

AdaPoinTr: Diverse Point Cloud Completion with Adaptive Geometry-Aware Transformers

Xumin Yu, *Student Member, IEEE*, Yongming Rao, *Student Member, IEEE*, Ziyi Wang, *Student Member, IEEE*, Jiwen Lu, *Senior Member, IEEE*, Jie Zhou, *Senior Member, IEEE*

Abstract—Point clouds captured in real-world scenarios are often incomplete due to the limited sensor resolution, single viewpoint, and occlusion. Therefore, recovering the complete point clouds from partial ones becomes an indispensable task in many practical applications. In this paper, we present a new method that reformulates point cloud completion as a set-to-set translation problem and design a new model, called **PoinTr**, which adopts a Transformer encoder-decoder architecture for point cloud completion. By representing the point cloud as a set of unordered groups of points with position embeddings, we convert the input data to a sequence of point proxies and employ the Transformers for generation. To facilitate Transformers to better leverage the inductive bias about 3D geometric structures of point clouds, we further devise a geometry-aware block that models the local geometric relationships explicitly. The migration of Transformers enables our model to better learn structural knowledge and preserve detailed information for point cloud completion. Taking a step towards more complicated and diverse situations, we further propose **AdaPoinTr** by developing an adaptive query generation mechanism and designing a novel denoising task during completing a point cloud. Coupling these two techniques enables us to train the model efficiently and effectively: we reduce training time (by 15x or more) and improve completion performance (over 20%). Additionally, we propose two more challenging benchmarks with more diverse incomplete point clouds that can better reflect real-world scenarios to promote future research. We also show our method can be extended to the scene-level point cloud completion scenario by designing a new geometry-enhanced semantic scene completion framework. Extensive experiments on the existing and newly-proposed datasets demonstrate the effectiveness of our method, which attains 6.53 CD on PCN, 0.81 CD on ShapeNet-55 and 0.392 MMD on real-world KITTI, surpassing other work by a large margin and establishing new state-of-the-arts on various benchmarks. Most notably, AdaPoinTr can achieve such promising performance with higher throughputs and fewer FLOPs compared with the previous best methods in practice. The code and datasets are available at <https://github.com/yuxumin/PoinTr>.

Index Terms—Point Cloud, Transformers, Point Cloud Completion.

1 INTRODUCTION

RECENT developments in 3D sensors largely boost researches in 3D computer vision. One of the most commonly used 3D data format is the point cloud, which requires less memory to store but convey detailed 3D shape information. However, point cloud data from existing 3D sensors are not always complete and satisfactory because of inevitable self-occlusion, light reflection, limited sensor resolution, *etc.* Therefore, recovering complete point clouds from partial and sparse raw data becomes an indispensable task with ever-growing significance.

Over the years, researchers have tried many approaches to tackle this problem in the realm of deep learning. Early attempts on point cloud completion [9], [23], [30], [31], [35], [46], [49], [56], [60], [68] try to migrate mature methods from 2D completion tasks to 3D point clouds by voxelization and 3D convolutions. However, these methods suffer from a heavy computational cost that grows cubically as the spatial resolution increases. With the success of PointNet [41] and PointNet++ [42], directly processing 3D coordinates becomes the mainstream of point cloud based 3D analysis. The technique is further applied to many pioneer work [1],

[20], [24], [34], [45], [54], [72] in point cloud completion task, in which an encoder-decoder based architecture is designed to generate complete point clouds. However, the bottleneck of such methods lies in the single feature vector produced in the encoding phase, where fine-grained information is lost and can hardly be recovered in the decoding phase.

Reconstructing complete point clouds is a challenging problem since the structural information required in the completion task runs counter to the unordered and unstructured nature of point cloud data. Therefore, learning structural features and long-range correlations among local parts of the point cloud becomes the key ingredient towards better point cloud completion. In this paper, we propose to adopt Transformers [57], one of the most successful architectures in Natural Language Processing (NLP), to learn the structural information of pairwise interactions and global correlations for point cloud completion. Our model, named *PoinTr*, is characterized by five key components: 1) *Encoder-Decoder Architecture*: We adopt the encoder-decoder architecture to convert point cloud completion as a set-to-set translation problem. The self-attention mechanism of Transformers models all pairwise interactions between elements in the encoder, while the decoder reasons about the missing elements based on the learnable pairwise interactions among features of the input point cloud and queries; 2) *Point Proxy*: We represent the set of point clouds in a local region as a feature vector called *Point Proxy*. The input point cloud is converted to a sequence of Point Proxies, which are

- The authors are with Beijing National Research Center for Information Science and Technology (BNRist), the State Key Lab of Intelligent Technologies and Systems, and the Department of Automation, Tsinghua University, Beijing, 100084, China. Email: yuxm20@mails.tsinghua.edu.cn; raoyongming95@gmail.com; wziyi22@mails.tsinghua.edu.cn; lujiwen@tsinghua.edu.cn; jzhou@tsinghua.edu.cn.

