

# PIVOT-Net: Heterogeneous Point-Voxel-Tree-based Framework for Point Cloud Compression

Jiahao Pang Kevin Bui\* Dong Tian  
InterDigital, New York, NY, USA

jiahao.pang@interdigital.com, kevinb3@uci.edu, dong.tian@interdigital.com

## Abstract

The universality of the point cloud format enables many 3D applications, making the compression of point clouds a critical phase in practice. Sampled as discrete 3D points, a point cloud approximates 2D surface(s) embedded in 3D with a finite bit-depth. However, the point distribution of a practical point cloud changes drastically as its bit-depth increases, requiring different methodologies for effective consumption/analysis. In this regard, a heterogeneous point cloud compression (PCC) framework is proposed. We unify typical point cloud representations—point-based, voxel-based, and tree-based representations—and their associated backbones under a learning-based framework to compress an input point cloud at different bit-depth levels. Having recognized the importance of voxel-domain processing, we augment the framework with a proposed context-aware upsampling for decoding and an enhanced voxel transformer for feature aggregation. Extensive experimentation demonstrates the state-of-the-art performance of our proposal on a wide range of point clouds.

## 1. Introduction

As affordable depth-sensing devices develop, the point cloud format increasingly attracts interest in the ecosystem of 3D applications, such as AR/VR, robotics, autonomous driving, *etc.* A 3D point cloud describes the surface of an object or a scene by sampling a set of 3D points. To cater to real-world usages in representation and analysis, a point cloud needs to store a huge number of points. For instance, a 10-bit point cloud frame for AR/VR applications typically contains over a million of points. There is a practical demand for point cloud compression (PCC) [25] to alleviate the transmission or storage of point cloud information. In this work, we particularly focus on the lossy compression of point cloud geometry.

A point cloud to be compressed has finite precision indicated by its *bit-depth*, the number of bits needed to de-

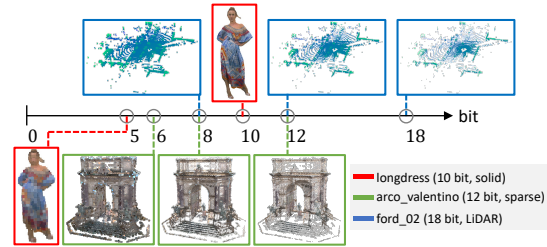


Figure 1. Several 3D point clouds at different bit-depth levels.

scribe the 3D coordinates of its points. For example, a 10-bit point cloud is a point set confined in a 3D box of size  $1024 \times 1024 \times 1024$ , where its  $(x, y, z)$ -coordinates are integers ranging between 0 to  $2^{10} - 1 = 1023$ . In other words, the point cloud coordinates are represented by a 10-bit binary number, where the first few bits correspond to the *coarse* geometry/shape while the last few bits delineate the *fine* details established on top of the coarse shape. By removing the last few bits from the coordinates followed by removing the duplicate points, a point cloud is quantized to its coarser version. For instance, removing the last 2 bits from a 10-bit point cloud results in an 8-bit point cloud with coordinates ranging from 0 to 255. Thus, the compression of a point cloud means to compress all bits of its points' coordinates, starting from its coarse representation (first few bits) to its finer details (last few bits). With every additional bit that is compressed, the resolution of the whole 3D space increases by a factor of 2; meanwhile, the precision of the point cloud also increases by 2 times.

However, it is non-trivial to compress the bits of the point cloud coordinates. As analyzed by [20, 33], a practical point cloud has very different point distributions when it is inspected at different bit-depth levels, as demonstrated by the examples in Fig. 1. When inspected at the first few bits, the points are *densely distributed* in the space. As bit-depth/granularity increases, distances between the 3D points gradually increase. At the last few bits representing the fine details, the 3D points become farther apart from one another. In other words, the points are *sparsely distributed*, as opposed to the distributions at the first few bits.

\*Work done while the author was an intern at InterDigital.

Therefore, the varying point distributions request different methodologies/strategies to effectively process a point cloud at different bit-depths.

In this work, we unify the merits of different point cloud representations—the point-based, the voxel-based, and the tree-based representations—under one learning-based PCC framework that we call the PIVOT-Net (PoInt, VOxel and Tree). It novelly considers the point distributions across bit-depth levels under the Rate-Distortion (RD) restriction of compression problems. Similar to related works [20, 24, 30], for the first few bits where the neighboring points are highly correlated and compressible, a tree-based method is applied to compress the quantized version losslessly. This lays a faithful foundation for compressing the remaining bits at a low cost. For the last few bits where the 3D points are sparsely distributed and less correlated, we apply a point-based neural network to extract the local geometric features like in [20].

Different from [20], to compress the middle-range bits, where neighboring points are still considerably correlated, a 3D convolutional neural network (CNN) is applied for processing in the voxel domain. Notably, PIVOT-Net augments the voxel-domain processing via a proposed context-aware upsampling for decoding/synthesis and via an enhanced voxel transformer for advanced feature aggregation.

The main contributions of the proposed PIVOT-Net include the following:

- (i) We propose the first learning-based PCC framework *unifying* the point-based, voxel-based, and tree-based representations of point clouds to efficiently compress different bit-depth levels of a point cloud.
- (ii) The voxel-domain processing is augmented by a proposed context-aware upsampling module and an enhanced voxel transformer module.
- (iii) The proposed PIVOT-Net demonstrates the state-of-the-art compression performance on a wide spectrum of practical point clouds.

## 2. Related Work

Unlike an image whose pixels are naturally organized on a 2D grid, a point cloud contains *unordered* 3D points in space. To consume the challenging point cloud data, several representations—point-based, voxel-based, and tree-based—are exploited. We first review these representations, followed by illustrating how they facilitate the state-of-the-art PCC proposals. To this end, related techniques for voxel-domain processing are discussed.

**Point cloud representations:** *Voxel-based* representation organizes 3D points in the Euclidean domain, which enables them to be processed like images by 3D CNNs [25]. It is achieved by uniformly quantizing the 3D coordinates to voxel grids with a given quantization step size  $s$ . For example, a point  $(x, y, z)$  is quantized to a voxel

$(\lfloor x/s \rfloor, \lfloor y/s \rfloor, \lfloor z/s \rfloor)$  where  $\lfloor \cdot \rfloor$  is the floor function. Then for indication, a scalar “1” is assigned to the voxel to mark it as occupied while empty voxels are marked by “0” [25]. During voxelization, a smaller  $s$  creates many empty voxels that make the subsequent processing inefficient, whereas a larger  $s$  compromises the representability of the geometry. Thus, voxel-based representation is less suitable to delineate intricate details.

With a voxelized point cloud, the *tree-based* representation views the occupied voxels as leaf nodes and organizes them under a hierarchical tree structure [13]. For instance, a voxelized point cloud with 10-bit precision can be organized as an octree with depth level of 10, where each level is represented by a bit. Thus, encoding the whole octree means encoding the whole point cloud *losslessly*; and encoding only the first few levels of the tree—the first few bits of the point cloud—is to encode the quantized version of the input losslessly [9].

