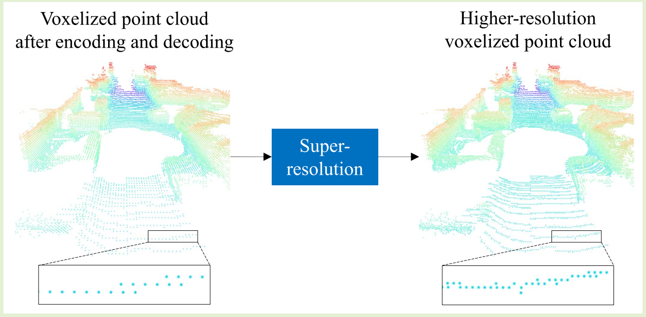
Efficient Deep Super-Resolution of Voxelized Point Cloud in Geometry Compression

Kohei Matsuzaki and Satoshi Komorita

***Abstract*—Point cloud compression is an essential task for practical applications using point clouds. Most of the previous approaches rely on octree compression which involves voxelization in the coding itself. Distortions derived from voxelization can be reduced without increasing the bitrate by postprocessing. In this article, we propose a super-resolution method for a decoded voxelized point cloud as a postprocessing step in the geometry compression. The proposed method increases the resolution of the voxelized point cloud by predicting the occupancy of higher resolution voxels than those used to compress the original point cloud. For efficient prediction, we propose a deep neural network for super-resolution based on sparse convolution. It can be highly efficient even for a large point cloud since the network applies convolution only to nonempty space. The proposed method predicts the occupancies represented by continuous values for each point and estimates the binary occupancies through a thresholding procedure. We design a dynamic threshold to ensure that at least one of all voxels is predicted to be occupied in order to prevent the generation of regions with missing points. We also introduce an occupancy prediction method to address the sparsity of high-resolution occupied voxels. Experiments on the outdoor and indoor datasets demonstrate the effectiveness of the proposed method.**



***Index Terms*— Deep super-resolution, point cloud compression, sparse convolution, voxelization.**

I. Introduction

WITH the widespread utilization of 3-D sensing devices, point cloud compression has attracted a great deal of attention. Point cloud is one of the most common 3-D data formats used in many applications, such as augmented/virtual reality, autonomous navigation, and geographic information systems. Recent sensing technologies represented by light detection and ranging (LiDAR) can acquire a large amount of data in the order of more than one million points per second. In the processing of such data, an efficient compression method is essential to reduce storage and traffic requirements.

Point cloud compression has been actively studied in recent years with the aim of enabling it to handle a large number of point clouds. In compressing sparse point clouds, such as those obtained from 3-D sensing devices, most of the approaches rely on octree compression, which voxelizes the point cloud based on voxel grids [1], [2], [3], [4], [5], [6], [7], [8]. While many point cloud compression approaches perform voxelization before coding, the octree compression involves voxelization in the coding itself. The octree representation is of significant benefit for efficient encoding since the sparse point cloud is a set of points occupying only a small portion of 3-D space. It hierarchically represents the occupancy of voxels by recursively dividing the voxel space encompassing the point cloud into eight octants. Thus, the higher the level in the hierarchy, the higher the voxel resolution, and the more faithfully the shape can be preserved. On the other hand, there is a tradeoff that the higher the level, the higher the bitrate.

A typical approach to balancing the tradeoff is to truncate the octree at a user-specified level. The appropriate level can be selected to achieve the fidelity required by the application since the level corresponds to the voxel resolution used in the voxelization. Alternatively, a level can be set that corresponds to an acceptable bitrate. Another possible approach is to correct the point cloud distortion derived from the voxelization as a postprocessing step after encoding and decoding. The distortion can be reduced without increasing the bitrate by applying postprocessing to the decoded point cloud. This reduces the level at which the required fidelity can be achieved, resulting in a lower bitrate.

Point cloud correction has been studied for various purposes such as shape completion [9], [10], [11], upsampling [12], [13], [14], and noise reduction [15], [16]. They are applied to point clouds acquired by sensing devices in order to improve

Manuscript received 7 November 2022; accepted 15 November 2022. Date of publication 2 December 2022; date of current version 12 January 2023. The associate editor coordinating the review of this article and approving it for publication was Dr. Guofa Li. *(Corresponding author: Kohei Matsuzaki.)*

The authors are with the 3D Space Transmission Laboratory, KDDI Research, Inc., Saitama 356-8502, Japan (e-mail: ko-matsuzaki@kddi.com).

Digital Object Identifier 10.1109/JSEN.2022.3225170

fidelity to the shape of the target object. Furthermore, a point cloud super-resolution method that increases the resolution of point clouds acquired by low-resolution LiDAR sensors is also being studied [17], [18]. However, these methods are not always effective for a voxelized point cloud since they do not take compression into account. Recently, a point coordinate refinement method for correcting distortion has been proposed as a postprocessing step for octree compression [7]. Although it reduces point cloud distortion by predicting the offset vector for correcting the coordinates, it is not designed to reconstruct data lost in the voxelization.

In this article, we propose a super-resolution method for voxelized point clouds to reduce distortions as a postprocessing step in geometry compression. The proposed method increases the resolution of the voxelized point cloud by predicting occupancy of higher resolution voxels than those used to compress the original point cloud. This reduces distortion derived from the voxelization without increasing the bitrate. Processing large point clouds such as those acquired by LiDAR sensors may be computationally expensive. Therefore, we propose a deep super-resolution network for super-resolution based on sparse convolution. It is highly efficient even for large point clouds since convolution is applied only to nonempty space. We also propose an occupancy prediction method to address the sparsity of high-resolution occupied voxels. The proposed method may be applied to both static and dynamic point clouds.

The contributions of this article can be summarized as follows.

1) We propose a super-resolution method to reduce distortion of a voxelized point cloud in geometry compression by predicting pointwise occupancies corresponding to high-resolution voxels.
2) We construct a deep neural network for super-resolution based on efficient sparse convolution. High-resolution point clouds are reconstructed from the network output by estimating binary occupancies.
3) Evaluation experiments showed the effectiveness of the proposed method on reducing point cloud distortion as a postprocessing step in geometry compression. The proposed method achieves the state-of-the-art results in both compression performance and efficiency.

The rest of this article is organized as follows. In Section II, we review related work on point cloud compression, correction, and super-resolution. In Section III, we propose a super-resolution method to reduce point cloud distortion in geometry compression. In Section IV, we evaluate the effectiveness of the proposed method on outdoor and indoor datasets. Section V concludes this article.

II. Related Work

A. Point Cloud Compression

It has been an important research topic in computer graphics and signal processing due to the excellent representational capabilities and high data capacity of the point cloud. The point cloud consists of geometry information and attribute information. The geometry information represents the position of the point, and the attribute information represents the associated attributes of the point, such as color values, reflection intensity, and normal vectors. Therefore, point cloud compression is divided into geometry compression and attribute compression [19]. In this article, we concentrate on geometry compression.

Traditional point cloud compression approaches effectively encode geometry information by extracting dependencies between points using handcrafted techniques [2], [3], [4], [20], [21], [22], [23], [24]. The Moving Picture Experts Group (MPEG) has developed two compression standards called geometry-based point cloud compression (G-PCC) and video-based point cloud compression (V-PCC) [4]. V-PCC encodes point clouds projected from 3-D space to 2-D plane using a video codec [e.g., high efficiency video coding (HEVC)], while G-PCC encodes point clouds directly in 3-D space using efficient data structures such as octree. Although V-PCC is suitable for dense point clouds that produce a continuous and smooth surface, it is not suitable for sparse point clouds. In several works [23], [24], effective compression was achieved with the help of video coding techniques by projecting sparse point clouds onto range images. However, these works are difficult to apply to other types of point clouds, such as those from multiview reconstruction since they are specialized for LiDAR point clouds.

In recent years, learning-based point cloud compression has attracted much attention, inspired by the success of deep learning techniques in image compression [25], [26] and video compression [27], [28]. The point cloud autoencoder [29], [30], [31], [32], [33], [34], [35], [36], [37], [38], [39] maps the input point cloud to a latent representation by an encoder network and reconstructs the point cloud by a decoder network. Most of the conventional point cloud autoencoders are based on voxelization, which introduces distortions [29], [30], [31], [32], [33], [34], [35]. On the other hand, several methods have been proposed to directly encode point clouds without voxelization [36], [37], [38]. These methods may suffer from huge memory usage and high computational costs for large-scale point clouds since they predict the coordinates of point clouds based on a fully connected network. Aiming at efficient compression of dense point clouds, an autoencoder is also proposed, which performs downsampling and upsampling of points during encoding and decoding, respectively [39]. The reconstruction error caused by each iteration accumulates and large artifacts may occur since this method reconstructs the point cloud by iterative upsampling.

The other learning-based approaches [5], [6], [7], [8] proposed learning an entropy model using deep neural networks in an octree and entropy compression schemes. The octree structure has been widely used for point cloud compression since it flexibly and efficiently models arbitrary point clouds, including sparse point clouds. The learning-based approaches design deep entropy models that encode context information with the local structure of each point. They showed better compression performance than octree-based G-PCC, which uses a handcrafted entropy model. However, while these learning-based approaches improve the performance of entropy compression, the point cloud distortion caused by octree compression is not reduced. In contrast, our goal is to reduce the distortion in a postprocessing step.

B. Point Cloud Correction

Point cloud correction has been studied to improve the fidelity of the point cloud to the target shape. Shape completion [9], [10], [11] reconstructs the complete shape from the point cloud representing the incomplete shape captured by the sensing device. This approach has applicability for many applications in the field of recognition and reconstruction. However, it is not suitable for reducing distortion since it reconstructs a significantly different shape from the input point cloud. Point cloud upsampling [12], [13], [14] generates a higher density point cloud from the input point cloud. The approach uniformly increases the number of points to reconstruct the object surface. However, it is not always effective for reducing the distortion of decoded point clouds in which decoded resolution is lower than the input resolution since it is designed to interpolate input points rather than to reconstruct detailed shape. Noise reduction [15], [16] removes noise that occurred during sensing and reconstructs a clean point cloud. This approach aims to reconstruct a point cloud located on smooth surfaces from the input point cloud containing measurement noise. Therefore, it is difficult to apply this approach to sparse point clouds that cannot exploit the underlying surface, such as those acquired by a LiDAR sensor.

A recent work [7] has proposed a method of refining the coordinates of a voxelized point cloud. It predicts pointwise offset vectors to correct errors from the original point cloud using a deep neural network called the coordinate refinement module (CRM). It can reduce distortion without increasing the bitrate. However, CRM cannot deal with the case where multiple original points exist inside the voxel used in voxelization since it predicts only a single offset for each point. In other words, it is not designed to reconstruct data lost in voxelization. In contrast, we propose to explicitly reduce distortions derived from artifacts associated with compression methods, which is compressed by performing further voxelization such as octree-based methods.

C. Point Cloud Super-Resolution

It takes an original point cloud with any spatial resolution as input and reconstructs a point cloud with a higher resolution than that of the input. It can be regarded as a specific type of point cloud correction in geometry compression, which involves voxelization.

An optimization-based approach [40], [41] super-resolves the original point cloud based on optimization of an objective function to promote piecewise smoothness of the underlying surface. As this approach is aimed at densifying the point cloud for rendering images on high-resolution displays, it does not take care of the geometry distortions due to compression.