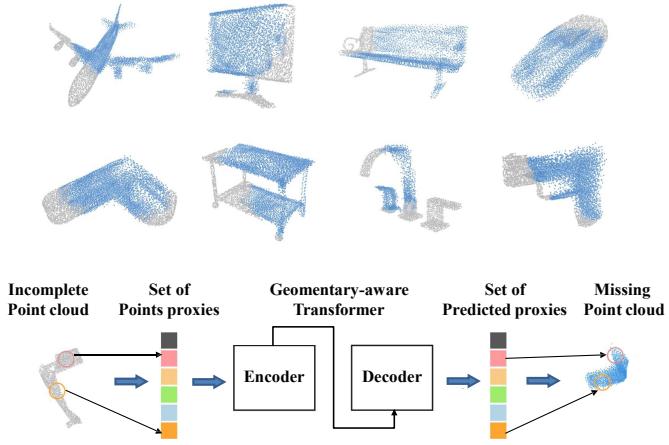


Fig. 1. *PoinTr* is designed for point cloud completion task. It takes the downsampled partial point clouds as inputs (gray points), predicts the missing parts (blue points) and upsamples the known parts simultaneously. We propose to formulate the point cloud completion task as a set-to-set translation task and use a Transformer encoder-decoder architecture to learn the complex dependencies among the point groups.

used as the inputs of our Transformer model; 3) *Geometry-aware Transformer Block*: To facilitate Transformers to better leverage the inductive bias about 3D geometric structures of point clouds, we design a geometry-aware block that models the geometric relations explicitly; 4) *Query Generator*: Queries serve as the initial state of predicted proxies and we use dynamic queries instead of fixed queries in the decoder, which are generated by a query generation module that summarizes the features produced by the encoder and represents the initial sketch of the missing points; 5) *Multi-Scale Point Cloud Generation*: We devise a multi-scale point generation module to recover the missing point cloud in a coarse-to-fine manner.

Based on the proposed architecture, point cloud completion is reformulated as a set-to-set translation problem, as shown in Fig. 1. Given the incomplete point cloud (gray points) as the known part, we translate it into the unknown part (blue points) with our *PoinTr*, which enables us to fully exploit the interactions between known and unknown sets. Then a simple concatenation operation is performed to combine these two partial point clouds into a complete one. However, since these two parts are considered separately during the completion process, the concatenation operation in the \mathcal{R}^3 space brings the issue that the final prediction may be discontinuous in appearance. Therefore, we further propose *AdaPoinTr* based on *PoinTr*, which combines the known and unknown parts by concatenating corresponding queries in the embedding space \mathcal{R}^Q rather than in the \mathcal{R}^3 space (See § 3.6 for detailed explanation). In this way, the known and unknown parts can be considered jointly during the completion process in a unified manner. Technically, we develop an adaptive query generation mechanism to deal with diverse situations. For example, a car from the LiDAR-scan may miss over 90% points while another one may miss only 50% points. Moreover, an auxiliary denoising task is designed to effectively make the optimization more stable and efficient, as it alleviates the problem caused by low-

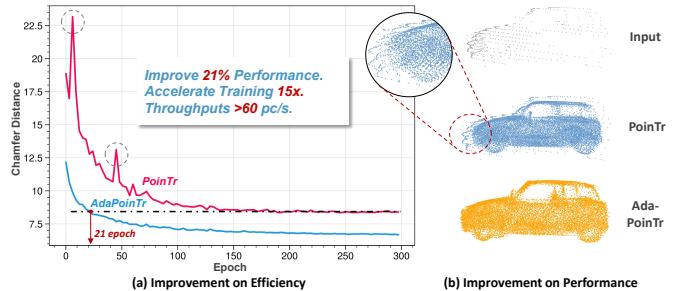


Fig. 2. We further propose *AdaPoinTr* by developing an adaptive query generation mechanism and designing a novel denoising task. We achieve improvement on both efficiency and performance with less training epochs, excellent throughputs and more stable training curve on wide-used PCN dataset.

quality queries at the very beginning of the training. As shown in Fig. 2, *AdaPoinTr* can achieve better performance with only 8% training epochs compared with *PoinTr*. Meanwhile, *AdaPoinTr* can produce more reasonable completion results without appearance issues even in challenging real-world situations.

As another contribution, we argue that existing benchmarks are not representative enough to cover real-world scenarios of incomplete point clouds. Therefore, we introduce two more challenging benchmarks, ShapeNet-55 and ShapeNet-34, in short *SN-55/34*, which contain more diverse tasks (*i.e.*, joint upsampling and completion of point cloud), more object categories (*i.e.*, from 8 categories to 55 categories), more diverse views points (*i.e.*, from 8 viewpoints to all possible viewpoints) and more diverse level of incompleteness (*i.e.*, missing 25% to 75% points of the ground-truth point clouds). In *SN-55/34*, we adopt an online-cropping method to generate diverse incomplete point clouds, which is more flexible and efficient than other offline back-projecting benchmarks like PCN [72], Completion3D [54] and MVP [37]. Even though their samples are more realistic than those in our efficient *SN-55/34*, they still did not take noises into consideration. At this point, we propose noised back-projecting by adding a scaled random noise to the depth map before generating incomplete point clouds. We change the online cropping method with this noised back-projecting method and obtain two new variants: Projected-ShapeNet-55 and Projected-ShapeNet-34. We show some samples of incomplete point cloud generation with different methods (blue columns) in Fig. 3. The noised back-projecting makes the completion task more difficult and avoids the trivial solution of simply combining the incomplete point clouds from different views, which are shown in the last two columns in the figure.

Besides, we also explore the application potential of our proposed architecture in scene-level completion task. Due to the large scale of a point cloud scene, scene-level completion is always formulated as a voxel-based completion task, predicting the occupancy for the given voxel, which always lacks of considering detailed geometric information. Luckily, our model demonstrates its outstanding performance on learning point-wise structural features and building long-range correlations among local regions. Therefore, we introduce the proposed geometry-aware Transformer

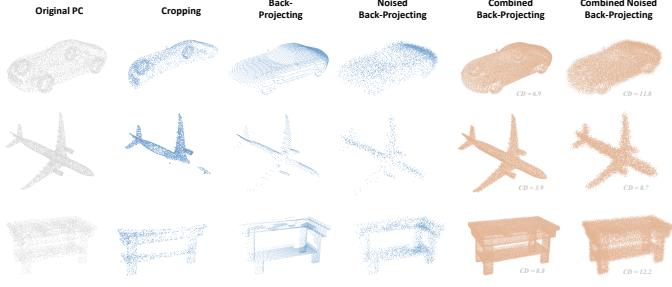


Fig. 3. There are three main-stream methods to generate incomplete point clouds from 3D objects, cropping, back-projecting, and noised back-projecting. We propose ShapeNet-55/34 and Projected-ShapeNet-55/34, which generate incomplete point clouds by cropping and noised back-projecting, respectively. Online cropping is more flexible and efficient, while noised back-projecting is a better approximation to real scans.

architecture to the scene-level completion task. Specifically, we propose a geometry-enhanced semantic scene completion framework, which can supplement voxel-based models with fine-grained geometric information, thus improving the completion performance on scenes.

