3QNet: 3D Point Cloud Geometry Quantization Compression Network

TIANXIN HUANG, Zhejiang University, China
JIANGNING ZHANG, Zhejiang University, China
JUN CHEN, Zhejiang University, China
ZHONGGAN DING, Tencent youtu Lab, China
YING TAI, Tencent youtu Lab, China
ZHENYU ZHANG, Tencent youtu Lab, China
CHENGJIE WANG, Tencent youtu Lab, China
YONG LIU*, Zhejiang University, China

Since the development of 3D applications, the point cloud, as a spatial description easily acquired by sensors, has been widely used in multiple areas such as SLAM and 3D reconstruction. Point Cloud Compression (PCC) has also attracted more attention as a primary step before point cloud transferring and saving, where the geometry compression is an important component of PCC to compress the points geometrical structures. However, existing non-learning-based geometry compression methods are often limited by manually pre-defined compression rules. Though learning-based compression methods can significantly improve the algorithm performances by learning compression rules from data, they still have some defects. Voxel-based compression networks introduce precision errors due to the voxelized operations, while point-based methods may have relatively weak robustness and are mainly designed for sparse point clouds. In this work, we propose a novel learning-based point cloud compression framework named 3D Point Cloud Geometry Quantiation Compression Network (3QNet), which overcomes the robustness limitation of existing point-based methods and can handle dense points. By learning a codebook including common structural features from simple and sparse shapes, 3QNet can efficiently deal with multiple kinds of point clouds. According to experiments on object models, indoor scenes, and outdoor scans, 3QNet can achieve better compression performances than many representative methods.

CCS Concepts: • **Computing methodologies** → *Computer vision representations*; **Image compression**.

Additional Key Words and Phrases: point cloud, geometry compression, network, learning

ACM Reference Format:
Tianxin Huang, Jiangning Zhang, Jun Chen, Zhonggan Ding, Ying Tai, Zhenyu Zhang, Chengjie Wang, and Yong Liu. 2022. 3QNet: 3D Point Cloud Geometry Quantization Compression Network. *ACM Trans. Graph.* 37, 4, Article 187 (August 2022), 13 pages. https://doi.org/10.1145/3550454.3555481

*denotes that Yong Liu is the corresponding author.

Authors' addresses: Tianxin Huang, Zhejiang University, China; Jiangning Zhang, Zhejiang University, China; Jun Chen, Zhejiang University, China; Zhonggan Ding, Tencent youtu Lab, China; Ying Tai, Tencent youtu Lab, China; Zhenyu Zhang, Tencent youtu Lab, China; Chengjie Wang, Tencent youtu Lab, China; Yong Liu, Zhejiang University, China, yongliu@iipc.zju.edu.cn.

0730-0301/2022/8-ART187 $15.00
https://doi.org/10.1145/3550454.3555481

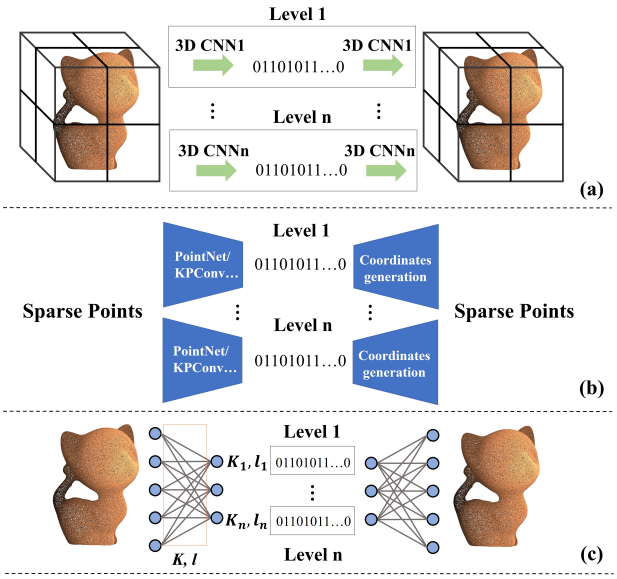

Fig. 1. Main differences between existing learning-based compression frameworks and ours. Voxel-based methods (a) introduce extra precision loss from voxelized 3D CNN, while point-based methods (b) are mainly designed for sparse points and have relatively poor robustness. Besides, both of them need to train a separate network for each compression level, which is quite inconvenient. As a point-based method, 3QNet (c) can flexibly change compression levels with the same network by adjusting the hyper-parameter K and l after training. It has good robustness and can process dense point clouds with different spatial distributions.

1 INTRODUCTION

With the rapid development of real-time 3D sensors like LiDAR and depth camera, 3D point clouds have been widely used in applications such as SLAM [Cadena et al. 2016], 3D reconstruction [Gropp et al. 2020] and object detection [Qi et al. 2018; Shi et al. 2019]. Increasement of points leads to the increment of transmission burden and saving cost. In this case, well-performed point cloud compression (PCC) algorithms with high efficiency and low cost are required. PCL [Rusu and Cousins 2011] is a widely-used library including lots of typical operations on point clouds. It provides an octree-based track for point cloud compression. Draco [Galligan et al. 2018] is proposed by Google for high efficient 3D model compression, which can efficiently handle both 3D mesh models and point clouds. MPEG

Group starts two representative point cloud compression methodologies, including geometry-based PCC (G-PCC) and video-based PCC (V-PCC) [Graziosi et al. 2020]. G-PCC encodes point clouds based on spatial structures of 3D point clouds with the octree, while V-PCC works by processing 2D images projected from 3D point Clouds. These non-learning-based compression algorithms use different methods to quantize and encode coordinates directly. They have good robustness for point clouds with different spatial distributions, while the pre-defined compression rules may limit the further improvements of compression performances.

Since the rise of deep learning-based methods, many researches have explored to compress 3D point clouds with neural networks. Most of them [Quach et al. 2020; Wang et al. 2021a,b] are based on the voxel representation, while the others [He et al. 2022; Huang and Liu 2019] try to deal with the coordinates directly. Learning-based methods can often achieve better compression performances than non-learning-based algorithms by learning a more memory-saving encoding strategy from the training data. However, there is still room for improvements in these methods. Generally speaking, voxel-based methods often suffer from voxel precision loss due to the coordinate-voxel transformation. Though point-based methods avoid the precision loss, existing techniques [He et al. 2022; Huang and Liu 2019] mainly work on sparse point clouds with similar spatial distributions as training data. In real applications, it is not realistic to always acquire training data with the same distribution as the test scenario, where the compression of relatively dense point clouds is more valuable and challenging. Besides, we often need multiple compression levels to balance the encoding bits and distortions. These learning-based methods need to train a separate network for each compression level, which is quite inconvenient.

In this work, we propose a new learning-based point cloud compression framework named 3QNet to get over mentioned problems. The main differences between 3QNet and existing compression methods are presented in Fig. 1. 3QNet has better robustness than existing point-based methods and can flexibly change the compression level with the same trained network. After training on simple and sparse models, 3QNet can achieve good performances on dense point clouds with quite different spatial distributions, such as objects, indoor and outdoor scenes. For dense models, we propose a model breaking strategy (MBS) to divide the whole models into smaller patches, where the acquired patches are further processed separately with compression and decompression networks. As 3D shapes are composed of multiple similar tiny local shapes, we propose Hierarchical Compression and Progress Decompression to compress and decompress the point clouds by their local geometrical shapes. By designing detachable multilevel compression and decompression networks, we can naturally control the balance between encoding bits and shape distortions by adjusting the network level after a single training. The generic codebook composed of multiple common features is learned from different models during the training process. During the compression period, local features are extracted with Hierarchical Compression and encoded with their indexes of nearest neighbors in the codebook. In other words, local features are quantized and encoded with the codebook in the feature space. Unlike numerical quantization operations in [He et al. 2022; Wang et al. 2021a,b; Wen et al. 2020], quantization with the codebook can use indexes including fewer bits to encode extended features, which is appropriate to encode multiple local features. Then during the decompression period, local features are decoded through the codebook with the encoded indexes, which are used to recover local shapes with Progress Decompression. The local shapes are finally combined back into decompressed results. The main contributions of this work can be summarized as follows.

- We propose a novel point-based 3D point cloud geometry compression framework, which can be efficiently applied to different dense point clouds;
- We introduce a codebook-based encoding method to transform features into more memory-saving binary codes;
- By defining detachable hierarchical compression and progress decompression networks, our framework is flexible to change the compression level after only one training.
- According to the experiments, after training on small and sparse models, our method can work well on point clouds with different spatial distributions without any further fine-tuning, which shows its good robustness.

2 RELATED WORKS

2.1 Non-learning-based Geometry Compression

Meshes and point clouds are two widely-used 3D geometric representations. Mesh models are composed of vertexes and faces constructed by the connectivity/topology between vertexes. Mesh compression methods [Rossignac 1999; Taubin and Rossignac 1998; Touma and Gotsman 1998] are often achieved by saving the memory of topology through intermediate representation, where the vertexes can also be compressed with the topological information. Point clouds are only composed of discrete points in 3D space without any extra topology. Non-learning-based point cloud compression methods [Cao et al. 2019] usually works by encoding the point clouds with manually-defined rules and data structures such as octree and kd-tree. PCL [Rusu and Cousins 2011] and the octree geometry codec in MPEG standard G-PCC [Graziosi et al. 2020] are two commonly used methods based on octree, while Draco [Galligan et al. 2018] released by Google uses a kd-tree to divide the space and encodes points according to the occupancy of divided spaces. Since point clouds can be regarded as sampling results from surfaces, the trisoup geometry codec in MPEG standard G-PCC [Graziosi et al. 2020] improves the subjective quality at less encoding bits by considering the underlying triangles. Some algorithms do not compress point clouds based on 3D geometry structures directly. They project point clouds to 2D images and then compress them with video compression algorithms, such as MPEG standard V-PCC [Graziosi et al. 2020]. Some works [Thanou et al. 2016] also explore the compression of dynamic point cloud sequences by introducing the motion estimation between successive frames. As these non-learning-based methods encode coordinates by pre-defined rules, their encoding performances may be limited, while they are often quite robust for different kinds of point clouds.