An example-based approach [42] searches for local volumes based on similarity from downsampled versions of time-adjacent high-resolution point clouds. The original point cloud is super-resolved using an upsampled version of the searched local volumes. Although this approach targets voxelized point clouds in geometry compression, it cannot be applied without the decoded high-resolution point clouds.

Super-resolution by neighborhood inheritance [43] predicts high-resolution voxel occupancy based on neighborhood

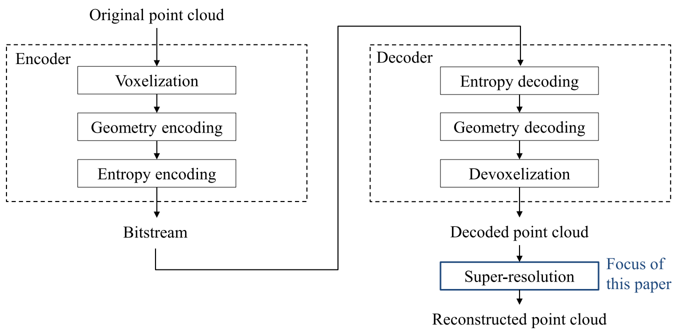


Fig. 1. Overview of the point cloud geometry compression framework. The focus of this article is super-resolution, which is introduced as a postprocessing step after encoding and decoding.

occupancy at the voxel level and a classification of voxel division results. It reconstructs high-resolution points for each voxelized point using prebuilt lookup tables. This approach may only be applicable to dense point clouds since it is restricted to searching very small neighborhoods in order to save memory and computational effort.

LiDAR-SR [17] predicts the high-resolution point cloud from the low-resolution point cloud. The resolution of these point clouds depends on the number of vertical channels of a LiDAR sensor. This approach converts the problem of point cloud super-resolution into an image super-resolution by projecting a LiDAR point cloud onto a range image. The super-resolved range image is backprojected into 3-D space and treated as the high-resolution point cloud. Fusion of high-resolution range images with image-based segmentation results has also been proposed to address depth discontinuities between objects [18]. This approach does not always accurately predict the range image when a voxelized point cloud is input since it usually takes a raw point cloud as input.

Voxel super-resolution [44], [45] reconstructs a high-resolution volume from a low-resolution volume. It may be exploited for point cloud super-resolution purposes by transforming the volume to a point cloud. However, this approach is based on implicit representation of the 3-D shape and requires watertight meshes to determine whether a point is inside the mesh for training of the model. Therefore, this approach limits the applicable point cloud.

Point cloud geometry prediction (PCGP) [46] super-resolves voxelized point clouds based on occupancy predictions of high-resolution voxels. This method divides the point cloud into small local patches and feeds each patch into a deep neural network. Therefore, the efficiency of processing may decrease for large-scale point clouds as the number of patches increases. Furthermore, occupancy prediction for sparse point clouds remains a challenge since this method is designed for dense point clouds.

III. Proposed Method

In this section, we propose a point cloud super-resolution method as a postprocessing step in point cloud geometry compression. Fig. 1 shows an overview of the geometry compression framework assumed in this article. In this framework, the original point cloud is voxelized, which corresponds to the voxelization that most point cloud compression methods perform as a preprocessing before coding to reach lower bitrates. Then, any geometry encoding is performed on the voxelized point cloud. Here, geometry encoding that involves voxelization in the encoding itself may be performed, such as an octree-based method. In that case, further voxelization is performed on the voxelized point cloud during encoding to make its resolution equal to or less than the input to this process. After entropy coding is performed, the bitstream is output from the encoder. At the decoder, the point cloud is decoded from the bitstream by the decoding processes corresponding to the encoding processes. There is distortion derived from voxelization between the original point cloud and the decoded point cloud. In order to reduce this distortion, we introduce a process of super-resolution of the decoded point cloud. This process reconstructs the high-resolution point cloud by predicting the occupancy of higher resolution voxels than those used to voxelize the original point cloud. The degree of resolution increase is set by the user and may not always target the recovery of the resolution of the original point cloud. Although other types of distortion can occur when geometry coding is a lossy scheme, the proposed method concentrates on reducing the distortion derived from voxelization. Finally, the framework outputs a reconstructed point cloud.

A. Preliminaries

In the following, we briefly explain the voxelization of the point cloud. Voxelization is a process in which the coordinates of the point cloud are transformed into coordinates that represent the corners of the voxel. This is equivalent to quantization using a quantization step δ that represents the voxel size. Let $\mathcal{P} = \{\boldsymbol{p}_i \in \mathbb{R}^3\}_{i=1}^N$ denote the original point cloud. The quantization using δ is defined as follows:

$$\boldsymbol{q}_i = \left\lfloor \frac{s\boldsymbol{p}_i + \boldsymbol{t}}{\delta} \right\rfloor \tag{1}$$

where $\boldsymbol{q}_i \in \mathbb{R}^3$ is a quantized point, $\boldsymbol{t} \in \mathbb{R}^3$ is an offset to make all coordinates nonnegative, and $s \in \mathbb{R}$ is a scale factor. Devoxelization is a process of reconstructing a point in the original coordinate system defined as follows:

$$\tilde{\boldsymbol{p}}_i = \frac{\delta\boldsymbol{q}_i - \boldsymbol{t}}{s} \tag{2}$$

where $\tilde{\boldsymbol{p}}_i \in \mathbb{R}^3$ is a reconstructed point. The reconstructed point has distortions with respect to the original point since this process does not recover the data lost in voxelization. Although these processes are based on a floor function, some variants based on other functions, such as ceil, round, and truncation, are also available.

Some geometric encodings themselves involve voxelization. We give an explanation with octree encoding, one of the most popular schemes, as a specific example. In the octree representation, the point cloud is first encompassed by a bounding cube, and then, the cube is recursively divided into eight octants. The octant is considered a voxel and its size per side is halved as the octree level increases. Let L be the edge length of the bounding cube, and the voxel size at the lth level ($l = 1, 2, \ldots$) is represented as $\delta^{(l)} = L/2^l$. If an octree is truncated at the lth level by octree pruning, the point cloud to be encoded is represented by voxels with size $\delta^{(l)}$. It corresponds to the voxelization of the original point cloud using $\delta^{(l)}$ as δ in (1). The original point cloud may already be voxelized using δ. Even in that case, it is further voxelized using $\delta^{(l)}$, which is equal to or greater than δ when octree pruning is performed. Points in the original coordinate system are reconstructed using $\delta^{(l)}$ as δ in (2).

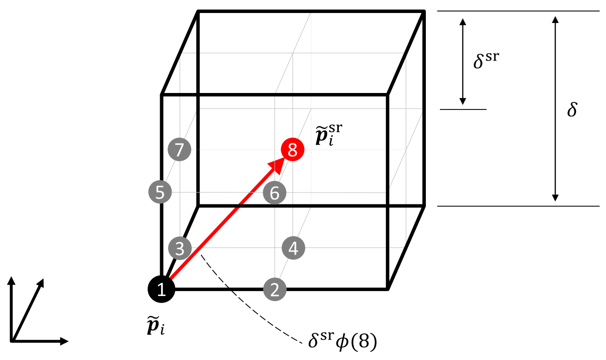

δ^{sr}
δ
$\tilde{\boldsymbol{p}}_i^{\text{sr}}$
7
8
5
6
3
4
1
2
$\tilde{\boldsymbol{p}}_i$
$\delta^{\text{sr}}\phi(8)$

Fig. 2. Illustration of a voxel for voxelization and its high-resolution voxels. The corners of high-resolution voxels drawn as circles are candidate positions where the point $\tilde{\boldsymbol{p}}_i^{\text{sr}}$ is reconstructed from $\tilde{\boldsymbol{p}}_i$. The number in the circle represents the index of the high-resolution voxels.

Geometry coding with voxelization usually stores only the code representing whether or not points exist inside a voxel. Therefore, all points in the same voxel are merged into a single point. An extension that stores side information representing the number of points per voxel is possible. However, it is not considered in this article since it increases the bitrate.

B. Reconstruction of High-Resolution Voxels

We consider reconstructing $\tilde{\boldsymbol{p}}_i^{\text{sr}}$ from $\tilde{\boldsymbol{p}}_i$, where $\tilde{\boldsymbol{p}}_i^{\text{sr}}$ represents super-resolved points. A volume consisting of m voxels with high-resolution can be constructed by dividing the voxel used for voxelization. Let $\boldsymbol{y}_i \in \{0, 1\}^m$ denote occupancies of the high-resolution voxels corresponding to $\tilde{\boldsymbol{p}}_i$. Then, $\tilde{\boldsymbol{p}}_i^{\text{sr}}$ can be reconstructed by generating a point if the jth element ($j = 1, 2, \ldots, m$) of $\boldsymbol{y}_i$ is occupied as follows:

$$\tilde{\boldsymbol{p}}_i^{\text{sr}} = \tilde{\boldsymbol{p}}_i + \delta^{\text{sr}}\phi(j) \tag{3}$$

where δ^{sr} is the voxel size of a high-resolution voxel and $\phi(\cdot)$ is a transformation function from the index of $\tilde{\boldsymbol{p}}_i$ to a spatial index vector representing the relative position of the occupied voxels. Multiple $\tilde{\boldsymbol{p}}_i^{\text{sr}}$ may be reconstructed from single $\tilde{\boldsymbol{p}}_i$ since $\boldsymbol{y}_i$ may contain multiple occupied elements.

Fig. 2 shows an example of a voxel and its high-resolution voxels where $m = 8$. The black cube represents a voxel with size δ. Any point inside this voxel is quantized as a point $\tilde{\boldsymbol{p}}_i$. The gray cubes represent the voxels with size δ^{sr} obtained by dividing the black voxel. The corners of these voxels drawn as circles are candidate positions where the point $\tilde{\boldsymbol{p}}_i^{\text{sr}}$ is reconstructed. The number in the circle corresponds to the index of $\boldsymbol{y}_i$, namely j. Here, it is assumed that a voxel is occupied corresponding to $j = 8$. In the case where $\tilde{\boldsymbol{p}}_i^{\text{sr}}$ is reconstructed as the red circle, the vector $\delta^{\text{sr}}\phi(8)$ representing the relative position is visualized as the red arrow. Also, $\tilde{\boldsymbol{p}}_i^{\text{sr}}$

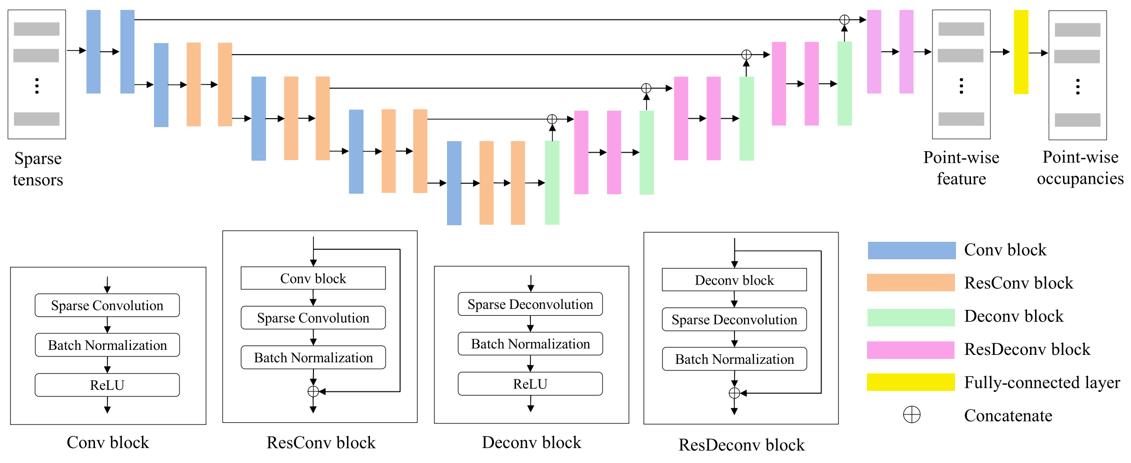


Fig. 3. Network architecture of the proposed method. Given sparse tensors constructed from a voxelized point cloud, pointwise features are extracted by a U-shaped network. The network consists of sparse convolution layers, sparse deconvolution layers, and skip connections. The pointwise occupancies are predicted from the pointwise features with a fully connected layer.

could be reconstructed to the positions of the gray circles or the black circle representing $\tilde{\boldsymbol{p}}_i$ itself.

