Point Cloud Compression with Sibling Context and Surface Priors

Zhili Chen, Zian Qian, Sukai Wang, and Qifeng Chen

The Hong Kong University of Science and Technology
{zchenei,zqianaa,swangcy,cqf}@ust.hk

Abstract. We present a novel octree-based multi-level framework for large-scale point cloud compression, which can organize sparse and unstructured point clouds in a memory-efficient way. In this framework, we propose a new entropy model that explores the hierarchical dependency in an octree using the context of siblings' children, ancestors, and neighbors to encode the occupancy information of each non-leaf octree node into a bitstream. Moreover, we locally fit quadratic surfaces with a voxel-based geometry-aware module to provide geometric priors in entropy encoding. These strong priors empower our entropy framework to encode the octree into a more compact bitstream. In the decoding stage, we apply a two-step heuristic strategy to restore point clouds with better reconstruction quality. The quantitative evaluation shows that our method outperforms state-of-the-art baselines with a bitrate improvement of 11-16% and 12-14% on the KITTI Odometry and nuScenes datasets, respectively.

Keywords: Point Cloud Compression, Autonomous Driving

1 Introduction

LiDAR is undergoing rapid development and becoming popular in robots and self-driving vehicles. As the eyes of those machines, LiDAR sensors can generate point clouds that provide accurate 3D geometry in diverse environments. However, the large number of point clouds incur a heavy burden on storage and bandwidth usage. For example, a single Velodyne HDL-64E can generate more than 450 gigabytes of data over 30 billion points in just eight hours of driving. Therefore, developing effective algorithms for 3D point cloud compression is imperative.

Unlike image and video compression, point cloud compression is technically challenging due to the sparseness of orderless point clouds. In the early years, researchers utilize different data structures such as octrees [18] and KD-trees [4] to organize unstructured data and ignore the sparsity of point clouds. However, these methods did not reduce the information redundancy hidden in the data structure representations. Therefore, the potential of reducing the bitrate of a point cloud is still not well explored.

Recent works have shown that learning-based methods have great potential in reducing information redundancy. By encoding point cloud information into an embedded representation, a neural network is used to further reduce the bitrate by predicting a better probability distribution in entropy coding [11,24]. Their approaches can be summarized into three steps: (1) constructing an octree; (2) fusing different features such as ancestor information [11] or neighbor information [24] to construct a contextual feature map; (3) and training a shared weight entropy model to reduce the length of the bitstream. However, their approaches have three limitations. First, although previous approaches largely focus on the spatial dependencies such as ancestor information [11] or neighbor information [24], they do not utilize prior knowledge of decoded siblings' children to reduce the information redundancy during the entropy coding. Second, they ignore local geometric information such as quadratic surface. Third, as illustrated in Fig. 3, since the distributions of occupancy symbols at different levels in an octree are very diverse, training a shared weight entropy model can overfit the octants at deeper levels and lead to low prediction accuracy on octants at shallower levels.

To address these issues, we propose a novel octree-based multi-level framework for point cloud compression. Our framework encodes the point cloud data into the non-leaf octants as eight-bit occupancy symbols. The entropy model encodes each occupancy symbol into a more compact bitstream through entropy coding by accurately estimating symbol occurrence probabilities. Compared to the previous methods, our entropy model is the first to utilize the siblings' children as strong priors when inferring the current octant's symbol probabilities. Our entropy model also incorporates the context of the current octant's neighbor and ancestor to explore the hierarchical contextual information of the octree fully. We also propose a quadratic surface fitting module to provide geometric priors, which is empirically proven to be beneficial in lowering the bitrate. In our compression framework, we train an independent entropy model for each octree level to capture resolution-specific context flow across different levels. At the decoding stage, we further improve the reconstructed point cloud quality by a two-step heuristic strategy that first predicts leaf octants' occupancies to retrieve the missing points by a specific level of entropy model and then apply a refinement module on the aggregated point cloud.

The contributions of this work can be summarized as follows.

- We build a novel octree-based multi-level compression framework for the point cloud. Our approach is the first one that incorporates sibling context for point cloud compression.
- We introduce surface priors into our entropy model, which effectively reduces octree-structured data's overall bitrate.
- In the decoding stage, we propose a two-step heuristic strategy by first predicting the missing points and then refining the aggregated point clouds to achieve a better reconstruction quality.

- Our proposed multi-level compression framework outperforms previous state-of-the-art methods on compression performance and reconstructs high-quality point clouds on two large-scale LiDAR point cloud datasets.

2 Related Work

2.1 Traditional Point Cloud Compression

Traditional point cloud methods usually utilize tree-based data structure to organize unstructured point cloud data, especially on KD-tree [4,9] and Octree [7,10,13,14,15,25,26]. Meagher et al. [18] directly utilize octree to encode the point cloud without reducing the information redundancy. Huang et al. [13] utilize the neighborhood information for better entropy encoding. To reduce spatial redundancy, Kammerl et al. [15] use XOR to encode sibling octants. In recent years, MPEG also developed an octree-based standard point cloud compression method G-PCC [10]. Moreover, Google utilizes KD-tree in its open-source point cloud compression software Draco [9]. Since all of the methods are hand-crafted, they cannot be implicitly optimized end-to-end. As such, the reduction of information redundancy is likely to be sub-optimal.

