VoxelContext-Net: An Octree based Framework for Point Cloud Compression

Zizheng Que
Beihang University
quezizheng@buaa.edu.cn

Guo Lu*
Beijing Institute of Technology
guo.lu@bit.edu.cn

Dong Xu
University of Sydney
dong.xu@sydney.edu.au

Abstract

In this paper, we propose a two-stage deep learning framework called VoxelContext-Net for both static and dynamic point cloud compression. Taking advantages of both octree based methods and voxel based schemes, our approach employs the voxel context to compress the octree structured data. Specifically, we first extract the local voxel representation that encodes the spatial neighbouring context information for each node in the constructed octree. Then, in the entropy coding stage, we propose a voxel context based deep entropy model to compress the symbols of non-leaf nodes in a lossless way. Furthermore, for dynamic point cloud compression, we additionally introduce the local voxel representations from the temporal neighbouring point clouds to exploit temporal dependency. More importantly, to alleviate the distortion from the octree construction procedure, we propose a voxel context based 3D coordinate refinement method to produce more accurate reconstructed point cloud at the decoder side, which is applicable to both static and dynamic point cloud compression. The comprehensive experiments on both static and dynamic point cloud benchmark datasets(e.g., ScanNet and Semantic KITTI) clearly demonstrate the effectiveness of our newly proposed method VoxelContext-Net for 3D point cloud geometry compression.

1. Introduction

Due to the rapid population of 3D sensors such as LiDAR, there is an increasing research interest to compress tremendous amount of 3D point cloud data for a broad range of applications (*e.g.,* autonomous driving). When compared with image and video compression [35, 32, 4, 37, 31, 19], it is a more challenging task to compress a set of orderless 3D points from point clouds.

Recently, several deep learning methods were developed for point cloud compression. For example, Wang *et al.* [36] transformed the point cloud data to the voxel representation in order to capture the spatial dependency, and then they employed the existing image compression method [2] for point cloud compression. Other recent works [39] and [16] directly compressed the raw point cloud data by using the existing backbone networks (*e.g.,* PointNet/PointNet++ [25, 26]) for feature encoding. These voxel-based or point-based methods can take advantage of the existing image compression or point cloud analysis techniques. However, the voxel-based methods ignore the sparsity characteristic of point clouds and thus suffer from the relatively high computational complexity [36] while the point based methods are inefficient when processing large point cloud data [39]. In [15], Huang *et al.* used the octree to organize the point cloud data and proposed an entropy model to exploit the dependency between multiple parent nodes and each child node (see Figure 1(a)). Although their approach inherits the benefits for efficiently processing octree structured point cloud data, the strong dependency among the neighboring octree nodes at the same depth level is unfortunately ignored in their octree-based entropy model [15]. Furthermore, the distortions are also introduced after converting the raw point cloud to the octree structured data, which further degrades the compression performance. Besides, their approach is only designed for static point cloud compression, which may limit the performance for dynamic point cloud compression.

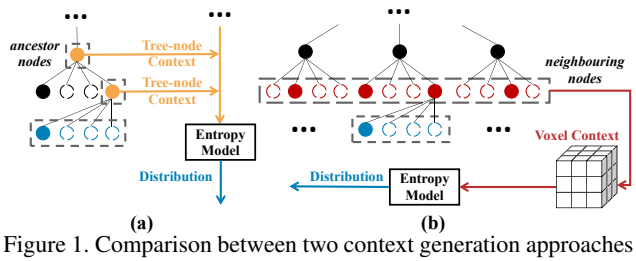

Figure 1. Comparison between two context generation approaches for learning the entropy model based on the octree structured data. (a) Tree-node context used in [15]. (b) Local voxel context employed in our approach.

To address these issues, we propose a new learning based point cloud compression method VoxelContext-Net by exploiting the voxel context in the octree based framework. Our approach takes advantage of the efficient data organization ability from the octree based methods and the spatial modeling capability from the voxel based methods, which

*Guo Lu is the corresponding author.

can be applied to both static and dynamic point cloud geometry compression. Specifically, the input point cloud is first organized by using the octree structure, where the symbol of each non-leaf node represents the occupancy status of its eight children. In the entropy coding stage, we propose a new learning based entropy model to compress these symbols in a lossless way. To effectively produce context information for the entropy model, we exploit the local binary voxel representation for each node, where the entries of our voxel representation indicate the existence of neighbouring nodes at the same depth level (see Figure 1(b)). Furthermore, to reduce temporal redundancy for dynamic point cloud compression, we additionally include the co-located voxel representations from the previous and the subsequent point clouds to generate richer context information. In the reconstruction stage, we further propose a coordinate refinement method based on the local voxel representations at the decoder side to produce more accurate 3D coordinate for each leaf node in both static and dynamic point clouds.

We evaluate the performance of our newly proposed method on the large-scale 3D static and dynamic point cloud datasets (*e.g.,* ScanNet [7] and Semantic KITTI [11, 3]). The comprehensive experiments demonstrate that our method outperforms both hand-crafted point cloud compression methods and the learning-based point cloud compression methods.

The contributions of our work are highlighted as follows:

- By taking the advantage of both voxel based methods and octree based schemes, we introduce local voxel context in the deep entropy model for better compression of octree structured data. Our approach can be applied to both static and dynamic point cloud compression.
- We develop a voxel context based coordinate refinement module to produce accurate coordinates of leaf nodes at the decoder side.
- Our simple and effective approach achieves the state-of-the-art compression performance on several large-scale datasets for both static and dynamic point cloud geometry compression.

2. Related Work

2.1. Traditional Point Cloud Compression Methods

In the past several years, a few point cloud compression methods [27, 12, 29, 17, 28, 8, 9] have been proposed and most of them are based on the tree representations. For example, the MPEG group developed a standard point cloud compression method [29, 13, 13] G-PCC (geometry based point cloud compression) for static point clouds, which includes an octree-structure based method for point cloud compression. However, they are all based on hand-crafted techniques and thus cannot be optimized in an end-to-end fashion by using large-scale data.

In addition, although some learning based image and video compression approaches [1, 2, 23, 20, 38, 21, 14] have been proposed, it is still a non-trial task to employ the standard CNN operations for compressing point clouds consisting of a sparse set of orderless 3D points.