2.2 Learning-based Geometry Compression

Recently, with the development of deep learning architectures, networks have been developed to reconstruct [Puang et al. 2022;

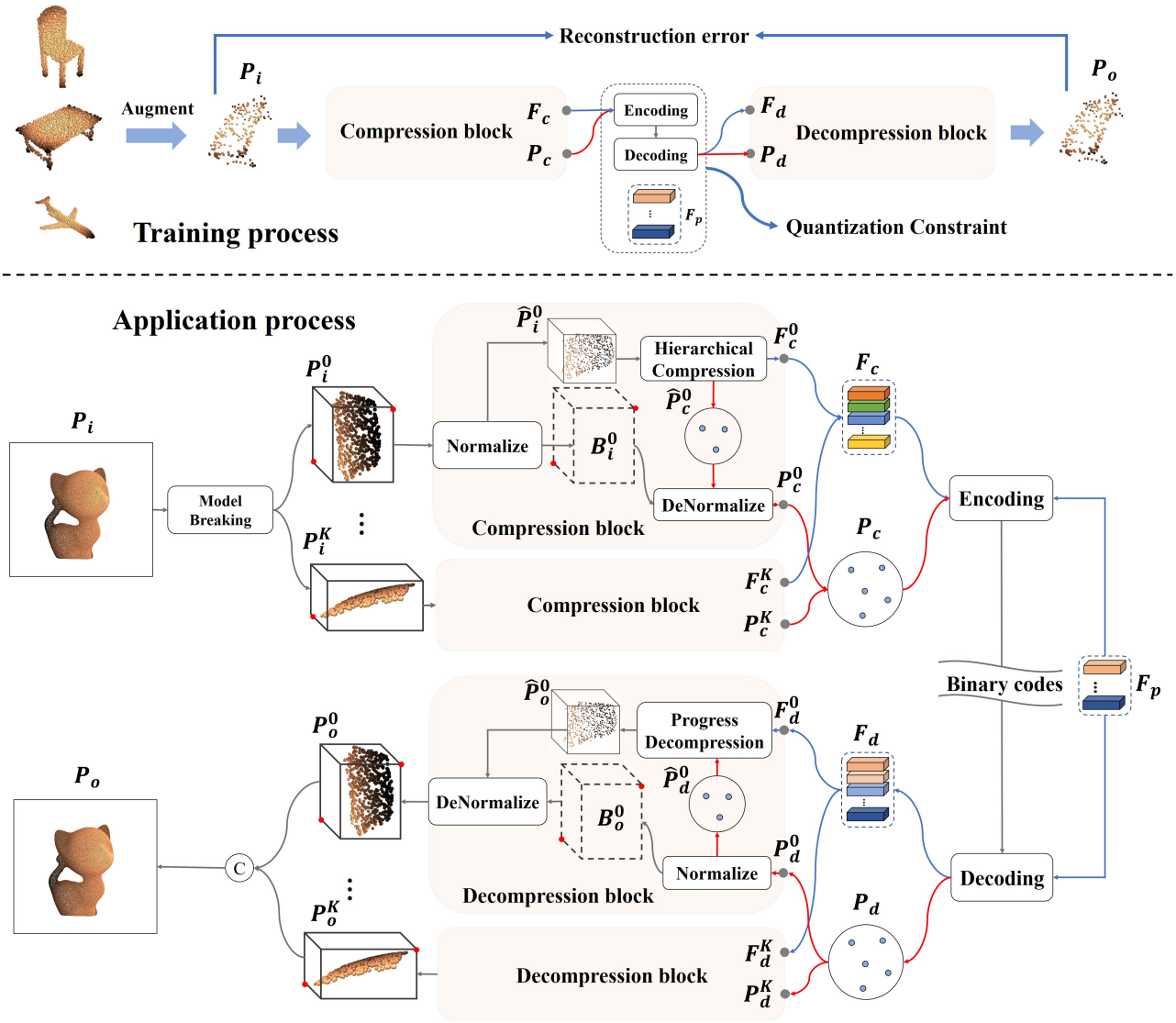

Fig. 2. The training and application processes of 3QNet. © denotes concatenation. Model Breaking Strategy (MBS) is our proposed strategy to break the whole point clouds into patches, while Hierarchical Compression and Progress Decompression are the networks for the compression or decompression of each patch, respectively. **During the application process**, The input model P_i is broken into patches $P_i^0 \sim P_i^K$ with MBS and parallel processed with Compression and Decompression blocks. Take patch P_i^0 as an example, it is normalized into $\hat{P}_i^0$ with bounding box B_i^0. Then $\hat{P}_i^0$ is transformed into local features F_c^0 and center points $\hat{P}_c^0$. Center points $\hat{P}_c^0 \sim \hat{P}_c^K$ are denormalized with corresponding bounding boxes and combined into P_c. $F_c^0 \sim F_c^K$ are combined into F_c, which would be encoded by the codebook F_p together with P_c to binary codes. During decompression, decoded local features $F_d^0 \sim F_d^K$ and center points $P_d^0 \sim P_d^K$ will be used to reconstruct patches $P_o^0 \sim P_o^K$ and concatenated into decompressed results P_o. **During the training process**, the codebook F_p as well as networks in compression and decompression blocks are trained with sparse and simple shapes by the reconstruction error and Quantization constraint, where the training data are augmented with random sampling and rotation. More details about the encoding and decoding processes can be found in Sec. 3.3.

Tatarchenko et al. 2017], complete [Dai et al. 2017b] or generate [Luo and Hu 2021] point clouds in an auto-encoder style. Based on similar point cloud auto-encoder approaches, learning-based geometry compression methods are proposed to improve compression performances by learning to extract memory-saving representations from point clouds. Some of them [Nguyen et al. 2021; Quach et al. 2020; Wang et al. 2021a,b] transform the original point clouds to voxel models and adopt 3D CNN to encode their geometry structures to binaries, which introduces extra precision loss. On the opposite side, [Huang and Liu 2019] attempts to deal with coordinates directly based on PointNet++ [Qi et al. 2017b]. The method performs well on small and sparse models, while it may have limited robustness because the global feature adopted for reconstruction may have difficulty generating accurate contours on unseen shapes. Though

[He et al. 2022] improves the robustness by preserving the contours with center points and introducing more complicated operations such as the attention mechanism, its application is still limited to sparse points in blocks partitioned from original dense point clouds. These point-based compression methods avoid precision loss in voxel-based networks, while their complex network structures limit the further adoption on dense points. Octsqueeze [Huang et al. 2020] and VoxelcontextNet [Que et al. 2021] are a series of works for the outdoor scenes compression, which propose networks to improve the entropy encoding performances in the octree-based compression. Muscle [Biswas et al. 2020] further introduces spatio-temporal relationship between multiple scans of outdoor scenes to reduce the bitrate of both geometry and intensity values. Depoco [Wiesmann et al. 2021] develops a auto-encoder styled compression framework for dense point cloud maps, which directly uses the extracted features and associated points from encoder as the compressed data. These works perform well on outdoor scenes, while they still need to train the networks on outdoor scans with similar distributions as test scenario. Besides, most existing learning-based compression algorithms use a hyper-parameter in the training loss to balance encoding bits and distortions for different compression levels. It means they need multiple training processes to acquire networks for different compression levels. In this work, we aim to build a method capable of processing point clouds with different spatial distributions after a single training process, which does not further require training data corresponding to test scenarios. We focus on the lossy compression [Huang and Liu 2019; Quach et al. 2020; Wang et al. 2021a,b; Wen et al. 2020] with limited encoding bits.

3 METHODOLOGY

In this section, we introduce the technical details of 3QNet. The input point cloud to be compressed P_i is broken into multiple patches $P_i^0 \sim P_i^K$ with the proposed Model Breaking Strategy (MBS), where K is the desired partition number. The details of MBS will be presented in Sec. 3.1. Then the partitioned patch $P_k, k \in \{0, 1, 2 \cdots K\}$ are normalized to $-1 \sim 1$ with the bounding box B_k as:

$$\hat{P}_i^k = \frac{2P_i^k - (B_i^k[1] + B_i^k[0])}{B_i^k[1] - B_i^k[0]}, \tag{1}$$

where $B_i^k = [min(P_i^k), max(P_i^k)]$ denotes the bounding box of P_i^k determined by the bottom and top corner points. For convenience, all normalized patches are marked with a hat. Center points $\hat{P}_c^k$ and local representations F_c^k are extracted from $\hat{P}_i^k$ with Hierarchical Compression. To encode the center points of multiple partitioned patches together, we combine the denormalized center points from different regions as a coarse point clouds P_c and encode P_c as a whole. Each patch will be assigned an id as an extra attribute during compression to help distinguish points from different patchs. The id is just presented from 1 to K during compression, which will be attached and compressed as an attribute together with coordinates. The same id does not get shared information between different models. The denormalization process can be defined as:

$$P_c^k = \frac{(B_i^k[1] - B_i^k[0])\hat{P}_c^k + B_i^k[1] + B_i^k[0]}{2}. \tag{2}$$

P_c and F_c are then encoded to binary codes with the learned codebook F_p. During the decompression, quantized center points P_d and local features F_d are decoded from the binary codes. P_d is then partitioned to $P_d^0 \sim P_d^K$ again by the patch id. By the application of patch id, all quantized center points can be processed together as P_c. As the encoding of local representations can also be parallel encoded with the codebook F_p, our method can process multiple patches simultaneously instead of processing patch by patch like subdivision-based works such as [He et al. 2022; Quach et al. 2020]. The bounding box B_o^k for k_{th} partitioned patch is then recovered by normalizing P_d^k following Eq. 1 with:

$$B_o^k = [min(P_d^k), max(P_d^k)], \tag{3}$$

where $k \in \{1, 2, 3, ..., K\}$. The normalized $\hat{P}_d^k$ and decoded F_d^k would be used to recover normalized patch $\hat{P}_o^k$, which is denormalized back to the original scale again with B_o^k. Note that B_i^k and B_o^k are ensured consistent with Boundary Sampling (BDSam) mentioned in Sec. 3.2.1. Finally, all denormalized results are concatenated to reconstruct decompressed results P_o. By dealing with only normalized point clouds with networks in Hierarchical Compression and Progressive Decompression, the algorithm robustness is guaranteed.