*Point-based* representation is a native point cloud representation, where a point cloud is simply a set of 3D points specified by their  $(x, y, z)$ -coordinates [20, 25]. Unlike other representations, it does not require any pre-processing. Thus, it accurately represents intricate details—the last few bits of a point cloud—without compromising known geometric information. Though it is non-trivial to digest an unordered point set, recent progress in point-based deep neural networks has shed light on this problem. The seminal work PointNet [21] combines multi-layer perceptron (MLP) and pooling operators to extract permutation-invariant features. Inspired by the convolutional layer in Euclidean domain, subsequent works, such as PointCNN [17] and PointConv [36], are proposed. Point-based representation also applies to reconstruction. With a series of MLP layers, Latent GAN [2] directly learns the 3D coordinates of a point cloud. Other works, such as FoldingNet [39] and TearingNet [19], embed a topology for reconstruction.

**Point cloud compression:** As the state-of-the-art, non-learning-based PCC method, MPEG G-PCC utilizes the octree representation for lossless point cloud coding [9]. It is achieved by a hand-crafted context model that predicts the voxel occupancy status, followed by an arithmetic coder for entropy coding. Replacing its hand-crafted context model with a learnable one establishes a family of works called deep octree coding. Representative works in this thread include OctSqueeze [13] and VoxelContext-Net [26].

Based on the seminal work on end-to-end learning for image compression [4, 5], another family of learning-based PCC methods is proposed to perform rate-distortion optimization for lossy point cloud geometry compression. Different from deep octree coding that compresses occupancy status, this paradigm compresses geometric features generated by deep neural networks. As first attempts of this thread, the works [10, 23, 24, 31] apply regular 3D CNN

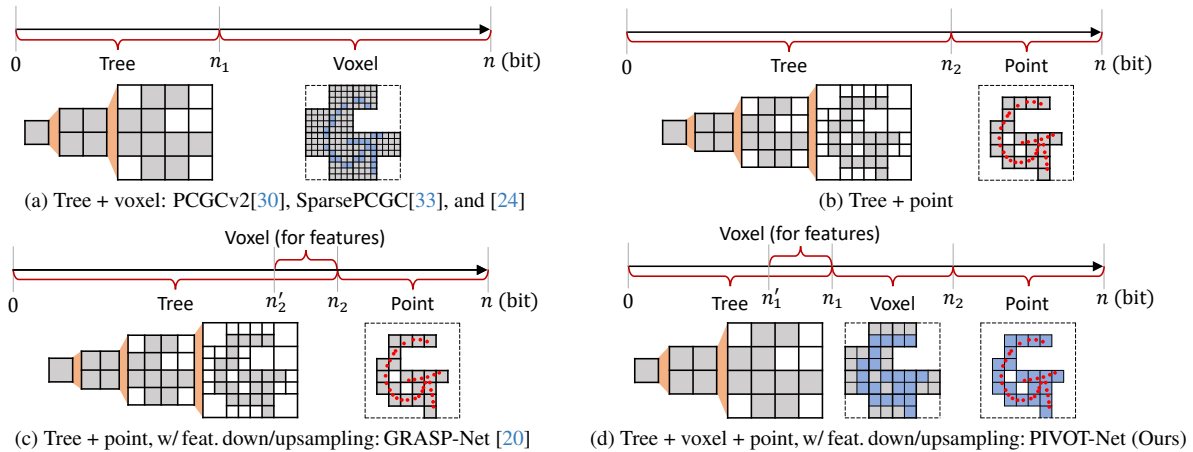


Figure 2. Comparisons of lossy PCC frameworks utilizing different point cloud representations.

to compress voxelized point cloud. Since most of the voxels are empty, regular 3D convolution is memory inefficient. To counter this issue, PCGCv2 [30] and its follow-up effort SparsePCGC [33, 33] apply 3D sparse CNN [7] for lossy compression based on an octree-coded coarse partitioning.

Compared to voxel representation and the associated CNN backbone, point-based representation is more effective for consuming local geometric details. However, there is limited progress in using point-based representation for learning-based PCC. Many existing works, such as [3, 14, 35, 38, 40], fail to justify their performance on real-world data, *e.g.*, the MPEG group’s recommended test point clouds [34]. The recently proposed GRASP-Net [20] has successfully used point-based learning and octree coding for promising compression performance. However, GRASP-Net is *sub-optimal*. It only digests the very last few bits (*e.g.*, the last 2 bits for a 12-bit point cloud) with a point-based neural network, while the rest (*e.g.*, the first 10 bits for a 12-bit point cloud) are coded losslessly with an octree coder that can be very inefficient.

**Voxel-domain processing:** Unlike GRASP-Net, in this work, we relieve the burden of the octree coder by introducing additional voxel-based processing to consume the middle-range bits. The aforementioned works PCGCv2 [30] and SparsePCGC [33] utilize sparse CNNs to achieve upsampling, followed by binary classification and pruning in the voxel domain. Following a similar methodology, while being inspired by the context modeling in octree coding [13, 26], we propose an adaptive voxel upsampling process for decoding that is *context aware*.

Feature aggregation also plays a critical role in the processing and analysis of voxel geometry. In [24], Quach *et al.* applies the classic ResNet [12] for analysis and synthesis in the voxel domain. Wang *et al.* [30, 33] then extend the Inception ResNet (IRN) proposed for 2D images [28] to 3D. The recently proposed *vision transformer* module [8] for 2D image analysis adaptively counts the contributions from

nearby pixels according to a self-attention model, which enables a large receptive field with affordable computational cost. It is later adapted for point cloud processing, leading to point transformer [41] and voxel transformer [18]. In the PIVOT-Net, we enhance the voxel transformer for advanced geometric feature aggregation.

### 3. Point-Voxel-Tree-based PCC Framework

First we analyze typical lossy PCC frameworks to understand how they utilize different point cloud representations. Then we introduce the proposed PIVOT-Net that combines point-based, voxel-based, and tree-based representations.

#### 3.1. Heterogeneous PCC Frameworks

**Overview:** Fig. 2 compares several lossy PCC frameworks to encode an  $n$ -bit point cloud. For each framework, different point cloud representations are applied to compress different intervals along the bit axis. First and foremost, in all frameworks, a *tree-based* (*e.g.*, octree-based) coder is first applied to code the first few bits losslessly as a bit-stream, which results in a coarse, quantized representation of the input point cloud. For instance, see the occupied leaf voxels (in gray) of the tree at the bottom-left of Fig. 2a where the point cloud is partitioned as occupied blocks at the bit-depth level of  $n_1$ . And this *partitioning* information is losslessly recorded by compressing the octree. Thus, the associated bit-stream of the octree is called the *partitioning* bit-stream. After partitioning the point cloud into blocks, how do we encode the point coordinates within each block? In other words, how do we encode the remaining bits?