The proposed method predicts voxel occupancies for $\tilde{\boldsymbol{p}}_i$ from the decoded point cloud. Let $\boldsymbol{x}_i \in \mathbb{R}^m$ denote the predicted occupancies corresponding to the ground truth $\boldsymbol{y}_i$. Then, the proposed method increases the resolution of the decoded point cloud with $\boldsymbol{x}_i$. Finally, we obtain super-resolved points $\hat{\boldsymbol{p}}_i^{\mathrm{sr}}$.

C. Network Architecture

We use a deep neural network to predict high-resolution occupancies for each point from the decoded point cloud. In large-scale point cloud processing, the network may suffer from high memory consumption and expensive computation. To address the problem, we adopt the framework of a sparse 3-D convolutional neural network [47] to construct the network for super-resolution. While it is developed for perceptual tasks such as 3-D semantic segmentation, we apply it to the super-resolution task on sparse point clouds. We adopt this framework to build a backbone network for efficient feature extraction from point clouds. To achieve super-resolution, we propose a predictor to predict the occupancies of high-resolution voxels from these features. For this predictor, we introduce a novel dynamic threshold (DT) to avoid the generation of empty regions.

Fig. 3 shows the network architecture of the proposed method. The input to the network is sparse tensors constructed from a voxelized point cloud $\tilde{\mathcal{P}} = \{\tilde{\boldsymbol{p}}_i \in \mathbb{R}^3\}_{i=1}^n$. The sparse tensor is a data structure for storing pointwise coordinates and features. The network extracts pointwise features by a U-shaped network consisting of sparse convolution layers, sparse deconvolution layers, and skip connections. In order to achieve efficient processing, we adopt the U-shaped networks based on the lightweight architecture proposed in [47] as a feature extractor backbone, unlike the one in [46] introducing inception-residual blocks [48]. Then, a predictor based on a fully connected layer predicts the pointwise occupancies $\boldsymbol{x}_i$ from the features.

The convolution is performed using Conv blocks and ResConv blocks shown at the bottom of Fig. 3. In the convolution, the sparse tensors are downsampled by four Conv blocks with stride 2. The sparse convolution can achieve much higher efficiency than the standard convolution for the point cloud since it applies convolution only to nonempty space.

Deconvolution is performed in the opposite fashion to convolution, concatenating features layer by layer with skip connections. The sparse tensors are upsampled and the coordinates of input are completely reconstructed. Therefore, a set of pointwise features corresponding to each point in the voxelized point cloud is extracted.

In inference, we estimate binary occupancies by thresholding from the occupancies $\boldsymbol{x}_i$ represented by continuous values in the range of [0, 1]. We obtain m binary occupancies for each point corresponding to high-resolution voxels. If all occupancies are below a fixed threshold due to the selection of an inappropriate threshold, all voxels may be estimated as unoccupied. In such a case, empty regions are generated since no points are reconstructed in their voxel spaces. A naive solution for obtaining an appropriate threshold is to find the threshold that maximizes the estimation accuracy for each point at the encoder side and transmit it to the decoder side as side information. However, this solution increases the bitrate and requires additional processing at the encoder side. To avoid such a case without increasing the bitrate, we use the DT defined as follows:

$$\beta = \begin{cases} \max\limits_j(x_{i,j}), & \text{if } \max\limits_j(x_{i,j}) < \alpha \\ \alpha, & \text{otherwise} \end{cases} \tag{4}$$

where α is the fixed threshold and β is the DT. It is guaranteed that at least one voxel is occupied for each point by deciding occupancy of β or less as occupied. This threshold can be computed efficiently without requiring side information.

D. Multilevel Occupancy Prediction

The proposed method can predict the occupancies of higher resolution voxels from the decoded point cloud by increasing

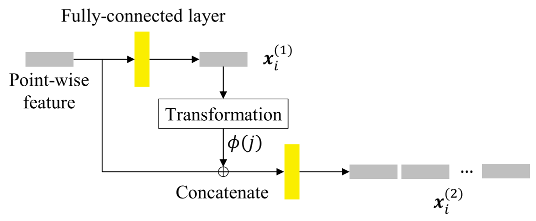


Fig. 4. Illustration of the hierarchical predictor. The predictor first predicts the voxel occupancies at one level higher ($k = 1$) than the one used for voxelization. It then predicts voxel occupancies at further one level higher ($k = 2$) for the voxels predicted as occupied.

the number of voxel divisions m. However, a larger m may increase the sparsity of the high-resolution occupied voxels, which may become difficult to predict. To address this difficulty, we design a hierarchical predictor to predict occupancies of recursively divided voxels. We first divide a voxel used for voxelization into m voxels and predict their occupancies. Then, we recursively divide the occupied voxels and predict the occupancies of the divided voxels. We store these occupancies hierarchically, such as the octree representation. Let $k \in \mathbb{N}$ denote the number of level increases in the hierarchy, corresponding to the number of recursive voxel divisions.

Fig. 4 shows this hierarchical predictor for $k = 2$. First, it predicts occupancies $\boldsymbol{x}_i^{(1)} \in \mathbb{R}^m$ through fully connected layers from a feature corresponding to the ith point. Here, the superscript of $\boldsymbol{x}_i$ represents the level. Next, binary occupancies are estimated from $\boldsymbol{x}_i^{(1)}$. In the case where its jth element is "1," it is transformed into a spatial index vector $\phi(j)$ that represents the relative position of the occupied voxels. Then, it concatenates $\phi(j)$ with the feature and predicts m-dimensional occupancies through another fully connected layer. Depending on the number of "1" in $\boldsymbol{x}_i^{(1)}$, multiple occupancies can be predicted. Unpredicted occupancies are considered zero vectors. Finally, these are concatenated and regarded as occupancies $\boldsymbol{x}_i^{(2)}$. The volume can be reconstructed from $\boldsymbol{x}_i^{(1)}$ and $\boldsymbol{x}_i^{(2)}$ such as octree decoding. For larger k, the predictor makes predictions in a similar hierarchical fashion.

When training the model, the spatial index vector $\phi(j)$ is transformed from the ground truth, not from the estimation result. This ensures that the hierarchical structure is always correctly reconstructed during training. On the other hand, the hierarchical structure may be incorrectly reconstructed during testing since the estimation result is used.

E. Loss Function

The network loss function is the sum of the weighted binary cross entropy between predicted occupancies and the ground truth. While a binary cross entropy was considered in a previous work [46] for dense point clouds, we introduce a weight parameter to account for occupancy label imbalance in sparse point clouds. A sigmoid function $\sigma(\cdot)$ is applied to the predicted values to stabilize training. Thus, the loss function is defined as follows:

$$\mathcal{L}^{(k)} = -\frac{1}{nm}\sum_{i=1}^{n}\sum_{j=1}^{m}\Big\{\lambda y_{i,j}^{(k)} \log\Big(\sigma\left(x_{i,j}^{(k)}\right)\Big) + \left(1 - y_{i,j}^{(k)}\right)\log\left(1 - \sigma\left(x_{i,j}^{(k)}\right)\right)\Big\} \quad (5)$$

where λ is a weight parameter for adjusting the balance between "0" and "1," n is the number of points, and m is the number of occupancies. Superscripts represent the level.

When using the hierarchical predictor described in Section III-D, a model consisting of a single U-shaped network and k fully connected layers is trained. In this case, the weighted binary cross entropy at each level is accumulated over k levels as follows:

$$\mathcal{L}_{\mathrm{mul}}^{(k)} = \sum_{i=1}^{k}\mathcal{L}^{(i)}. \quad (6)$$

The ground truths corresponding to the unpredicted occupancies are set to zero vectors.

IV. Experiments

In this section, we experimentally demonstrate that the proposed method reduces distortion derived from voxelization in point cloud geometry compression. We first construct point clouds voxelized in various resolutions by encoding and decoding the original point clouds using a geometry compression method. Then, we super-resolve these decoded voxelized point clouds by postprocessing to reduce the distortion without increasing the bitrate. To construct the decoded voxelized point clouds, we use G-PCC in the latest version (TMC13 v14.0) [4], [49] with an octree coding configuration. It is suitable for our experiments since it can naturally decode point clouds voxelized at various resolutions by octree pruning. As the proposed method is applicable to any decoded voxelized point cloud, other geometry compression methods, such as learning-based methods that perform downsampling on the initial voxelized point clouds, may be used.

A. Experimental Setup

1) Datasets: We use the SemanticKITTI [50], Ford [51], QNX [52], and ScanNet [53] datasets to evaluate the proposed method in outdoor and indoor environments. Fig. 5 shows the example point clouds from the datasets.

The SemanticKITTI dataset is a large-scale point cloud dataset acquired by a LiDAR sensor in outdoor environments such as urban areas, residential areas, and highways. The LiDAR sensor is a Velodyne HDL-64E, which can acquire point cloud data within a 360° circular area with a radius of approximately 100 m with the sensor at the center. The dataset consists of 43 552 point clouds collected from 22 sequences (00-21) of LiDAR data acquired along different vehicle trajectories in city areas. We use sequences from 00 to 10 except 08 as a training set, sequence 08 as a validating set, and sequences from 11 to 21 as a testing set. These sets contain 19 130/4071/20 351 point clouds, respectively.

The Ford and QNX datasets are LiDAR point cloud sequences acquired in outdoor environments used in MPEG. The Ford dataset consists of three sequences using the Velodyne HDL-64E LiDAR sensor, namely "ford_01_q1mm," "ford_02_q1mm," and "ford_03_q1mm." Each sequence contains 1500 point clouds. We use the first and second sequences for training, and the third sequence is split into two and used for validating/testing.