3.1 Model Breaking Strategy

To apply the algorithm to dense point cloud models, some existing methods [Wang et al. 2021b; Wen et al. 2020] have proposed available algorithms to partition dense models into small blocks. However, these methods partition model and cluster patches directly on original dense models, which have quite limited efficiency. We design a Model Breaking Strategy (MBS) to partition models more efficiently. The pipeline of MBS is presented in Fig. 4. We divide the point cloud P into multiple sub-areas with 8 planes perpendicular to the x-y plane as shown in Fig. 4, where the subsequent operations are parallel implemented on each sub-area to improve the efficiency. Then, we down-sample the j_{th} divided point clouds P_j to sparser point sets and find K_j centers C_j by the K-means algorithm, while K_j is calculated with the Space estimation module by considering the sizes of sub-areas, $j \in \{0, 1, 2, \cdots, 7\}$. Applying K-means to downsampled P_j would improve the efficiency over clustering on P_j directly. Specifically, if we define the desired divided number for P as K, then for each sub point cloud P_j, the estimated divided number can be defined as:

$$K_j = \frac{sum \|P_j - mean(P_j)\|_2}{\sum_{j=0}^{7} sum \|P_j - mean(P_j)\|_2} \cdot K, \tag{4}$$

where $mean(\cdot)$ denote the average operation to calculate the center of P_j. We can see that bigger sub-areas will get more clustering centers to cover corresponding shapes better. Finally, P_j are divided into K_j parts according to their neighbors in C_j. Original point cloud P is broken into K patches efficiently under the parallel partitioning.

3.2 Compression and Decompression Networks

3QNet works by compressing multiple sub point clouds partitioned by MBS. In this section, we present the specific structures of the

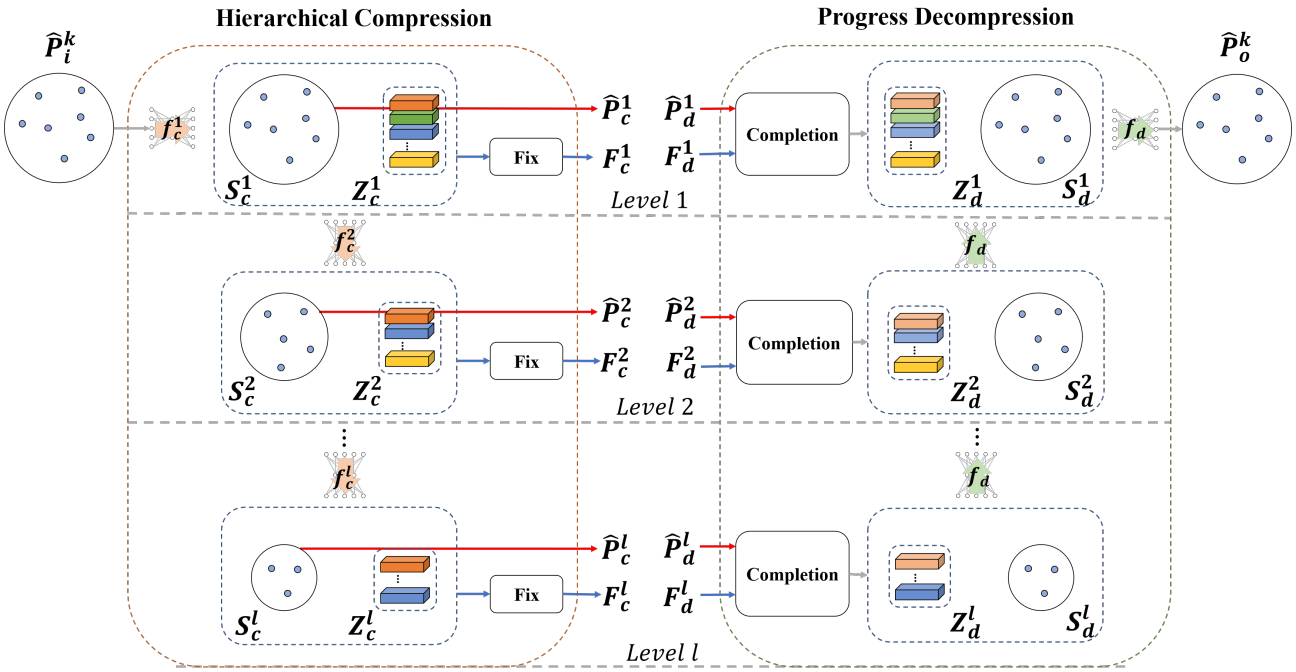

Fig. 3. The structures of Hierarchical Compression and Progress Decompression. Fix and completion are two modules including shallow networks to refine the center points or local representations. The geometrical features are more and more abstract from level $1 \sim l$. $f_c^1(\cdot) \sim f_c^l(\cdot)$ are compression modules in levels $1 \sim l$, while $f_d(\cdot)$ is the parameter-shared decompression module. The structures are detachable to adjust the compression level. One network level would be selected during application to switch between different compression levels, which means $\{\hat{P}_c^k, F_c^k, \hat{P}_d^k, F_d^k\} \in \{\{\hat{P}_c^1, F_c^1, \hat{P}_d^1, F_c^1\}, \{\hat{P}_c^2, F_c^2, \hat{P}_d^2, F_c^2\}, \cdots, \{\hat{P}_c^l, F_c^l, \hat{P}_d^l, F_c^l\}\}$, while all networks are optimized together during training.

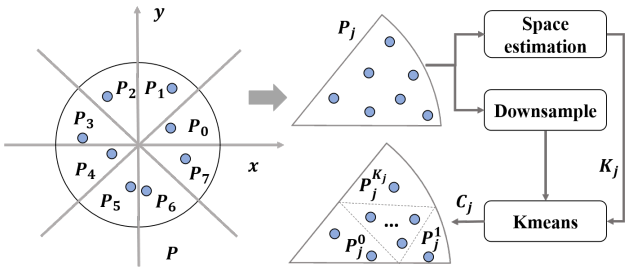

Fig. 4. The process of Model Breaking Strategy. K_j and C_j are the number of center points and coordinates of center points in j_{th} divided point cloud $P_j, j \in \{0, 1, 2, \cdots, 7\}$. We divide it at the center like a pie to ensure that there are points in each sub-areas.

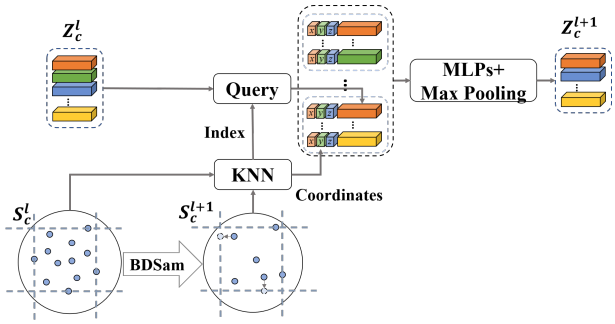

Fig. 5. The structure of compression module in level l, which is $f_c^l(\cdot)$ in Hierarchical Compression as shown in Fig 3. S_c^l is sampled into center points S_c^{l+1} with Boundary Sampling (BDsam) by moving the farthest point sampling (FPS) sampled points nearest to boundaries towards boundary positions. Then Z_c^l, S_c^l are aggregated around S_c^{l+1} with KNN like PointNet++ [Qi et al. 2017b] and abstracted into Z_c^{l+1} by MLPs and pooling.

compression networks for k_{th} sub point cloud $\hat{P}_i^k$ and decompression networks for $\hat{P}_o^k$ from Fig. 2, where $k \in \{1, 2, 3, ..., K\}$. The whole structures of our compression and decompression networks are presented in Fig. 3. Both Hierarchical Compression and Progressive Decompression are composed of multiple levels $1 \sim l$. Take level l as an example, Hierarchical Compression use compression modules $f_c^1(\cdot) \sim f_c^l(\cdot)$ to extract abstract representations Z_c^l and center points S_c^l during compression, while a fix module is used to refine Z_c^l into F_c^l. $\hat{P}_c^l$ is kept same as S_c^l during compression. In Progress Decompression, decoded local representations F_d^l and center points P_d^l are refined to intermediate representations Z_d^l and S_d^l with completion module, which are then expanded with multiple decompression modules $f_d(\cdot)$ to acquire decompressed results P_o^k. Networks in all levels are trained together during training. As shown in Fig. 2, we use Hierarchical Compression to extract P_c^k and F_c^k and Progress Decompression to process P_d^k and F_d^k. During application, we select a specific level between $1 \sim l$, that is $\{\hat{P}_c^k, F_c^k, \hat{P}_d^k, F_d^k\} \in \{\{\hat{P}_c^1, F_c^1, \hat{P}_d^1, F_c^1\}, \{\hat{P}_c^2, F_c^2, \hat{P}_d^2, F_c^2\}, \cdots, \{\hat{P}_c^l, F_c^l, \hat{P}_d^l, F_c^l\}\}$.

3.2.1 Hierarchical Compression. Hierarchical compression plays the role of aggregating features around multiple sampled centers. To

aggregate features from the original models, we adopt local feature extraction structure in PointNet++ [Qi et al. 2017b] as compression module, which is shown as $f_c^1(\cdot) \sim f_c^l(\cdot)$ in Fig. 3. The specific structure of compression module is presented in Fig. 5. For level l, given the center points S_c^l and local structural representations Z_c^l, compression module finds center points for the next level S_c^{l+1} by Boundary Sampling (BDSam), which moves the farthest point sampling (FPS) sampled points nearest to boundaries towards boundary positions as shown in Fig. 5. BDSam is introduced to guarantee boundaries invariant between different levels so that the boundary of $\hat{P}_c^k = \hat{P}_c^l = S_c^l$ is $-1 \sim 1$ as $\hat{P}_i^k$. As we have $P_c^k \approx P_d^k$ by lossless coordinates encoding and decoding, then we can ensure $B_o^k = normalize(P_d^k) \approx normalize(P_c^k) = B_i^k$. It can avoid extra memory to save the boundaries, which is unaffordable if there are many patches. In this work, we sample 1/4 center points in each level of the compression module. Local coordinates S_c^l and representations Z_c^l are aggregated with KNN around center points S_c^{l+1}, which are abstracted into new local representations Z_c^{l+1} for next level with Multi Layer Perceptrons (MLPs). Note that we only move a few points nearest boundaries to boundaries. Other points still follow farthest point sampling to have a uniform distribution around the patch. With slightly bigger K for KNN, we can ensure that almost all points can be covered during compression. As there is not local representation at level 1 in Fig. 3, only coordinates will be used to extract Z_c^{l+1} in this condition.