2.2. Deep Learning for Point Cloud Compression

Taking the point cloud data as the input, Yan *et al.* [39] built an auto-encoder network by using PointNet, in which the latent representation is quantized and further compressed by using an entropy coding model. These point based methods [39, 16] may suffer from the huge memory usage issue and high computational costs. Wang *et al.* [36] extended the existing image compression method [1] for voxelized point cloud compression. Unfortunately, their approach ignores the sparsity characteristic of point clouds and thus the computational complexity is relatively high when compared with the octree based methods.

Recently, an octree-based method OctSqueeze [15] was proposed. While the OctSqueeze method [15] avoids the issues related to high memory usage and slow encoding/decoding speed, their approach still suffer from the following drawbacks. First, they only exploited context information from its *ancestor nodes* (as shown in Figure 1(a)) to predict the probability model in the entropy model, which ignores strong prior information between *spatial neighbouring nodes* at the same depth level. In addition, their work does not consider the distortion introduced in the octree construction procedure, and their method is only designed for static point cloud compression. Although there is a concurrent work [5] for dynamic point cloud compression, it follows the existing framework from [15] and thus suffers from similar limitations.

In contrast to these works [39, 16, 36, 15, 5, 34], we propose to exploit context information between neighbouring nodes by using local voxel representation in our deep entropy model and our work also refines the 3D coordinate at the decoder side in order to achieve better reconstruction results. Besides, we further extend our approach by additionally exploit the local context representation from neighbour frames for dynamic point cloud compression.

3. Methodology

3.1. Overview

The overall architecture of the proposed point cloud compression method is shown in Fig 2. In this section, we first take the static point cloud compression as an example to illustrate our proposed method and then introduce how to

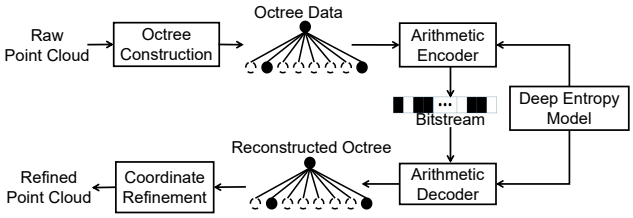

Figure 2. The overall architecture of our proposed static point cloud compression approach. The symbols of the non-leaf nodes from the octree are losslessly compressed by using the proposed deep entropy model, while the coordinate refinement module is used to predict more accurate coordinates at the decoder side.

extend the proposed method for dynamic point cloud compression.

Specifically, in the first stage, we organize the input static point cloud by using the octree structure, in which our approach aims to encode these symbols of non-leaf nodes in a lossless way. To improve the compression performance, we propose a voxel context based deep entropy model to accurately predict the probability distributions of these symbols. Furthermore, to compensate the distortion in the octree construction procedure, a local voxel context based coordinate refinement module is proposed to produce more accurate reconstructed point cloud at the decoder side.

3.2. Octree Construction

In Figure 3, we provide a toy example to illustrate the octree construction procedure. Specifically, an octree can be constructed from any 3D point cloud by first partitioning the 3D space into 8 cubes with the same size, and then recursively partitioning each non-empty cube in the same way until the maximum depth level is reached. The 3D coordinate of each node represents the cube center. For each non-leaf node, a 8-bit symbol is used to represent the occupancy status of its eight children, with each bit corresponding to one specific child.

In the octree construction procedure, the quality of the reconstructed point cloud depends on the maximum depth level in the octree structure. Therefore, the coordinate of the current leaf node (*i.e.,* the cube center) is not always consistent with the original 3D coordinate of the corresponding point in the raw point cloud. For example, the coordinate of one input point r_i is $(0.6, 0.7, 0.7)$, while the coordinate for the corresponding leaf node n_i is quantized to $(0.625, 0.625, 0.625)$, thus inevitable distortion is introduced in the octree construction procedure. In this work, we will compress the symbols of octree nodes losslessly and recover the accurate decoded coordinates at the decoder side.

3.3. Local Voxel Context in Octree

In the octree structure, the parent node will generate 8 children nodes, which is equivalent to bisecting the 3D space along x-axis, y-axis and z-axis. Thus, the partition

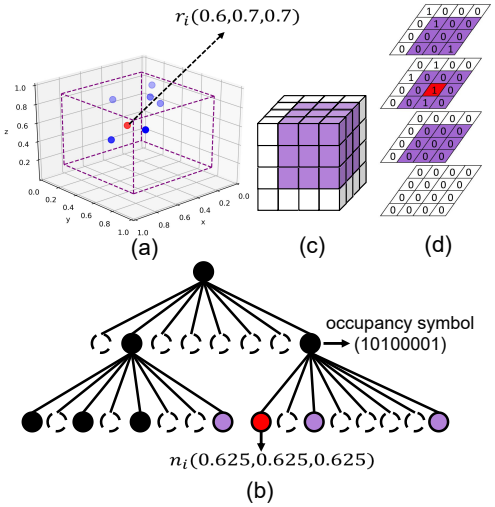

Figure 3. A toy example for constructing the octree and extracting the local voxel context representation. (a) Raw input point cloud. (b) The corresponding octree. (c) The voxel representation for the input point cloud at the depth level of 2. (d) The detailed binary voxel representation.

of the original space at the kth depth level in the octree is equivalent to dividing the corresponding 3D space 2^k times along x-axis, y-axis and z-axis, respectively. Then we produce a binary voxel representation with the shape of $2^k * 2^k * 2^k$ based on the existence of points in each cube. Here we assume the corresponding local voxel representations centered at the node n_i is $\boldsymbol{V_i} \in \mathcal{R}^{M \times M \times M}$, where M represents the size of the local voxel representation. $\boldsymbol{V_i}$ will be used in our approach as strong prior information to improve the compression performance.

In Figure 3(c), the purple region represents the local voxel context for the current node n_i and the detailed binary values for the local voxel representation are depicted in Figure 3(d). In Figure 3(a), we also provide the 3D coordinate range of the corresponding local region in the 3D space(see the purple dash line). It is noted that the local voxel context $\boldsymbol{V_i}$ of the current node n_i represents the distribution information of neighbouring nodes at the same depth level. In contrast, the previous method [15] only exploit the information from its ancestor nodes without considering strong spatial neighbouring prior information(see Figure 1(a)).