**Tree+voxel:** Under the general tree+voxel design in Fig. 2a, several works, including PCGCv2 [30], SparsePCGCv1 [33], and [24], view the points within each block as voxels, allowing them to apply a CNN to extract a geometric feature for each block. Unlike traditional compression paradigm (*e.g.*, HEVC [27]) where information rarely flows across block boundaries, utilizing deep neural

networks naturally aggregates information across blocks. Depending on the receptive field of the networks, the generated feature  $f_A$  associated with the block  $A$  may not only abstract the geometry of  $A$  but also contain information from other blocks surrounding  $A$ . Hence, CNNs can uncover the underlying correlation between nearby voxels to generate compact, informative features that can facilitate better coding performance. The extracted features are organized as a 3D feature map of size  $2^{n_1} \times 2^{n_1} \times 2^{n_1}$ , where  $n_1$  is the number of bits already processed in tree-based representation. The nonempty entries of the feature map only correspond to the occupied blocks (in gray) as they are capable of generating features. The feature map is then entropy coded as another bit-stream that we call the *feature* bit-stream to represent the geometry of the last  $n - n_1$  bits.

**Tree+point:** However, the tree+voxel design fails on sparse point cloud [20] such as LiDAR sweeps because the inherent lack of neighbors undermines the performance of CNNs. To address this problem, the points of each block are represented as raw 3D points in  $(x, y, z)$ -coordinates so that they are fed into *point-based* neural networks to extract blockwise features. This process leads to the tree+point design in Fig. 2b. This design is motivated by the fact that point-based neural networks are flexible in representing intricate details without heavily relying on neighboring points, as mentioned in Section 2. Extracting descriptive features with point-based networks should only be applied to the very last few bits where the point cloud gets very sparse. Thus, compared with the tree+voxel design, the tree+point design requires more bits of the point cloud to be coded in tree-based representation. In other words,  $n_2$  in Fig. 2b is larger than  $n_1$  in Fig. 2a, leaving a smaller bit interval to be coded in point-based representation.

**GRASP-Net [20]:** The tree+point design is not optimal for PCC. Compared with the tree+voxel design, the tree+point design requires a larger bit-stream because of two reasons: (i) the feature map generated from the last few bits is finer, and (ii) the tree-based representation is deeper. To resolve (i), GRASP-Net additionally downsamples (or upsamples when decoding) the feature map in the voxel domain by a CNN, as shown in Fig. 2c. Via downsampling with a CNN, the resolution of the feature map is reduced from  $2^{n_2} \times 2^{n_2} \times 2^{n_2}$  to  $2^{n'_2} \times 2^{n'_2} \times 2^{n'_2}$  (when encoding) where  $n'_2 < n_2$ . The downsampled feature map is further entropy coded. In this design, the point cloud geometry is not consumed by the CNN. Differently, the known geometry at bit-depth level  $n_2$  serves as a support to down-/up-sample the features attached to the point cloud blocks.

**Our proposal:** Note that GRASP-Net still encodes the finer-grained block partitioning of the first  $n_2$  bits. Motivated by the CNN in the tree+voxel design (Fig. 2a) for digesting geometry, we propose PIVOT-Net as the first framework for the *unified* tree+voxel+point design in Fig. 2d. In-

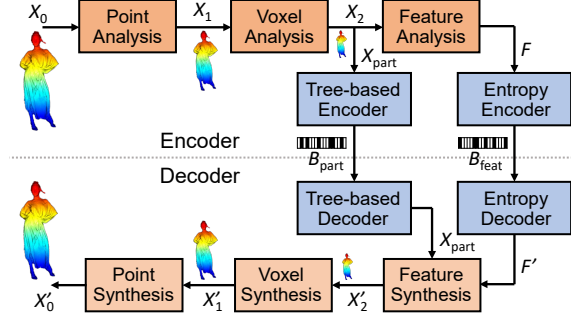


Figure 3. Architecture of our PIVOT-Net. The orange blocks contain learnable neural network layers, where point analysis/synthesis are point-based neural networks while voxel and feature analysis/synthesis are sparse CNNs.

stead of handling bit interval  $[n_1, n_2]$  with an octree coder in GRASP-Net, the proposed PIVOT-Net represents these middle-range bits with voxels and consumes them with a CNN, thereby reducing the burden of the octree coder.

Based on a novel processing of the fine, middle and coarse levels of point clouds with the point-, voxel-, and tree-based modules, our PIVOT-Net considers the point distributions across bit-depth levels under the Rate-Distortion (RD) restriction in compression. Thus, each main component of PIVOT-Net *can be configured* to handle a specific range of point distribution to achieve superior PCC.

### 3.2. The PIVOT-Net Architecture

The block diagram of the PIVOT-Net is shown in Fig. 3, where the neural network modules are colored in orange.

**Encoder:** An input point cloud  $X_0$  is fed into the *Point Analysis Network*, a point-based neural network akin to PointNet++ [22]. In contrast to PointNet++ relying on farthest point sampling to obtain a coarse point cloud, the Point Analysis Network first generates the coarse representation  $X_1$  by uniform quantization with a constant step size  $s_1 > 1$ , where  $n - n_2 = \log_2 s_1$ . Then with the set abstraction layer from PointNet++, it extracts a geometric feature for each point in  $X_1$  with the nearest neighbor search followed by a shared PointNet. Thus, the last  $\log_2 s_1$  bits representing the fine geometric details are consumed with this point-based network (the bit interval  $[n_2, n]$  of Fig. 2d).

Next, in the 3D tensor format, the coarse point cloud  $X_1$  (appended with the geometric features generated from the Point Analysis Network) is fed into the *Voxel Analysis Network* that uses a sparse CNN for downsampling and feature aggregation similarly done in PCGCv2 [30]. By downsampling  $X_1$  by  $s_2$  times to obtain the output  $X_2$  and its features, the Voxel Analysis Network abstracts from  $X_1$  the last  $\log_2 s_2$  bits, which corresponds to the bit interval  $[n_1, n_2]$  in Fig. 2d, *i.e.*,  $n_2 - n_1 = \log_2 s_2$ .

The voxel geometry, or occupancy status, of  $X_2$  is denoted as  $X_{part}$ , which contains the block partitioning in-

formation.  $X_{\text{part}}$  is a coarse version of the input  $X_0$  with an overall quantization step size  $s = s_1 s_2$ . It is losslessly coded with the tree-based encoder as a partitioning bit-stream  $B_{\text{part}}$ , corresponding to the bit interval  $[0, n_1]$  in Fig. 2d. Moreover, the feature map of  $X_2$  shrinks by  $s_3$  times using the *Feature Analysis Network*, another sparse CNN for downsampling. Thus, the Feature Analysis Network is applied to the bit interval  $[n'_1, n_1]$  in Fig. 2d, i.e.,  $n_1 - n'_1 = \log_2 s_3$ . The downsampled feature map  $F$  is finally entropy encoded as the feature bit-stream  $B_{\text{feat}}$ .