3.2.2 Progress Decompression. To reconstruct models with multiple resolutions and reduce the parameters, we propose a multilevel Progress Decompression structure which is composed of multiple parameter-shared decompression modules $f_d(\cdot)$ as shown in Fig. 3. The structure of decompression module is presented in Fig. 6. For level l, the local representations Z_d^l are divided into 4 parts symmetric to the 1/4 sampling in the compression module, which are combined with center points S_d^l to make hybrid representations by MLPs. The hybrid representations are then concatenated with the corresponding global feature acquired by pooling and used to predict offsets for S_d^l and Z_d^l in order to expand them to higher resolution S_d^{l-1} and Z_d^{l-1} in compression level $l-1$.

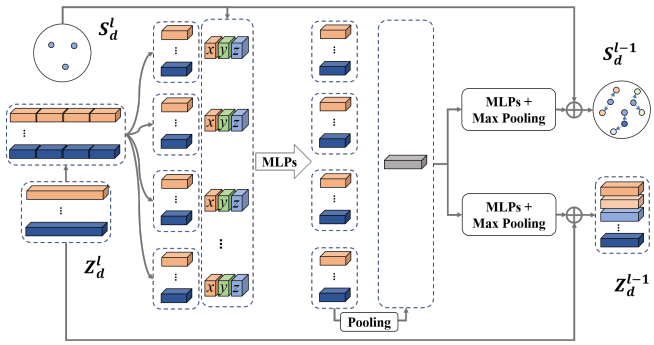

Fig. 6. The structure of decompression module, which is $f_d(\cdot)$ in Progress Decompression as shown in Fig 3. S_d^l and Z_d^l in compression level l are expanded to higher resolution S_d^{l-1} and Z_d^{l-1} in compression level $l-1$.

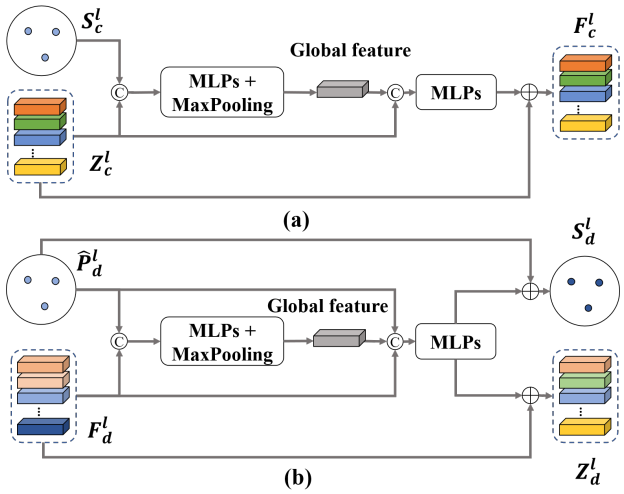

Fig. 7. Structures of the fix module (a) and completion module (b). Fix module introduce global information to feature F_c^l from the compression module by fusing features of S_c^l and Z_c^l, while completion module refine both decoded center points $\hat{P}_d^l$ and F_d^l to predict S_d^l and Z_d^l.

3.2.3 Fix and Completion. Though the whole algorithm seems to work well with just mentioned modules, there are still some problems. Compression module extracts representations separately in multiple local regions. For level l, the extracted representations Z_c^l have relatively weak relationships with the global shape. Besides, the encoding and decoding process may introduce extra information loss to local representations F_d^l and center points $\hat{P}_d^l$. Considering these problems, we introduce fix module to refine the extracted local representation with global shape representation, while we propose completion module to make up for the information loss from encoding and decoding processes. The structures of fix module and completion module are presented in Fig. 7. Their organizations are relatively similar, where both fix and completion modules extract global representations to estimate the offsets of coordinates or local representations. Fix module does not change the coordinates in S_c^l because the original sampled results of BDsam mentioned in Sec. 3.2.1 are needed to guarantee boundaries invariant, while completion module changes both coordinates and representations to make up for their possible information loss.

3.3 Encoding and Decoding

The encoding and decoding operations are used to achieve the transformation between extracted structural information and binary codes. The transformation includes operations for center points P_c and local representations F_c. In this work, we introduce a different encoding method. Unlike numerical quantization for features in other works, we quantize continuous local representations F_c into discrete representations F_d by their nearest neighbors in the learned codebook F_p. Then the probabilities of each item in F_p are counted and used to encode the indexes of F_d with Arithmetic coding [Witten et al. 1987] into C_f. As different point clouds may have variant statistic probabilities of quantized representations, we also encode the probability sequence with sparse coding following [Huang and Liu 2019] to C_p. Except for the encoding of representations, center

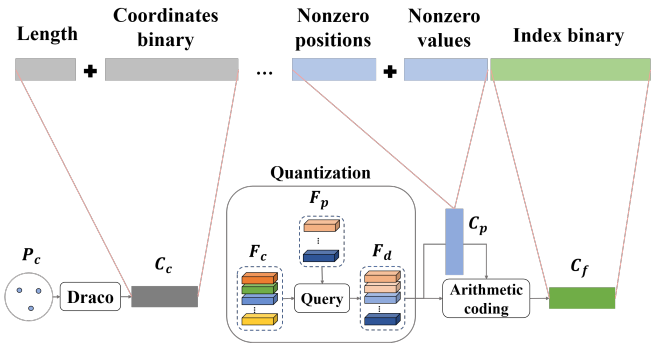

Fig. 8. The encoding process from center points and local representations to binary codes. P_c and F_c denote the center points and local representations from original point clouds, respectively, while F_p is the learned codebook to quantize F_c to F_d. The binary codes consist of binaries from coordinates C_c, index probability C_p, and feature index C_f.

points P_c are encoded into binary C_c by Draco algorithm [Galligan et al. 2018] under the lossless encoding with high precision, where a 16 bits binary length is used to record the length of coordinates binary. Finally, the binary codes are consist of C_c, C_p, and C_f as shown in Fig. 8. The decoding process is the reverse process of encoding, which is presented as follows.

1. Read the first 16 bits in binary codes as the coordinate length;
2. Decode center points P_d following the read length;
3. Recover the probability sequence by len(F_p) bits nonzero positions and following nonzero values;
4. Decode the Indexes with the probability sequence in later binary codes by Arithmetic decoding and transform decoded indexes to local representations F_d according to F_p.

As the encoding and decoding processes are non-differentiable, we use the quantized representations during the training process to optimize the codebook as well as compression and decompression networks. We relax the quantization operation based on the nearest neighbor in the codebook to k neighbors during back-propagation, which can be defined as:

$$F_d = f_{sg}(\sum_{i=1}^{N_k} \frac{exp(-\frac{\|F_k^i - F_c\|_2}{\sigma + |\delta_k|^2})}{\sum_{i=1}^{N_k} exp(-\frac{\|F_k^i - F_c\|_2}{\sigma + |\delta_k|^2})} \cdot F_k^i - F_n) + F_n, \tag{5}$$

where $f_{sg}(\cdot)$ means to stop gradient during the inference. F_k^i and F_n denote the i_{th} representation in the k neighbors and the nearest neighbor of F_c in the codebook, respectively. With Eq. 5, the quantization is accurately calculated during the forward process with F_n, then approximated with F_k^i to optimize items in the codebook during back-propagation. N_k is the pre-defined number of k neighbors. δ_k is a hyper-parameter controlling the distribution of weights for F_k^i, while σ is a tiny value to protect denominators from being 0.

3.4 Compression Level Selection

To adapt the compression algorithm to different conditions, we need to balance the decompressed shape distortions and compressed encoding bits. For example, encoding bits should be sacrificed for fewer distortions in applications with higher precision requirements, while distortions are allowed when fewer encoding bits are needed under limited transmission bandwidth. In other words, we need to define multiple compression levels to adapt to different applications. Existing learning-based methods [Huang and Liu 2019; Quach et al. 2020; Wang et al. 2021a,b] use different pre-defined hyper-parameters to adjust the trade-off between encoding bits and distortions, which needs a separate training for each compression level and is quite inconvenient. By designing detachable multilevel compression and decompression networks, the compression level can be naturally adjusted by selecting different network level l after training. As the network level is limited, we further change the number of patches K divided by MBS to get more compression levels. Specifically, we select the compression level as:

$$K, l = D_{cl}(level), \tag{6}$$

where $D_{cl}\{\cdot\}$ is a manually-defined dictionary, K and l are the number of partitioned patches and selected levels in Fig. 3, respectively. $level$ is the compression level. By changing K and l following the pre-defined dictionary according to the selected $level$, we can flexibly adjust the balance between distortions and encoding bits without repeated training process.

3.5 Loss Function

The Loss Function of 3QNet is the weighted sum of multiple components, including multi-scale reconstruction error, quantization constraint, and expansion constraint.

3.5.1 Multi-scale reconstruction error. The commonly used Chamfer Distance (CD) [Fan et al. 2017] mainly works on the contours of models, which will create quite non-uniform results. Earth Mover Distance (EMD) tries to find a bijection between two point sets with optimal algorithms. Though EMD performs well in constraining overall shapes, it suffers from great memory and time cost. In this work, we follow the multi-scale loss proposed in [Huang and Liu 2019] to construct our multi-scale reconstruction error, which can be presented as follows:

$$\mathcal{L}_{MCD}(S_1, S_2) = \sum_i \sum_k \frac{\xi_k}{N} \sum_{j=1}^{N} \mathcal{L}_{CD}(S_{i,1}^{j,k}, S_{i,2}^{j,k}), \tag{7}$$

where $S_{i,1}^{j,k}$ and $S_{i,2}^{j,k}$ are the local point sets with k points around j_{th} sampled center under i_{th} resolution, respectively. N is the number of captured local regions and ξ_k is the coefficient of errors of k points local regions. As we train the Hierarchical Compression and Progressive Decompression networks of multi levels together, we add the multi-scale errors in different levels together to get the final reconstruction errors as:

$$\mathcal{L}_{MS} = \sum_l \mathcal{L}_{MCD}(S_i^l, S_o^l), \tag{8}$$

where S_i^l and S_o^l denote the input compressed and output decompressed point clouds in l_{th} level, respectively. Note that we actually use all network levels during training, where the cost is affordable according to our experiments.