3.4. Our Deep Entropy Model

3.4.1 Formulation

Let $\boldsymbol{s} = [\boldsymbol{s_1}, ..., \boldsymbol{s_i}, ...]$ denote a sequence of 8-bit occupancy symbols from all non-leaf octree nodes where $\boldsymbol{s_i}$ represents the symbol for node n_i in the octree. For example, when $\boldsymbol{s_i} = [0, 0, 0, 1, 0, 0, 0, 1]$, it means node n_i has two children and the corresponding indexes of the two children are 4 and 8, respectively.

According to the information theory [30], when com-

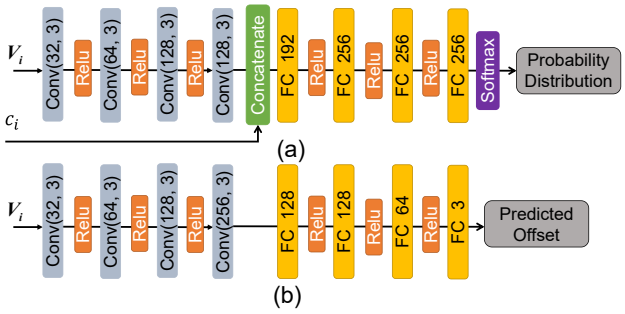

Figure 4. Network architecture of (a) our proposed deep entropy model and (b) our coordinate refinement module. 'Conv(128,3)' represents the 3D convolution operation when the number of channels is 128 and the kernel size is $3 \times 3 \times 3$.

pressing the occupancy information, the lower bound of bitrates is the Shannon entropy. However, the actual distribution P is unknown in the practical applications. Therefore, we use the deep neural network to estimate the probability distribution which can be employed to approximate the actual distribution P. Based on the learned deep entropy model, we can compress these occupancy symbols from the octree in a lossless manner. Specifically, the objective of our learning based entropy model is to minimize the cross-entropy loss $E_{\boldsymbol{s}\sim P}[-\log Q_s(\boldsymbol{s})]$, where $Q_s(\boldsymbol{s})$ is the estimated probability of $\boldsymbol{s}$.

It is noted that the children's probability distribution $q_s(\boldsymbol{s_i})$ at the current node n_i may rely on the previously decoded nodes as well as the neighbouring nodes at the current depth level, so it is very difficult to model this complex relationship. In this work, we assume the occupancy symbol $\boldsymbol{s_i}$ for the current node n_i only depends on the node's local voxel context $\boldsymbol{V_i}$ and the node feature $\boldsymbol{c_i}$ that includes the node's 3D coordinate and the depth level of the node. Since $\boldsymbol{V_i}$ represents local context information at the same depth level, it is reasonable to infer the node n_i's children distribution (*i.e.,* $\boldsymbol{s_i}$) based on $\boldsymbol{V_i}$. Then we can simplify the complex dependence relationship for each original occupancy symbol $\boldsymbol{s}$ and further factorize $Q_s(\boldsymbol{s})$ into the following way,

$$Q_s(\boldsymbol{s}) = \prod_i q_s(\boldsymbol{s_i}|\boldsymbol{V_i}, \boldsymbol{c_i}) \tag{1}$$

where $q_s(\boldsymbol{s_i}|\boldsymbol{V_i}, \boldsymbol{c_i})$ is the predicted probability of $\boldsymbol{s_i}$ when the local voxel context $\boldsymbol{V_i}$ and the node feature $\boldsymbol{c_i}$ are given at node n_i. Here, we use the same probability distribution model for each node.

3.4.2 Network Architecture

In this work, we use deep neural networks to parameterize the entropy model in Eq. (1). The whole network architecture is shown in Figure 4. Specifically, for the current node n_i, we first extract its local voxel context $\boldsymbol{V_i}$ based on the method described in Section 3.3. Then $\boldsymbol{V_i}$ is fed to a multi-layer convolutional neural networks (CNN) and the corresponding output is $\boldsymbol{f_i}$. In this procedure, we adopt the popular and off-shelf CNN structure to effectively exploit context information in the 3D space, which has not been exploited by the existing octree based point cloud compression methods. Then we concatenate the context feature $\boldsymbol{f_i}$ and the node feature $\boldsymbol{c_i}$. After that, a multi-layers perception (MLPs) is adopted to generate a 256-dimensional hidden feature, which fuses both local voxel contextual information and node information. Finally, a softmax layer is used to produce the probabilities $q_s(\boldsymbol{s_i}|\boldsymbol{V_i}, \boldsymbol{c_i})$ of the 8-bit occupancy symbol for each given node n_i. Our approach takes advantage of both octree based methods and voxel based methods and achieves better point cloud compression performance.

3.5. Our Coordinate Refinement Method

To reduce the distortion in the octree construction procedure, we propose a local voxel context based coordinate refinement method to produce more accurate reconstructed point cloud at the decoder side. A key intuition behind our coordinate refinement method is that we can better predict the coordinate of the current node by exploiting local context information of its neighbouring voxels.

Given the decoded octree, for each leaf node n_i, the goal of our coordinate refinement method is to predict the refined output coordinate (x_i^r, y_i^r, z_i^r) in the following way,

$$(x_i^r, y_i^r, z_i^r) = (x_i^d, y_i^d, z_i^d) + R(\boldsymbol{V_i}) \tag{2}$$

where $\boldsymbol{V_i}$ is the local voxel context for the leaf node n_i and $R(\cdot)$ is a learnable function for coordinate refinement. The decoded coordinate (x_i^d, y_i^d, z_i^d) represents the coordinate of node n_i after octree decoding.

As shown in Figure 4(b), the network architecture of the proposed coordinate refinement method is similar to the deep entropy model. Specifically, for a decoded octree, we firstly transform the whole octree into a global binary voxel representation, from which we can readily produce the local voxel context for each leaf node. Given the binary voxel context $\boldsymbol{V_i}$ for node n_i, we firstly extract context information in the 3D space by using multi-layer CNNs, and then predict the offset by using a set of fully connected layers. We then calculate the decoded coordinate of the leaf node n_i and the final reconstructed coordinate of n_i is the sum of its decoded coordinate and the predicted offset (see Eq. (2)).