**Decoder:** The partitioning information  $X_{\text{part}}$  is decoded from the partitioning bit-stream  $B_{\text{part}}$ . Meanwhile, the feature bit-stream  $B_{\text{feat}}$  is decoded for the (decoded) downsampled feature map  $F'$ . Next, both  $X_{\text{part}}$  and  $F'$  are fed to the *Feature Synthesis Network*, a sparse CNN similar to the one in GRASP-Net [20]. It upsamples the input feature map  $F'$  while refining it to match the occupancy status of  $X_{\text{part}}$ , leading to an output tensor  $X'_2$  with the upsampled feature map attached to it. Note that  $X'_2$  and  $X_2$  have different features, but they share the same geometry as  $X_{\text{part}}$ .

$X'_2$  is then fed into the *Voxel Synthesis Network*. Like the Feature Synthesis Network, the Voxel Synthesis Network is also a sparse CNN for upsampling. However, while the Feature Synthesis Network intends to upsample the features  $F'$  to a known geometry  $X_{\text{part}}$ , the Voxel Synthesis Network aims at upsampling the *geometry* of  $X'_2$  based on its features. Outputted from the Voxel Synthesis Network,  $X'_1$  has the same resolution as  $X_1$ , i.e.,  $2^{n_2} \times 2^{n_2} \times 2^{n_2}$ .

In the end, consisting of a series of multi-layer perceptron (MLP) layers, the *Point Synthesis Network* [20] recovers the raw 3D points for the final reconstruction. For each geometric feature in  $X'_1$ , it generates a 3D point set that delineates the local geometric details in the neighborhood of the point cloud block  $A$ . The decoded point cloud  $X'_0$  is the aggregation of all the 3D point sets generated by the Point Synthesis Network. Please refer to the supplementary material for additional details on architecture designs.

## 4. Voxel Geometry Processing

Our proposed PIVOT-Net stands out among the other designs in Fig. 2 because it utilizes voxel-domain processing for consuming the middle-range geometric bits. This section illustrates our efforts in capitalizing voxel-domain processing. We incorporate two major components for voxel-domain processing: (i) a context-aware upsampling process for adaptive voxel synthesis and (ii) an Enhanced Voxel Transformer inspired by [18] for feature aggregation.

### 4.1. Adaptive Voxel Synthesis

Downsampling and feature aggregation are straightforward to perform with sparse CNN, such as using convolutional layers with stride 2 as done in [24] and [30]. For simplicity, our work adopts the “down-scale” module from

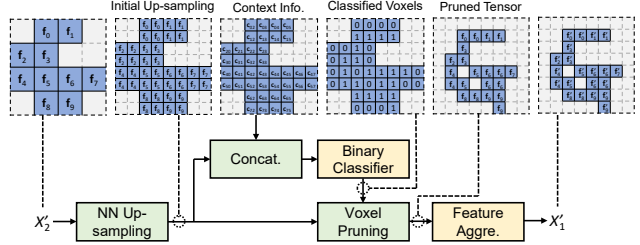


Figure 4. Context-aware upsampling for adaptive voxel synthesis. Learnable modules are colored in yellow.

PCGCv2 [30] for the Voxel Analysis Network.

On the other hand, upsampling in the voxel domain is non-trivial. The naïve nearest-neighbor (NN) upsampling always splits a parent voxel into 8 child voxels, which leads to not only inaccurate upsampled geometry but also a waste of computation because many irrelevant occupied voxels are created. To address these issues, PCGCv2 [30] and SparsePCGC [33] perform binary classification on the newly created child voxels and remove some of them that are less likely to be occupied in the ground truth.

Inspired by the *context modeling* in octree coding, we additionally utilize the known knowledge about the child voxels to determine which child voxels should be pruned and which to be kept. The block diagram of the proposed Voxel Synthesis Network is shown in Fig 4. For simplicity, we assume the voxel analysis/synthesis network digests only 1 bit. Thus, the voxel synthesis network performs upsampling by a factor of 2, though it can be cascaded a few times if more middle-range bits are coded.

To upsample the point cloud  $X'_2$  and its associated feature map, they are first fed to an NN upsampling module using transposed convolution [11] to obtain an initial upsampled tensor that is two times larger along each dimension. After that, they are concatenated with the context information, followed by being fed to a binary classification module [30] to determine which voxels should be removed. In this work, the context information of a child voxel includes its  $(x, y, z)$ -coordinate and the current bit-depth level. Next, the geometry of the initial upsampled point cloud is refined using voxel pruning according to the classification output. In the end, a feature aggregation module with sparse convolutions is appended to improve the features based on the refined geometry, resulting in the output point cloud  $X'_1$ .

### 4.2. Enhanced Voxel Transformer

Compared to point-based neural network that usually has a large receptive field for capturing neighboring relationships, voxel-based processing with convolutional layers appears to be more “rigid” as it is unable to discover long-range dependency, especially for sparse point clouds. However, CNN-based feature aggregation, e.g., with 3D Inception ResNet (IRN) [20, 30], in the voxel-domain is still critical for generating descriptive geometric features. To overcome this in-

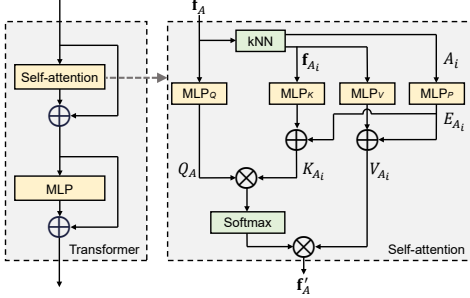


Figure 5. Enhanced Voxel Transformer (left) and its self-attention module (right). Learnable modules are colored in yellow.

herent issue of CNN-based feature aggregation, we propose the *Enhanced Voxel Transformer* inspired by [18] as a better alternative for feature aggregation.

An Enhanced Voxel Transformer block is shown in the left of Fig. 5, which consists of a self-attention block and an MLP block. The self-attention block is detailed in the right of Fig. 5. Given a current feature vector  $\mathbf{f}_A$  associated with a voxel  $A$  and its neighboring  $k$  features  $\mathbf{f}_{A_i}$ 's associated with the  $k$  nearest voxels  $A_i$ 's, the self-attention block updates  $\mathbf{f}_A$  based on the neighboring features  $\mathbf{f}_{A_i}$ 's using learnable MLP blocks  $\text{MLP}_Q$ ,  $\text{MLP}_K$ ,  $\text{MLP}_V$  and  $\text{MLP}_P$ .

Firstly, the query embedding of  $A$  is obtained via  $Q_A = \text{MLP}_Q(\mathbf{f}_A)$ . The key embedding  $K_{A_i}$  and the value embedding  $V_{A_i}$  of all the nearest neighbors of  $A$  are computed as

$$K_{A_i} = \text{MLP}_K(\mathbf{f}_{A_i}) + E_{A_i}, V_{A_i} = \text{MLP}_V(\mathbf{f}_{A_i}) + E_{A_i}, \quad (1)$$

for  $0 \leq i \leq k-1$ . The term  $E_{A_i}$  is the positional encoding between the voxels  $A$  and  $A_i$  calculated by

$$E_{A_i} = \text{MLP}_P(P_A - P_{A_i}), \quad (2)$$

where  $P_A$  and  $P_{A_i}$  are the 3D coordinates of the voxels  $A$  and  $A_i$ , respectively. Then the updated feature of  $A$  is

$$\mathbf{f}'_A = \sum_{i=0}^{k-1} \sigma \left( Q_A^T K_{A_i} / c\sqrt{d} \right) \cdot V_{A_i}, \quad (3)$$

where  $\sigma(\cdot)$  is the softmax function for computing the attention map,  $d$  is the length of  $\mathbf{f}'_A$ , and  $c$  is a constant.