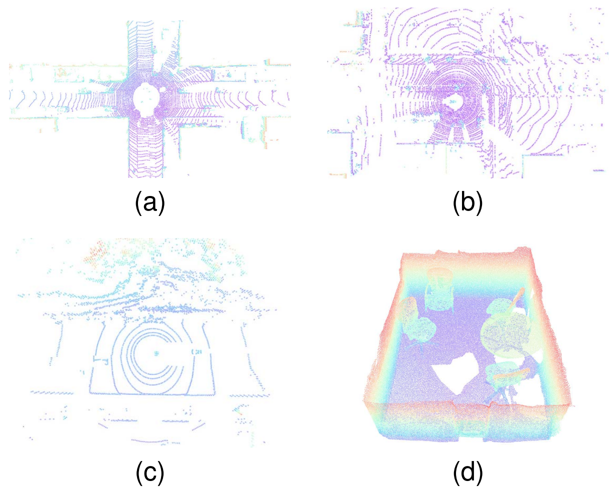


Fig. 5. Examples of point clouds from the datasets. The color of the points corresponds to the vertical height. (a) SemanticKITTI. (b) Ford. (c) QNX. (d) ScanNet.

The QNX dataset consists of four sequences using the Velodyne VLP-16 LiDAR sensor, namely, "qnxadas-junction-approach," "qnxadas-junction-exit," "qnxadas -motorway-join," and "qnxadas-navigating-bends," and these sequences contain 74/74/811/300 point clouds, respectively. We use the first/second sequences for validating/testing and the others for training.

The ScanNet dataset is a large-scale 3-D dataset acquired by a handheld RGB-D sensor in indoor environments such as offices, apartments, and kitchens. It contains 1513 mesh data reconstructed from the RGB-D sequences acquired in those environments. In our experiment, we spatially sampled 100k points with Poisson disk sampling [54] from each mesh data. We also scaled the points to fit into a cube with a side of 100 m to align with the scale of the SemanticKITTI dataset. We split the data into 1045/156/312 point clouds for training/validating/testing, respectively, according to the index list defined in the dataset.

The number of points consisting of each point cloud in the SemanticKITTI, Ford, QNX, and ScanNet datasets is approximately 120k, 80k, 30k, and 100k, respectively. The density of these point clouds is sparse since they are acquired by scanning with LiDAR sensors in the former three datasets and constructed by sampling in the ScanNet dataset. The point clouds in the former three datasets representing the outdoor environment contain objects such as roads, buildings, and vegetation, while the point clouds in the ScanNet dataset representing the indoor environment contain objects such as floors, walls, and chairs.

2) Evaluation Metrics: We use bit per point (bpp) to measure the compression ratio of the encoder. In our experiments, the bpp represents the average number of bits per original point. We use D1 PSNR [55] to evaluate the distortion between the original point cloud and the reconstructed point cloud. We use Bjøntegaard delta bitrate (BD-BR) and Bjøntegaard delta PSNR (BD-PSNR) [56] to evaluate the distortion across various bpp values compared to G-PCC. The G-PCC only performs encoding and decoding of the original point cloud and does not perform any postprocessing corrections. The BD-BR is measured as an average bpp difference in percentage over the whole range of PSNR. The BD-PSNR is measured as an average PSNR difference in decibels over the whole range of bpp. We also evaluate the prediction performance of occupancy in voxel representations. Let $\tilde{\mathcal{P}}$ and $\tilde{\mathcal{P}}^{\mathrm{sr}}$ denote the voxelized point cloud with voxel sizes δ and δ^{sr}, respectively. We assume that $\hat{\mathcal{P}}^{\mathrm{sr}}$ is generated from $\tilde{\mathcal{P}}$ by postprocessing. We construct voxels $\hat{\mathcal{V}}^{\mathrm{sr}}$ from $\hat{\mathcal{P}}^{\mathrm{sr}}$. At this time, its ground truth is represented as $\tilde{\mathcal{V}}^{\mathrm{sr}}$. In order to evaluate the prediction performance, we use intersection over union (IoU) that is represented as follows:

$$\mathrm{IoU} = \frac{\Sigma_i \mathbb{1}[\tilde{\mathcal{V}}^{\mathrm{sr}}(i)\hat{\mathcal{V}}^{\mathrm{sr}}(i) > 0]}{\Sigma_i \mathbb{1}[\tilde{\mathcal{V}}^{\mathrm{sr}}(i) + \hat{\mathcal{V}}^{\mathrm{sr}}(i) > 0]} \tag{7}$$

where i is a voxel index and $\mathbb{1}$ is an indicator function. The IoU is a measure of the degree of overlap between $\tilde{\mathcal{V}}^{\mathrm{sr}}$ and $\hat{\mathcal{V}}^{\mathrm{sr}}$. A higher IoU indicates a more precise prediction of the high-resolution point cloud.

TABLE I
SUMMARY OF VOXELIZATION SETTINGS. λ REPRESENTS THE WEIGHT PARAMETER IN THE LOSS FUNCTION

Octree level		8	9	10	11	12	13
Voxel size [mm]		1024	512	256	128	64	32
λ	SemanticKITTI	2.24	2.48	2.97	3.80	4.88	6.02
	Ford	2.46	2.88	3.57	4.78	6.09	6.77
	QNX	3.21	3.92	4.70	5.26	5.46	5.72
	ScanNet	1.57	2.91	5.16	6.71	6.98	6.99

3) Implementation Details: We voxelize the point clouds using voxel size 1 mm as preprocessing for coding. We set the edge length of the bounding cube for constructing the octree to $L = 2^{18}$ mm to ensure that it encompasses the original point cloud. Then, we construct point clouds further voxelized at six levels corresponding to voxel sizes from 1024 to 32 mm. Table I summarizes the voxelization settings. In this table, the first row shows the level l to truncate the octree. The second row shows the voxel size δ corresponding to each level. We encode and decode the original point cloud using G-PCC at each level to construct the voxelized point cloud. We perform the postprocessing on these point clouds. We set the voxel division in the proposed method to halve the voxel size along each axis, as in the octree representation, that is, we set $m = 8$ and $\delta^{\mathrm{sr}} = \delta/2$. We also construct bit strings representing voxel occupancies from the point cloud voxelized with δ^{sr} and assign them as the ground-truth label of the point cloud voxelized with δ. Here, manual labeling is not required. We use the ratio of the number of "0" and "1" in these labels in the training set as the coefficient λ in (5). The third and subsequent rows of Table I show the coefficients λ used for each dataset. We implemented the proposed method using PyTorch library [57]. To train our model, we used the Adam optimizer [58] with a learning rate of 10^{-3}. The cosine annealing learning rate strategy is used to decay the learning rate. We train each model for 100 epochs and select a model that achieves the best IoU on the validating set. Training is

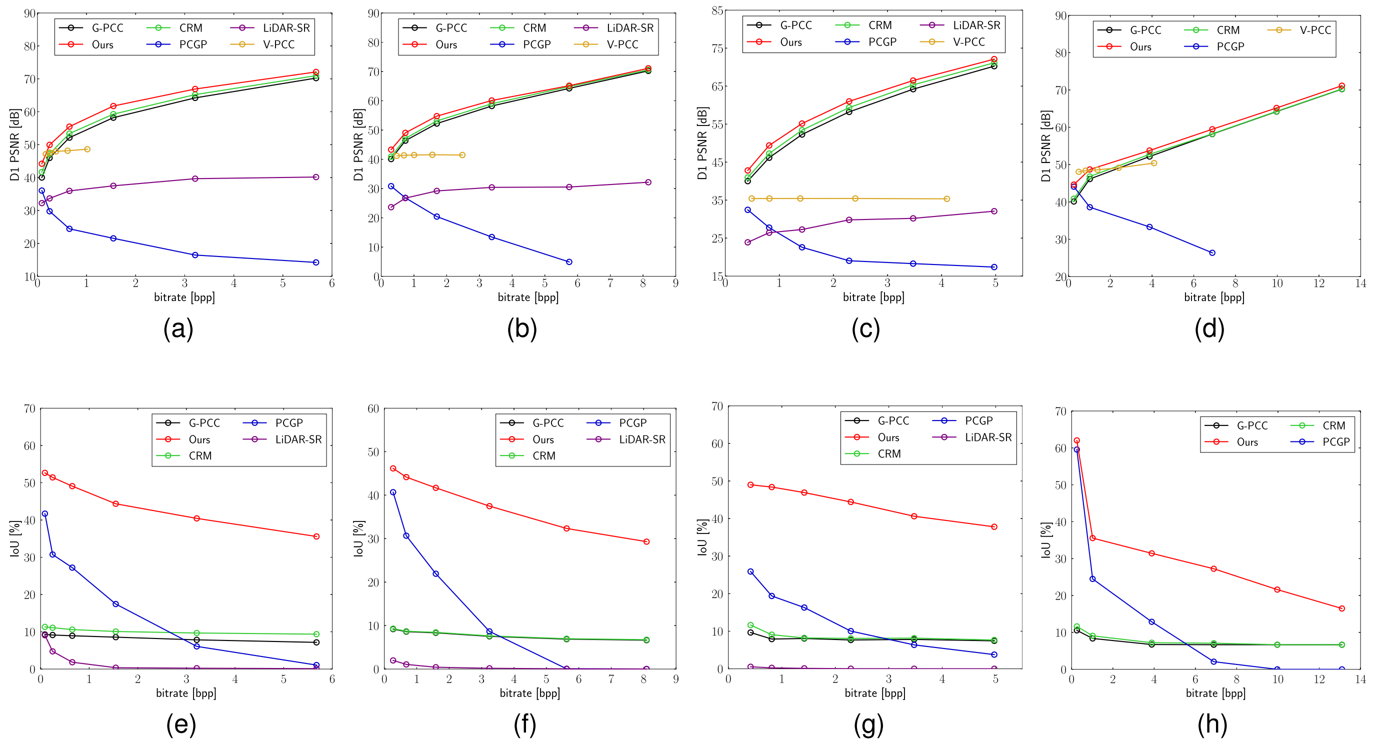


Fig. 6. Quantitative comparison in D1 PSNR and IoU as a function of bitrate. (a) PSNR-bitrate on SemanticKITTI. (b) PSNR-bitrate on Ford. (c) PSNR-bitrate on QNX. (d) PSNR-bitrate on ScanNet. (e) IoU-bitrate on SemanticKITTI. (f) IoU-bitrate on Ford. (g) IoU-bitrate on QNX. (h) IoU-bitrate on ScanNet.

performed for each combination of voxel size and number of level increases k. For example, if we set k to three values (i.e., $k = 1$, 2, and 3), a total of $3 \times 6 = 18$ models are trained for the six sizes listed in Table I. The fixed threshold in (4) is set to $\alpha = 0.5$. All experiments are conducted on a computer equipped with a NVIDIA Quadro GV100 GPU, Intel Core i9-9900X CPU (3.60 GHz), and 16 GB of RAM.

B. Comparison to Baseline Methods

1) Baseline Methods: We compare the proposed method with the conventional point cloud correction methods, namely, CRM [7], PCGP [46], and LiDAR-SR [17].

For CRM, we set the size of local voxel representation to 9^3 voxels as in [7] for all voxel sizes δ. In the construction of the ground truth, we calculated the offset toward the original point, rather than the point voxelized with δ^{sr}. When multiple points exist in a voxel, an offset toward their center is set as the ground truth.

PCGP first divides the point cloud into local patches of 128^3 voxels as in [46]. Then, the occupancies of high-resolution voxels are predicted for each patch, and all patches are aggregated to construct a super-resolved point cloud. We used a threshold of 0.5 to estimate binary occupancies from the predicted continuous occupancies.

