction error at close bitrates for both SemanticKITTI and Ford test samples, e.g., as for the #000000 frame in SemanticKITTI #11 sequence, the Bpp of our method is 4.78 and the corresponding PSNR of the point-to-point (D1) error is 73.45 dB, while *SparsePCGC* Wang et al. (2021a) and *G-PCC* present lower PSNR at respective 71.99 dB & 70.22 dB and cost more bits, e.g., 4.90 Bpp & 4.86 Bpp. The similar observation happens for other testing frames.

4.3 Ablation Studies

This section examines the impact of neighborhood size k and number of attention heads #ah to get more insights on the proposed NPA.

Number of Attention Heads #ah. We adapt #ah at different bitrates to understand its impact on compression efficiency. To this aim, we adjust the scaling factor S from 1 (lossless) to 1/256 to reach

Table 3: Comparisons of encoding and decoding time measured in seconds. Various octree depths Ds are evaluated and SemanticKITTI sequences are exemplified. Time is measured for each sequence.

Encoding Time										
Methods	D=8	D=9	D=10	D=11	D=12	D=13	D=14	D=15	D=16	Ave.
G-PCC	0.50	0.62	0.82	1.07	1.61	2.05	2.53	2.78	3.02	1.67
OctAttention	0.40	0.49	0.54	0.44	0.66	1.51	1.91	2.22	2.58	1.19
SparsePCGC	2.08	2.91	4.36	7.11	10.58	14.70	18.68	22.79	20.88	11.57
Ours	1.18	1.38	1.79	2.62	4.11	5.35	8.05	10.65	13.34	5.39

Decoding Time										
Methods	D=8	D=9	D=10	D=11	D=12	D=13	D=14	D=15	D=16	Ave.
G-PCC	0.05	0.09	0.16	0.31	0.54	0.72	1.02	1.19	1.29	0.60
OctAttention	79.17	227.79	633.66	782.11	1569.79	6066.57	6865.55	6628.52	8914.32	3529.72
SparsePCGC	2.88	3.54	4.76	6.97	11.19	14.51	18.28	22.43	21.39	11.77
Ours	1.56	1.89	2.21	2.93	3.85	5.87	7.47	11.68	12.48	5.55

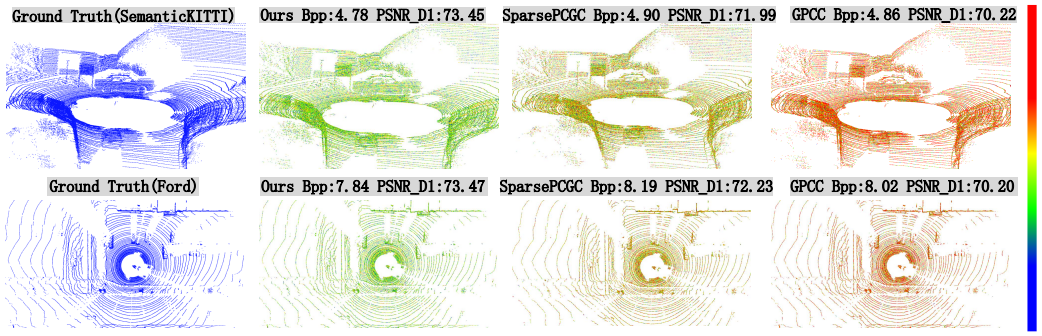

Figure 6: Qualitative visualization of LiDAR PCG reconstructions at typical bit rates. Color map is placed to reflect the reconstruction error distribution.

Table 4: Impact of using various number of attention heads #ah and nearest neighbors k on Bpp using KITTI sequences. Various scaling factors Ss means different precision of PCGs, the larger the value of S, the higher the precision of PCGs.

#ah	Bpp								
	S=1/256	S=1/128	S=1/64	S=1/32	S=1/16	S=1/8	S=1/4	S=1/2	S=1
1	**0.52**	**1.21**	**2.48**	**4.34**	6.71	9.23	11.92	14.74	17.57
2	**0.52**	1.22	2.51	4.37	**6.68**	9.06	11.55	14.24	16.78
4	0.53	1.23	2.52	4.38	6.70	**9.04**	**11.47**	**14.12**	**16.62**

k	Bpp								
	S=1/256	S=1/128	S=1/64	S=1/32	S=1/16	S=1/8	S=1/4	S=1/2	S=1
8	0.54	1.25	2.59	4.56	6.95	9.36	11.85	14.55	17.05
16	**0.53**	**1.23**	**2.52**	**4.38**	**6.70**	**9.04**	**11.47**	**14.12**	**16.62**
24	**0.53**	**1.23**	2.53	4.41	6.73	9.09	11.56	14.24	16.67
32	**0.53**	1.24	2.55	4.45	6.76	9.14	11.63	14.32	16.89