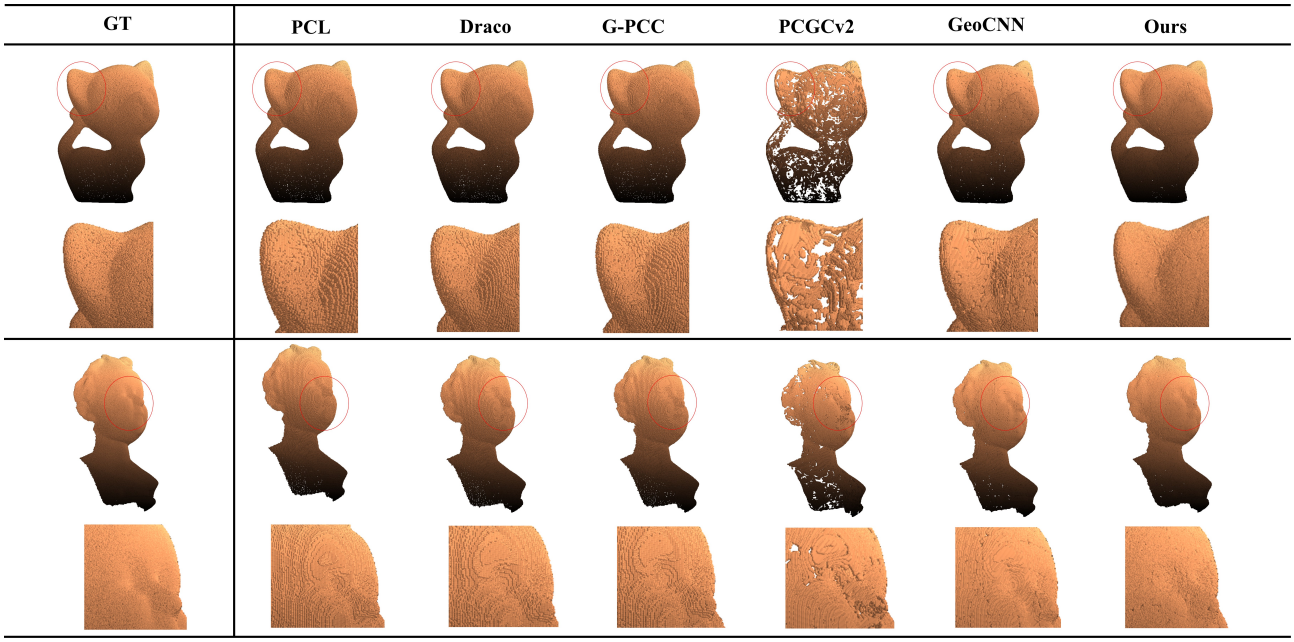

Fig. 9. The qualitative comparison on object models. All results are acquired under the compression level with about 1 bpp. We can see that our method tends to produce smoother shapes with fewer defects.

Table 1. Comparisons with representative compression methods on object models. The '+' and '-' denote the average improvements or depreciation of 3QNet against the compared methods, while the percentages denote the relative changes.

		Non-learning			Learning	
Data	Metrics	PCL	Draco	G-PCC	PCGCv2	GeoCNN
duck	D1	+3.22 (16.40%)	+8.95 (62.58%)	+1.28 (8.53%)	+8.37 (135.93%)	+2.17 (14.77%)
	D2	+5.12 (23.64%)	+9.71 (67.70%)	+3.47 (17.36%)	+5.19 (36.18%)	+3.68 (17.84%)
pierrot	D1	+2.38 (12.63%)	+7.99 (57.98%)	+0.38 (2.44%)	+8.93 (143.63%)	+1.47 (9.86%)
	D2	+3.09 (15.08%)	+7.66 (55.20%)	+1.34 (6.37%)	+4.79 (35.61%)	+1.68 (7.96%)
julius	D1	+3.23 (15.17%)	+8.78 (53.91%)	+1.19 (8.92%)	+4.79 (64.72%)	+1.02 (7.37%)
	D2	+3.36 (14.56%)	+7.72 (47.57%)	+0.84 (4.63%)	+0.58 (3.84%)	+0.38 (1.95%)
pig	D1	+3.37 (15.33%)	+8.98 (52.57%)	+1.54 (11.59%)	+3.99 (44.21%)	+1.62 (11.77%)
	D2	+4.93 (20.92%)	+9.43 (55.21%)	+2.55 (14.09%)	+0.59 (3.60%)	+2.26 (12.25%)
nicolo	D1	+3.20 (14.93%)	+9.01 (55.46%)	+1.19 (8.82%)	+4.50 (54.85%)	+0.93 (6.55%)
	D2	+4.12 (17.71%)	+8.69 (53.74%)	+1.67 (9.13%)	+0.53 (3.35%)	+0.40 (2.02%)
eros	D1	+2.66 (12.89%)	+8.17 (52.31%)	+0.87 (5.86%)	+6.21 (86.87%)	+1.15 (7.64%)
	D2	+2.52 (11.47%)	+7.05 (45.25%)	+0.70 (3.54%)	+2.33 (15.78%)	+0.85 (4.17%)
boy01	D1	+4.44 (19.22%)	+9.75 (52.51%)	+2.31 (19.98%)	+0.65 (6.00%)	+0.99 (7.93%)
	D2	+5.28 (21.45%)	+9.34 (49.89%)	+2.88 (18.54%)	-2.35 (14.13%)	+1.67 (10.44%)
kitten	D1	+3.05 (14.40%)	+8.65 (54.10%)	+0.92 (6.06%)	+5.45 (62.77%)	+1.22 (7.89%)
	D2	+3.65 (15.78%)	+8.06 (49.90%)	+1.06 (5.30%)	+1.15 (6.86%)	+0.94 (4.46%)
big_girl	D1	+4.34 (18.79%)	+9.81 (53.52%)	+2.23 (18.62%)	+0.66 (5.84%)	+1.04 (8.09%)
	D2	+5.20 (21.07%)	+9.44 (51.59%)	+2.86 (17.97%)	-1.99 (11.64%)	+1.68 (10.12%)
star	D1	+3.27 (16.15%)	+11.30 (88.85%)	+1.00 (7.04%)	+6.75 (88.41%)	+1.72 (12.04%)
	D2	+4.90 (21.54%)	+11.45 (89.79%)	+2.13 (11.16%)	+3.24 (22.09%)	+2.15 (10.81%)
mean	D1	+3.45 (16.36%)	+9.34 (59.52%)	+1.44 (10.48%)	+5.21 (64.70%)	+1.63 (11.55%)
	D2	+4.43 (19.43%)	+9.06 (57.63%)	+2.11 (11.48%)	+1.57 (10.07%)	+1.96 (10.15%)

3.5.2 Quantization constraint. Quantization constraint is used to manage the codebook and help learn an appropriate encoding strategy. According to the encoding process mentioned in Sec. 3.3, we should ensure the local structural representation are as close as possible to items of the codebook. Besides, as we use weighted results of N_k neighbors to approximate the nearest neighbor and train the variables in the codebook, we should constrain the distribution of weights to approximate the nearest neighbor for more efficient training, which can be achieved by reducing the hyper-parameter δ_k in Eq. 5. In consideration of all these constraints, the quantization constraint can be defined as:

$$\mathcal{L}_{QC} = \|F_c - F_d\|_2 + \|\delta_k\|_2, \tag{9}$$

3.5.3 Expansion constraint. In Progressive Decompression, high-resolution models gradually grow from the low-resolution points in an expansion process of each decompression module as shown in Fig. 6. We constrain the expansion distances between different resolution models to avoid large deviation from low-resolution shapes. The expansion constraint can be defined as:

$$\mathcal{L}_{EC} = \frac{1}{|S_d^{l-1}|} \sum_{x \in S_d^{l-1}, f: S_d^l \to S_d^{l-1}} \|x - f^{-1}(x)\|_2, \tag{10}$$

where the $f^{-1}(\cdot)$ means to find the corresponding center points of decompressed points. S_d^{l-1} and S_d^l are the point sets in level $l-1$ and l of decompression module as shown in Fig. 6.

3.5.4 Overall loss function. The final training loss is defined as:

$$\mathcal{L}_{all} = \mathcal{L}_{MS} + \lambda_1 \mathcal{L}_{QC} + \lambda_2 \mathcal{L}_{EC}, \tag{11}$$

where we set λ_1 and λ_1 as both 0.01 in this work.

4 EXPERIMENTS

4.1 Dataset and Implementation Details

To present a comprehensive evaluation, we conduct comparisons on 3 kinds of commonly used data including point cloud object models [Kopecki et al. 2011] following [Yu et al. 2018], indoor scenes [Dai et al. 2017a], outdoor scans [Behley et al. 2019] to evaluate the general performances for point clouds with different spatial distributions. We train our network on 12288 sparse models from 16 categories in ShapeNet dataset [Wu et al. 2015] containing 2048 points randomly sampled from the corresponding mesh models following [Yang et al. 2018]. This work is implemented with Tensorflow. The networks are optimized with Adam optimizer with a learning rate of 2e-4, where it may take about 200 epochs to converge to the final result. More details can be found in the supplement. In this work, we adopt the non-learning-based PCL [Rusu and Cousins 2011], Draco [Galligan et al. 2018], and G-PCC [Graziosi et al. 2020] as well as the learning-based PCGCv2 [Wang et al. 2021a] and GeoCNN [Quach et al. 2020] for comparison. Note that we do not introduce any extra fine-tuning on new datasets before evaluation.

4.2 Metrics

In this section, we describe the metrics adopted to evaluate the performances of our compression algorithm. Following the former related works [Quach et al. 2020; Wang et al. 2021a,b], we adopt bits per point (bpp) to evaluate the bit rate after compression and Peak Signal-to-Noise (PSNR) to measure the reconstruction performance after decompression. The bpp is defined as the average length of encoding bits taken by each point, which is:

$$bpp = \frac{L}{N}, \tag{12}$$

where L is the size of compressed binary and N is the number of points in the compressed point cloud.
Motivated by [Tian et al. 2017], in this work, PSNR is defined as:

$$PSNR = 10log_{10}(\frac{max_{x \in S_1} \|x - \hat{x}\|_2}{Dis(S_1, S_2)}), \tag{13}$$

where S_1 and S_2 are the ground truths and decompressed point clouds, respectively. $\hat{x}$ denotes the nearest neighbor of x in S_1. $Dis(S_1, S_2)$ is the two-way average distance between S_1 and S_2. There are two kinds of distance measurements. One calculates the point-to-point error, the other computes the point-to-plane error. The PSNRs calculated by point-to-point and point-to-plane distances are denoted as $PSNR_{D_1}$ and $PSNR_{D_2}$, respectively.