We conduct experiments on both object point cloud completion and semantic scene completion. For object point cloud completion, we evaluate our method on the newly proposed benchmarks, the widely used dataset PCN [72], and the LiDAR-based dataset KITTI [16]. Our AdaPoinTr establishes new state-of-the-art on all the mentioned benchmarks. Additionally, we evaluated our model on the MVP competition benchmark [38] and won the first place of the MVP Completion challenges in the ICCV 2021 Workshop [39]. For scene-level point cloud completion, we follow the standard protocol of the semantic scene completion task [43] and evaluate our method on NYU Depth V2 [47] and NYUCAD [14]. Extensive experiments demonstrate the effectiveness of our method in various application scenarios.

This paper is an extended version of our conference paper [70]. We make several new contributions: 1) We propose an improved version of original PoinTr, AdaPoinTr, by adopting an adaptive query generation mechanism and designing a novel auxiliary denoising task during completion, which solves the appearance issue and enables us to establish new state-of-the-arts with less training times and satisfactory throughputs in practice. 2) we extend our method to the scene-level point cloud completion by designing a geometry-enhanced semantic scene completion framework and performing point-to-voxel translation with PoinTr; 3) we present Projected-ShapeNet-55/34 dataset by generating incomplete samples with proposed noised back-projecting to better approximate the real scans. We also provide more in-depth analysis and visualization of our method.

2 RELATED WORK

3D Shape Completion. Traditional methods for 3D shape completion tasks often adopt voxel grids or distance fields to describe 3D objects [9], [23], [49]. Based on such structured 3D representations, the powerful 3D convolutions are used and achieve a great success in the tasks of 3D reconstruction [7], [17] and shape completion [9], [23], [65]. However,

this group of methods [18], [50], [59] suffers from heavy memory consumption and computational burden. Different from the above methods, researchers gradually start to use unstructured point clouds as the representation of 3D objects, given the small memory consumption and strong ability to represent fine-grained details. Nevertheless, the migration from structured 3D data understanding to point cloud analysis is non-trivial, since the commonly used convolution operator is no longer suitable for unordered points clouds. PointNet and its variants [41], [42] are the pioneering work to directly process 3D coordinates and inspire the researches in many downstream tasks. In the realm of point cloud completion, PCN [72] is the first learning-based architecture, which proposes an Encoder-Decoder framework and adopts a FoldingNet to map the 2D points onto a 3D surface by mimicking the deformation of a 2D plane. After PCN, many other methods [24], [28], [54], [67], [70] spring up, pursuing point clouds completion in higher resolution with better robustness. VRCNet [37] proposes a variational framework with probabilistic modeling and relational enhancement, which effectively exploits 3D structural relations to predict complete shapes. SnowflakeNet [66] proposes to model the generation of point clouds as a snowflake-like growth based on certain points in 3D space, where the point clouds are generated progressively from some parent points. And recently, LAKeNet [52] proposes to consider predicting structured and topological information of 3D shape, which follow a keypoints-skeleton-shape prediction manner. SeedFormer [78] introduces a new shape representation, Patch Seeds, for point clouds and devises a novel Upsample Transformer during the generation.

Semantic Scene Completion. Semantic Scene Completion is an important task in 3D scene understanding, which aims to predict volumetric occupancy and semantic labels simultaneously from a single-view depth map or RGB-D image. For a long time, many methods consider these two tasks separately. SSCNet [48] is the first to introduce Semantic Scene Completion by combining scene completion and semantic segmentation in an end-to-end way, proving that these two tasks can promote each other in the meantime. ESSCNet [74] proposes Spatial Group Convolution (SGC) to group the input volume and then utilizes 3D sparse convolution for feature extraction. VVNet [22] further proposes to bridge 2D domain and 3D domain through a differentiable projection layer, which efficiently reduces the computational cost and makes it possible to extract the feature from multi-channel inputs. ForkNet [62] proposes to build a multi-branch architecture and alleviates problems caused by the limited training samples of real scenes by adopting generative models. CCPNet [75] argues to progressively restore the details of objects by adopting a cascaded context pyramid model, which improves the labeling coherence. Then some methods [15], [29] try to seek a way to leverage the complementary information of depth map, like RGB images. 3D CNN is employed to fuse two-stream inputs. DDRNet [26] proposes a light-weight dimensional decomposition residual block to fuse multi-scale RGB-D features. AICNet [25] modifies the standard 3D convolution so that the kernels can be of various sizes. Sketch [6] proposes to guide the semantic prediction with