2.2 Learning-based Point Cloud Compression

Since point cloud data are in $n \times 3$ unstructured floating points, it is almost impossible to directly apply convolutions to raw point cloud data. Thus, researchers proposed several kinds of methods to organize raw point clouds. Point-based methods [12,35] are usually built based on PointNet framework [22,23]. Some point-based methods [34] directly downsample a point cloud to a small set of points, then use deconvolution to rebuild the point cloud. However, point-based methods usually suffer from the sparsity of point cloud data and incur high memory costs in processing. Therefore, it cannot handle extremely large LiDAR point cloud data. Range-image-based point cloud compression methods [1,20,27,28,31,32,33] first transfer point cloud into depth map or range image, then utilize state-of-the-art image compression methods such as [29] for compression. However, such kinds of methods highly depend on the quality of image compression algorithms and could not utilize spatial information. Voxel-based methods [2,11,21,24] ignore the sparsity of the point cloud data, and can utilize diverse geometric and spatial information. However, Huang et al. [11] only utilize the ancestor contexts, Que et al. [24] only use the neighbor contexts, and Fu et al. [6] ignore the local geometric information.

Unlike previous methods, our model is the first to reduce the inter-voxel redundancy by utilizing the context of decoded siblings' children octants. Moreover, local geometric information is also provided as the features for entropy model encoding. We also incorporate both ancestor and neighbor information into our entropy model to enrich the contextual information.

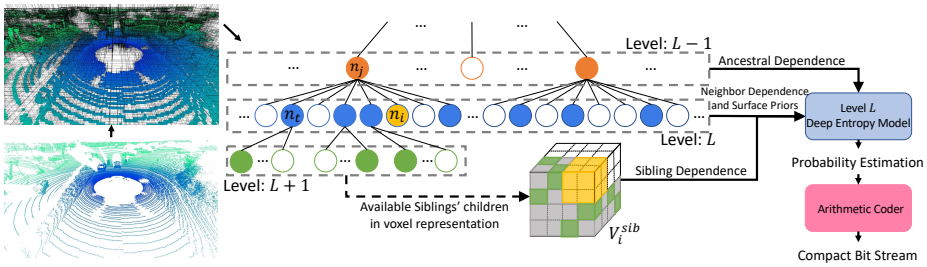

Fig. 1. The overview of our method. The input point cloud is first encoded into an octree. The ancestors, neighbors, and siblings' children octants are represented in orange, blue, and green colors. Our entropy model estimates occupancy probability distribution for each non-empty octant, conditioning on ancestral dependence, neighbor dependence, sibling dependence, and surface priors. Finally, the octree symbols are encoded into a more compact bitstream with arithmetic coding. The voxel block V_i^{sib} is formed by decoded siblings' children. The green and the gray voxels in V_i^{sib} represent the existent and absent siblings' children, respectively. The white voxels represent the unknown occupancies. After encoding the octant n_i in yellow, its occupancy symbol will be filled in the yellow voxels of V_i^{sib}.

3 Methodology

3.1 Overview

Our compression framework is shown in Fig. 1. Given an input point cloud, we first organize it using an octree with occupancy symbols stored at each non-leaf octant. A corresponding deep entropy model takes the hierarchical contexts as input for each octant to predict its occupancy symbol probabilities with 256 classes. The hierarchical contexts consist of local neighbors, ancestors, and siblings' children. The local neighbor context and sibling context are represented in binary voxel blocks. The ancestor information is an extracted feature map passed from the upper level. Moreover, we also incorporate our entropy model with surface priors by locally fitting quadratic surfaces among the local neighbors. Our entropy model can accurately predict symbol probabilities with these strong priors and further compress the symbols into a more compact bitstream by entropy coding.

In the decoding stage, we first reconstruct an octree with L levels from the compressed bitstream. We then propose a two-step heuristic strategy to produce a point cloud with better quality. As illustrated in Fig. 4, for each leaf octant, we first reuse our deep entropy model at level $L+1$ to retrieve its missing points by decoding the top-1 prediction of the occupancy symbol. Finally, we apply a separately trained refinement module [24] to further refine the aggregated points by adding the predicted offsets.

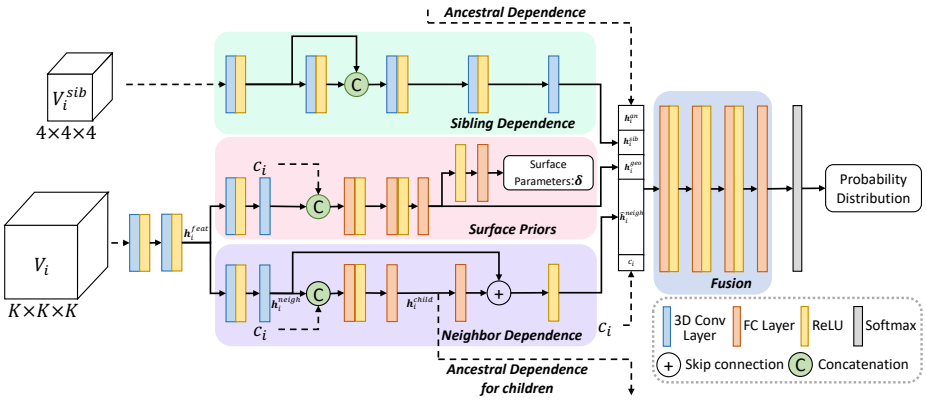

Fig. 2. The network architecture of our deep entropy model. The dash lines indicate the contextual input to our entropy model. The sibling and neighbor contexts are present as V_i^{sib} and V_i in voxel representations, c_i is the octant information (located tree level and corresponding coordinates) and h_i^{an} is the ancestor context. Note that the dimension K of the neighbor context V_i is empirically set to 9 in experiments.