3.6. Training Strategy

In the proposed method, we separately train our local voxel context based deep entropy model and our coordinate refinement module. For the deep entropy model, based on the predicted distribution $q_s(\boldsymbol{s_i}|\boldsymbol{V_i}, \boldsymbol{c_i})$ at each non-leaf node, we use the following loss function:

$$L_{entropy} = -\sum_i \log q_s(\boldsymbol{s_i}|\boldsymbol{V_i}, \boldsymbol{c_i}) \tag{3}$$

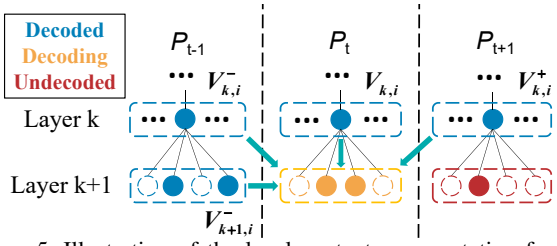

Figure 5. Illustration of the local context representation for dynamic point clouds. Four voxel representations from the decoded layers(see the blue boxes) are used as local voxel context to estimate the probability distribution of each node in the current decoding layer(see the yellow box).

where $q_s(\boldsymbol{s_i}|\boldsymbol{V_i}, \boldsymbol{c_i})$ is defined after Eq. (1).

For the coordinate refinement module, we aim to generate the precise coordinate of each point. Towards this goal, a MSE based loss is used as the distortion in the training procedure:

$$L_{mse} = \sum_i \|(x_i^r, y_i^r, z_i^r) - (x_i^g, y_i^g, z_i^g)\|_2^2 \qquad (4)$$

where (x_i^g, y_i^g, z_i^g) is the ground-truth coordinate of node n_i before octree construction.

3.7. Dynamic Point Cloud Compression

The proposed point cloud compression framework is very general and can be readily extended for dynamic point cloud compression. Considering that each dynamic point cloud consists of a sequence of redundant point clouds that are captured at different time, it is necessary to exploit temporal information.

Specifically, given the point cloud $\mathcal{P}_t$ at time step t and its neighbouring point clouds $\mathcal{P}_{t-1}$ and $\mathcal{P}_{t+1}$ at time step $t-1$ and $t+1$, we first align these point clouds to the same coordinate system based on their pose information of the sensor. It should be mentioned that we decode *all* point clouds from a sequence in a depth by depth fashion. Thus, when we decode the node n_i at the kth depth level for the current point cloud $\mathcal{P}_t$, the voxel representation at the kth depth level from other two point clouds $\mathcal{P}_{t-1}$ and $\mathcal{P}_{t+1}$ is available. Meanwhile, the voxel representation at the $(k+1)$th depth level from the point cloud $\mathcal{P}_{t-1}$ is also available.

Here, as shown in Fig. 5, we assume the local voxel representation for node n_i at the kth depth level in $\mathcal{P}_t$ is $\boldsymbol{V_{k,i}}$ and the corresponding co-located local voxel representations at the kth depth level in $\mathcal{P}_{t-1}$ and $\mathcal{P}_{t+1}$ are $\boldsymbol{V^-_{k,i}}$ and $\boldsymbol{V^+_{k,i}}$, respectively. Furthermore, the voxel representation at the $(k+1)$th depth level in the previous point cloud is denoted as $\boldsymbol{V^-_{k+1,i}}$. To exploit temporal information in the entropy model, we adopt a simple fusion strategy and use the similar network architecture as shown in Figure 4 to extract the features from each local voxel representation. Then we concatenate the features extracted from four different voxel representations (*i.e.,* $\boldsymbol{V_{k,i}}$, $\boldsymbol{V^-_{k,i}}$, $\boldsymbol{V^+_{k,i}}$ and $\boldsymbol{V^-_{k+1,i}}$) and feed the aggregated feature to the *Softmax* layer to produce the final probability distribution. In our implementation, we set the size of $\boldsymbol{V_{k,i}}$, $\boldsymbol{V^-_{k,i}}$ and $\boldsymbol{V^+_{k,i}}$ as $9 \times 9 \times 9$, and set the size of $\boldsymbol{V^-_{k+1,i}}$ as $10 \times 10 \times 10$. Finally, the coordinates from the reconstructed octree are refined by using the voxel context based method discussed in Section 3.5. It is noted that we only use the voxel representation $\boldsymbol{V_{k,i}}$ as context information for coordinate refinement as it is sufficient to use this voxel representation to achieve promising results.

4. Experiments

4.1. Datasets

ScanNet: ScanNet [7] is a large-scale dataset with a few dense point cloud sequences from the indoor scenario. It consists of 1,513 scans. In our experiment, 50,000 points are sampled from each scan.

Semantic KITTI: Semantic KITTI [11, 3] is another large-scale dataset with several dense point cloud sequences captured from the self-driving scenario, which contains 23,311 scans with about 5 billion points. In our experiment, it is used for both static and dynamic point cloud compression.

For the ScanNet dataset, we use the official training/testing split, which includes 1,201 point clouds for training and 312 point clouds for testing. For evaluating the static point cloud compression methods on the Semantic KITTI dataset, we follow the default setting, where 11 point cloud sequences are used for training and the other 11 sequences are used for testing. For dynamic point cloud compression, considering that the ground-truth pose information is not available in the official testing sequences, in this work, we only use 11 training sequences for training and performance evaluation, in which 8 sequences are used for training while 3 sequences are used for performance evaluation.

4.2. Experimental Details

Baseline Methods. In our experiments, the two most representative hand-crafted point cloud compression methods, MPEG's standard point cloud compression method('G-PCC') [29, 13] and Google's KD-tree based method('Draco') [12] are used as the baseline algorithms. In addition, we also compare our method with the recent learning based static point cloud compression method OctSqueeze [15]. Since the source code of OctSqueeze is not publicly available, we re-implemented it by ourselves.

Training and Testing Strategy. In our training procedure for static point cloud compression, the maximum depth levels on ScanNet and SemanticKITTI are empirically set as 9 and 12, respectively. And the entropy model is optimized by using all nodes from the complete 9-level/12-level octree. In the testing stage, we truncate the octree at different levels to evaluate our deep entropy models at different bitrates.