In contrast to [18], we replace the linear projection layers with MLP layers for more flexibility when computing the attention map and the value embedding. In this work, we use the Enhanced Voxel Transformer blocks for feature aggregation in the Feature Synthesis Network (Fig. 3) to have a good trade-off between complexity and performance, where we cascade 3 blocks with *shared* weights. Empirically, we find that our weight-shared transformers lead to a smaller model with better results.

## 5. Experimentation

### 5.1. Experimental Setup

**Datasets:** Compression of the PIVOT-Net is verified on a comprehensive set of selected point clouds suggested by the

MPEG group for learning-based PCC [1]. With the taxonomy of [1], selected point clouds are categorized into four types—(i) 4 *solid* surface point clouds (10–11 bits), (ii) 3 *dense* surface point clouds (12 bits), (iii) 5 *sparse* surface point clouds (12–13 bits), and (iv) “ford\_02” & “ford\_03” *LiDAR* sequences with 3000 point cloud frames collected by a spinning LiDAR (18 bits). The surface point clouds have a number of points ranging from 272K to 4.8M; the LiDAR point clouds have about 80K points. Example point clouds in [1] are visualized in Fig. 1.

To compress surface point clouds, PIVOT-Net is trained with the ModelNet40 [37] dataset which contains 12k CAD models from 40 categories of objects. As recommended by [1], for the case of LiDAR point clouds, we train the PIVOT-Net with the LiDAR sequence “ford\_01” containing 1500 point cloud frames.

**Implementation details:** PIVOT-Net is implemented based on *PccAI* [15]—a testbed for learning-based PCC. It is then end-to-end trained with the rate-distortion (R-D) loss  $L = L_D + \lambda L_R$ . Here  $L_D$  measures the geometric distortion and  $L_R$  is the estimated bitrate of  $B_{\text{feat}}$  while  $\lambda$  controls their trade-off. The distortion is computed as  $L_D = \alpha L_{CD} + \beta L_{BCE}$ , where  $L_{CD}$  is the augmented Chamfer distance between the ground-truth and the decoded point clouds [6, 20, 39] and  $L_{BCE}$  is the average binary cross entropy loss between the classification output (before thresholding) and the voxelized ground-truth point cloud.

We achieve different rate points by adjusting the interval  $[n_1, n]$  (Fig. 2d) and train one model per rate point, where shorter interval  $n - n_1$  means a deeper octree (or smaller block size) and leads to a larger rate point. For surface point clouds, we sample 5 sizes of  $n - n_1$  in  $[1, 4]$  for 5 rate points. Within  $[n_1, n]$ , we configure the sub-interval of point-based processing  $[n_2, n]$  to have a size from 0 to 4 according to the surface point cloud categories. Note that  $n - n_2 = 0$  implies the point-based processing is turned off and the whole range  $[n_1, n]$  is consumed by voxel-domain processing. For the sparse LiDAR point clouds, we let  $n - n_1$  ranges within  $[2, 9]$  and pick 6 rate points covering a typical operation range. We also turn off the voxel-domain processing due to the highly sparse nature of LiDAR sweeps, *i.e.*,  $n_1 = n_2$ . In all cases, we let  $n_1 - n'_1 = 1$ , meaning that the feature map is  $2 \times$  downsampled/upsampled for compression.

We apply the Adam optimizer [16] to train the PIVOT-Net for 50 epochs with a learning rate  $8 \times 10^{-4}$ . The batch size is 8 when training on ModelNet40 and is 2 when training on LiDAR data. Please refer to the supplementary material for more details.

**Benchmarking:** State-of-the-art lossy PCC methods are used for comparisons: (i) G-PCC octree (lossy), which is a non-learning-based method standardized by MPEG [9]; (ii) GRASP-Net [20] and (iii) SparsePCGC [33] which are learning-based methods. For fair comparisons, the octree

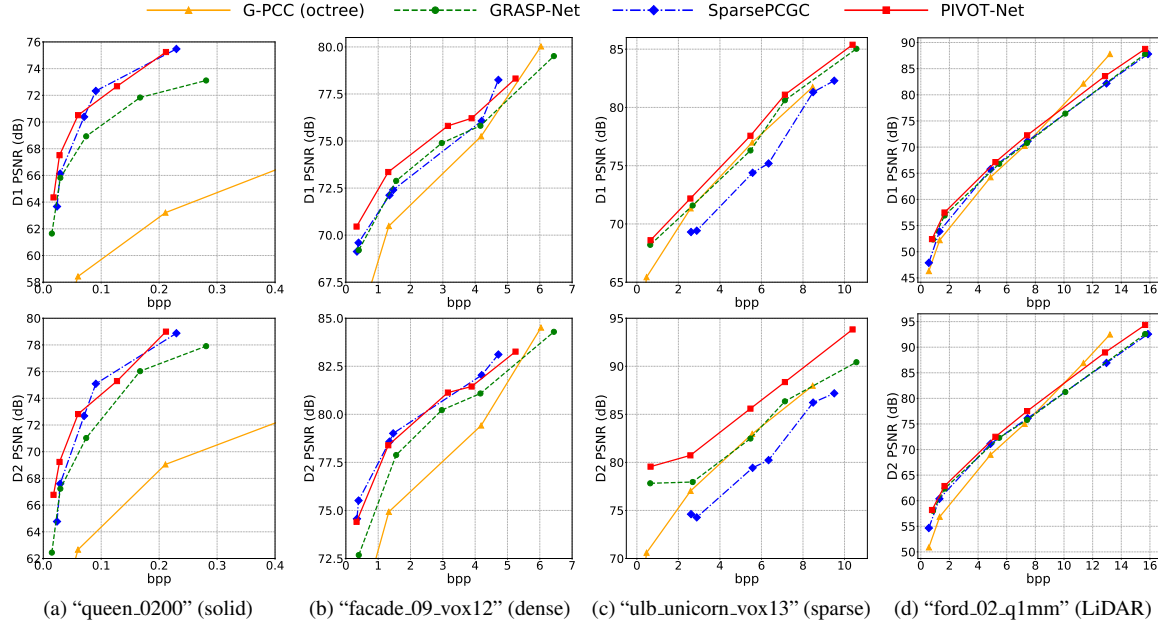


Figure 6. R-D performance on different point clouds measured in both D1-PSNR and D2-PSNR. Our proposed PIVOT-Net provides competitive performance among all the methods.