3.2 Octree Construction

Due to the sparseness of the orderless point cloud data, we have to first organize it in a well-structured representation before subsequent processing. Unlike other popular structures, such as voxel-based methods, an octree is more memory efficient as it only partitions the non-empty 3D space and ignores the empty space. Moreover, the tree structure can provide hierarchical spatial information for a better reduction of information redundancy during compression. Therefore, we utilize octree to organize the point clouds.

An octree is built by recursively partitioning the input space into eight non-empty subspaces of the same size until the predefined depth is reached. The octant represents a bounding subspace and uses its center as the point coordinate. Each non-leaf occupied octant consists of an 8-bit occupancy symbol, and each bit represents the existence of a corresponding child (one for existence and zero for the absence). With such representation, a point cloud can be expressed as a serialized 8-bit occupancy symbols stream level-by-level and facilitate further lossless compression by an entropy encoder.

3.3 Hierarchical Context Entropy Model

Given an occupancy symbol stream $\mathbf{s} = [s_1, s_2, ..., s_n]$ with the number of non-leaf octants as its length n, the goal of our entropy model is to minimize the bitstream length. According to information theory, this goal can be reached by minimizing cross-entropy loss $\mathbb{E}_{s\sim p}[-\log q(\mathbf{s})]$, where $q(s)$ is the estimated probability distribution of occupancy symbol s and p is the actual distribution.

The accuracy of the probability distribution $q(s)$ determines the effectiveness of the entropy model. For example, if the actual probability distribution of s is

known, no bit is needed to encode the whole point cloud. However, formulating the actual $q(s)$ could be very difficult because it has many complex dependencies such as ancestor or neighbor context. A good formulation of $q(s)$ can provide strong priors for the entropy model and therefore reduce the information redundancy in a bitstream. As illustrated in Fig. 1, our entropy model utilizes hierarchical dependencies from coarse to fine with ancestors, neighbors, and decoded siblings' children. Moreover, we utilize surface information as geometric priors while estimating $q(s)$. Therefore, the formulation of $q(s)$ in our entropy model can be defined as

$$q(\mathbf{s}) = \prod_i q(s_i \mid h_i^{an}, h_i^{neigh}, h_i^{sib}, h_i^{geo}, c_i; \theta), \tag{1}$$

where h_i^{an} is the ancestral dependence, h_i^{neigh} is the neighbor dependence, h_i^{sib} is the siblings' children dependence, h_i^{geo} is the surface priors, c_i is the octant information (located tree level and corresponding coordinates), and θ is the parameters of our entropy model.

3.3.1 Neighbor Dependence

The octree at depth l ($l \in [1, L]$) can be considered as a discretization of the 3D space at the resolution of $2^l \cdot 2^l \cdot 2^l$. Inspired by [24], we construct the neighbor context of an octant n_i by locally forming a $K \times K \times K$ binary voxel block V_i centered at n_i and K is empirically set to 9 in experiments. Each binary value indicates the existence of its corresponding neighbors. The neighbor voxel V_i is transformed by a feature extraction function f_{neigh}:

$$h_i^{neigh} = f_{neigh}(V_i), \tag{2}$$

where h_i^{neigh} is the output feature vector that represents the neighbor contextual information. The feature vector h_i^{neigh} is provided as part of the prior knowledge for inferring the n_i's children distribution.

3.3.2 Ancestral Dependence

The ancestral information brings coarser geometric information from a shallower level to the current octant [11], which can enlarge the entropy model's receptive field. As illustrated in Fig. 2, since the entropy model is trained level-by-level, the neighbor context features h_i^{neigh} of current octant n_i can be further passed to its children as ancestral features.

To avoid ancestral features overwhelming the feature space of its children octants, we first concatenate the h_i^{neigh} with the octant information c_i and then apply an MLP denoted as φ to extract more condensed features h_i^{child}. The processed features h_i^{child} will then be passed to its children as their ancestral dependence. A following linear projection layer γ is applied on h_i^{child} to scale back the features to the original dimension. Then the feature is added with h_i^{neigh} through skip connection to recover the original neighbor feature $\hat{h}_i^{neigh}$ of

the current octant:

$$h_i^{child} = \varphi(\text{Concat}(h_i^{neigh}, c_i)), \hat{h}_i^{neigh} = \gamma(h_i^{child}) + h_i^{neigh}. \tag{3}$$

Note that the ancestral features received from the upper level for the current octant are denoted as h_i^{an}. For the root octant, we initialize its ancestral features with zero values.