For LiDAR-SR, we project the point cloud voxelized with δ and δ^{sr} onto the range images. We modified the method to predict the range image corresponding to δ^{sr} from the range image corresponding to δ. Therefore, the resolution of both the input and output range images to the neural network is set to 64×1024. We evaluated the performance of LiDAR-SR only on the SemanticKITTI dataset since it is designed for point clouds acquired by a LiDAR sensor.

2) D1 PSNR: The top row of Fig. 6 summarizes the D1 PSNR values corresponding to the bitrate. In this figure, the smaller the voxel size, the higher the bitrate. The proposed method, PCGP, and LiDAR-SR are set to predict the point cloud voxelized with a voxel size obtained by dividing the voxel of G-PCC once (i.e., $k = 1$). Thus, the point clouds predicted by these methods at a given bitrate have the same resolution as the point clouds corresponding to one bitrate higher in G-PCC.

At all bitrates, the proposed method improves D1 PSNR compared to G-PCC. This indicates that our postprocessing reduces the distortion resulting from the point cloud compression. Although CRM also outperforms G-PCC at all bitrates, the improvement is slight. It is difficult for CRM to deal with the case where multiple points exist in a voxel since it corrects the coordinates with an offset for each point. PCGP tends to predict high-resolution voxels as unoccupied since the higher the level, the sparser the points in the local patches. In the case where all high-resolution voxels are predicted to be unoccupied, no points are reconstructed in those voxel spaces. Therefore, sparser point clouds are reconstructed compared to G-PCC and performance degrades as the voxel size decreases. In some voxel sizes, the D1 PSNR cannot be measured since there are no cases that are predicted to be occupied at all points, with the result that the data points in the figures are missing. LiDAR-SR significantly degrades the performance compared to G-PCC. It reconstructs high-resolution range images according to the encoder–decoder architecture,

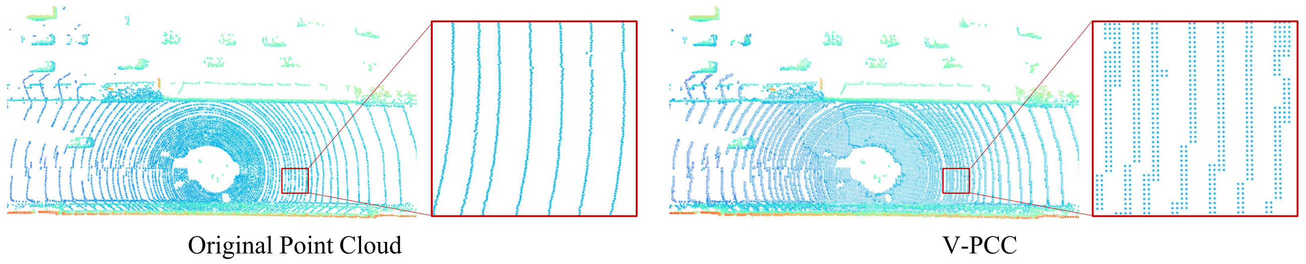


Fig. 7. Comparison of point clouds reconstructed by V-PCC with the original point clouds. The color of the points corresponds to the vertical height.

resulting in reconstruction errors. Range images generated from voxelized point clouds tend to have more missing data compared to those from raw point clouds acquired from sensors, resulting in larger restoration errors. These reconstruction errors cause greater distortion than those due to voxelization.

3) V-PCC Performance: We also provide a PSNR-bitrate comparison to V-PCC [4], a compression method that encodes images on which points are projected. We use the latest version software (TMC2 v18.0) [59] and its parameters. As these are originally designed for dense point clouds, the projection of extensive and sparse point clouds used in the experiments requires larger images, which requires a significantly larger memory and computational effort for coding. For reasonable computational efficiency with small images, we rescale the point clouds by a factor of 0.01 before encoding and reconstruct the scale after decoding.

V-PCC may achieve the best D1 PSNR at low bitrates, while D1 PSNR hardly improves with increasing bitrate. As shown in Fig. 7, V-PCC reconstructs coarse grids corresponding to the pixel values since the points are encoded after being projected onto the image. In this experiment, these grids are reconstructed even with the highest bitrate setting, limiting the improvement in D1 PSNR as a result. As shown in Fig. 5, the point clouds used in this experiment are all sparse, which is very different from the dense point clouds used in MPEG [4]. This accounts for the poor V-PCC performance.

4) IoU: The bottom row of Fig. 6 summarizes the IoU values corresponding to the bitrate. The smaller the voxel size, the higher the bitrate. Here, the IoU of G-PCC is measured between the ground-truth voxels $\tilde{\mathcal{V}}$ and $\tilde{\mathcal{V}}^{\mathrm{sr}}$ instead of the predicted voxels. This is a reference value to show the degree to which the IoU degrades as the voxel size decreases.

As can be seen, the proposed method achieves the best IoU at all bitrates. This shows the effectiveness of the reconstruction of high-resolution voxels. IoU represents the occupancy prediction accuracy for voxels with higher resolution than that used for voxelization. The smaller the voxel size, the more difficult it is to predict accurately since the point cloud becomes more detailed. Therefore, the IoU tends to be lower at higher bitrates. CRM has a significantly lower IoU than the proposed method since it does not improve the resolution of the point cloud. The IoU of PCGP decreases as the bitrate increases. This is because the predicted value of occupancy decreases as the sparsity of the point cloud increases, and most voxels are estimated to be unoccupied. LiDAR-SR cannot accurately predict voxel occupancy due to the image reconstruction errors.

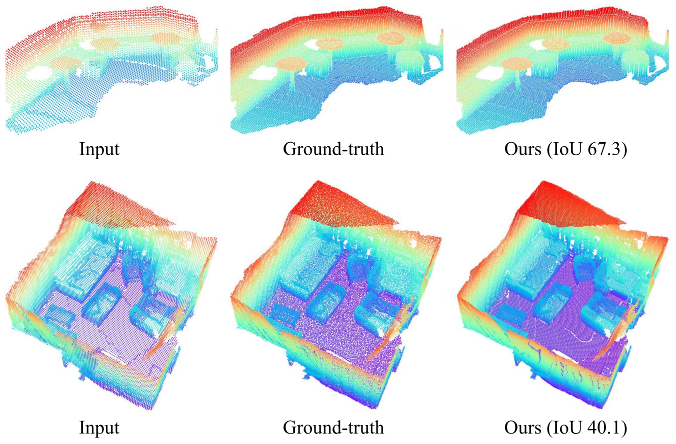


Fig. 8. Examples of the prediction results of the proposed method on the ScanNet dataset. The color of the points corresponds to the vertical height.

In ScanNet, the IoU is decreased more significantly for smaller voxel size compared to other datasets. This is due to the fact that we construct the point clouds from ScanNet by spatial sampling. For these point clouds, it is more difficult to predict the occupancy of the high-resolution voxels since they have higher irregularities than the point clouds acquired by the LiDAR sensor. However, the proposed method is able to reduce the distortion even for these point clouds.

Fig. 8 shows the examples of the prediction results of the proposed method on the ScanNet dataset. The left column represents the input to the postprocessing, and the middle column represents the ground truth, namely, the point cloud voxelized with δ^{sr}. The right column represents the prediction results of the proposed method, together with the IoU. The example in the top row is a relatively dense point, making it easy to predict and resulting in a high IoU. On the other hand, the example in the bottom row is more difficult to predict due to the large spacing of the sampled points, resulting in a low IoU.

5) BD-BR and BD-PSNR: Table II shows BD-BR and BD-PSNR for all methods. The proposed method reduces the bitrate by 20.62%–38.41% and improves the D1 PSNR by 2.25–3.42 dB on average, compared to G-PCC. In CRM, these values are 6.48%–15.08% and 0.53–1.17 dB. It can be confirmed that PCGP and LiDAR-SR degraded performance in both metrics.

TABLE II
PERFORMANCE COMPARISON IN BD-BR (%) AND BD-PSNR (dB) WITH G-PCC AS REFERENCE

Method	SemanticKITTI		Ford		QNX		ScanNet	
	BD-BR	BD-PSNR	BD-BR	BD-PSNR	BD-BR	BD-PSNR	BD-BR	BD-PSNR
Ours	**-38.41%**	**3.42** dB	**-22.01%**	**2.25** dB	**-20.62%**	**2.78** dB	**-23.16%**	**2.28** dB
CRM	-15.08%	1.17 dB	-9.33%	0.86 dB	-8.56%	1.08 dB	-6.48%	0.53 dB
PCGP	1117.30%	-20.85 dB	1187.19%	-30.37 dB	436.58%	-29.80 dB	5220.56%	-10.44 dB
LiDAR-SR	5276.39%	-17.32 dB	8542.17%	-24.28 dB	2480.27%	-25.40 dB	-	-

TABLE III
POSTPROCESSING RUNTIME (SECONDS). (a) SEMANTICKITTI. (b) FORD. (c) QNX. (d) SCANNET

(a)

Voxel size [mm]	1024	512	256	128	64	32	mean
Ours	0.025	**0.028**	**0.036**	**0.051**	**0.078**	**0.111**	**0.055**
CRM	**0.022**	0.039	0.095	0.205	0.366	0.533	0.210
PCGP	0.218	0.573	1.531	5.541	16.625	57.750	13.706
LiDAR-SR	0.361	0.362	0.364	0.380	0.387	0.400	0.376

(b)

Voxel size [mm]	1024	512	256	128	64	32	mean
Ours	**0.036**	**0.040**	**0.052**	**0.069**	**0.090**	**0.101**	**0.065**
CRM	0.044	0.105	0.202	0.334	0.441	0.481	0.268
PCGP	0.264	0.855	2.643	8.682	25.101	69.520	17.845
LiDAR-SR	0.375	0.381	0.382	0.386	0.391	0.397	0.385

(c)

Voxel size [mm]	1024	512	256	128	64	32	mean
Ours	0.033	0.034	**0.035**	**0.035**	**0.037**	**0.040**	**0.036**
CRM	**0.019**	**0.027**	0.037	0.050	0.067	0.089	0.048
PCGP	0.289	0.706	2.423	5.727	13.814	30.325	8.881
LiDAR-SR	0.367	0.371	0.378	0.384	0.388	0.389	0.380

(d)

Voxel size [mm]	1024	512	256	128	64	32	mean
Ours	**0.039**	**0.062**	**0.082**	**0.099**	**0.117**	**0.119**	**0.087**
CRM	0.088	0.282	0.408	0.429	0.447	0.448	0.350
PCGP	0.087	0.412	1.417	5.108	20.022	77.674	17.453

TABLE IV
NUMBER OF MODEL PARAMETERS

Method	Ours	CRM	PCGP	LiDAR-SR
#Params	8.87M	1.22M	25.60M	34.53M

6) Runtime: Table III summarizes the runtime [seconds] of the postprocessing step for each voxel size. The encode and decode runtimes are not included in this table since this article does not focus on them. Table IV shows a comparison of the number of model parameters. The number of models is the same among all methods since we train a model at each voxel size. The proposed method achieves the fastest runtime compared to the other methods in most cases. This is due to the efficient convolutional operations using sparse convolution. In general, the smaller the voxel size, the greater the number of points to be processed. Thus, the runtime tends to increase as the voxel size decreases. However, the runtime of LiDAR-SR is almost independent of the voxel size since it projects all points to a range image. While range image processing is efficient, LiDAR-SR performs multiple feedforward passes according to Monte Carlo dropout [60]. Although CRM has fewer model parameters than the proposed method, the runtime of CRM tends to be longer since it is based on standard convolution. In PCGP, the smaller the voxel size, the larger the number of local patches obtained by dividing the point cloud. Therefore, smaller voxel sizes require longer runtimes.