As decompression distortions evaluated by PSNR dynamically change with bpp when the encoded information varies, performances of compression methods are often evaluated by the bpp-PSNR curves. To present an overall quantitative evaluation for the bpp-PSNR curve, we calculate the BD-PSNR of our method compared to other methods following [Bjontegaard 2001; Quach et al. 2020], which reflects the our relative improvements against other methods. All subsequent quantitative comparisons are based on BD-PSNR(D1) and BD-PSNR(D2) calculated with $bpp\text{-}PSNR_{D_1}$ and $bpp\text{-}PSNR_{D_2}$ curves, respectively.

4.3 Experiments on Object Models

In this section, we compare our method with some representative point cloud compression methods on point clouds with about 160k uniformly sampled points from object models in Visionair repository [Kopecki et al. 2011]. BD-PSNR is used to show the performances of 3QNet compared to other methods. The quantitative results are presented in Table 1 and Fig. 11. We can see that our methods can achieve obvious improvements over other methods on BD-PSNR metrics calculated with both $PSNR_{D_1}$ and $PSNR_{D_2}$, which confirms the effectiveness of 3QNet. To clearer find the differences between different algorithms, we also conduct a group of qualitative comparisons in Fig. 9. All presented results are acquired under the basically same bpp. We can see that non-learning-based methods PCL, G-PCC, and Draco may produce regular and obvious distortions because of direct coordinates encoding by structures such as octree, while other learning-based methods PCGCv2 and GeoCNN also have relatively obvious surface defects like unexpected holes. By achieving encoding with the codebook in the feature space, 3QNet can avoid regular distortions in non-learning-based methods, while its network designation also helps create smoother decompressed results with fewer defects than other learning-based methods.

As training models from ShapeNet have quite different shapes with the evaluated models like shown in Fig. 2, it confirms the 3QNet has excellent robustness. Except the preservation of contours with center points, the robustness may also benefit from the learned common local representations in the codebook. By training to recover shapes with the learned local representations, the codebook can include common characteristics from diverse local shapes. As different geometrical structures may be composed of similar local shapes, the codebook learned from simple models may be used to reconstruct more complex shapes.

4.4 Experiments on Indoor scenes

To give a more comprehensive evaluation of the algorithm performance on real scene data which may have more complicated shapes and non-uniform distributions, we conduct comparisons on indoor scenes from Scannet [Dai et al. 2017a], which is composed of 1513 scenes with about 50k ∼ 200k points collected by RGBD cameras.

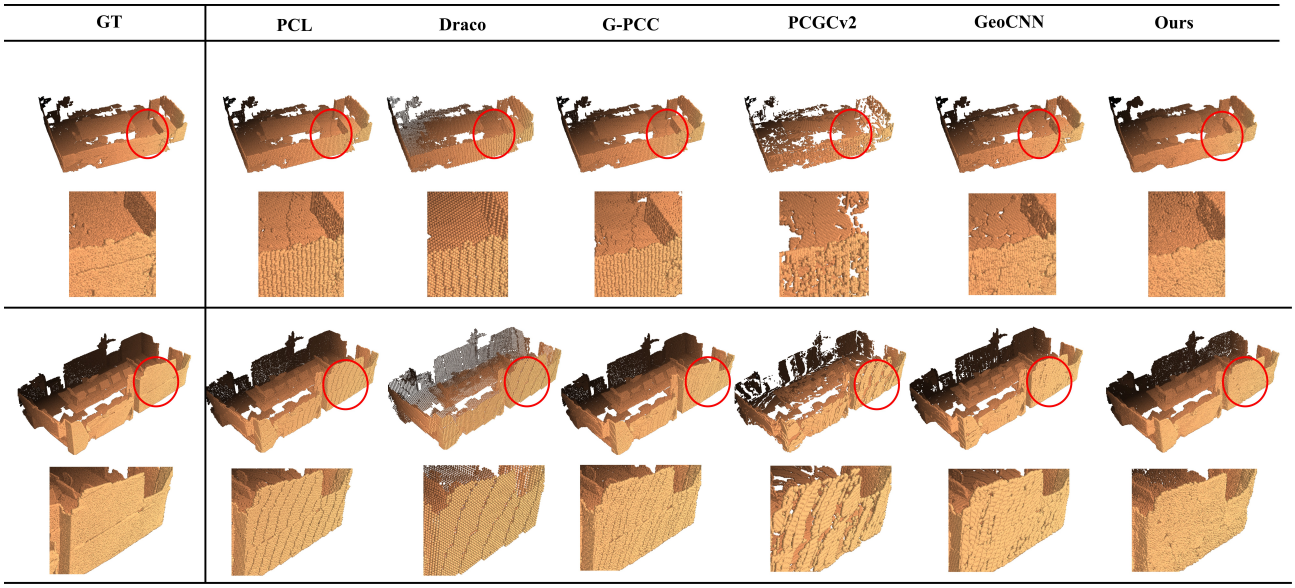

Fig. 10. The qualitative comparison on indoor scenes. All results are acquired under the compression level with about 1 bpp. Our method still creates smooth surfaces with fewer artifacts for the indoor walls.

Table 2. Comparisons with representative compression methods on Scannet.

		Non-learning			Learning	
Data	Metrics	PCL	Draco	G-PCC	PCGCv2	GeoCNN
Group 1	D1 D2	+3.12 (16.37%) +2.57 (13.51%)	+7.95 (54.60%) +7.40 (52.83%)	+1.16 (7.23%) +1.30 (6.10%)	+7.42 (107.70%) +4.44 (29.91%)	+1.86 (11.57%) +1.73 (8.02%)
Group 2	D1 D2	+3.53 (18.19%) +3.45 (18.12%)	+8.27 (55.34%) +7.96 (55.14%)	+1.51 (9.94%) +1.78 (8.67%)	+7.51 (114.02%) +4.48 (30.54%)	+2.11 (13.58%) +2.23 (10.57%)
Group 3	D1 D2	+2.78 (14.25%) +2.52 (13.19%)	+7.59 (50.19%) +7.27 (49.37%)	+0.86 (5.61%) +1.01 (4.85%)	+7.45 (115.99%) +4.46 (30.91%)	+1.26 (8.02%) +1.33 (6.24%)
Group 4	D1 D2	+3.10 (15.21%) +2.74 (13.67%)	+7.98 (50.18%) +7.43 (47.90%)	+1.14 (7.61%) +0.98 (4.84%)	+5.82 (73.48%) +2.56 (16.14%)	+1.21 (7.78%) +1.32 (6.35%)
Group 5	D1 D2	+3.13 (15.95%) +3.09 (15.79%)	+7.96 (52.03%) +7.72 (51.46%)	+1.30 (8.42%) +1.76 (8.52%)	+7.61 (114.32%) +4.33 (28.84%)	+1.88 (11.80%) +1.98 (9.18%)
mean	D1 D2	+2.88 (21.96%) +2.71 (14.60%)	+5.66 (46.84%) +5.79 (32.66%)	+1.22 (7.89%) +1.39 (6.71%)	+7.21 (104.99%) +4.12 (27.66%)	+1.69 (10.69%) +1.73 (8.14%)

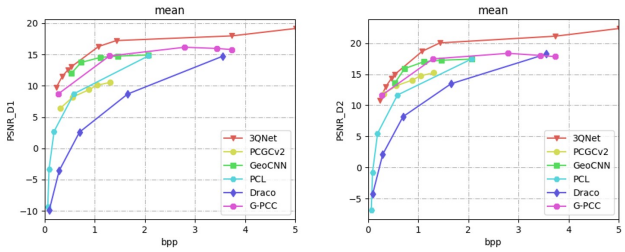

Fig. 11. The average bpp-PSNR curves on object models.

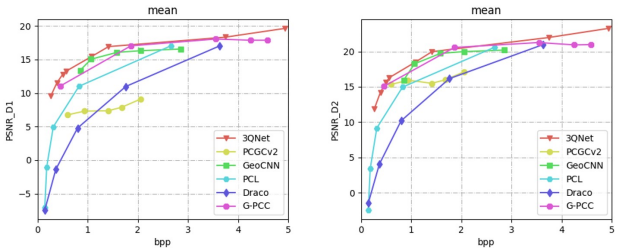

Fig. 12. The average bpp-PSNR curves on indoor scenes.

In this work, we follow [Qi et al. 2017a,b] and choose 312 scenes to evaluate our method, which is divided into 5 groups according to the number of points. The results evaluated on different groups are presented in Table 2 and Fig. 12. We can see that our method consistently achieves improvements over both non-learning and learning-based methods. Our method can produce smoother decompressed surfaces with fewer defects, as shown in Fig. 10.