3.3.3 Sibling Context

Sibling octants are adjacent in the original 3D space. The decoded children of siblings are represented in a finer voxel size and are even closer to the encoding/decoding octant. Similar to successive pixels in 2D images and videos, siblings' children are strongly correlated with the children of the current octant because the geometric structure of these siblings' children and the current octant's children can be quite similar. Therefore, having a condition on siblings' children can reduce the information redundancy while predicting the probability distribution of the current octant.

As illustrated in Fig. 1, an occupied octant n_j is located at the octree level $L-1$ and represents a space in the 3D world. Until the level of $L+1$, this space is partitioned into $4 \times 4 \times 4$ subspaces. We represent it as a binary voxel V_i^{sib}. Since our entropy model encodes the occupancy symbols of n_j's children sequentially, the previous siblings' occupancy symbols are available while predicting the probability of the current octant n_i. The available occupancy symbols are filled in V_i^{sib}. Note that the first occupied child octant of n_t has no available sibling context, and it will take V_i^{sib} with zero values as input. We parameterize another feature transformation function, f_{sib}, as a 3D convolution network to exploit the sibling context with V_i^{sib} as input. We formulate it as

$$h_i^{sib} = f_{sib}(V_i^{sib}). \tag{4}$$

The h_i^{sib} is the output feature vector that represents the sibling contextual information and will be further provided as a part of prior knowledge for entropy coding.

3.3.4 Geometric Priors

Since LiDAR is a time-of-flight device to estimate the round-trip time of the darting laser beams reflecting from a scene, most of the sampling LiDAR points are densely distributed on large surfaces, such as roads and vehicles. If an octant crosses the boundary of a surface, its children's octants are more likely to be occupied. Therefore, geometric information such as the surface can be a strong prior for reducing the data redundancies.

The neighbor context of n_i is defined as a voxel representation V_i in Section 3.3.1. We first transform the neighbor context V_i into a point representation $\{(x_i, y_i, z_i)\}_{i=1}^{N}$ with the coordinate of n_i as the origin. Then we train a module to locally fit a quadratic surface by minimizing the vertical distance from each

point to the surface:

$$\mathcal{L}_{sf} = \|\mathbf{z} - \boldsymbol{\delta}\left[\mathbf{x}^2, \mathbf{y}^2, \mathbf{xy}, \mathbf{x}, \mathbf{y}, \mathbf{1}\right]^T\|_2^2, \tag{5}$$

where $\boldsymbol{\delta} \in \mathbb{R}^6$ are six parameters of the quadratic surface. As illustrated in Fig. 2, the module that learns surface priors will take the low-level feature h_i^{feat} from the second layer of f_{neigh} as input. The surface estimation module f_{geo} extract the geometric features from h_i^{feat}, followed by a MLP to estimate the parameters $\boldsymbol{\delta}$ for the quadratic surface. The transformation can be defined as

$$h_i^{geo} = f_{geo}(h_i^{feat}), \tag{6}$$

$$\boldsymbol{\delta} = \mathbf{MLP}(h_i^{geo}), \tag{7}$$

where h_i^{geo} represents the geometric priors provided to the entropy model to infer the probability distribution.

3.3.5 Entropy Model Header

Eventually, the neighbor feature h_i^{neigh}, the ancestral feature h_i^{an}, siblings' children feature h_i^{sib}, the geometric feature h_i^{geo}, and the current octant's information c_i are aggregated to a four-layer MLP followed by a softmax to predict a 256 channels probability $q(s_i)$ for the 8-bit occupancy symbol. Then we minimize the cross-entropy loss on each octant:

$$\mathcal{L}_{CE} = -\sum_i p(s_i) \log q(s_i), \tag{8}$$

where $p(s_i)$ is the ground-truth probability distribution of the occupancy symbol s_i.

3.4 Multi-level Learning Framework

From our statistics on 12-level octrees constructed from the KITTI Odometry [8] dataset, the last level of the non-leaf octants accounts for more than 50% of the total number of all the non-leaf octants. We calculate the occupancy symbols distribution by levels and compute the Jensen–Shannon (JS) divergence among them. As shown in Fig. 3, the greater JS divergence means the larger difference between the two distributions.

Based on these two observations, we separately train entropy models for each level instead of training a single shared-weight entropy model on the entire octree. This multi-level learning framework facilitates our entropy model to better reduce each octant's spatial redundancies in different spatial resolutions.

We use the same objective function for the entropy models at different levels:

$$\mathcal{L} = \mathcal{L}_{CE} + \lambda \mathcal{L}_{sf}, \tag{9}$$

where λ is the weight of the loss and empirically set to 0.2 in experiments.

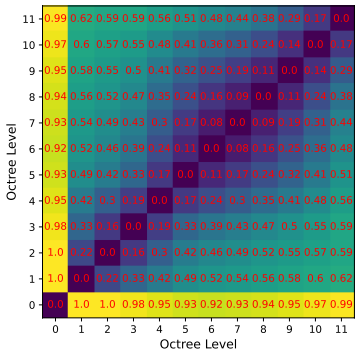

Fig. 3. The heat map of the Jensen-Shannon (JS) divergence of the occupancy symbol distributions among different levels of an octree. The distributions are computed from our training set on KITTI [8]. JS divergence is to measure the difference between two probability distributions. The greater JS divergence means the larger difference between the two distributions.