7) Qualitative Evaluation: Fig. 9 visualizes the point cloud after postprocessing with each method. The input to the postprocessing step and its ground truth are decoded voxelized point clouds with 1024 and 512 mm as voxel sizes, respectively. Each point is colored according to the geometry error, which is the distance to the nearest neighbor point in the original point cloud. The color bar represents the correspondence between the color and the error. Note that there are errors between the ground truth and the original point cloud since the ground truth is also a kind of decoded point cloud.

For the input point cloud, large errors occur throughout the shape due to the large voxel size. In contrast, the errors are reduced in the ground truth with smaller voxel size. The proposed method generates the most faithful point cloud on all datasets. Although CRM refines the position of the points, it does not reconstruct merged points through the voxelization. PCGP sometimes predicts all high-resolution voxels as unoccupied when the points are sparse, producing mostly empty spaces. LiDAR-SR causes larger distortions than the input point cloud due to reconstruction errors of the range image.

C. Impact of Increase in the Number of Levels

We explore the applicability of the proposed method to multilevel occupancy prediction. Here, we evaluate two types of methods: linear and hierarchical. The linear method predicts all the occupancies of a volume from pointwise features through a linear layer, that is, a voxel used for voxelization is divided into m^k voxels, and all of their m^k occupancies are predicted. The hierarchical method predicts the occupancies of the local volume obtained by dividing a voxel in the current level when it is occupied, as described in Section III-D. In contrast to the linear method, it does not necessarily predict

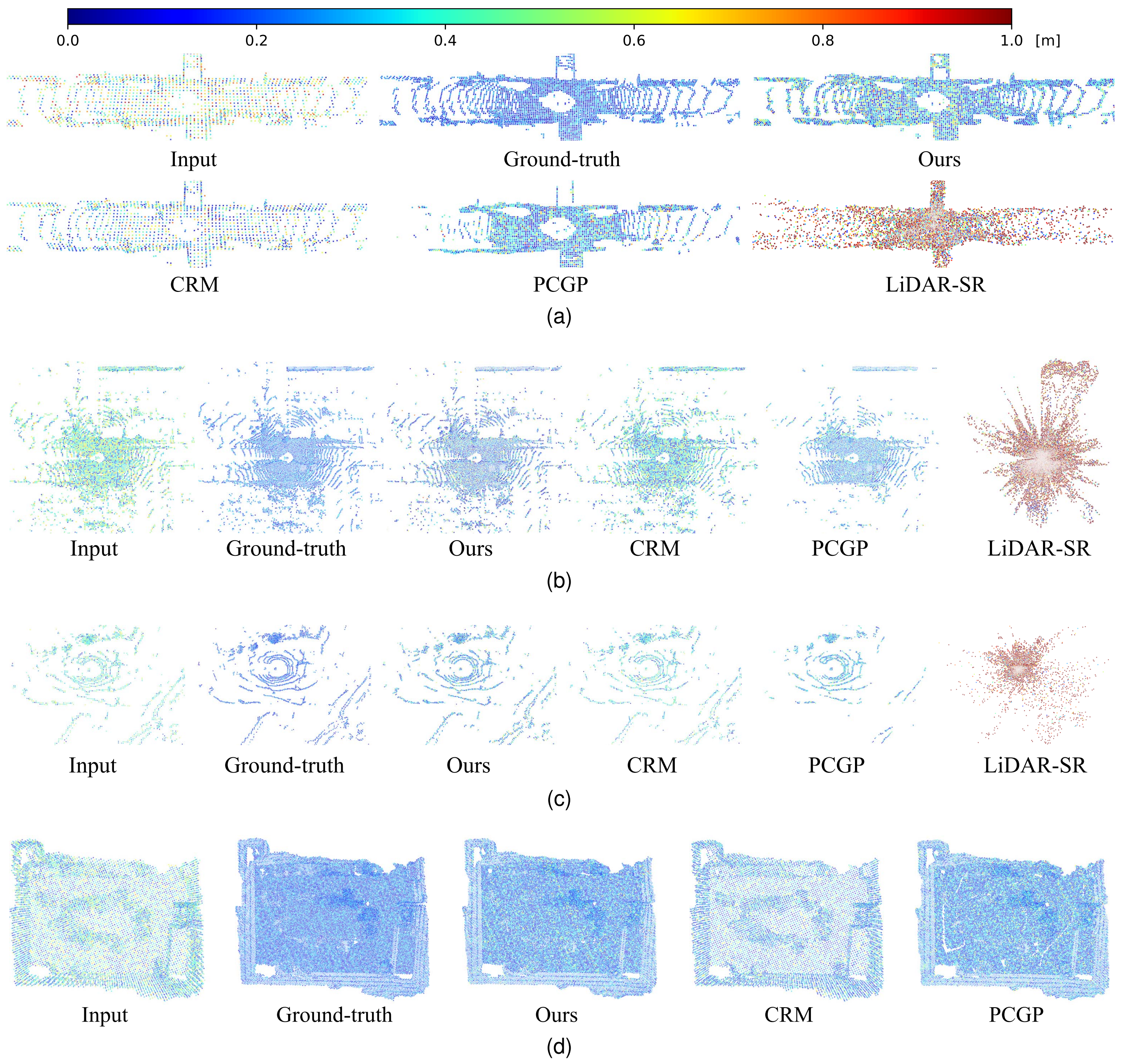


Fig. 9. Qualitative comparison visualizing geometry errors of the original point cloud. The color bar represents the correspondence between the color and the error. The ground truth for the super-resolution is a point cloud voxelized at one level higher than the input point cloud. (a) SemanticKITTI. (b) Ford. (c) QNX. (d) ScanNet.

the occupancy of all kth level voxels. We fix the voxel size to voxelize the input point cloud to 1024 mm and perform super-resolution with the number of level increases $k = 1$, 2, and 3.

Fig. 10 shows the D1 PSNR and IoU as functions of the number of level increases k, comparing these two methods. These methods are identical in the case of $k = 1$ since it represents single-level occupancy prediction. As can be seen, both methods show improvement in the D1 PSNR when $k > 1$ compared to the case where $k = 1$. This suggests that the proposed method may be applicable to multilevel occupancy prediction. However, it can be observed that the performances are hardly improved or slightly worse for $k = 3$ based on the case of $k = 2$. This is due to the significant decrease in IoU with increasing k, resulting in greater distortion. The higher resolution of voxel than that for voxelization, the more difficult it is to predict occupancies. We can also see that the hierarchical method achieves better performance than the linear method. It is considered that distortion can be reduced by predicting the occupancies for each local volume rather than the whole volume.

Fig. 11 shows a visual representation of the results of both methods in the case of $k = 2$. The points are colored in the same way as Fig. 9, and the areas of the red box are

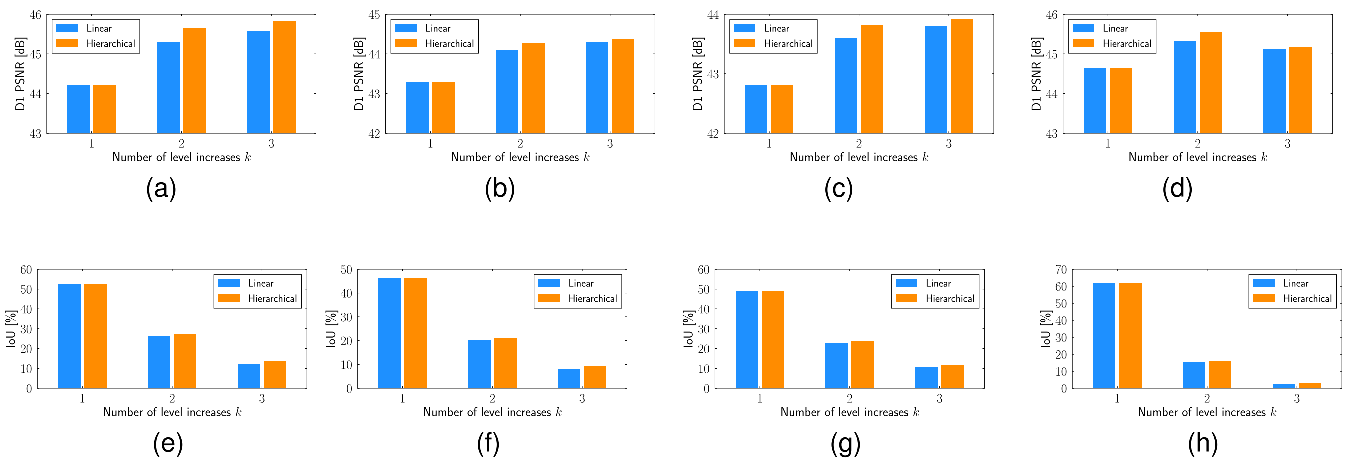


Fig. 10. Quantitative comparison of two types of multilevel occupancy predictors for each number of level increases *k*. (a) PSNR-*k* on SemanticKITTI. (b) PSNR-*k* on Ford. (c) PSNR-*k* on QNX. (d) PSNR-*k* on ScanNet. (e) IoU-*k* on SemanticKITTI. (f) IoU-*k* on Ford. (g) IoU-*k* on QNX. (h) IoU-*k* on ScanNet.

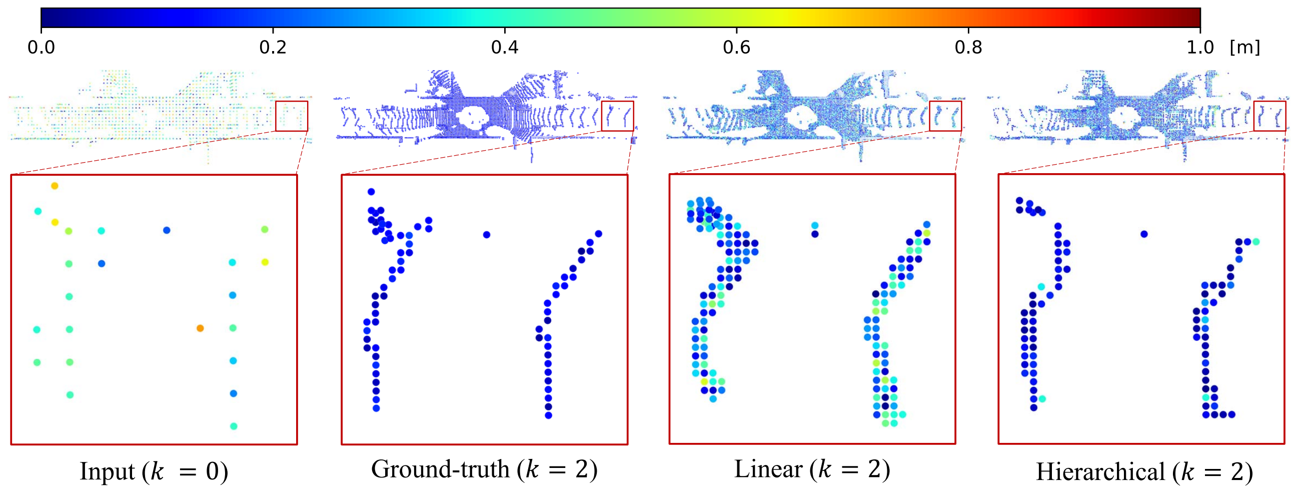


Fig. 11. Visualization of point clouds reconstructed by two types of multilevel occupancy predictors. The color bar represents the correspondence between the color and the error. The ground truth for the super-resolution is a point cloud voxelized at two higher levels than the input point cloud. The areas of the red box are enlarged to compare the detailed shapes.

enlarged. The input point cloud corresponds to $k = 0$. The linear method tends to reconstruct dilated point clouds since it predicts the occupancy of the whole volume. If most of the occupancy in the volume is predicted to be occupied, the detailed shape cannot be reconstructed. On the other hand, the hierarchical method may be able to reconstruct detailed shapes in some cases since it predicts occupancies for each local volume. It can suppress major distortion across the whole volume, even when accurate prediction is difficult. However, it is generally challenging to reconstruct detailed shapes when k is large. Therefore, large geometry errors are often observed in both methods.