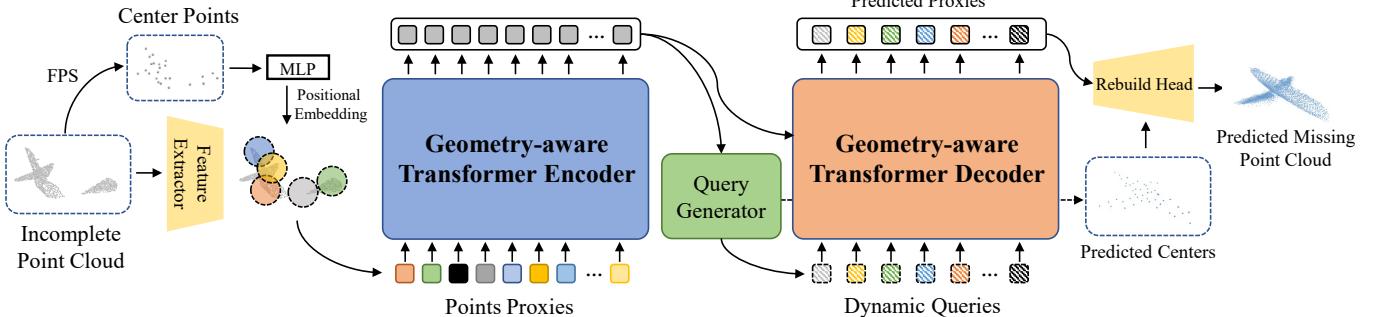


Fig. 4. The Pipeline of *PoinTr*. We first downsample the input partial point cloud to obtain the center points. Then, we use a lightweight DGCNN [63] to extract the local features around the center points. After adding the position embedding to the local feature, we use a Transformer encoder-decoder architecture to predict the point proxies for the missing parts. A simple MLP and a Rebuild Head are used to complete the point cloud based on the predicted point proxies in a coarse-to-fine manner.

an explicitly encoded 3D sketch-aware feature embedding, which contains rich geometric information. Recently, SISNet [4] exploits the relations between instance completion and scene completion in an interactive manner. They iteratively reconstruct instances within the scene and complete the entire scene, which effectively recovers the geometric patterns and semantic information compared with previous end-to-end methods.

Transformers. Transformers [57] are first introduced as an attention-based framework in Natural Language Processing (NLP). Transformer models often utilize the encoder-decoder architecture and are characterized by both self-attention and cross-attention mechanisms. Transformer models have proven to be very helpful to tasks that involve long sequences thanks to the self-attention mechanism. The cross-attention mechanism in the decoder exploits the encoder information to learn the attention map of query features, which makes Transformers powerful in generation tasks. By taking advantage of both self-attention and cross-attention mechanisms, Transformers have a strong capability to handle long sequence input and enhance information communications between the encoder and the decoder. In the past few years, Transformers have dominated the tasks that take long sequences as input and gradually replaced RNNs [58] in many domains. Now they begin their journey in computer vision [11], [33], [40], [51], [71]. ViT [12] introduces Transformers into 2D domains, and DeiT [55] explores a data-efficient training strategy for Vision Transformers. While most efforts focus on learning vision Transformers on 2D data, the applications on point clouds remain limited. Some preliminary explorations on recognition task have been implemented [21], [77], while we introduce this architecture into 3d generation tasks like point cloud completion.

3 POINT CLOUD COMPLETION

In this section, we will introduce the details of *PoinTr*. We first elaborate the five key components of *PoinTr* in Section 3.1-3.5. Then, we present the new adaptive query generation mechanism and auxiliary denoise task in Section 3.6. Lastly, we show the learning objectives in Section 3.7. The overall framework of our method is illustrated in Fig. 4.

3.1 Set-to-Set Translation with Transformers

The primary goal of our method is to leverage the impressive sequence-to-sequence generation ability of Transformer architecture for point cloud completion tasks. We propose to first convert the point cloud to a set of feature vectors, *point proxies*, that represent the local regions in the point clouds (we will describe in Section 3.2). By analogy to the language translation pipeline, we model point cloud completion as a set-to-set translation task, where the Transformers take the point proxies of the partial point clouds as the inputs and produce the point proxies of the missing parts. Specifically, given the set of point proxies $\mathcal{F} = \{F_1, F_2, \dots, F_N\}$ that represents the partial point cloud, we model the process of point cloud completion as a set-to-set translation problem:

$$\mathcal{V} = \mathcal{M}_E(\mathcal{F}), \quad \mathcal{H} = \mathcal{M}_D(\mathcal{Q}, \mathcal{V}), \quad (1)$$

where \mathcal{M}_E and \mathcal{M}_D are the encoder and decoder models, $\mathcal{V} = \{V_1, V_2, \dots, V_N\}$ are the output features of the encoder, $\mathcal{Q} = \{Q_1, Q_2, \dots, Q_M\}$ are the dynamic queries for the decoder, $\mathcal{H} = \{H_1, H_2, \dots, H_M\}$ are the predicted point proxies of the missing point cloud, and M is the number of the predicted point proxies. The recent success in NLP tasks like text translation and question answering [10] have clearly demonstrated the effectiveness of Transformers to solve this kind of problem. Therefore, we propose to adopt a Transformer-based encoder-decoder architecture to solve the point cloud completion problem.