3.5 Two-step Reconstruction Strategy

The quantization error of our point cloud compression framework comes from the octree construction and depends on the octree level L. Each leaf octant at level L represents a subspace in the large 3D space. Because of the quantization, we can only recover a single point from each leaf octant by taking the center coordinate of its corresponding subspace. VoxelContext [24] introduces a refinement module to reduce the quantization error by adding predicted offsets to these coordinates in the decoding stage. In contrast, we approach this problem by first retrieving missing points with the help of our trained entropy model at level $L+1$ and then further refining the aggregated points. With this strategy, we can reconstruct point clouds with a precision close to level $L+1$ while keeping the bitrate at level L.

As illustrated in Fig. 4, considering a reconstructed octree with L levels, we denote a set of octants at level L as $\mathcal{N}_L$ storing the occupancy symbols. The symbols from $\mathcal{N}_L$ consist of the occupancy information that forms the children's octants at level $L+1$. We first apply our entropy model at level $L+1$ on each children's octant to estimate its occupancy symbol probabilities. We take the predicted occupancy symbols $\mathcal{S}_{L+1}$ with top-1 accuracy to form a new set of octants $\mathcal{N}_{L+1}$ located at the predicted level $L+1$. We denote the corresponding 3D coordinate of $\mathcal{N}_{L+1}$ as $\mathbf{x}^p \in \mathbb{R}^{n\times 3}$. In the second step, we apply the refinement module at level $L+1$ [24] that takes neighbor context V_i and octant information c_i as the input to predict offsets for the coordinates. Our refinement module is predefined to output offsets for every child's octant of $\mathcal{N}_L$, where the offsets are denoted as $\mathbf{x}^o$. Then we obtain the final output of the coordinate by

$$\mathbf{x}^r = \mathbf{x}^p + \mathbf{I}(\mathcal{S}_{L+1}, \mathbf{x}^o), \tag{10}$$

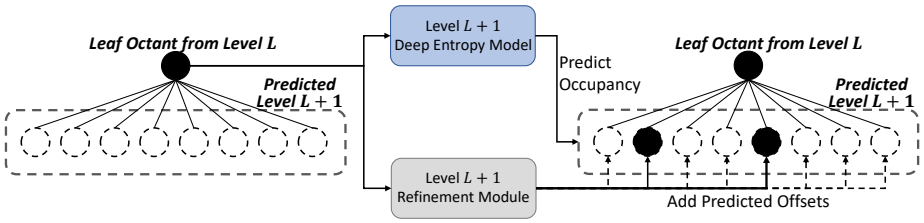

Fig. 4. The illustration of our two-step refinement strategy. The leaf octant of the decoded octree locates at the depth level L, and its occupancies are unknown. We first apply our entropy model on the octant to predict the occupancy symbols and calculate their coordinates. Then the coordinates are added with the corresponding offsets predicted from the refinement module.

where $\mathbf{I}(a, b)$ denote the operation of indexing b according to a. To be specifically, $\mathbf{I}(\mathcal{S}_{L+1}, \mathbf{x}^o)$ means that for those octants whose occupancy codes are zeros, we ignore their offsets. We only add the offsets to the octants whose occupancy codes are ones. $\mathbf{x}^r \in \mathbb{R}^{n\times 3}$ is the resulting refined 3D coordinates.

To obtain the ground-truth coordinate of N_{L+1} when training the entropy model, We build octrees with $L+1$ levels and calculate the coordinates $\mathbf{x}^{gt}$ from $\mathcal{N}_{L+1}$ as the ground truth. Note that the sibling context at level $L+1$ is not available during the encoding and decoding time, and we set V_i^{sib} as zeros voxel block to be the input of our entropy model. We train the refinement module $\mathcal{R}$ by minimizing the Chamfer Distance [5] of the $\mathbf{x}^{gt}$ and $\mathbf{x}^r$:

$$\mathcal{L}_{CD} = \max\left\{\mathrm{CD}(\mathbf{x}^r, \mathbf{x}^{gt}), \mathrm{CD}(\mathbf{x}^{gt}, \mathbf{x}^r)\right\}, \tag{11}$$

where $\mathcal{L}_{CD}$ is the objective function for the refinement module.

4 Experiments

4.1 Experimental Setup

Dataset. We evaluate the compression performance and reconstruction quality on two real-world datasets with different densities: KITTI Odometry [8] and nuScenes [3], which are collected by a Velodyne HDL-64 sensor and a Velodyne HDL-32 sensor. We also use the KITTI detection dataset [8] for evaluating the 3D object detection as a downstream task using the reconstructed point cloud.

To ensure the diversity of the training and testing scene, we randomly sample 6000 frames from sequences 00 to 10 in the KITTI Odometry dataset for training and 550 frames from sequences 11 to 21 for testing. For nuScenes, we randomly sample 1200 frames from each of the first five data batches (total 6000) for training, and randomly sampled 100 frames from each of the last five data batches (total 500) for testing. Note that we compare our compression performance and reconstruction quality with the original raw point cloud without any pre-processing in the following sections.

Baselines. To evaluate the performance of our framework, we compare our method with three state-of-the-art baselines: hand-drafted Octree-based point cloud compression method MPEG G-PCC [26], KD-tree based method Google Draco [9], and also a learning-based method VoxelContext [24]. VoxelContext has been faithfully self-implemented as the official code has not been released.