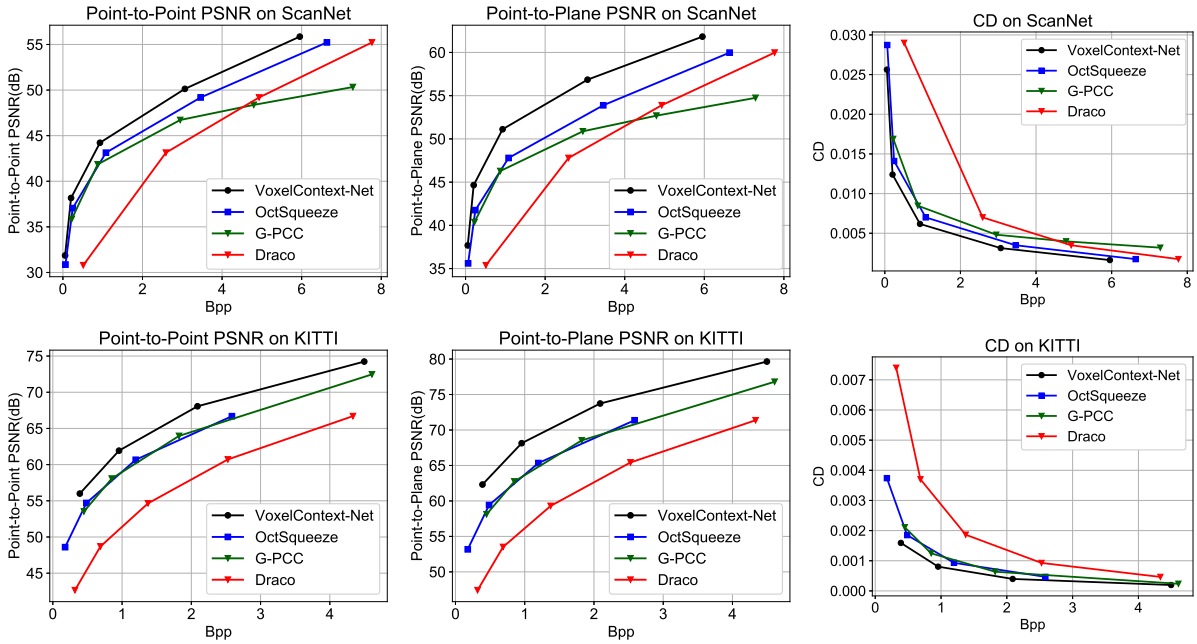

Figure 6. Results of different static point cloud compression methods on two benchmark datasets ScanNet & Semantic KITTI.

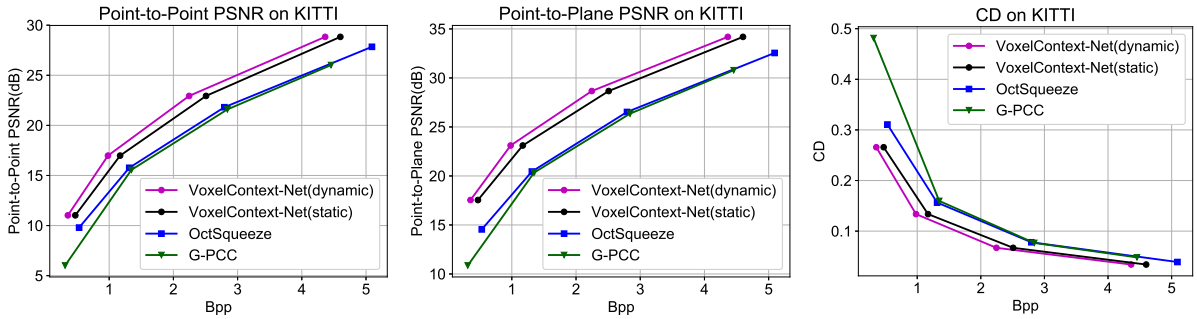

Figure 7. Results of different methods for dynamic point cloud compression on the Semantic KITTI dataset.

It is noted that we train a coordinate refinement model at each depth level to reduce the distortion in the octree construction procedure. For dynamic point cloud compression, we adopt the same training procedure.

We perform the experiments on the machine with one NVIDIA 2080TI GPU. Our whole network is implemented based on PyTorch and it takes 3 days and 5 days to train the model, for the static and dynamic point cloud compression tasks, respectively. In the training procedure, we use the Adam [18] optimizer and the learning rate is set as $1e-4$ for both entropy model and the coordinate refinement model.

Evaluation Metrics In our experiments, we use the Chamfer distance(CD) [10, 16], point-to-point PSNR and point-to-plane PSNR [33, 22], where p is set as 1 to measure the quality of the reconstructed point cloud. We simply use bits per point(Bpp) as the compression ratio metric. To compare different compression algorithms, we also include the BDBR [6] in our approach, which represents the average bitrate saving when the reconstructed quality (*e.g.,* PSNR) of these methods are the same.

4.3. Experiment Results

Results for Static Point Cloud Compression. The quantitative experimental results are provided in Table 1 and we use BDBR to evaluate the compression performance of different codecs, where G-PCC [29, 13] is used as the anchor algorithm. It is observed that our approach (*i.e.,* VoxelContext-Net) saves 43.66% bitrate on the ScanNet dataset when compared with G-PCC, while the corresponding bitrate saving for OctSqueeze is only 15.00%. Similar results are also observed on the Semantic KITTI dataset, where our approach achieves 31.15% bitrate saving, while OctSqueeze only saves 2.13% bitrates. These experimental results clearly demonstrate our approach outperforms the state-of-the-art learning based compression algorithm and the traditional codecs like G-PCC.

For static point cloud compression, the corresponding rate-distortion curves are provided in Figure 6. Our approach achieves better compression performance, especially at high bitrates. For example, our approach has more