The encoder-decoder architecture consists of L_E and L_D multi-head self-attention layers [57] in the encoder and decoder, respectively. The self-attention layer in the encoder first updates proxy features with both long-range and short-range information. Then a feed-forward network (FFN) further updates the proxy features with an MLP architecture. The decoder utilizes self-attention and cross-attention mechanisms to learn structural knowledge. The self-attention layer enhances the local features with global information, while the cross-attention layer explores the relationship between queries and outputs of the encoder. To predict the point proxies of the missing parts, we propose to use dynamic query embeddings, which is different from the learnable static query embedding in [5], [73] (as shown in Fig. 6 (a, b)). This dynamic query scheme makes our decoder more flexible and adjustable for different types of

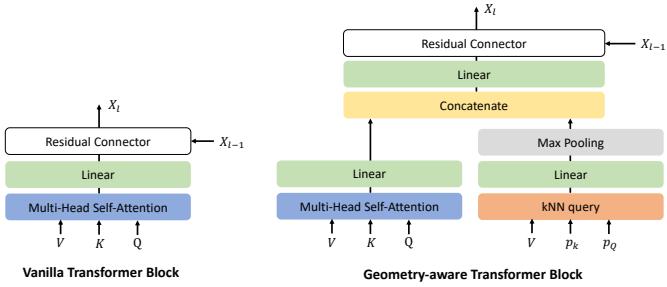


Fig. 5. Comparisons of the vanilla Transformer block and the proposed geometry-aware Transformer block.

objects and their missing information. More details about the Transformer architecture can be found in the supplementary material and [10], [57].

Note that benefiting from the self-attention mechanism in Transformers, the features learned by the Transformer network are invariant to the order of point proxies, which is also the basis of using Transformers to process point clouds. Considering the strong ability to capture data relationships, we expect the Transformer architecture to be a promising alternative for deep learning on point clouds.

3.2 Point Proxy

The Transformers in NLP take as input a 1D sequence of word embeddings [57]. To make 3D point clouds suitable for Transformers, the first step is to convert the point cloud to a sequence of vectors. A trivial solution is directly feeding the sequence of xyz coordinates to the Transformers. However, since the computational complexity of the Transformers is quadratic to the sequence length, this solution will lead to an unacceptable cost. Therefore, we propose to represent the original point cloud as a set of *point proxies*. A point proxy represents a local region of the point clouds. Inspired by the set abstraction operation in [42], we first conduct *furthest point sample (FPS)* to locate a fixed number N of point centers $\{p_1, p_2, \dots, p_N\}$ in the partial point cloud. Then, we use a lightweight DGCNN [63] with hierarchical downsampling to extract the feature of the point centers from the input point cloud. The point proxy F_i is a feature vector that captures the local structure around p_i , which can be computed as:

$$F_i = F'_i + \varphi(p_i), \quad (2)$$

where F'_i is the feature of point p_i that is extracted using the DGCNN model, and φ is another MLP to capture the location information of the point proxy. The first term represents the semantic patterns of the local region, and the second term is inspired by the position embedding [3] operation in Transformers, which explicitly encodes the global location of the point proxy. The detailed architecture of the feature extraction model can be found in Supplementary Material.

3.3 Geometry-aware Transformer Block

One of the key challenges of applying Transformers for vision tasks is that the self-attention mechanism in Transformers lacks some inductive biases inhered in conventional vision models like CNNs, which explicitly model the structures of vision data. To facilitate Transformers to better

leverage the inductive bias about 3D geometric structures of point clouds, we design a geometry-aware block that models the geometric relations, which can be a plug-and-play module to incorporate with the attention blocks in any Transformer architecture. The details of the proposed block are shown in Fig. 5. Different from the self-attention module that uses the feature similarity to capture the semantic relation, we propose to use kNN to capture the geometric relations in the point cloud:

$$\phi(V_i) = \mathcal{M}(\text{Linear}([V_i, V_j - V_i])), \forall j : p_k^j \in \kappa(p_Q^i), \quad (3)$$

where \mathcal{M} is max-pooling operation, V is the input features. Given the query coordinates p_Q , we gather the features whose coordinates p_k locates within the k -neighborhood of p_Q , represented as $\kappa(p_Q)$. Then we follow the practice of DGCNN [63] to learn the local geometric structures via feature aggregation with a linear layer followed by the max-pooling operation as shown in Equ. 3. The feature captured by kNN and the feature captured by multi-head self-attention are then concatenated and mapped to the original dimensions to form the output.

3.4 Query Generator

The queries Q serve as the initial state of the predicted proxies. To make sure the queries correctly reflect the sketch of the complete point cloud, we propose a query generator module to generate the query embeddings dynamically conditioned on the encoder outputs V . Specifically, we first summarize V with a linear projection to higher dimensions followed by the max-pooling operation. Similar to [72], we use a linear projection layer to directly generate $M \times 3$ dimension features that can be reshaped as the M coordinates $C = \{c_1, c_2, \dots, c_M\}$. Lastly, we concatenate the global feature of the encoder and the coordinates, and use an MLP to produce the query embeddings $Q = \{Q_1, Q_2, \dots, Q_M\}$, which can be formulated as follows:

$$C = \mathcal{P}(\mathcal{M}(\text{Linear}(V)), \quad Q = \text{MLP}([C, \mathcal{M}(\text{Linear}(V))]), \quad (4)$$

where \mathcal{M} and \mathcal{P} are max-pooling operation and coordinates projection, respectively.

3.5 Multi-Scale Point Cloud Generation

The goal of our encoder-decoder network is to predict the missing parts of incomplete point clouds. However, we can only get predictions for missing proxies from the Transformer decoder. Therefore, we propose a multi-scale point cloud generation framework to recover missing point clouds at full resolution. To reduce redundant computations, we reuse the M coordinates produced by the query generator as the local centers of the missing point cloud. Then, we utilize a reconstruction head like FoldingNet [69] f to recover detailed local shapes centered at the predicted proxies:

$$\mathcal{P}_i = f(\mathcal{H}_i) + c_i, \quad i = 1, 2, \dots, M. \quad (5)$$

where \mathcal{P}_i is the set of neighboring points centered at c_i . Following previous work [24], we only predict the missing parts of the point cloud and *concatenate* them with the input point cloud to obtain the complete objects. Both predicted proxies and recovered point clouds are supervised during the training process, and the detailed loss function will be introduced in the following section.