We also evaluate the effect of super-resolution on the point clouds voxelized with each voxel size from 1024 to 32 mm, with the number of level increases $k = 1, 2$, and 3. Here, we always use the hierarchical method. Fig. 12 shows a PSNR performance comparison over different k settings. A smaller voxel size corresponds to a higher bitrate in this figure. In all datasets, it can be observed that the performance is higher for $k = 2$ and 3 than $k = 1$ when the voxel size is large. This suggests the effectiveness of multilevel occupancy prediction. On the other hand, as voxel size becomes smaller, their performance becomes equal to or less than when $k = 1$. This is because point clouds voxelized with smaller voxel have higher sparsity, making it more difficult to predict the occupancies corresponding to the detailed shape. For a similar reason, when $k = 3$, there is little difference compared to $k = 2$. However, we can confirm that all k settings achieve better performance than the G-PCC.

D. Ablation Study

We verify the impact of the design choices made in the proposed method by performing an ablation study. Table V shows the performance of the proposed method that ablated the DT in the threshold determination of (4) and the weight (WT) in the loss function of (5). The performance metrics are BD-BR and BD-PSNR. The symbols "✓" and "✗" indicate whether or not to use the setting, respectively. Not using DT means that a fixed threshold is always used. If WT is not used,

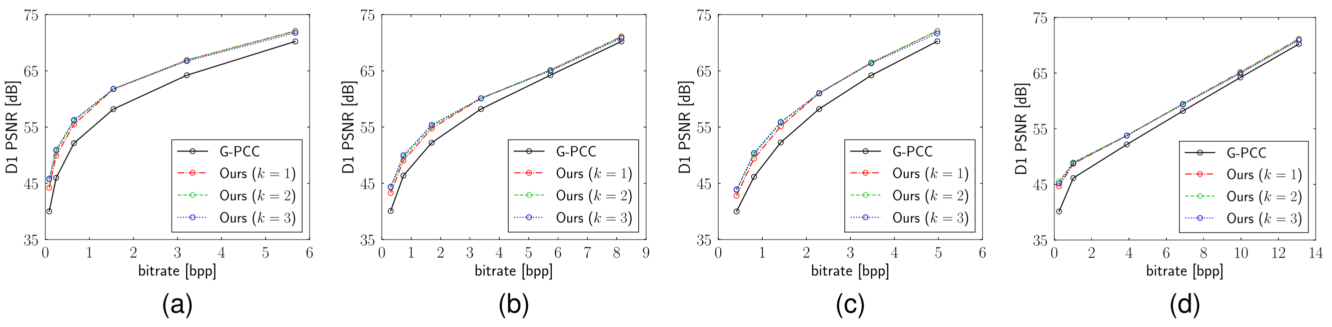


Fig. 12. PSNR performance comparison for multilevel occupancy prediction. *k* represents the number of level increases from the voxelization level of the point cloud. (a) SemanticKITTI. (b) Ford. (c) QNX. (d) ScanNet.

TABLE V
Ablation Study. DT and WT Represent the Use of DT and Weight, Respectively

Setting		SemanticKITTI		Ford		QNX		ScanNet	
DT	WT	BD-BR	BD-PSNR	BD-BR	BD-PSNR	BD-BR	BD-PSNR	BD-BR	BD-PSNR
✓	✓	**-38.41**%	**3.42** dB	**-22.01**%	**2.25** dB	**-20.63**%	**2.78** dB	**-23.16**%	**2.28** dB
✓	✗	-37.01%	3.28 dB	-17.39%	1.75 dB	-18.07%	2.42 dB	-21.17%	2.08 dB
✗	✓	88.18%	-6.06 dB	711.50%	-11.71 dB	313.56%	-15.49 dB	108.76%	-1.17 dB
✗	✗	1860.99%	-20.77 dB	2947.65%	-24.06 dB	835.60%	-24.93 dB	7065.43%	-18.81 dB

the weight parameter is set to $\lambda = 1$. We set the number of level increases to $k = 1$ in this experiment.

The first column that uses both settings represents the full proposed method. By comparing with or without DT, it can be seen that DT improves the performance significantly. If all the predicted values of occupancy are below the threshold, a region with missing points is generated. DT has the effect of suppressing distortion by preventing the generation of such regions. We can also see that the performance is restored by DT even when WT is not used. On the other hand, the performance is degraded the most if neither DT nor WT is used. It can be considered that training of the model is more difficult in the case without WT due to the imbalance between "1" and "0" in the occupancies constructed from the point cloud. In contrast, the best performance is obtained by using both settings.

V. Conclusion

In this article, we proposed a point cloud super-resolution method to reduce distortion caused by voxelization in geometry compression. The proposed method improves the fidelity of the point cloud by increasing the resolution of the voxelized point cloud as a postprocessing step after encoding and decoding. This reduces the bitrate required to achieve the same fidelity as without postprocessing. To achieve efficient super-resolution for large point clouds such as those acquired by LiDAR sensors, we predicted pointwise occupancies using a deep neural network based on sparse convolution. The network supports both single-level and multilevel occupancy prediction. The proposed method estimates binary occupancies from the output of this network while preventing the generation of regions with missing points using dynamic thresholding. Evaluation experiments on the outdoor and indoor datasets demonstrated that the proposed method is effective and efficient.

One limitation of the proposed method is that it assumes a voxel grid representation. From this, the super-resolved point cloud also has distortion due to the discrete representation of coordinates. Even if the original point cloud is voxelized, distortion occurs when the resolution of the super-resolved point cloud is lower than that of the original point cloud. We would like to explore ways to further improve the compression performance by removing this assumption. Another limitation is that our model represents only trained data. Performance may be degraded for data with a different resolution from the trained data, such as point clouds voxelized with different voxel sizes. While we found that our model was able to represent actual data distributions, reconstructing detailed shapes remains a challenge, especially in multilevel occupancy prediction.

We describe several research directions that could improve the performance of the proposed method. First, the use of temporal information for dynamic point clouds is promising. Although only spatial information was used in this article, the proposed method may be extended using the previous frame in a temporal sequence when it is available. Second, the application of neural implicit representation is expected. While deep neural networks based on sparse convolution play an important role in improving efficiency, neural implicit representation has the potential to provide more faithful 3-D shape reconstruction. Recent improvements [61] show that it is capable of representing general shapes. Finally, the compensation of coding artifacts is considered when using lossy geometry compression schemes. The quality of the super-resolved point clouds may be improved by performing the proposed method after compensating for coding artifacts in the decoded point clouds.

References

[1] D. Meagher, "Geometric modeling using octree encoding," *Comput. Graph. image Process.*, vol. 19, no. 2, pp. 129–147, Jun. 1982.

[2] T. Golla and R. Klein, "Real-time point cloud compression," in *Proc. IEEE/RSJ Int. Conf. Intell. Robots Syst. (IROS)*, Sep. 2015, pp. 5087–5092.

[3] D. C. Garcia, T. A. Fonseca, R. U. Ferreira, and R. L. de Queiroz, "Geometry coding for dynamic voxelized point clouds using octrees and multiple contexts," *IEEE Trans. Image Process.*, vol. 29, pp. 313–322, 2019.

[4] D. Graziosi, O. Nakagami, S. Kuma, A. Zaghetto, T. Suzuki, and A. Tabatabai, "An overview of ongoing point cloud compression standardization activities: Video-based (V-PCC) and geometry-based (G-PCC)," *APSIPA Trans. Signal Inf. Process.*, vol. 9, no. 1, pp. 1–17, 2020.

[5] L. Huang, S. Wang, K. Wong, J. Liu, and R. Urtasun, "OctSqueeze: Octree-structured entropy model for LiDAR compression," in *Proc. IEEE/CVF Conf. Comput. Vis. Pattern Recognit. (CVPR)*, Jun. 2020, pp. 1313–1323.

[6] S. Biswas, J. Liu, K. Wong, S. Wang, and R. Urtasun, "MuSCLE: Multi sweep compression of LiDAR using deep entropy models," in *Proc. Adv. Neural Inf. Process. Syst. (NIPS)*, vol. 33, 2020, pp. 22170–22181.

[7] Z. Que, G. Lu, and D. Xu, "VoxelContext-Net: An octree based framework for point cloud compression," in *Proc. IEEE/CVF Conf. Comput. Vis. Pattern Recognit. (CVPR)*, Jun. 2021, pp. 6042–6051.

[8] C. Fu, G. Li, R. Song, W. Gao, and S. Liu, "OctAttention: Octree-based large-scale contexts model for point cloud compression," in *Proc. Assoc. Advancement Artif. Intell. (AAAI)*, 2022, pp. 1–9.

[9] D. Stutz and A. Geiger, "Learning 3D shape completion from laser scan data with weak supervision," in *Proc. IEEE/CVF Conf. Comput. Vis. Pattern Recognit.*, Jun. 2018, pp. 1955–1964.

[10] X. Wang, M. H. Ang, and G. Hee Lee, "Point cloud completion by learning shape priors," in *Proc. IEEE/RSJ Int. Conf. Intell. Robots Syst. (IROS)*, Oct. 2020, pp. 10719–10726.

[11] A. Kulikajevas, R. Maskeliunas, and R. Damasevicius, "Adversarial 3D human pointcloud completion from limited angle depth data," *IEEE Sensors J.*, vol. 21, no. 24, pp. 27757–27765, Dec. 2021.

[12] L. Yu, X. Li, C.-W. Fu, D. Cohen-Or, and P.-A. Heng, "PU-Net: Point cloud upsampling network," in *Proc. IEEE/CVF Conf. Comput. Vis. Pattern Recognit.*, Jun. 2018, pp. 2790–2799.

[13] R. Li, X. Li, C.-W. Fu, D. Cohen-Or, and P.-A. Heng, "PU-GAN: A point cloud upsampling adversarial network," in *Proc. IEEE/CVF Int. Conf. Comput. Vis. (ICCV)*, Oct. 2019, pp. 7203–7212.