Implementation details. We construct the octree with a maximum level of 13 on both KITTI Odometry and nuScenes datasets. Note that we construct level 13 of the octree only for generating the ground-truth labels for our training. In our experiment section, we evaluate the performance of our framework by truncating octree levels ranging from 9 to 12 on both datasets to vary the compression bitrates. The corresponding spatial quantization error ranges from 14.94 cm to 1.87 cm for KITTI Odometry and 18.29 cm to 2.29 cm for nuScenes.

We implement our framework in PyTorch, and the models are trained on NVIDIA 3090 GPU. The total number of parameters in our deep entropy model is 1.77M. We use Adam optimizer [16] with a learning rate of 1e-4 to train our whole framework. The training epoch for the multi-level entropy model and the refinement module are 20 and 2, respectively. To be memory efficient, we do not backpropagate the gradients of an entropy model from the lower to the upper level during the training of multi-level entropy models. We freeze the weights of the entropy models while training the refinement module.

4.2 Evaluation Metrics

We evaluate our framework from two aspects, compression performance and reconstruction quality. Bits per point (BPP) is the most commonly used metric for evaluating compression performance. Since we only consider the geometric compression of a point cloud in this section, the size of the original point cloud data is calculated by $96 \times N$, where N is the number of points in the raw point cloud, and 96 is the size of the coordinates: x, y and z, where each coordinate is represented in a 32-bit floating-point. BPP is defined as $BPP = |bit|/N$, where the $|bit|$ is the bitstream length.

We utilize four metrics to evaluate the reconstruction quality of the point cloud: point-to-point and point-to-plane PSNR [19,30], F1 score, and maximum Chamfer Distance [5,30]. Same as the evaluation metrics defined in [2], for original input point cloud P and reconstruction point cloud $\hat{P}$, we use $\text{PSNR} = 10 \log_{10} \frac{3r^2}{\max\{\text{MSE}(P,\hat{P}),\text{MSE}(\hat{P},P)\}}$, $\text{F1} = \frac{TP}{TP+FP+FN}$, $\text{CD}_{max} = \max\{\text{CD}(P,\hat{P}), \text{CD}(\hat{P},P)\}$, where the peak constant value of r is 59.70 m, and the distance threshold of the F1 score is 2 cm.

4.3 Results

Quantitative results. To quantitatively evaluate our method, we report BPP versus four reconstruction quality metrics. As illustrated in Fig. 5, our method saves more bitrate and achieves higher reconstruction quality compared with all

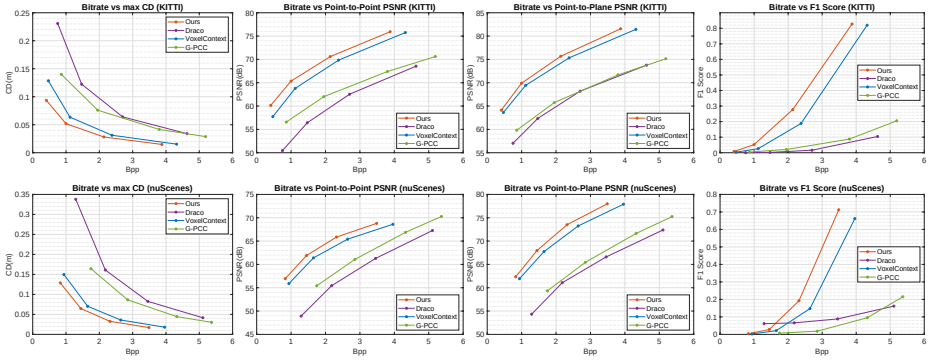

Fig. 5. The quantitative results of our method on the KITTI Odometry dataset (first row) and nuScenes dataset (second raw). From left to right: maximum Chamfer distance (↓), point-to-point PSNR (↑), point-to-plane PSNR (↑), and F1 score (↑). Note that both our method and VoxelContext are applied to the refinement module when calculating the evaluation metrics in this figure.

Dataset	Method	BPP↓				
		Level 8	Level 9	Level 10	Level 11	Level 12
KITTI	VoxelContext [24]	0.173	0.466	1.123	2.380	4.371
	Ours	**0.149**	**0.409**	**0.999**	**2.137**	**3.878**
nuScenes	VoxelContext [24]	0.474	0.937	1.650	2.642	3.960
	Ours	**0.424**	**0.829**	**1.445**	**2.319**	**3.487**

Table 1. The quantitative results when compared to VoxelContext without any refinement module. The first row is the result of the KITTI Odometry dataset. The second row is the result of the nuScenes dataset. The reconstructed point clouds of the two methods are the same at each level.

the state-of-the-art methods on both 64 channel KITTI Odometry dataset and 32 channel nuScenes dataset.

Since we use the same octree construction strategy as VoxelContext, we have the same reconstruction quality at the same octree level without applying any refinement module. As illustrated in Table 1, our method saves 11%-16% bitrate on the KITTI Odometry dataset and 12%-14% on the nuScenes dataset compared to VoxelContext. Our method reaches higher reconstruction quality than VoxelContext at the same octree level despite using the same refinement module, clearly demonstrating the benefit of our newly proposed two-step refinement strategy.