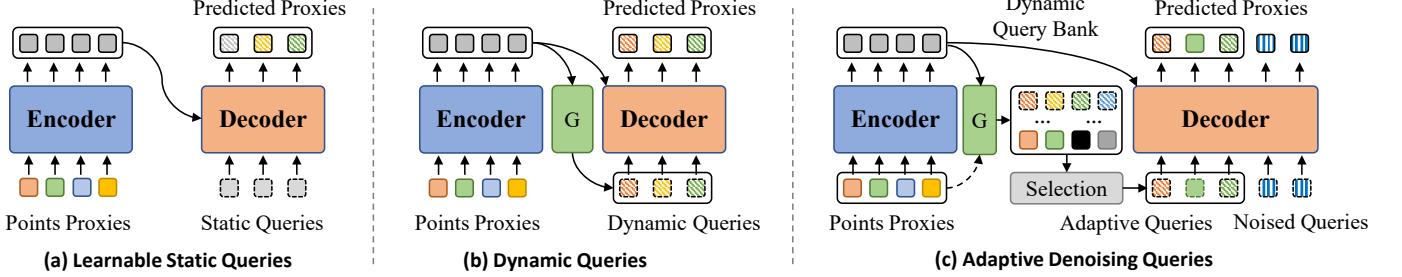


Fig. 6. Three types of queries for transformer decoder. (a) Learnable static queries, which are usually used in DETR-style framework. (b) Dynamic queries proposed in PoinTr, which are generated based on the output features of the encoder. (c) Proposed adaptive denoising queries, which are selected from a dynamic query bank.

3.6 Adaptive Denoising Queries

The concatenation operation in \mathcal{R}^3 space is a simple solution for combining input points and predicted missing points, which treats these two parts of a point cloud as two separate units rather than a unified whole. The missing part is generated by a reconstruction head and the known part is obtained from the input (generated by the back-projecting method or captured by LiDAR sensors). However, this will bring some issues, like discontinuous and uneven appearance for the final predicted point cloud, which impairs the performance. One straightforward approach [61] to address this issue is to add some refinement modules after the concatenation and generate a new complete point cloud, which requires extra parameters and longer latency. Different from theirs, we propose AdaPoinTr (PoinTr+Adaptive Denoising Queries), which takes the advantage of set-to-set translation formulation and point proxy representation, to address the issue by concatenating two sets of point proxies instead of concatenating two partial point clouds. The concatenated proxies are fed into a shared reconstruction model to produce the complete point clouds. In this way, the model reconstructs two parts in a unified manner and introduces no additional refinement parameters. Besides, we devise an advanced denoising task, which significantly improves the efficiency and robustness of our model. Two components of Adaptive Denoising Queries: 1) Adaptive queries generation mechanism and 2) Auxilliary denoising task are introduced in the following paragraphs.

Adaptive Query Generation. We modify the design of the original Query Generator and generate a dynamic query bank \mathcal{B} conditioned on the encoder outputs \mathcal{V} and the input point proxies \mathcal{F} , as shown in the Fig. 6(c). The dynamic query bank contains two types of queries: (a) $\mathcal{Q}_{\mathcal{I}}$, abstracted from the input point proxies \mathcal{F} . (b) $\mathcal{Q}_{\mathcal{O}}$, generated to serve as the initial state of the missing point proxies. Specifically, we summarize \mathcal{V} and \mathcal{F} with a linear projection to higher dimensions followed by the max-pooling, separately. Then the summarized features are projected into $M_I \times 3$ and $M_O \times 3$ dimension space and then reshaped as M_I and M_O coordinates $\mathcal{C}^{\mathcal{I}} = \{c_1^I, c_2^I, \dots, c_{M_I}^I\}$ and $\mathcal{C}^{\mathcal{O}} = \{c_1^O, c_2^O, \dots, c_{M_O}^O\}$. After concatenating coordinates and the corresponding summarized features, we use an MLP to produce queries, which can be written as:

$$\begin{aligned} \mathcal{C}^{\mathcal{I}} &= \mathcal{P}_{\mathcal{I}}(\mathcal{M}(W_I(\mathcal{F}))), & \mathcal{Q}_{\mathcal{I}} &= \text{MLP}([\mathcal{C}^{\mathcal{I}}, \mathcal{M}(W_I(\mathcal{F}))]), \\ \mathcal{C}^{\mathcal{O}} &= \mathcal{P}_{\mathcal{O}}(\mathcal{M}(W_O(\mathcal{V}))), & \mathcal{Q}_{\mathcal{O}} &= \text{MLP}([\mathcal{C}^{\mathcal{O}}, \mathcal{M}(W_O(\mathcal{V}))]), \end{aligned}$$

where \mathcal{M} , $\mathcal{P}_{\mathcal{I}}$, $\mathcal{P}_{\mathcal{O}}$, W_I and W_O are max-pooling operation, coordinates projection for two sets and linear projection for two sets, respectively. We further perform a query selection scheme to adaptively pick a portion of queries out. In this selection, we only constrain the total number of queries after selection without caring about the specific number of queries from $\mathcal{Q}_{\mathcal{I}}$ and $\mathcal{Q}_{\mathcal{O}}$, which makes our method more flexible in diverse situations. Technically, we design a light-weight scoring module \mathcal{S} to rank the queries in \mathcal{B} according to the predicted score and choose M queries with higher scores, represented as $\hat{\mathcal{Q}}$ with their coordinates $\hat{\mathcal{C}}$.