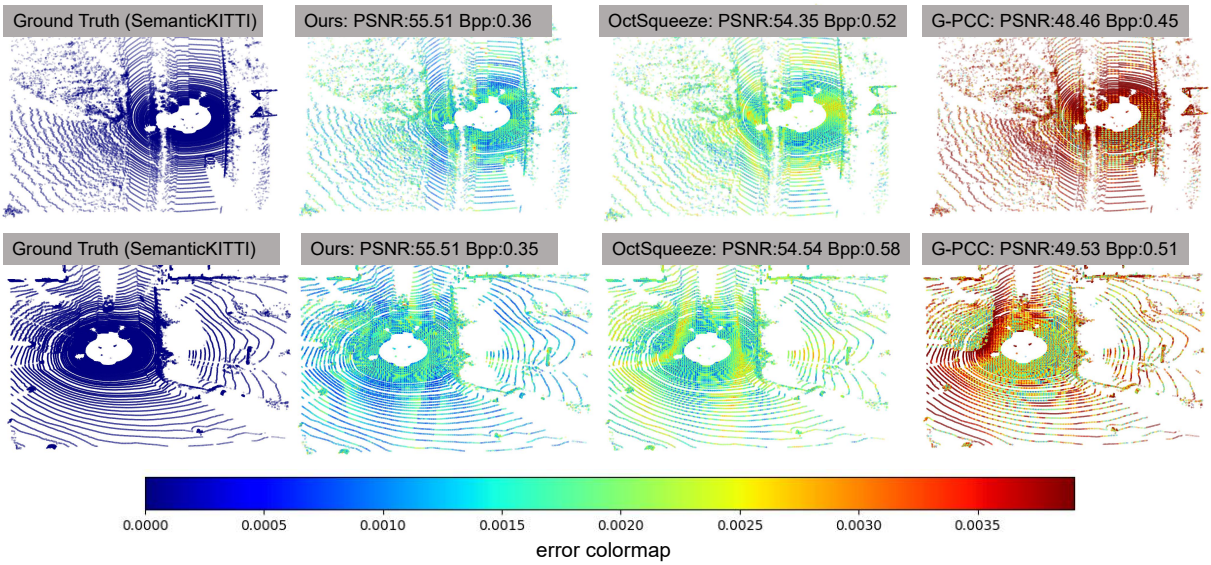

Figure 8. Visualization of our VoxelContext-Net and other baseline methods for static point cloud compression on Semantic KITTI.

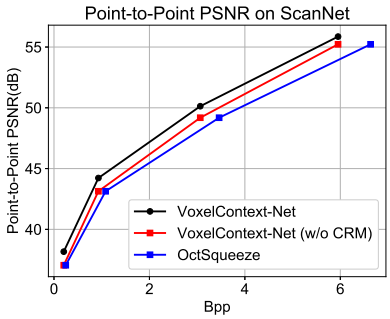

Figure 9. Ablation study on the Scannet dataset.

than 2dB improvement over OctSqueeze on the ScanNet dataset when the bpp is 4.

In Figure 8, we take the Semantic Kitti dataset as an example to provide the qualitative results. We observe that the errors between the ground-truth point clouds and our reconstructed point clouds are also smaller when compared with the baseline methods OctSqueeze and G-PCC. For example, the bpp of our approach is 0.36 and the corresponding PSNR is 55.51dB, while OctSqueeze achieves a lower PSNR(54.35dB) when using more bits(0.52bpp).

Results for Dynamic Point Cloud Compression. For dynamic point cloud compression, we provide the quantitative results in Table 2 and Figure 7. It is observed that our dynamic point cloud compression method (*i.e.,* Ours(dynamic)) outperforms the baseline methods G-PCC and OctSqueeze by a large margin. For example, Ours(dynamic) saves 38.10% bitrates when compared with the anchor algorithm G-PCC while the corresponding saving is only 5.01% for the OctSqueeze method. Furthermore, when compared with Ours(static), Ours(dynamic) saves additional 11.11% bitrates (-26.99% vs. -38.10%) on Semantic KITTI. Considering that the only difference between Ours(static) and Ours(dynamic) is the additional temporal voxel context in the entropy model (see Section 3.7 and Figure 5), the results demonstrate it is effective to exploit the temporal information for dynamic point cloud compression.

Since our approach requires the pose information of the sensor when performing point cloud alignment, we also provide the results on the KITTI dataset when our approach uses the estimated pose information [40], instead of the ground truth pose information. The experimental result in Table 2 demonstrates that our approach with the estimated pose information (*i.e.,* Ours*(dynamic)), also saves 41.77% bitrate and still outperforms the existing baseline methods. It is noted that Ours*(dynamic) performs even better than Ours(dynamic). One possible explanation is that the ground-truth pose represents motion information of the sensors while the estimated pose information is calculated based on the motion between actual point clouds, which may be more useful for the compression task.

4.4. Ablation Study and Analysis

In this section, we take our VoxelContext-Net for static point cloud compression on the ScanNet dataset as an example to perform the ablation study.

Effectiveness of Our Proposed Components. To demonstrate the effectiveness of our proposed two components, we consider a simplified version of our approach by removing the coordinate refinement module, which is referred to as VoxelContext-Net (w/o CRM).

Based on the experimental results (see the purple curve) in Figure 9, we have the following two observations. First, when compared with the octree based entropy model in OctSqueeze [15], our approach using the local voxel context boosts the performance and saves an additional 14.81% bitrate in terms of BDBR. It shows it is more effective to use context information provided by the local voxel represen-

Table 1. BDBR(%) results of our method Ours(static) and two baseline algorithms Draco [12] and OctSqueeze [15] when compared with G-PCC [29, 13] on two benchmark datasets for static point cloud compression.

Methods	Draco	OctSqueeze	Ours(static)
ScanNet	+133.32	-15.00	-43.66
Semantic KITTI	+138.58	-2.13	-31.15

Table 2. BDBR(%) results of our methods and the baseline algorithm OctSqueeze [15] when compared with G-PCC on Semantic KITTI for dynamic point cloud compression. In Ours(dynamic) and Ours*(dynamic), we use the ground-truth and the estimated pose information, respectively.

OctSqueeze	Ours(static)	Ours(dynamic)	Ours*(dynamic)
-5.01	-26.99	-38.10	-41.77

tation rather than that extracted from the parent and child nodes [15]. Second, when comparing our full model (see the black curve) and the simplified model (see the purple curve), it is observed that the coordinate refinement procedure further improves the performance and saves 21.50% bitrates in terms of BDBR. Therefore, it is beneficial to reduce the distortion from the octree construction procedure by using our proposed coordinate refinement module.

Voxel Size. In our implementation, the size of local voxel representation for each node is empirically set as $9 \times 9 \times 9$. As shown in Table 3, we provide more experimental results when the size varies. It is noted that the performance can be boosted by increasing the resolution of the local voxel representations. For example, when compared with our entropy model with the voxel size as $5 \times 5 \times 5$, our proposed approach saves 2.5% bitrates when the voxel size becomes $9 \times 9 \times 9$ at the depth level 9. Considering that the computational complexity will also increase by using larger voxel sizes, we choose $9 \times 9 \times 9$ as the default size in our approach for better trade-off between compression performance and computational complexity.