Qualitative results. We evaluate our method qualitatively by comparing the reconstruction error with state-of-the-art methods on two datasets. As illustrated in Fig. 6, the error between the original point cloud and our reconstructed

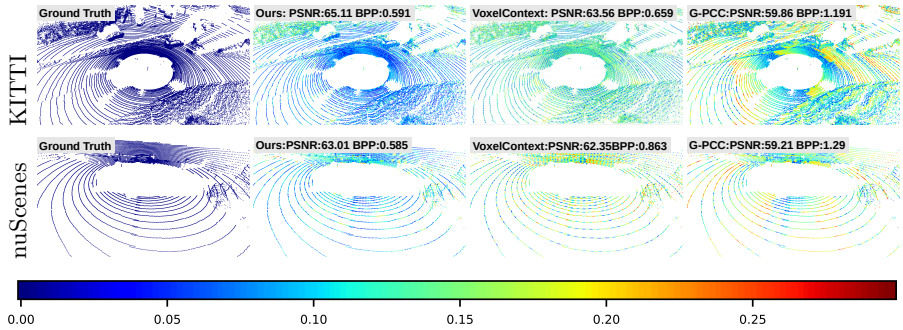

Fig. 6. The qualitative results of our method compared with VoxelContext and G-PCC on the KITTI Odometry dataset (first row) and the nuScenes dataset (second row). The color bar indicates the error in meters between the original and reconstructed point cloud. Our reconstructed point cloud has a lower error at a lower bitrate compared to all baselines.

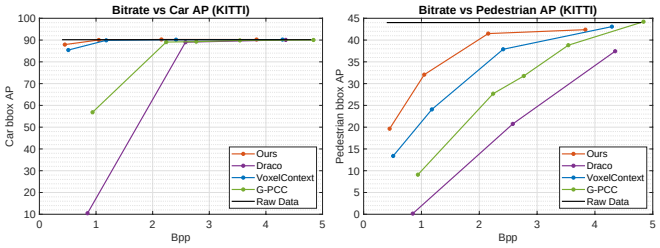

Fig. 7. The qualitative results of downstream tasks on the KITTI detection dataset.

point cloud is much smaller at an even lower bitrate as compared to all three baseline methods.

Downstream tasks. The performance on downstream tasks of the reconstructed point cloud is also a crucial evaluation metric for point cloud compression. We evaluate the 3D object detection method proposed in [17] on reconstructed point cloud data of different compression methods. We report the bitrate versus average precision (AP) at 0.5 IOU as a threshold for pedestrians and 0.7 IOU for cars. As illustrated in Fig. 7, our method outperforms all three baselines in different categories.

4.4 Ablation Study

We perform an ablation study on our deep entropy model to demonstrate the effectiveness of different contextual features. We ablate over the contextual features by training the model without exploiting features from siblings' children, ancestors, or surfaces. The models for ablation studies are trained on the KITTI Odometry dataset with the same training configurations. We evaluate the models at a level ranging from 8 to 12. From the quantitative results shown in Table 2,

we observe that the bitrate increases when not incorporating any dependencies into the entropy model. Moreover, the result also demonstrates that the sibling features are crucial to the compression performance of our entropy model.

Method	BPP↓				
	Level 8	Level 9	Level 10	Level 11	Level 12
Ours w/o Sibling	0.161 (+8.1%)	0.439 (+7.3%)	1.071 (+7.2%)	2.288 (+7.1%)	4.133 (+6.6%)
Ours w/o Ancestor	0.150 (+0.6%)	0.414 (+1.2%)	1.016 (+1.7%)	2.181 (+2.1%)	3.956 (+2.0%)
Ours w/o Surface	0.151 (+1.3%)	0.415 (+1.5%)	1.019 (+2.0%)	2.180 (+2.0%)	3.953 (+1.9%)
Ours	**0.149**	**0.409**	**0.999**	**2.137**	**3.878**

Table 2. Ablation study of our entropy model on KITTI [8] without using context from siblings' children, ancestors, or surface priors.

5 Conclusion

We have presented a novel octree-based point cloud compression method. Our method uses the hierarchical contexts of octree structures and surface priors to reduce the information redundancies in the bitstream. On the decoding side, our method utilizes a two-step heuristic strategy for reconstructing a point cloud with better quality. We have evaluated our method against three state-of-the-art baselines on two datasets. The result demonstrates that our method outperforms previous methods on compression performance, reconstruction quality, and downstream tasks. We hope our work can inspire researchers to further reduce the compression rate and improve the reconstruction quality in future work.