Table 1. BD-Rate (in %) and BD-PSNR (in dB) gains against G-PCC octree (lossy) on different categories of point clouds.

Category	Solid surface point clouds				Dense surface point clouds				Sparse surface point clouds				LiDAR point clouds			
	BD-Rate ↓		BD-PSNR ↑		BD-Rate ↓		BD-PSNR ↑		BD-Rate ↓		BD-PSNR ↑		BD-Rate ↓		BD-PSNR ↑	
	D1	D2	D1	D2	D1	D2	D1	D2	D1	D2	D1	D2	D1	D2	D1	D2
GRASP-Net	-88.80	-75.74	9.30	7.20	-41.08	-54.86	2.86	3.70	-10.19	-9.97	0.86	1.01	-5.56	-9.64	1.31	2.08
SparsePCGC	-91.10	-81.98	<b>10.99</b>	<b>9.01</b>	-52.23	<b>-68.10</b>	4.26	<b>6.11</b>	14.53	6.08	-1.48	-0.80	-5.40	-11.47	1.00	2.04
PIVOT-Net	<b>-92.34</b>	<b>-83.22</b>	10.88	8.97	<b>-58.42</b>	-66.61	<b>4.37</b>	5.20	<b>-17.01</b>	<b>-21.21</b>	<b>1.40</b>	<b>2.30</b>	<b>-11.85</b>	<b>-17.19</b>	<b>2.19</b>	<b>3.10</b>

coders in GRASP-Net, SparsePCGC, and our PIVOT-Net are all aligned as G-PCC octree (lossless). The code/results of SparsePCGC and GRASP-Net are provided by their authors, where the SparsePCGC we compared is a slightly improved version [32] over the original work [33].

For evaluation, bitrate is measured by *bits per input point* (bpp) while geometry distortion is measured by the peak signal-to-noise ratio (PSNR) based on *point-to-point distance* (D1) and *point-to-plane distance* (D2) [29]. Following the convention in MPEG [34], we evaluate BD-Rate (in %, the more negative it is, the more bitrate savings) and BD-PSNR (in dB, higher means better) by comparing an R-D curve with the one achieved by G-PCC octree (lossy), *i.e.*, G-PCC octree (lossy) serves as an *anchor*.

## 5.2. Performance Comparisons

The average R-D curves of the different methods on representative point clouds are plotted in Fig. 6. We also compute the BD-Rate and BD-PSNR gains against the G-PCC octree (lossy) anchor. The results are provided Table 1.

Overall, we see that our PIVOT-Net achieves the best R-D trade-off among the competing approaches. It performs on par with SparsePCGC [33] on solid and dense surface point clouds while outperforming SparsePCGC significantly on sparse surface point clouds and LiDAR point

clouds. That is because we account for the sparse nature of these point clouds with point-based processing (the  $[n_2, n]$  interval of Fig. 2d) while SparsePCGC does not. Compared to GRASP-Net [20], we achieve clear advantages on the solid and dense surface point clouds because of the consumption of the middle-range bits in the voxel domain (the  $[n_1, n_2]$  interval of Fig. 2d) while GRASP-Net ignores that.

Fig. 7 visualizes the 11-bit solid surface point cloud “dancer\_vox11\_00000001” decoded by different methods while Fig. 8 visualizes the decoded results of the LiDAR sweep “ford\_02\_vox1mm-0120”. The decoded point clouds in these figures are colored by their D1 errors. Compared with the other methods, our PIVOT-Net provides higher-quality reconstructions at smaller or similar bit-rates. Notably, it makes fewer errors in intricate details such as the corners in Fig. 7 and the line patterns in Fig. 8.

## 5.3. Ablation Study

We study how the coding performance of the PIVOT-Net is benefited by (i) the voxel processing of the middle-range bits and (ii) the transformer. On solid surface point clouds, the performance of several variations of the PIVOT-Net are shown in Table 2. We start from a configuration where both (i) and (ii) are removed (*i.e.*, GRASP-Net [20]). Then we enable (i) (first without, then with the context-aware upsam-

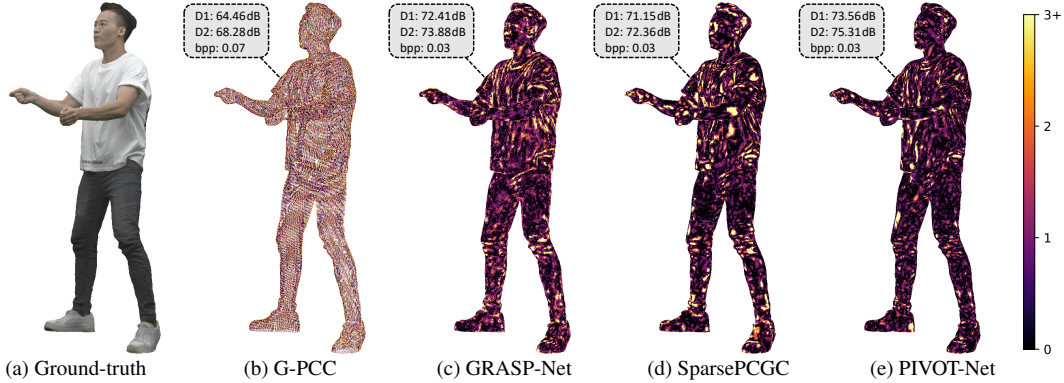


Figure 7. Visual comparisons of “dancer\_vox11\_00000001” (solid, 11-bit). The decoded point clouds are colored by the D1 errors.

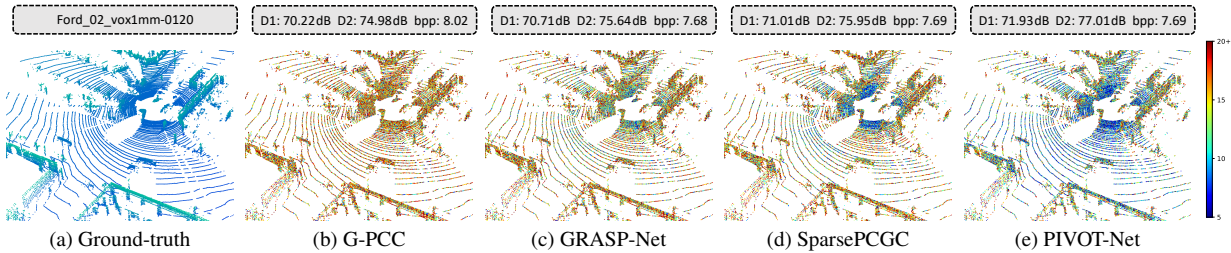


Figure 8. Visual comparisons of LiDAR point cloud “ford.02\_vox1mm-0120” (18-bit). Decoded point clouds are colored by D1 errors.

Table 2. BD-Rate (in %) and BD-PSNR (in dB) gains against G-PCC octree (lossy) on different variations of the PIVOT-Net.