Computational Complexity In Table 4, we provide the decoding time of different methods at various bitrates on two datasets. Our method is faster than G-PCC [29, 13]. While it is slower than OctSqueeze [15], we achieve better compression performance (see Fig. 6 and Table 1). The total number of parameters in our approach is 2.15M.

Experimental Results for the Downstream Tasks We evaluate the impact of different point cloud compression methods for two downstream tasks (*i.e.,* object detection and semantic segmentation). Specifically, we use VoteNet [24] as the object detection method and PointNet++ [26] as the semantic segmentation method. We train these models based on the uncompressed point clouds from the official training dataset of ScanNet. In the evaluation stage, the reconstructed point clouds at different bitrates are fed into the detection or the segmentation method. For the object detection task, we employ mAP@0.25 to measure the detection accuracy. Following the setting in [15], for the semantic segmentation task, we adopt the intersection-over-union (IOU) score for performance evaluation, which is computed based on the ground truth labels for each voxel.

Table 3. Results of our deep entropy model when using various sizes of local voxel representations on the ScanNet dataset.

Size	Bpp			
	Depth=6	Depth=7	Depth=8	Depth=9
$5 \times 5 \times 5$	0.207	0.950	3.137	6.119
$7 \times 7 \times 7$	0.205	0.937	3.090	6.009
$9 \times 9 \times 9$	0.204	0.930	3.065	5.952
$11 \times 11 \times 11$	0.205	0.935	3.061	5.932

Table 4. Decoding time(ms) on ScanNet/KITTI at five/four different bitrates reported in Figure 6 (from low to high).

Methods	Ours	OctSqueeze	G-PCC
ScanNet	49/52/64/80/109	6/6/7/7/7	306/309/316/324/348
KITTI	52/58/78/90	6/7/7/8	578/649/734/768

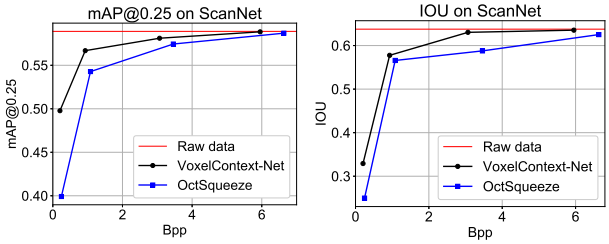

Figure 10. The results for the two downstream tasks (*i.e.,* object detection (left) and semantic segmentation (right)) on ScanNet.

The experimental results are shown in Fig 10. Specifically, we provide the compression ratios (i.e., bpps) and the corresponding detection/segmentation accuracy (i.e., mAP/IOU) when using the decoded point clouds from our VoxelContext-Net and the baseline method OctSqueeze as the input to the detection/segmentation method. At any given bpp, it is obvious that the detection/segmentation results based on the decoded point clouds from our VoxelContext-Net are higher, especially at low-bitrates. It demonstrates that the reconstructed point clouds from our VoxelContext-Net are more useful for the downstream tasks, like object detection or semantic segmentation.

5. Conclusion

In this work, we have proposed a new learning based point cloud geometry compression framework by exploiting the local voxel representation for each node in the octree structured point cloud. Specifically, we propose a new deep entropy model to losslessly compress the symbols of octree nodes and a new coordinate refinement module for reconstructing high-quality point clouds at the decoder side. Our simple and effective approach is applied to both static and dynamic point cloud compression and our method has achieved the state-of-the-art compression performance on two benchmark datasets.

Acknowledgement This work was supported by the National Key Research and Development Project of China (No. 2018AAA0101900).

References

[1] Johannes Ballé, Valero Laparra, and Eero P. Simoncelli. End-to-end optimized image compression. In *5th International Conference on Learning Representations, ICLR*, 2017. 2

[2] Johannes Ballé, David Minnen, Saurabh Singh, Sung Jin Hwang, and Nick Johnston. Variational image compression with a scale hyperprior. In *6th International Conference on Learning Representations, ICLR*, 2018. 1, 2

[3] J. Behley, M. Garbade, A. Milioto, J. Quenzel, S. Behnke, C. Stachniss, and J. Gall. SemanticKITTI: A Dataset for Semantic Scene Understanding of LiDAR Sequences. In *Proc. of the IEEE/CVF International Conf. on Computer Vision (ICCV)*, 2019. 2, 5

[4] Fabrice Bellard. Bpg image format. *URL https://bellard.org/bpg*, 2015. 1

[5] 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, 2020. 2

[6] Gisle Bjontegaard. Calculation of average psnr differences between rd-curves. *VCEG-M33*, 2001. 6

[7] Angela Dai, Angel X Chang, Manolis Savva, Maciej Halber, Thomas Funkhouser, and Matthias Nießner. Scannet: Richly-annotated 3d reconstructions of indoor scenes. In *Proceedings of the IEEE Conference on Computer Vision and Pattern Recognition*, pages 5828–5839, 2017. 2, 5

[8] Ricardo L De Queiroz and Philip A Chou. Compression of 3d point clouds using a region-adaptive hierarchical transform. *IEEE Transactions on Image Processing*, 25(8):3947–3956, 2016. 2

[9] Ricardo L de Queiroz and Philip A Chou. Transform coding for point clouds using a gaussian process model. *IEEE Transactions on Image Processing*, 26(7):3507–3517, 2017. 2

[10] Haoqiang Fan, Hao Su, and Leonidas J Guibas. A point set generation network for 3d object reconstruction from a single image. In *Proceedings of the IEEE conference on computer vision and pattern recognition*, pages 605–613, 2017. 6

[11] A. Geiger, P. Lenz, and R. Urtasun. Are we ready for Autonomous Driving? The KITTI Vision Benchmark Suite. In *Proc. of the IEEE Conf. on Computer Vision and Pattern Recognition (CVPR)*, pages 3354–3361, 2012. 2, 5

[12] Google. Draco 3d graphics compression. https://github.com/google/draco, 2017. 2, 5, 8

[13] MPEG Group. Mpeg g-pcc tmc13. https://github.com/MPEGGroup/mpeg-pcc-tmc13, accessed:2020. 2, 5, 6, 8

[14] Zhihao Hu, Zhenghao Chen, Dong Xu, Guo Lu, Wanli Ouyang, and Shuhang Gu. Improving deep video compression by resolution-adaptive flow coding. In *European Conference on Computer Vision*, pages 193–209. Springer, 2020. 2