4.5 Experiments on Outdoor Scenes

Kitti [Behley et al. 2019] is composed of 21351 real scans with 4.5 billion points collected by the Velodyne HDL-64 sensor. For outdoor scans and maps, most points belong to the ground, which contains little useful structural information and is sometimes removed in applications such as [Kong et al. 2019]. As we are mainly concerned about the objects above the ground such as buildings or humans,

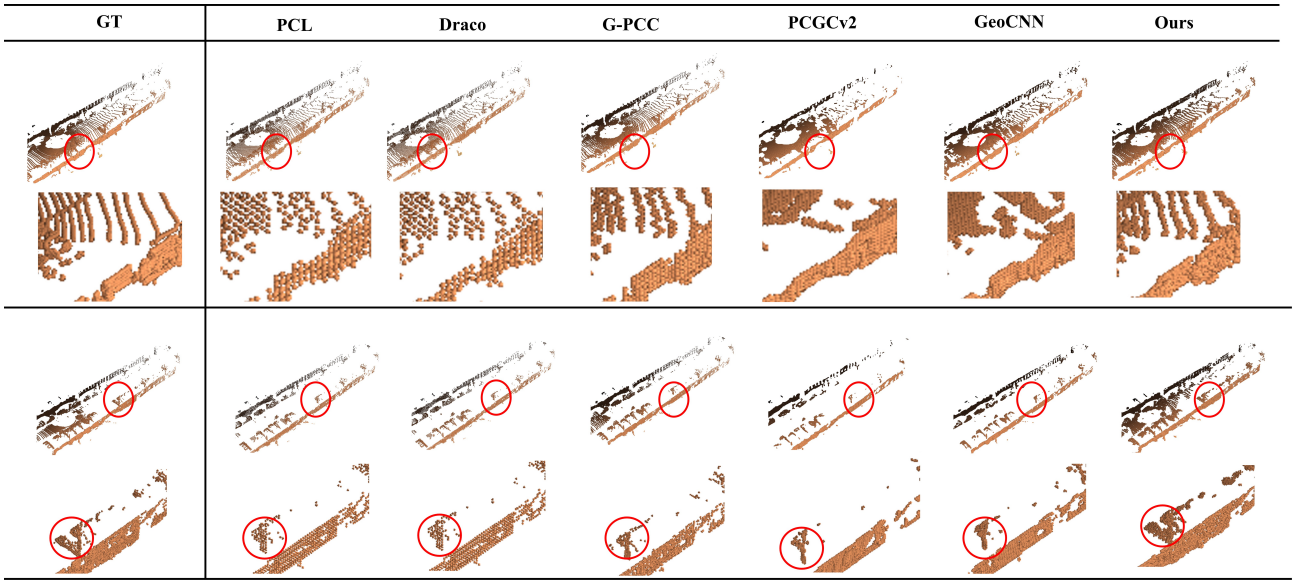

Fig. 13. The qualitative comparison on outdoor scenes. The two rows above and two lines below demonstrate a real scan before and after removing the ground with RANSAC, respectively. All results are acquired under the compression level with about 1 bpp. Our method can preserve some shape characteristics even without any requirements for outdoor training data.

we remove the ground with RANSAC [Derpanis 2010; Fischler and Bolles 1981] and evaluate distortions with remaining up-ground parts like shown in Fig. 14.

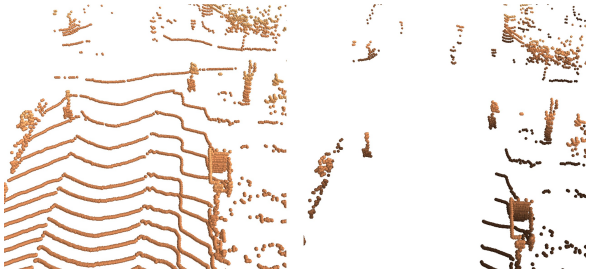

Fig. 14. An example of removing the ground with RANSAC. Most points under ground are removed to evaluate the distortions of above ground structures such as the wall, human and tree.

In this work, we use sequences 00, 02, 04, 06 from Kitti to construct 4 groups of test data by uniformly selecting 20 scans from each sequence. The results are presented in Table 3 and Fig. 15. The qualitative results are presented in Fig. 13. We can see that non-learning-based algorithms still have regular distortions, while learning-based methods may create over-smoothed results to lose some details. They may also lose some details close to the ground as circled parts, while our method can preserve these details well.

4.6 Ablation Study

In this section, we conduct the ablation study for proposed components. All performances are evaluated with the BD-PSNR(D1) and BD-PSNR(D2) against GeoCNN based on the models from Visionair repository [Kopecki et al. 2011] as mentioned in Sec. 4.3.

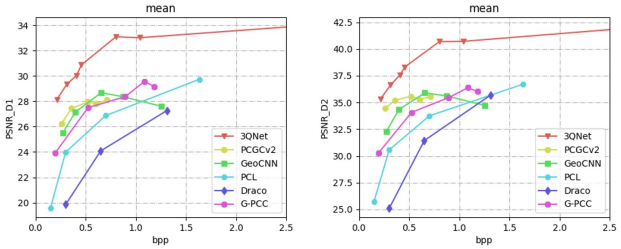

Fig. 15. The average bpp-PSNR curves on outdoor scans of Kitti.

4.6.1 Ablation study for the fix and completion modules. Fix module introduces global structural information to local representations in Hierarchical Compression, while completion module is adopted to make up information loss from encoding and decoding in Progress Decompression. To further confirm the effection of fix and completion modules, we compare the model performances with or without these modules. The results are presented in Table 4. We can see that both fix and completion module contribute to the final results.

4.6.2 Ablation study for the Decompression Module. Decompression module is an essential network in Progress Decompression to expand decoded local representations and center points to higher resolutions. In this section, we compare our decompression module with the commonly used expansion operations achieved by fully connected network [Achlioptas et al. 2018] and FoldingNet [Yang et al. 2018] to validate its effectiveness. The results are presented in Table 5. We can see the decompression module gets much better performances in our framework.

4.6.3 Ablation study for the loss function. In this section, we conduct an ablation study for both the basic organization and each component in the loss function. The results are presented in Table 6.

Table 3. Comparisons with representative compression methods on outdoor scans.

		Non-learning			Learning	
Data	Metrics	PCL	Draco	G-PCC	PCGCv2	GeoCNN
Group 1	D1	+5.98 (22.00%)	+8.36 (33.77%)	+8.27 (33.26%)	+2.48 (8.45%)	+4.62 (16.20%)
	D2	+6.44 (19.10%)	+9.24 (29.83%)	+9.64 (31.60%)	+3.01 (8.43%)	+4.80 (13.58%)
Group 2	D1	+9.75 (39.08%)	+11.49 (52.33%)	+6.49 (25.29%)	+8.13 (32.85%)	+6.23 (22.91%)
	D2	+10.74 (34.17%)	+11.73 (40.11%)	+7.55 (23.51%)	+7.22 (21.95%)	+6.19 (17.87%)
Group 3	D1	+1.51 (5.42%)	+2.35 (8.88%)	+0.06 (0.20%)	-0.02 (0.06%)	+1.18 (4.27%)
	D2	+1.84 (5.17%)	+2.52 (7.34%)	+1.00 (2.77%)	-0.72 (1.94%)	+1.58 (4.50%)
Group 4	D1	+5.83 (23.34%)	+9.34 (40.80%)	+14.91 (91.47%)	+4.53 (17.90%)	+4.61 (16.79%)
	D2	+5.88 (18.78%)	+8.75 (28.78%)	+12.90 (50.99%)	+3.43 (10.28%)	+4.63 (13.51%)
mean	D1	+5.57 (21.21%)	+7.90 (32.88%)	+3.97 (14.62%)	+3.39 (12.44%)	+4.03 (14.55%)
	D2	+6.08 (18.41%)	+8.16 (26.11%)	+4.81 (14.25%)	+2.76 (7.86%)	+4.31 (12.35%)

Table 4. Ablation study for the fix and completion modules. Fix and Com denote the fix and completion modules, respectively, while w/o and w/ mean to remove or preserve modules.

Metrics	BD-PSNR(D1)	BD-PSNR(D2)
w/o Com	+1.16 (8.18%)	+0.46 (2.39%)
w/o Fix&Com	+0.33 (2.32%)	-0.47 (-2.45%)
Ours	+1.63 (11.55%)	+1.96 (10.15%)

Table 5. Ablation study for the decompression module.

Metrics	BD-PSNR(D1)	BD-PSNR(D2)
FC	-0.52 (-3.7%)	-1.31 (6.79%)
Folding	-3.05 (-21.53%)	-5.61 (29.04%)
Ours	+1.63 (11.55%)	+1.96 (10.15%)

Our loss outperforms all basic constraints like CD or EMD, while removing any of its components will reduce the final performance. It means each component of the loss contributes to the results.

Table 6. Ablation study for the loss function. CD and EMD denotes using CD or EMD constraints to train the networks. Aug means the data augmentation operation during training. L_{EC} and L_{QC} are expansion constraint and quantization constraint, respectively.

Metrics	BD-PSNR(D1)	BD-PSNR(D2)
CD	+0.33 (2.30%)	+1.24 (6.44%)
EMD	-0.38 (2.70%)	-0.56 (2.90%)
w/o Aug	+0.76 (5.40%)	+1.12 (10.98%)
w/o Aug, L_{EC}	+0.61 (4.34%)	+1.92 (9.96%)
w/o Aug, L_{EC}, L_{QC}	+0.59 (4.16%)	+1.72 (8.90%)
Ours	+1.63 (11.55%)	+1.96 (10.15%)

4.6.4 Ablation Study for Model Breaking Strategy. The Model Breaking Strategy (MBS) is adopted to break dense models into small patches. As MBS divides original models by their distances toward clustering centers, we conduct a comparison of MBS under different clustering strategies in this section. The results are presented in Table 7. Though Random and Cmeans have lower time costs than Kmeans adopted, they also perform worse. In contrast, Gaussian clustering has a little better performance than Kmeans, while it costs hundreds times higher consumption. Considering both performances and costs, we use MBS based on Kmeans.

Table 7. Ablation Study for Model Breaking Strategy. Random means random sampling, while Cmeans, Gaussian and Kmeans denote Cmeans fuzzy clustering, Gaussian clustering and Kmeans clustering, respectively.

Metrics	BD-PSNR(D1)	BD-PSNR(D2)	Time cost(s)
Random	-5.38 (38.05%)	-5.17 (26.74%)	0.84
Cmeans	+1.54 (10.89%)	+1.61 (8.34%)	0.86
Gaussian	+1.73 (12.20%)	+2.09 (10.83%)	115.22
Kmeans	+1.63 (11.55%)	+1.96 (10.15%)	0.93

4.7 Comparison of Algorithm Cost

In this section, we compare the cost of our method with other learning-based compression algorithms under the lowest compression levels based on a 160k points duck model from Visionair [Kopecki et al. 2011]. The results are illustrated in Table 8. Though GeoCNN has a relatively lower memory cost by recurrently compressing points in each block, it works much slower than other methods due to the circulation to compress multiple blocks. Our method achieves lowest costs during decoding, which may benefit from our decompression module to generate coordinates directly. Though MBS causes a higher time cost than PCGCv2 during the pre-processing, our method takes relatively low cost during the network-based encoding. Besides, we also believe our method can be further accelerated by migrating it to C-based programs.

Table 8. Algorithm cost comparison. All time and memory costs are measured on a GTX 1080ti with a i7-6700 CPU. The "Pre/Time" denotes time costs of pre-processing and encoding processes, respectively.