References

1. Ahn, J.K., Lee, K.Y., Sim, J.Y., Kim, C.S.: Large-scale 3d point cloud compression using adaptive radial distance prediction in hybrid coordinate domains. JSTSP **9**(3) (2014)
2. Biswas, S., Liu, J., Wong, K., Wang, S., Urtasun, R.: Muscle: Multi sweep compression of lidar using deep entropy models. In: NIPS (2020)
3. Caesar, H., Bankiti, V., Lang, A.H., Vora, S., Liong, V.E., Xu, Q., Krishnan, A., Pan, Y., Baldan, G., Beijbom, O.: nuscenes: A multimodal dataset for autonomous driving. In: CVPR (2020)
4. Devillers, O., Gandoin, P.M.: Geometric compression for interactive transmission. In: VIS 2000 (Cat. No. 00CH37145) (2000)
5. Fan, H., Su, H., Guibas, L.J.: A point set generation network for 3d object reconstruction from a single image. In: CVPR (2017)
6. Fu, C., Li, G., Song, R., Gao, W., Liu, S.: Octattention: Octree-based large-scale contexts model for point cloud compression. In: AAAI (2022)
7. Garcia, D.C., de Queiroz, R.L.: Intra-frame context-based octree coding for point-cloud geometry. In: ICIP (2018)
8. Geiger, A., Lenz, P., Urtasun, R.: Are we ready for autonomous driving? the kitti vision benchmark suite. In: CVPR (2012)
9. Google: Draco. In: https://github.com/google/draco (2017)
10. Graziosi, D., Nakagami, O., Kuma, S., Zaghetto, A., Suzuki, T., Tabatabai, A.: An overview of ongoing point cloud compression standardization activities: video-based (v-pcc) and geometry-based (g-pcc). APSIPA Transactions on Signal and Information Processing **9** (2020)
11. Huang, L., Wang, S., Wong, K., Liu, J., Urtasun, R.: Octsqueeze: Octree-structured entropy model for lidar compression. In: CVPR (2020)
12. Huang, T., Liu, Y.: 3d point cloud geometry compression on deep learning. In: MM (2019)
13. Huang, Y., Peng, J., Kuo, C.C.J., Gopi, M.: A generic scheme for progressive point cloud coding. TVCG **14**(2) (2008)
14. Jackins, C.L., Tanimoto, S.L.: Oct-trees and their use in representing three-dimensional objects. CGIP **14**(3) (1980)
15. Kammerl, J., Blodow, N., Rusu, R.B., Gedikli, S., Beetz, M., Steinbach, E.: Real-time compression of point cloud streams. In: ICRA (2012)
16. Kingma, D.P., Ba, J.: Adam: A method for stochastic optimization. In: ICLR (2015)
17. Lang, A.H., Vora, S., Caesar, H., Zhou, L., Yang, J., Beijbom, O.: Pointpillars: Fast encoders for object detection from point clouds. In: CVPR (2019)
18. Meagher, D.: Geometric modeling using octree encoding. CGIP **19**(2) (1982)
19. Mekuria, R., Laserre, S., Tulvan, C.: Performance assessment of point cloud compression. In: VCIP (2017)
20. Nenci, F., Spinello, L., Stachniss, C.: Effective compression of range data streams for remote robot operations using h. 264. In: IROS (2014)
21. Nguyen, D.T., Quach, M., Valenzise, G., Duhamel, P.: Multiscale deep context modeling for lossless point cloud geometry compression. In: ICMEW (2021)
22. Qi, C., Yi, L., Su, H., Guibas, L.J.: Pointnet++: Deep hierarchical feature learning on point sets in a metric space. In: NIPS (2017)
23. Qi, C.R., Su, H., Mo, K., Guibas, L.J.: Pointnet: Deep learning on point sets for 3d classification and segmentation. In: CVPR (2017)

24. Que, Z., Lu, G., Xu, D.: Voxelcontext-net: An octree based framework for point cloud compression. In: CVPR (2021)
25. Schnabel, R., Klein, R.: Octree-based point-cloud compression. In: PBG@ SIGGRAPH (2006)
26. Schwarz, S., Preda, M., Baroncini, V., Budagavi, M., Cesar, P., Chou, P.A., Cohen, R.A., Krivokuća, M., Lasserre, S., Li, Z., et al.: Emerging standards for point cloud compression. JETCAS **9**(1) (2018)
27. Sun, X., Ma, H., Sun, Y., Liu, M.: A novel point cloud compression algorithm based on clustering. RA-L **4**(2) (2019)
28. Sun, X., Wang, S., Liu, M.: A novel coding architecture for multi-line lidar point clouds based on clustering and convolutional lstm network. T-ITS (2020)
29. Theis, L., Shi, W., Cunningham, A., Huszár, F.: Lossy image compression with compressive autoencoders. In: 5th International Conference on Learning Representations, ICLR (2017)
30. Tian, D., Ochimizu, H., Feng, C., Cohen, R., Vetro, A.: Geometric distortion metrics for point cloud compression. In: ICIP (2017)
31. Tu, C., Takeuchi, E., Carballo, A., Takeda, K.: Point cloud compression for 3d lidar sensor using recurrent neural network with residual blocks. In: ICRA (2019)
32. Wang, J., Zhu, H., Ma, Z., Chen, T., Liu, H., Shen, Q.: Learned point cloud geometry compression. CoRR **abs/1909.12037** (2019)
33. Wang, S., Jiao, J., Cai, P., Liu, M.: R-PCC: A baseline for range image-based point cloud compression. CoRR **abs/2109.07717** (2021)
34. Wiesmann, L., Milioto, A., Chen, X., Stachniss, C., Behley, J.: Deep compression for dense point cloud maps. RA-L **6**(2) (2021)
35. Yan, W., Shao, Y., Liu, S., Li, T.H., Li, Z., Li, G.: Deep autoencoder-based lossy geometry compression for point clouds. CoRR **abs/1905.03691** (2019)