different bitrate points. Since the distortion is the same for different #ah at the same S, we only present the bpp in Table 4. to report the performance, i.e., the smaller bpp the better compression efficiency. In the end, we choose #ah = 4 in this work. Although the setting of #ah = 4 provides marginal loss at lower bitrates (e.g., $\approx 1\%$ BD-Rate loss when S is from 1/256 to 1/32), it improves the coding efficiency noticeably at higher bitrates, e.g., 5.41% bpp reduction when S = 1 (lossless mode), and $> 2.5\%$ BD-Rate improvement when S is from 1/16 to 1/2.

Number of Nearest Neighbors k. Table 4 gives averaged Bpps at different Ss when adapting the k in proposed NPFormer for the compression of LiDAR PCGs. Similarly, the smaller Bpp the better coding efficiency (for the same distortion at a given S). As seen, k = 16 gives the best compression performance. Having a k smaller than 16, performance is deteriorated because of insufficient neighbors used for information aggregation; while having a larger k may include more irrelevant (or uncorrelated) neighbors, making the attentive weighting compromised. Also, larger k expects extra computation overhead potentially. Recalling the comparison with the *OctAttention* in Fig. 4 and 5, a small scale of local neighborhood context may be already sufficient in compression.

5 Discussion

This paper suggested the use of Multistage Occupancy Probability Approximation (MOPA) for exploiting cross-scale and same-scale correlations under a multiscale representation framework. The MOPA is driven by a novel NPFormer taking advantage of both sparse convolution and self-attention to effectively aggregate local neighbors for better occupancy probability estimation. Such NPA adaptively weighs the contributions from k nearest neighbors that are constructed dynamically conditioned on the input point for effective information aggregation and embedding, fundamentally resolving the limitations of pretrained convolutions (e.g., fixed receptive field, and fixed kernels), Extensive experiments on Object PCG and LiDAR PCG demonstrate the effectiveness and superiority of the proposed method, reporting state-of-the-art efficiency in both lossy and lossless compression modes against the latest models including the *G-PCC*, *SparsePCGC*, and *OctAttention*.

REFERENCES

Mpeg pcc dataset. http://mpegfs.int-evry.fr/mpegcontent/. Accessed: 2022.

Mpeg-pcc-tmc13. https://github.com/MPEGGroup/mpeg-pcc-tmc13. Accessed: 2022.

Jens Behley, Martin Garbade, Andres Milioto, Jan Quenzel, Sven Behnke, Cyrill Stachniss, and Jurgen Gall. Semantickitti: A dataset for semantic scene understanding of lidar sequences. In Proceedings of the IEEE/CVF International Conference on Computer Vision (ICCV), October 2019.

Sourav Biswas, Jerry Liu, Kelvin Wong, Shenlong Wang, and Raquel Urtasun. Muscle: Multi sweep compression of lidar using deep entropy models. Advances in Neural Information Processing Systems, 33:22170–22181, 2020.

G. Bjøntegaard. Calculation of average PSNR differences between rd-curves. In ITU-T SG 16/Q6, 13th VCEG Meeting. document VCEG-M33, April 2001.

Gisle Bjontegaard. Calculation of average psnr differences between rd-curves. VCEG-M33, 2001.

Chao Cao, Marius Preda, Vladyslav Zakharchenko, Euee S. Jang, and Titus Zaharia. Compression of sparse and dense dynamic point clouds—methods and standards. Proceedings of the IEEE, 109 (9):1537–1558, 2021. doi: 10.1109/JPROC.2021.3085957.

Loop Charles, Cai Qin, O.Escolano Sergio, and A. Chou Philip. Microsoft voxelized upper bodies - a voxelized point cloud dataset. ISO/IEC JTC1/SC29 Joint WG11/WG1 (MPEG/JPEG) m38673/M72012, May 2016.

Christopher Choy, JunYoung Gwak, and Silvio Savarese. 4d spatio-temporal convnets: Minkowski convolutional neural networks. In IEEE CVPR, 2019.

Eugene d'Eon, Bob Harrison, Taos Myers, and Philip A. Chou. 8i voxelized full bodies - a voxelized point cloud dataset. ISO/IEC JTC1/SC29 Joint WG11/WG1 (MPEG/JPEG) m38673/M72012, May 2016.

Nico Engel, Vasileios Belagiannis, and Klaus Dietmayer. Point transformer. IEEE Access, 9: 134826–134840, 2021.