Auxiliary Denoising Task. The Transformer decoder utilizes self-attention and cross-attention mechanisms to enhance the structural knowledge. It builds the relationship between queries and output features from the encoder to predict the missing point proxies. However, we observe an extremely unstable training curve during the early stage of model optimization (Shown in Fig. 2(a)) caused by the low-quality queries initialization. Although we propose a multi-scale point cloud generation scheme to directly supervise the coordinates \mathcal{C} corresponding to queries before feeding them into the decoder, it can only alleviate the problem a little. We further design a denoising task by feeding some newly introduced noised queries along with the adaptive queries to the decoder, as shown in the Fig. 6(c). Specifically, we produce k noised queries by adding a scaled random noise to k groundtruth center points (obtained by FPS operation) in the input, $\hat{\mathcal{C}}^{gt} = \{\hat{c}_1^{gt}, \hat{c}_2^{gt}, \dots, \hat{c}_k^{gt}\}$, where $\hat{c}_i^{gt} = n_i + c_i^{gt}$. The generation can be formulated as follows:

$$\hat{\mathcal{Q}} = \text{MLP}([\hat{\mathcal{C}}, \mathcal{M}(\text{Linear}(\mathcal{V}))]), \quad (6)$$

which can be generated together with $\mathcal{Q}_{\mathcal{I}}$ and $\mathcal{Q}_{\mathcal{O}}$, since the MLP is shared for these sets of queries. This design benefits from two aspects. Firstly, it guarantees that there are always some high-quality queries fed into the decoder. Secondly, the denoising task improves the robustness of the decoder to initial queries. To avoid knowledge leakage from those noised queries $\hat{\mathcal{Q}}$ to \mathcal{Q} , we introduce an attention mask in self-attention layers. Technically, we set the attention mask for noised queries $\hat{\mathcal{Q}}$ and \mathcal{Q} to zero and keep the other attention unchanged, which can be written as follows:

$$m_{ij} = \begin{cases} 0, & q_i \in \hat{\mathcal{Q}}, q_j \in \mathcal{Q} \\ 1, & \text{otherwise.} \end{cases} \quad (7)$$

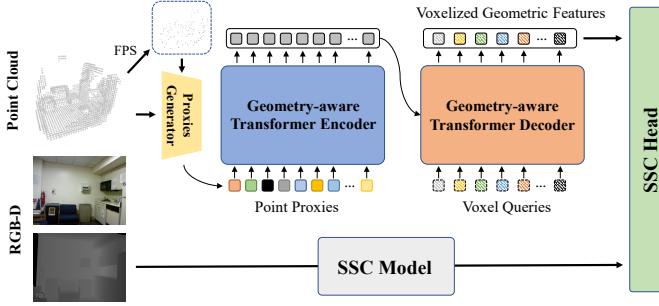


Fig. 7. The proposed geometry-enhanced block (**top branch**) for semantic scene completion. The input point clouds for this block are obtained from the depth map. It acts as a plug-in block for any SSC models, capturing the detailed geometric information and performing global interactions for the whole scene.

\hat{Q} is translated into predicted point proxies $\hat{\mathcal{H}}$ together with other normal queries \mathcal{Q} , which can be expressed as:

$$[\mathcal{H}, \hat{\mathcal{H}}] = \mathcal{M}_D([\mathcal{Q}, \hat{\mathcal{Q}}], \mathcal{V}),$$

where \mathcal{M}_D is Transformer decoder. Then, $\hat{\mathcal{H}}$ will be rebuilt as detailed local shapes centered at the predicted proxies:

$$\hat{\mathcal{P}}_i = f(\hat{\mathcal{H}}_i) + \hat{c}_i^{gt}, \quad i = 1, 2, \dots, k. \quad (8)$$

where $\hat{\mathcal{P}}_i$ is the set of neighboring points centered at \hat{c}_i^{gt} .

3.7 Optimization

The loss function for point cloud completion should provide a quantitative measurement for the quality of output. However, since the point clouds are unordered, many loss functions that directly measure the distance between two points (*i.e.* ℓ_2 distance) are unsuitable. Fan *et al.* [13] introduce two metrics that are invariant to the permutation of points, which are Chamfer Distance (CD) and Earth Mover's Distance (EMD). We adopt Chamfer Distance as our loss function for its $\mathcal{O}(N \log N)$ complexity. We use \mathcal{C} to represent the n_c local centers and \mathcal{P} to represent n_p points of the complete point cloud. Give the ground-truth complete point cloud \mathcal{G} , the loss functions for these two predictions can be written as:

$$\begin{aligned} J_0 &= \frac{1}{n_c} \sum_{c \in \mathcal{C}} \min_{g \in \mathcal{G}} \|c - g\| + \frac{1}{n_g} \sum_{g \in \mathcal{G}} \min_{c \in \mathcal{C}} \|g - c\|, \\ J_1 &= \frac{1}{n_p} \sum_{p \in \mathcal{P}} \min_{g \in \mathcal{G}} \|p - g\| + \frac{1}{n_g} \sum_{g \in \mathcal{G}} \min_{p \in \mathcal{P}} \|g - p\|. \end{aligned}$$

We directly use the high-resolution point cloud \mathcal{G} to supervise the sparse point cloud \mathcal{C} to encourage them to have similar distributions. As for the auxiliary denoising task, giving the noised queries $\hat{\mathcal{Q}}_i$ and the corresponding noised center $\hat{c}_i^{gt} = n_i + c_i^{gt}$, we expect our model can be robust to the random noise n_i and rebuild detailed local shapes centered at c_i^{gt} . If we use $\mathcal{G}_{c_i^{gt}}^{gt}$ to represent the ground-truth local shapes centered at c_i^{gt} and use $\hat{\mathcal{P}}_i$ to represent the predicted local shapes by $\hat{\mathcal{Q}}_i$, the auxiliary loss function can be written as:

$$J_{denoise} = \frac{1}{|\hat{\mathcal{P}}_i|} \sum_{c \in \hat{\mathcal{P}}_i} \min_{g \in \mathcal{G}_{c_i^{gt}}^{gt}} \|c - g\| + \frac{1}{|\mathcal{G}_{c_i^{gt}}^{gt}|} \sum_{g \in \mathcal{G}_{c_i^{gt}}^{gt}} \min_{c \in \hat{\mathcal{P}}_i} \|g - c\|.$$

Our final objective function for point cloud completion is the sum of these two objectives $J_{PC} = J_0 + J_1 + \lambda J_{denoise}$.