Voxel for mid. bits		Transformer		BD-Rate ↓		BD-PSNR ↑	
w/o ctx.	w/ ctx.	VT	Enh. VT	D1	D2	D1	D2
×	×	×	×	-88.80	-75.74	9.30	7.20
✓	×	×	×	-91.31	-81.46	10.52	8.70
×	✓	×	×	-91.72	-82.01	10.64	8.80
×	✓	×	×	-91.85	-82.40	10.58	8.79
×	✓	×	✓	<b>-92.34</b>	<b>-83.22</b>	<b>10.88</b>	<b>8.97</b>

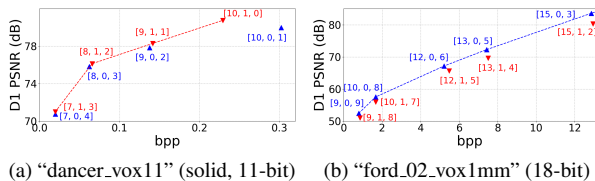


Figure 9. Changing the interval lengths in PIVOT-Net.

pling) which leads to around 3% and 6% BD-Rate gains in terms of D1 and D2, respectively. Next, we enable (ii) (with the voxel transformer [18], then with the Enhanced Voxel Transformer), bringing an extra 1% BD-Rate saving. Thus, we verify the importance of (ii) for PIVOT-Net.

We also adjust the intervals for solid (most dense) and LiDAR (most sparse) point clouds, showing the trade-offs in Fig. 9. In Fig. 9, each operation point in the R-D plot is denoted by  $[c, m, f]$ , where  $c = n_1$ ,  $m = n_2 - n_1$ , and  $f = n - n_2$  bits are the coarsest-, middle-, and finest-bit intervals (Fig. 2d), respectively. From Fig. 9a,  $\blacktriangledown$  operation points (with  $m > 0$ , *i.e.*, voxel processing for middle bits) are preferable for solid point clouds, where the  $\blacktriangledown$  points delineate the best R-D curve. But for sparse point clouds (Fig. 9b), point-based coding ( $\blacktriangle$ ) is preferred.

## 5.4. Complexity

PIVOT-Net has a reasonable computational cost. For instance, on the 3000 test LiDAR frames, to encode/decode a frame at around 8 bpp (the rate point of the examples in Fig. 8), PIVOT-Net takes about 1.8 sec/2.0 sec. On the same platform, G-PCC takes 3.8 sec/1.8 sec, GRASP-Net takes 1.6 sec/1.7 sec and SparsePCGC takes 2.1 sec/11.5 sec. Note that running time is measured with the `time.monotonic()` function in Python.

PIVOT-Net also has a small model size. When compressing different point cloud categories, neural network models of different sizes are applied. On average, the models of PIVOT-Net have 482 K parameters, being slightly larger than those of the GRASP-Net (430 K). However, both of them are much smaller than the model size of SparsePCGC, which on average has 3363 K parameters.

## 6. Conclusion

We propose PIVOT-Net, a heterogeneous PCC framework unifying three point cloud representations—the point-based, voxel-based, and tree-based representations—to altogether digest the least-significant, the middle-range, and the most-significant bits. The voxel-domain processing is augmented with a proposed context-aware upsampling procedure for decoding and an Enhanced Voxel Transformer for feature aggregation. The state-of-the-art performance of PIVOT-Net is confirmed on a wide spectrum of point clouds. A future research direction is to optimize the allocation of the bits assigned to the three representations.



## References

- [1] MPEG 3DGH. Preliminary data set collection for AI experiments. [https://www.mpeg.org/wp-content/uploads/mpeg\\_meetings/141\\_OnLine/w22431.zip](https://www.mpeg.org/wp-content/uploads/mpeg_meetings/141_OnLine/w22431.zip), 2023. 6
- [2] Panos Achlioptas, Olga Diamanti, Ioannis Mitliagkas, and Leonidas Guibas. Learning representations and generative models for 3D point clouds. In *International Conference on Machine Learning*, pages 40–49, Vancouver, Canada, 2018. PMLR. 2
- [3] Md Ahmed Al Muzaddid and William J Beksi. Variable rate compression for raw 3d point clouds. In *2022 International Conference on Robotics and Automation (ICRA)*, pages 8748–8755. IEEE, 2022. 3
- [4] Johannes Ballé, Valero Laparra, and Eero P Simoncelli. End-to-end optimized image compression. In *International Conference on Learning Representations*, pages 1–27, San Juan, Puerto Rico, USA, 2017. ICLR. 2
- [5] Johannes Ballé, David Minnen, Saurabh Singh, Sung Jin Hwang, and Nick Johnston. Variational image compression with a scale hyperprior. In *International Conference on Learning Representations*, pages 1–47, Vancouver, Canada, 2018. ICLR. 2
- [6] Siheng Chen, Baoan Liu, Chen Feng, Carlos Vallespi-Gonzalez, and Carl Wellington. 3D point cloud processing and learning for autonomous driving: Impacting map creation, localization, and perception. *IEEE Signal Processing Magazine*, 38(1):68–86, 2020. 6
- [7] Christopher Choy, JunYoung Gwak, and Silvio Savarese. 4d spatio-temporal convnets: Minkowski convolutional neural networks. In *Proceedings of the IEEE Conference on Computer Vision and Pattern Recognition*, pages 3075–3084, 2019. 3
- [8] Alexey Dosovitskiy, Lucas Beyer, Alexander Kolesnikov, Dirk Weissenborn, Xiaohua Zhai, Thomas Unterthiner, Mostafa Dehghani, Matthias Minderer, Georg Heigold, Sylvain Gelly, et al. An image is worth 16x16 words: Transformers for image recognition at scale. *arXiv preprint arXiv:2010.11929*, 2020. 3
- [9] 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 Transactions on Signal and Information Processing*, 9:e13, 2020. 2, 6
- [10] André FR Guarda, Nuno MM Rodrigues, and Fernando Pereira. Adaptive deep learning-based point cloud geometry coding. *IEEE Journal of Selected Topics in Signal Processing*, 15(2):415–430, 2020. 2
- [11] JunYoung Gwak, Christopher B Choy, and Silvio Savarese. Generative sparse detection networks for 3d single-shot object detection. In *European conference on computer vision*, 2020. 5
- [12] Kaiming He, Xiangyu Zhang, Shaoqing Ren, and Jian Sun. Deep residual learning for image recognition. In *IEEE Conference on Computer Vision and Pattern Recognition*, pages 770–778, Las Vegas, Nevada, USA, 2016. IEEE. 3
- [13] Lila Huang, Shenlong Wang, Kelvin Wong, Jerry Liu, and Raquel Urtasun. OctSqueeze: Octree-structured entropy model for LiDAR compression. In *IEEE Conference on Computer Vision and Pattern Recognition*, pages 1313–1323, Seattle, WA, USA, 2020. IEEE. 2, 3
- [14] Tianxin Huang and Yong Liu. 3D point cloud geometry compression on deep learning. In *ACM International Conference on Multimedia*, pages 890–898, Nice, France, 2019. ACM. 3
- [15] InterDigital. PccAI: A framework for ai-based point cloud compression. <https://github.com/InterDigitalInc>, 2022. 6
- [16] Diederick P Kingma and Jimmy Ba. Adam: A method for stochastic optimization. In *International Conference on Learning Representations*, pages 1–15, San Diego, CA, USA, 2015. ICLR. 6
- [17] Yangyan Li, Rui Bu, Mingchao Sun, Wei Wu, Xinhan Di, and Baoquan Chen. PointCNN: Convolution on X-transformed points. *Advances in Neural Information Processing Systems*, 31:1–11, 2018. 2
- [18] Jiageng Mao, Yujing Xue, Minzhe Niu, Haoyue Bai, Jiashi Feng, Xiaodan Liang, Hang Xu, and Chunjing Xu. Voxel transformer for 3D object detection. In *Proceedings of the IEEE/CVF International Conference on Computer Vision*, pages 3164–3173, 2021. 3, 5, 6, 8
- [19] Jiahao Pang, Duanshun Li, and Dong Tian. TearingNet: Point cloud autoencoder to learn topology-friendly representations. In *IEEE Conference on Computer Vision and Pattern Recognition*, pages 7453–7462, Nashville, Tennessee, USA, 2021. IEEE. 2
- [20] Jiahao Pang, Muhammad Asad Lodhi, and Dong Tian. GRASP-Net: Geometric residual analysis and synthesis for point cloud compression. In *Proceedings of the 1st International Workshop on Advances in Point Cloud Compression, Processing and Analysis*, pages 11–19, 2022. 1, 2, 3, 4, 5, 6, 7
- [21] Charles R Qi, Hao Su, Kaichun Mo, and Leonidas J Guibas. PointNet: Deep learning on point sets for 3D classification and segmentation. In *IEEE Conference on Computer Vision and Pattern Recognition*, pages 652–660, Honolulu, Hawaii, USA, 2017. IEEE. 2
- [22] Charles Ruizhongtai Qi, Li Yi, Hao Su, and Leonidas J Guibas. PointNet++: Deep hierarchical feature learning on point sets in a metric space. *Advances in Neural Information Processing Systems*, 30:1–10, 2017. 4
- [23] Maurice Quach, Giuseppe Valenzise, and Frederic Dufaux. Learning convolutional transforms for lossy point cloud geometry compression. In *IEEE International Conference on Image Processing*, pages 4320–4324, Taipei, Taiwan, 2019. IEEE. 2
- [24] Maurice Quach, Giuseppe Valenzise, and Frederic Dufaux. Improved deep point cloud geometry compression. In *International Workshop on Multimedia Signal Processing*, pages 1–6, Tampere, Finland, 2020. IEEE. 2, 3, 5
- [25] Maurice Quach, Jiahao Pang, Dong Tian, Giuseppe Valenzise, and Frédéric Dufaux. Survey on deep learning-based point cloud compression. *Frontiers in Signal Processing*, 2, 2022. 1, 2