	Encode		Decode	
Metrics	Pre/Time(s)	Memory(MB)	Time(s)	Memory(MB)
PCGCv2	**0.52**/0.42	821	0.43	1963
GeoCNN	- / 196	**430**	127.91	428
Ours	0.93/**0.41**	647	**0.21**	**423**

5 CONCLUSION

In this work, we propose a novel point-based point cloud compression framework named 3QNet. Point Clouds are broken into multiple patches for parallel processing, while each patch is separately compressed and encoded with a learned codebook in Hierarchical Compression. The encoded features are then used to reconstruct point clouds with Progress Decompression. 3QNet avoids the precision loss in voxel-based compression networks and can be used

on dense points. With the quantization operation in the feature space and the direct generation of coordinates in the decompression period, 3QNet produces smoother decompressed results with higher decompression efficiency. Besides, 3QNet can flexibly change the compression level by operating the network level and the number of partitioned patches in MBS after training instead of optimizing a separate network for each compression level. According to the experiments on objects, indoor and outdoor scenes, 3QNet can achieve good performances for point clouds with different spatial distributions after training on sparse models without further fine-tuning.

ACKNOWLEDGMENTS

Thanks for the excellent works of the chair, editors, reviewers, and authors. This work is supported by the Key Research and Development Project of Zhejiang Province under Grant 2021C01035.

REFERENCES

Panos Achlioptas, Olga Diamanti, Ioannis Mitliagkas, and Leonidas Guibas. 2018. Learning representations and generative models for 3d point clouds. In *International conference on machine learning*. PMLR, 40–49.

Jens Behley, Martin Garbade, Andres Milioto, Jan Quenzel, Sven Behnke, Cyrill Stachniss, and Jurgen Gall. 2019. Semantickitti: A dataset for semantic scene understanding of lidar sequences. In *Proceedings of the IEEE/CVF International Conference on Computer Vision*. 9297–9307.

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

Gisle Bjontegaard. 2001. Calculation of average PSNR differences between RD-curves. *VCEG-M33* (2001).

Cesar Cadena, Luca Carlone, Henry Carrillo, Yasir Latif, Davide Scaramuzza, José Neira, Ian Reid, and John J Leonard. 2016. Past, present, and future of simultaneous localization and mapping: Toward the robust-perception age. *IEEE Transactions on robotics* 32, 6 (2016), 1309–1332.

Chao Cao, Marius Preda, and Titus Zaharia. 2019. 3D point cloud compression: A survey. In *The 24th International Conference on 3D Web Technology*. 1–9.

Angela Dai, Angel X Chang, Manolis Savva, Maciej Halber, Thomas Funkhouser, and Matthias Nießner. 2017a. Scannet: Richly-annotated 3d reconstructions of indoor scenes. In *Proceedings of the IEEE conference on computer vision and pattern recognition*. 5828–5839.

Angela Dai, Charles Ruizhongtai Qi, and Matthias Nießner. 2017b. Shape completion using 3d-encoder-predictor cnns and shape synthesis. In *Proceedings of the IEEE conference on computer vision and pattern recognition*. 5868–5877.

Konstantinos G Derpanis. 2010. Overview of the RANSAC Algorithm. *Image Rochester NY* 4, 1 (2010), 2–3.

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

Martin A Fischler and Robert C Bolles. 1981. Random sample consensus: a paradigm for model fitting with applications to image analysis and automated cartography. *Commun. ACM* 24, 6 (1981), 381–395.

Frank Galligan, Michael Hemmer, Ondrej Stava, Fan Zhang, and Jamieson Brettle. 2018. Google/Draco: a library for compressing and decompressing 3D geometric meshes and point clouds.

D Graziosi, O Nakagami, S Kuma, A Zaghetto, T Suzuki, and A Tabatabai. 2020. An overview of ongoing point cloud compression standardization activities: Video-based (V-PCC) and geometry-based (G-PCC). *APSIPA Transactions on Signal and Information Processing* 9 (2020).

Amos Gropp, Lior Yariv, Niv Haim, Matan Atzmon, and Yaron Lipman. 2020. Implicit geometric regularization for learning shapes. *arXiv preprint arXiv:2002.10099* (2020).

Yun He, Xinlin Ren, Danhang Tang, Yinda Zhang, Xiangyang Xue, and Yanwei Fu. 2022. Density-preserving Deep Point Cloud Compression. In *Proceedings of the IEEE/CVF Conference on Computer Vision and Pattern Recognition (CVPR)*.

Lila Huang, Shenlong Wang, Kelvin Wong, Jerry Liu, and Raquel Urtasun. 2020. Octsqueeze: Octree-structured entropy model for lidar compression. In *Proceedings of the IEEE/CVF conference on computer vision and pattern recognition*. 1313–1323.

Tianxin Huang and Yong Liu. 2019. 3d point cloud geometry compression on deep learning. In *Proceedings of the 27th ACM international conference on multimedia*. 890–898.

Xin Kong, Guangyao Zhai, Baoquan Zhong, and Yong Liu. 2019. Pass3d: Precise and accelerated semantic segmentation for 3d point cloud. In *2019 IEEE/RSJ International Conference on Intelligent Robots and Systems (IROS)*. IEEE, 3467–3473.

Andreas Kopecki, Uwe WOSSNER, Dimitris Mavrikios, Loukas Rentzos, Christian Weidig, Lionel Roucoules, Okung-Dike Ntofon, Martin Reed, Georges Dumont, Daniel BUNDGENS, et al. 2011. Visionair vision advanced infrastructure for research. (2011).

Shitong Luo and Wei Hu. 2021. Diffusion probabilistic models for 3d point cloud generation. In *Proceedings of the IEEE/CVF Conference on Computer Vision and Pattern Recognition*. 2837–2845.

Dat Thanh Nguyen, Maurice Quach, Giuseppe Valenzise, and Pierre Duhamel. 2021. Learning-based lossless compression of 3d point cloud geometry. In *ICASSP 2021-2021 IEEE International Conference on Acoustics, Speech and Signal Processing (ICASSP)*. IEEE, 4220–4224.

En Yen Puang, Hao Zhang, Hongyuan Zhu, and Wei Jing. 2022. Hierarchical Point Cloud Encoding and Decoding With Lightweight Self-Attention Based Model. *IEEE Robotics and Automation Letters* 7, 2 (2022), 4542–4549.

Charles R Qi, Wei Liu, Chenxia Wu, Hao Su, and Leonidas J Guibas. 2018. Frustum pointnets for 3d object detection from rgb-d data. In *Proceedings of the IEEE conference on computer vision and pattern recognition*. 918–927.

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

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

Maurice Quach, Giuseppe Valenzise, and Frederic Dufaux. 2020. Improved deep point cloud geometry compression. In *2020 IEEE 22nd International Workshop on Multimedia Signal Processing (MMSP)*. IEEE, 1–6.

Zizheng Que, Guo Lu, and Dong Xu. 2021. Voxelcontext-net: An octree based framework for point cloud compression. In *Proceedings of the IEEE/CVF Conference on Computer Vision and Pattern Recognition*. 6042–6051.

Jarek Rossignac. 1999. Edgebreaker: Connectivity compression for triangle meshes. *IEEE transactions on visualization and computer graphics* 5, 1 (1999), 47–61.

Radu Bogdan Rusu and Steve Cousins. 2011. 3d is here: Point cloud library (pcl). In *2011 IEEE international conference on robotics and automation*. IEEE, 1–4.

Shaoshuai Shi, Xiaogang Wang, and Hongsheng Li. 2019. Pointrcnn: 3d object proposal generation and detection from point cloud. In *Proceedings of the IEEE/CVF conference on computer vision and pattern recognition*. 770–779.

Maxim Tatarchenko, Alexey Dosovitskiy, and Thomas Brox. 2017. Octree generating networks: Efficient convolutional architectures for high-resolution 3d outputs. In *Proceedings of the IEEE international conference on computer vision*. 2088–2096.

Gabriel Taubin and Jarek Rossignac. 1998. Geometric compression through topological surgery. *ACM Transactions on Graphics (TOG)* 17, 2 (1998), 84–115.

Dorina Thanou, Philip A Chou, and Pascal Frossard. 2016. Graph-based compression of dynamic 3D point cloud sequences. *IEEE Transactions on Image Processing* 25, 4 (2016), 1765–1778.

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

Costa Touma and Craig Gotsman. 1998. Triangle mesh compression. In *Proceedings-Graphics Interface*. Canadian Information Processing Society, 26–34.

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

Jianqiang Wang, Hao Zhu, Haojie Liu, and Zhan Ma. 2021b. Lossy point cloud geometry compression via end-to-end learning. *IEEE Transactions on Circuits and Systems for Video Technology* 31, 12 (2021), 4909–4923.

Xuanzheng Wen, Xu Wang, Junhui Hou, Lin Ma, Yu Zhou, and Jianmin Jiang. 2020. Lossy geometry compression of 3d point cloud data via an adaptive octree-guided network. In *2020 IEEE International Conference on Multimedia and Expo (ICME)*. IEEE, 1–6.

Louis Wiesmann, Andres Milioto, Xieyuanli Chen, Cyrill Stachniss, and Jens Behley. 2021. Deep compression for dense point cloud maps. *IEEE Robotics and Automation Letters* 6, 2 (2021), 2060–2067.

Ian H Witten, Radford M Neal, and John G Cleary. 1987. Arithmetic coding for data compression. *Commun. ACM* 30, 6 (1987), 520–540.

Zhirong Wu, Shuran Song, Aditya Khosla, Fisher Yu, Linguang Zhang, Xiaoou Tang, and Jianxiong Xiao. 2015. 3d shapenets: A deep representation for volumetric shapes. In *Proceedings of the IEEE conference on computer vision and pattern recognition*. 1912–1920.

Yaoqing Yang, Chen Feng, Yiru Shen, and Dong Tian. 2018. Foldingnet: Point cloud auto-encoder via deep grid deformation. In *Proceedings of the IEEE Conference on Computer Vision and Pattern Recognition*. 206–215.

Lequan Yu, Xianzhi Li, Chi-Wing Fu, Daniel Cohen-Or, and Pheng-Ann Heng. 2018. Pu-net: Point cloud upsampling network. In *Proceedings of the IEEE Conference on Computer Vision and Pattern Recognition*. 2790–2799.