[15] Lila Huang, Shenlong Wang, Kelvin Wong, Jerry Liu, and Raquel Urtasun. Octsqueeze: Octree-structured entropy model for lidar compression. In *Proceedings of the IEEE/CVF Conference on Computer Vision and Pattern Recognition (CVPR)*, June 2020. 1, 2, 3, 5, 7, 8

[16] Tianxin Huang and Yong Liu. 3d point cloud geometry compression on deep learning. In *Proceedings of the 27th ACM International Conference on Multimedia*, pages 890–898, 2019. 1, 2, 6

[17] Chris L Jackins and Steven L Tanimoto. Oct-trees and their use in representing three-dimensional objects. *Computer Graphics and Image Processing*, 14(3):249–270, 1980. 2

[18] Diederik P Kingma and Jimmy Ba. Adam: A method for stochastic optimization. *arXiv preprint arXiv:1412.6980*, 2014. 6

[19] Guo Lu, Chunlei Cai, Xiaoyun Zhang, Li Chen, Wanli Ouyang, Dong Xu, and Zhiyong Gao. Content adaptive and error propagation aware deep video compression. In *European Conference on Computer Vision*, pages 456–472. Springer, 2020. 1

[20] Guo Lu, Wanli Ouyang, Dong Xu, Xiaoyun Zhang, Chunlei Cai, and Zhiyong Gao. DVC: An end-to-end deep video compression framework. In *Proceedings of the IEEE Conference on Computer Vision and Pattern Recognition,CVPR*, pages 11006–11015, 2019. 2

[21] Guo Lu, Xiaoyun Zhang, Wanli Ouyang, Li Chen, Zhiyong Gao, and Dong Xu. An end-to-end learning framework for video compression. *IEEE Transactions on Pattern Analysis and Machine Intelligence*, 2020. 2

[22] Rufael Mekuria, Sebastien Laserre, and Christian Tulvan. Performance assessment of point cloud compression. In *2017 IEEE Visual Communications and Image Processing (VCIP)*, pages 1–4. IEEE, 2017. 6

[23] David Minnen, Johannes Ballé, and George D Toderici. Joint autoregressive and hierarchical priors for learned image compression. In *Advances in Neural Information Processing Systems*, pages 10771–10780, 2018. 2

[24] Charles R Qi, Or Litany, Kaiming He, and Leonidas J Guibas. Deep hough voting for 3d object detection in point clouds. In *Proceedings of the IEEE International Conference on Computer Vision*, pages 9277–9286, 2019. 8

[25] Charles R Qi, Hao Su, Kaichun Mo, and Leonidas J Guibas. Pointnet: Deep learning on point sets for 3d classification and segmentation. In *Proceedings of the IEEE conference on computer vision and pattern recognition*, pages 652–660, 2017. 1

[26] Charles Ruizhongtai Qi, Li Yi, Hao Su, and Leonidas J Guibas. Pointnet++: Deep hierarchical feature learning on point sets in a metric space. In *Advances in neural information processing systems*, pages 5099–5108, 2017. 1, 8

[27] Radu Bogdan Rusu and Steve Cousins. 3D is here: Point Cloud Library (PCL). In *IEEE International Conference on Robotics and Automation (ICRA)*, Shanghai, China, May 9-13 2011. 2

[28] Ruwen Schnabel and Reinhard Klein. Octree-based point-cloud compression. *Spbg*, 6:111–120, 2006. 2

[29] Sebastian Schwarz, Marius Preda, Vittorio Baroncini, Madhukar Budagavi, Pablo Cesar, Philip A Chou, Robert A Cohen, Maja Krivokuća, Sébastien Lasserre, Zhu Li, et al. Emerging mpeg standards for point cloud compression.

IEEE Journal on Emerging and Selected Topics in Circuits and Systems, 9(1):133–148, 2018. 2, 5, 6, 8

[30] Claude E Shannon. A mathematical theory of communication. *The Bell system technical journal*, 27(3):379–423, 1948. 3

[31] Gary J Sullivan, Jens-Rainer Ohm, Woo-Jin Han, Thomas Wiegand, et al. Overview of the high efficiency video coding(hevc) standard. *TCSVT*, 22(12):1649–1668, 2012. 1

[32] David S Taubman and Michael W Marcellin. Jpeg2000: Standard for interactive imaging. *Proceedings of the IEEE*, 90(8):1336–1357, 2002. 1

[33] Dong Tian, Hideaki Ochimizu, Chen Feng, Robert Cohen, and Anthony Vetro. Geometric distortion metrics for point cloud compression. In *2017 IEEE International Conference on Image Processing (ICIP)*, pages 3460–3464. IEEE, 2017. 6

[34] Chenxi Tu, Eijiro Takeuchi, Alexander Carballo, and Kazuya Takeda. Real-time streaming point cloud compression for 3d lidar sensor using u-net. *IEEE Access*, 7:113616–113625, 2019. 2

[35] Gregory K Wallace. The jpeg still picture compression standard. *IEEE transactions on consumer electronics*, 38(1):xviii–xxxiv, 1992. 1

[36] Jianqiang Wang, Hao Zhu, Zhan Ma, Tong Chen, Haojie Liu, and Qiu Shen. Learned point cloud geometry compression. *arXiv preprint arXiv:1909.12037*, 2019. 1, 2

[37] Thomas Wiegand, Gary J Sullivan, Gisle Bjontegaard, and Ajay Luthra. Overview of the h. 264/avc video coding standard. *TCSVT*, 13(7):560–576, 2003. 1

[38] Chao-Yuan Wu, Nayan Singhal, and Philipp Krahenbuhl. Video compression through image interpolation. In *ECCV*, September 2018. 2

[39] Wei Yan, Shan Liu, Thomas H Li, Zhu Li, Ge Li, et al. Deep autoencoder-based lossy geometry compression for point clouds. *arXiv preprint arXiv:1905.03691*, 2019. 1, 2

[40] Deyu Yin, Qian Zhang, Jingbin Liu, Xinlian Liang, Yunsheng Wang, Jyri Maanpää, Hao Ma, Juha Hyyppä, and Ruizhi Chen. Cae-lo: Lidar odometry leveraging fully unsupervised convolutional auto-encoder for interest point detection and feature description. *arXiv preprint arXiv:2001.01354*, 2020. 7