- [26] Zizheng Que, Guo Lu, and Dong Xu. VoxelContext-Net: An octree based framework for point cloud compression. In *IEEE Conference on Computer Vision and Pattern Recognition*, pages 6042–6051, Nashville, Tennessee, USA, 2021. IEEE. 2, 3
- [27] Gary J Sullivan, Jens-Rainer Ohm, Woo-Jin Han, and Thomas Wiegand. Overview of the high efficiency video coding (HEVC) standard. *IEEE Transactions on circuits and systems for video technology*, 22(12):1649–1668, 2012. 3
- [28] Christian Szegedy, Sergey Ioffe, Vincent Vanhoucke, and Alexander A Alemi. Inception-v4, inception-resnet and the impact of residual connections on learning. In *Thirty-first AAAI conference on artificial intelligence*, 2017. 3
- [29] Dong Tian, Hideaki Ochimizu, Chen Feng, Robert Cohen, and Anthony Vetro. Geometric distortion metrics for point cloud compression. In *IEEE International Conference on Image Processing (ICIP)*, pages 3460–3464, Beijing, China, 2017. IEEE. 7
- [30] Jianqiang Wang, Dandan Ding, Zhu Li, and Zhan Ma. Multi-scale point cloud geometry compression. In *Data Compression Conference*, pages 73–82. IEEE, 2021. 2, 3, 4, 5
- [31] Jianqiang Wang, Hao Zhu, Haojie Liu, and Zhan Ma. Lossy point cloud geometry compression via end-to-end learning. *IEEE Transactions on Circuits and Systems for Video Technology*, 31(12):4909–4923, 2021. 2
- [32] Jianqiang Wang, Zhan Ma, Honglian Wei, Yue Yu, Vladyslav Zakharchenko, and Dong Wang. SparsePCGCv1 update: Improvements on dense/sparse/LiDAR point clouds. m60352, MPEG 3DGH, 2022. 7
- [33] Jianqiang Wang, Dandan Ding, Zhu Li, Xiaoxing Feng, Chuntong Cao, and Zhan Ma. Sparse tensor-based multi-scale representation for point cloud geometry compression. *IEEE Transactions on Pattern Analysis and Machine Intelligence*, 45(7):9055–9071, 2023. 1, 3, 5, 6, 7
- [34] MPEG WG7. Common test conditions for G-PCC. w20358, 2022. 3, 7
- [35] Louis Wiesmann, Andres Milioto, Xieyuanli Chen, Cyrill Stachniss, and Jens Behley. Deep compression for dense point cloud maps. *IEEE Robotics and Automation Letters*, 6(2):2060–2067, 2021. 3
- [36] Wenxuan Wu, Zhongang Qi, and Li Fuxin. PointConv: Deep convolutional networks on 3D point clouds. In *IEEE Conference on Computer Vision and Pattern Recognition*, pages 9621–9630, Long Beach, CA, USA, 2019. IEEE. 2
- [37] Zhirong Wu, Shuran Song, Aditya Khosla, Fisher Yu, Linguang Zhang, Xiaoou Tang, and Jianxiong Xiao. 3D ShapeNets: A deep representation for volumetric shapes. In *IEEE Conference on Computer Vision and Pattern Recognition*, pages 1912–1920, Boston, MA, USA, 2015. IEEE. 6
- [38] 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*, pages 1–8, 2019. 3
- [39] Yaoqing Yang, Chen Feng, Yiru Shen, and Dong Tian. FoldingNet: Point cloud auto-encoder via deep grid deformation. In *IEEE Conference on Computer Vision and Pattern Recognition*, pages 206–215, Salt Lake City, Utah, USA, 2018. IEEE. 2, 6
- [40] Kang You and Pan Gao. Patch-based deep autoencoder for point cloud geometry compression. In *ACM Multimedia Asia*, pages 1–7, Gold Coast, Australia, 2021. ACM. 3
- [41] Hengshuang Zhao, Li Jiang, Jiaya Jia, Philip HS Torr, and Vladlen Koltun. Point transformer. In *Proceedings of the IEEE/CVF International Conference on Computer Vision*, pages 16259–16268, 2021. 3