Chunyang Fu, Ge Li, Rui Song, Wei Gao, and Shan Liu. Octattention: Octree-based large-scale contexts model for point cloud compression. In Proceedings of the AAAI Conference on Artificial Intelligence (AAAI), 2022.

André F. R. Guarda, Nuno M. M. Rodrigues, and F. Pereira. Adaptive deep learning-based point cloud geometry coding. IEEE Journal of Selected Topics in Signal Processing, 15:415–430, 2021.

Jianyuan Guo, Kai Han, Han Wu, Yehui Tang, Xinghao Chen, Yunhe Wang, and Chang Xu. Cmt: Convolutional neural networks meet vision transformers. In Proceedings of the IEEE/CVF Conference on Computer Vision and Pattern Recognition, pp. 12175–12185, 2022.

Meng-Hao Guo, Jun-Xiong Cai, Zheng-Ning Liu, Tai-Jiang Mu, Ralph R. Martin, and Shi-Min Hu. Pct: Point cloud transformer. Computational Visual Media, 7(2):187–199, Apr 2021. ISSN 2096-0662. doi: 10.1007/s41095-021-0229-5. URL http://dx.doi.org/10.1007/s41095-021-0229-5.

Kaiming He, Xiangyu Zhang, Shaoqing Ren, and Jian Sun. Deep residual learning for image recognition. In IEEE CVPR, pp. 770–778, 2016.

Lila Huang, Shenlong Wang, K. Wong, Jerry Liu, and R. Urtasun. Octsqueeze: Octree-structured entropy model for lidar compression. 2020 IEEE/CVF Conference on Computer Vision and Pattern Recognition (CVPR), pp. 1310–1320, 2020.

Ze Liu, Yutong Lin, Yue Cao, Han Hu, Yixuan Wei, Zheng Zhang, Stephen Lin, and Baining Guo. Swin transformer: Hierarchical vision transformer using shifted windows. In Proceedings of the IEEE/CVF International Conference on Computer Vision, pp. 10012–10022, 2021.

Ming Lu, Peiyao Guo, Huiqing Shi, Chuntong Cao, and Zhan Ma. Transformer-based image compression. IEEE Data Compression Conference, 2022.

Donald Meagher. Geometric modeling using octree encoding. Computer graphics and image processing, 19(2):129–147, 1982.

Xuran Pan, Chunjiang Ge, Rui Lu, Shiji Song, Guanfu Chen, Zeyi Huang, and Gao Huang. On the integration of self-attention and convolution. In Proceedings of the IEEE/CVF Conference on Computer Vision and Pattern Recognition, pp. 815–825, 2022.

Namuk Park and Songkuk Kim. How do vision transformers work? arXiv preprint arXiv:2202.06709, 2022.

Maurice Quach, Giuseppe Valenzise, and F. Dufaux. Improved deep point cloud geometry compression. 2020 IEEE MMSP Workshop, 2020.

Maurice Quach, Jiahao Pang, Tian Dong, Giuseppe Valenzise, and Frédéric Dufaux. Survey on deep learning-based point cloud compression. Frontiers in Signal Processing, 2022.

Zizheng Que, Guo Lu, and Dong Xu. Voxelcontext-net: An octree based framework for point cloud compression. 2021 IEEE/CVF Conference on Computer Vision and Pattern Recognition (CVPR), 2021.

S. Schwarz, M. Preda, V. Baroncini, et al. Emerging mpeg standards for point cloud compression. IEEE Journal on Emerging and Selected Topics in Circuits and Systems, 9:133–148, 2019.

C E Shannon. A mathematical theory of communication, 1948. Bell System Technical Journal, 27 (3):3 – 55, 1948.

Jianqiang Wang, Dandan Ding, Zhu Li, Xiaoxing Feng, Chuntong Cao, and Zhan Ma. Sparse tensor-based multiscale representation for point cloud geometry compression. arXiv preprint arXiv:2111.10633, 2021a.

Jianqiang Wang, Dandan Ding, Zhu Li, and Zhan Ma. Multiscale point cloud geometry compression. In IEEE Data Compression Conference (DCC), pp. 73–82, 2021b.

Jianqiang Wang, Hao Zhu, Haojie Liu, and Zhan Ma. Lossy point cloud geometry compression via end-to-end learning. IEEE Transactions on Circuits and Systems for Video Technology, 31(12): 4909–4932, Dec. 2021c.

WG7. Common test conditions for g-pcc. ISO/IEC JTC1/SC29/WG11 N00106, 2021.

Hengshuang Zhao, Li Jiang, Jiaya Jia, Philip HS Torr, and Vladlen Koltun. Point transformer. In Proceedings of the IEEE/CVF International Conference on Computer Vision, pp. 16259–16268, 2021.