[14] S. Ye, D. Chen, S. Han, Z. Wan, and J. Liao, "Meta-PU: An arbitrary-scale upsampling network for point cloud," *IEEE Trans. Vis. Comput. Graphics*, vol. 28, no. 9, pp. 3206–3218, Sep. 2021.

[15] P. H. Casajus, T. Ritschel, and T. Ropinski, "Total denoising: Unsupervised learning of 3D point cloud cleaning," in *Proc. IEEE/CVF Int. Conf. Comput. Vis. (ICCV)*, Oct. 2019, pp. 52–60.

[16] S. Luo and W. Hu, "Score-based point cloud denoising," in *Proc. IEEE/CVF Int. Conf. Comput. Vis. (ICCV)*, Oct. 2021, pp. 4583–4592.

[17] T. Shan, J. Wang, F. Chen, P. Szenher, and B. Englot, "Simulation-based lidar super-resolution for ground vehicles," *Robot. Auto. Syst.*, vol. 134, Dec. 2020, Art. no. 103647.

[18] J. Yue, W. Wen, J. Han, and L.-T. Hsu, "3D point clouds data super resolution-aided LiDAR odometry for vehicular positioning in urban canyons," *IEEE Trans. Veh. Technol.*, vol. 70, no. 5, pp. 4098–4112, May 2021.

[19] C. Cao, M. Preda, and T. Zaharia, "3D point cloud compression: A survey," in *Proc. Int. Conf. 3D Web Technol. (Web3D)*, 2019, pp. 1–9.

[20] R. Mekuria, K. Blom, and P. Cesar, "Design, implementation, and evaluation of a point cloud codec for tele-immersive video," *IEEE Trans. Circuits Syst. Video Technol.*, vol. 27, no. 4, pp. 828–842, Apr. 2016.

[21] R. L. de Queiroz and P. A. Chou, "Motion-compensated compression of dynamic voxelized point clouds," *IEEE Trans. Image Process.*, vol. 26, no. 8, pp. 3886–3895, Aug. 2017.

[22] F. Song, Y. Shao, W. Gao, H. Wang, and T. Li, "Layer-wise geometry aggregation framework for lossless LiDAR point cloud compression," *IEEE Trans. Circuits Syst. Video Technol.*, vol. 31, no. 12, pp. 4603–4616, Dec. 2021.

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

[24] X. Sun, Y. Sun, W. Zuo, S. S. Cheng, and M. Liu, "A novel coding scheme for large-scale point cloud sequences based on clustering and registration," *IEEE Trans. Autom. Sci. Eng.*, vol. 19, no. 3, pp. 2384–2396, Jul. 2021.

[25] J. Ballé, V. Laparra, and E. P. Simoncelli, "End-to-end optimized image compression," in *Proc. Int. Conf. Learn. Represent. (ICLR)*, 2017, pp. 1–27.

[26] J. Ballé, D. Minnen, S. Singh, S. J. Hwang, and N. Johnston, "Variational image compression with a scale hyperprior," in *Proc. Int. Conf. Learn. Represent. (ICLR)*, 2018, pp. 1–23.

[27] C.-Y. Wu, N. Singhal, and P. Krahenbuhl, "Video compression through image interpolation," in *Proc. Eur. Conf. Comput. Vis. (ECCV)*, 2018, pp. 416–431.

[28] G. Lu et al., "Content adaptive and error propagation aware deep video compression," in *Proc. Eur. Conf. Comput. Vis. (ECCV)*. Cham, Switzerland: Springer, 2020, pp. 456–472.

[29] A. Brock, T. Lim, J. M. Ritchie, and N. J. Weston, "Generative and discriminative voxel modeling with convolutional neural networks," in *Proc. Neural Inf. Process. Conf., 3D Deep Learn.*, 2016, pp. 1–9.

[30] M. Quach, G. Valenzise, and F. Dufaux, "Learning convolutional transforms for lossy point cloud geometry compression," in *Proc. IEEE Int. Conf. Image Process. (ICIP)*, Sep. 2019, pp. 4320–4324.

[31] J. Wang, H. Zhu, H. Liu, and Z. Ma, "Lossy point cloud geometry compression via end-to-end learning," *IEEE Trans. Circuits Syst. Video Technol.*, vol. 31, no. 12, pp. 4909–4923, Dec. 2021.

[32] J. Wang, D. Ding, Z. Li, and Z. Ma, "Multiscale point cloud geometry compression," in *Proc. Data Compress. Conf. (DCC)*, Mar. 2021, pp. 73–82.

[33] J. Xu et al., "Point AE-DCGAN: A deep learning model for 3D point cloud lossy geometry compression," in *Proc. Data Compress. Conf. (DCC)*, Mar. 2021, p. 379.

[34] A. F. R. Guarda, N. M. M. Rodrigues, and F. Pereira, "Adaptive deep learning-based point cloud geometry coding," *IEEE J. Sel. Topics Signal Process.*, vol. 15, no. 2, pp. 415–430, Feb. 2020.

[35] A. F. R. Guarda, N. M. M. Rodrigues, and F. Pereira, "Neighborhood adaptive loss function for deep learning-based point cloud coding with implicit and explicit quantization," *IEEE Multimedia Mag.*, vol. 28, no. 3, pp. 107–116, Jul. 2020.

[36] W. Yan, Y. Shao, S. Liu, T. H Li, Z. Li, and G. Li, "Deep AutoEncoder-based lossy geometry compression for point clouds," 2019, *arXiv:1905.03691*.

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

[38] K. Matsuzaki and K. Tasaka, "Binary representation for 3D point cloud compression based on deep auto-encoder," in *Proc. IEEE 8th Global Conf. Consum. Electron. (GCCE)*, Oct. 2019, pp. 489–490.

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

[40] C. Dinesh, G. Cheung, and I. V. Bajic, "3D point cloud super-resolution via graph total variation on surface normals," in *Proc. IEEE Int. Conf. Image Process. (ICIP)*, Sep. 2019, pp. 4390–4394.

[41] C. Dinesh, G. Cheung, and I. V. Bajic, "Super-resolution of 3D color point clouds via fast graph total variation," in *Proc. IEEE Int. Conf. Acoust., Speech Signal Process. (ICASSP)*, May 2020, pp. 1983–1987.

[42] D. C. Garcia, T. A. Fonseca, and R. L. De Queiroz, "Example-based super-resolution for point-cloud video," in *Proc. 25th IEEE Int. Conf. Image Process. (ICIP)*, Oct. 2018, pp. 2959–2963.

[43] T. M. Borges, D. C. Garcia, and R. L. de Queiroz, "Fractional super-resolution of voxelized point clouds," *IEEE Trans. Image Process.*, vol. 31, pp. 1380–1390, 2022.

[44] L. Mescheder, M. Oechsle, M. Niemeyer, S. Nowozin, and A. Geiger, "Occupancy networks: Learning 3D reconstruction in function space," in *Proc. IEEE/CVF Conf. Comput. Vis. Pattern Recognit. (CVPR)*, Jun. 2019, pp. 4460–4470.

[45] S. Peng, M. Niemeyer, L. Mescheder, M. Pollefeys, and A. Geiger, "Convolutional occupancy networks," in *Proc. Eur. Conf. Comput. Vis. (ECCV)*. Cham, Switzerland: Springer, 2020, pp. 523–540.

[46] A. Akhtar, W. Gao, X. Zhang, L. Li, Z. Li, and S. Liu, "Point cloud geometry prediction across spatial scale using deep learning," in *Proc. IEEE Int. Conf. Vis. Commun. Image Process. (VCIP)*, Dec. 2020, pp. 70–73.

[47] C. Choy, J. Gwak, and S. Savarese, "4D spatio–temporal ConvNets: Minkowski convolutional neural networks," in *Proc. IEEE/CVF Conf. Comput. Vis. Pattern Recognit. (CVPR)*, Jun. 2019, pp. 3075–3084.

[48] C. Szegedy, S. Ioffe, V. Vanhoucke, and A. Alemi, "Inception-v4, inception-ResNet and the impact of residual connections on learning," in *Proc. Assoc. Advancement Artif. Intell. (AAAI)*, 2017, pp. 1–7.

[49] MPEG Group. *Geometry Based Point Cloud Compression (G-PCC) Test Model*. Accessed: 2022. [Online]. Available: https://github.com/MPEGGroup/mpeg-pcc-tmc13

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

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

[52] D. Flynn, S. Lasserre, and G. Martin-Cocher, "PCC Cat3 test sequences from BlackBerry|QNX," ISO/IEC JTC1/SC29/WG11, Ljubljana, Slovenia, Tech. Rep. m23647, 2018.

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

[54] M. Corsini, P. Cignoni, and R. Scopigno, "Efficient and flexible sampling with blue noise properties of triangular meshes," *IEEE Trans. Vis. Comput. Graphics*, vol. 18, no. 6, pp. 914–924, Jun. 2012.

[55] *Common Test Conditions for Point Cloud Compression*, document ISO/IEC JTC1/SC29/WG11 Doc. N18474, 3DG Group, 2019.

[56] G. Bjøntegaard, "Calculation of average PSNR differences between RD-curves," ITU-T SG16/Q.6, Austin, TX, USA, Tech. Rep. VCEG-M33, 2001.

[57] A. Paszke et al., "PyTorch: An imperative style, high-performance deep learning library," in *Proc. Adv. Neural Inf. Process. Syst. (NIPS)*, vol. 32, 2019, pp. 1–12.

[58] D. P. Kingma and J. Ba, "Adam: A method for stochastic optimization," 2014, *arXiv:1412.6980*.

[59] MPEG Group. *Video Codec Based Point Cloud Compression (V-PCC) Test Model*. Accessed: 2022. [Online]. Available: https://github.com/MPEGGroup/mpeg-pcc-tmc2

[60] Y. Gal and Z. Ghahramani, "Dropout as a Bayesian approximation: Representing model uncertainty in deep learning," in *Proc. Int. Conf. Mach. Learn. (ICML)*, 2016, pp. 1050–1059.

[61] J. Ye, Y. Chen, N. Wang, and X. Wang, "GIFS: Neural implicit function for general shape representation," in *Proc. IEEE Conf. Comput. Vis. Pattern Recognit. (CVPR)*, Jun. 2022, pp. 12829–12839.

Kohei Matsuzaki received the B.E. degree in electrical, information and physics engineering and the M.E. degree in information sciences from Tohoku University, Sendai, Japan, in 2010 and 2012, respectively, and the Ph.D. degree in informatics from Nagoya University, Nagoya, Japan, in 2021.

In 2012, he joined KDDI Research, Inc., Saitama, Japan, where he has been engaged in research and development in the field of computer vision and image processing. He is currently a Core Researcher with the 3D Space Transmission Laboratory, KDDI Research, Inc.

Satoshi Komorita received the B.E. and M.E. degrees in information and communication engineering from The University of Tokyo, Tokyo, Japan, in 2004 and 2006, respectively.

He joined KDDI Corporation, Tokyo, Japan, in 2006 and engaged in mobile network research, IEEE Standardization, and smartphone development. He is currently the Senior Manager of the 3D Space Transmission Laboratory, KDDI Research, Inc., Saitama, Japan. His current research interests are position estimation from images and 3-D data compression.