4 PoinTr FOR SEMANTIC SCENE COMPLETION

Taking a step towards scene-level completion is essential and meaningful for the rapidly growing 3D community. Compared with object-level point cloud completion, scene-level tasks bring more challenges, since interactions among the elements within a scene should be considered besides intra-object relations. We propose a new semantic scene completion framework and design a new *Geometry-Enhanced block* based on PoinTr as a plug-and-play module for any semantic scene completion models. The overall framework is illustrated in Fig. 7.

4.1 Problem Formulation

Semantic Scene Completion (SSC) aims to simultaneously complete a 3D voxelized scene and predict the semantic labels of objects, given a single-view depth map or RGB-D image. Each scene is voxelized into 3D volumes of $60 \times 36 \times 60$ resolution, and the model is required to perform classification on each voxel in the scene:

$$\hat{S} = \mathcal{M}_{SSC}(\mathcal{I}, \mathcal{D}),$$

where \mathcal{I} and \mathcal{D} represent the input images and depth maps, respectively. \mathcal{M}_{SSC} is a general model for semantic scene completion. The output of the model, \hat{S} , represents the predicted probability over all classes for all voxels. In particular, $\hat{S} = \{\hat{s}_{i,j,k}\} \in \mathcal{R}^{\mathcal{N}+1}$, where \mathcal{N} is the number of object categories and the first dimension of $\hat{s}_{i,j,k}$ represents the possibility of being an empty voxel.

4.2 Geometry-Enhanced Semantic Scene Completion

There are two key challenges for semantic scene completion. The first issue is how to encode the inputs from 2D to 3D. Most recent work [4], [6], [15], [26], [29], [48] utilizes depth maps to encode the geometric information of the seen surface and employs RGB images to compensate for semantic knowledge. Depth maps are encoded as Truncated Signed Distance Function (TSDF), where each voxel stores the distance d to the closest surface with the sign indicating the voxel is free or occluded. This encoding method directly converts the 2D observation into 3D space and bridges the gaps between 2D inputs and 3D outputs. RGB images are processed with a separate branch for 2D feature maps and then projected into 3D voxelized space according to pixel-to-pixel correspondence with depth maps. Then, the encoded inputs are fed into a network consisting of 3D convolutions.

The second issue is how to compensate for the lack of geometric information in the occluded space, since an overall understanding of the whole scene including the occluded part is crucial to avoid the ambiguity caused by the incompleteness of some local areas. However, it still remains an open problem in this area. To explore the solutions to the above issue, we propose a new *geometry-enhanced* semantic scene completion framework by inferring the absent geometric information from the incomplete scene and introducing it to the conventional SSC models, as shown in Fig. 7. We propose to convert the depth maps into 3D point clouds, and adopt a geometry-enhance block based on a modified PoinTr model to learn structural features, building the point-wise interactions for the whole scene. In our geometry-enhanced semantic scene completion framework, there are two main components: 1) conventional semantic

scene model consisting of 3D convolutions; 2) geometry-enhanced block built with Transformers.

4.3 Point-to-Voxel Translation with Transformers

Most previous work doesn't consider point clouds as an input stream since the final complete scenes are required to be in the voxel format. Based on point clouds, we seek a way to capture more precise geometric information and build point-wise interactions as a supplement to existing SSC models. We do not introduce any extra input data compared with other SSC models for the input point clouds are converted from depth maps. It is hard to directly adopt PoinTr for this task because the original PoinTr can not handle the point-to-voxel translation. Since the final scene is voxelized into a fixed size and the spatial location of each voxel is pre-defined, which runs counter to the *Dynamic Queries* scheme in the original PoinTr, we modify the decoder of the original PoinTr to help it adapt to the SSC task. We pre-define K *Voxel Queries* as the inputs of the Transformer decoder instead of generating the queries in an online manner. Meanwhile, we use the same process to encode the incomplete scene as described in Sec. 3. Specifically, we represent the input point clouds of the scene with S point proxies and use the Transformer encoder to build the long-range relationship among all the point proxies by the self-attention mechanism and then feed these outputs into the Transformer decoder. The Transformer decoder takes pre-defined *Voxel Queries* and the outputs of the Transformer encoder as inputs and utilizes the attention mechanism to build the interactions between points and voxels to realize a point-to-voxel translation. After that, we feed the output of the decoder (i.e., *Voxelized Geometric Features*) to other SSC models to provide the detailed geometric information for the entire pipeline.

4.4 Optimization

In semantic scene completion, the predicted scenes and the ground-truth scenes are voxelized before measuring the difference between two scenes. We follow the previous work [48] to adopt the sum of voxel-wise cross-entropy loss as the loss function, which can be written as:

$$\mathcal{J}_{SSC}(\hat{S}, S) = \sum_{i,j,k} \mathcal{L}_{CE}(\hat{s}_{i,j,k}, s_{i,j,k}), \quad (9)$$

where \mathcal{L}_{CE} is cross-entropy loss, $\hat{s}_{i,j,k}$ and $s_{i,j,k}$ are the predicted probability over all classes and the ground-truth label of the voxel at coordinates (i, j, k) , respectively.

5 EXPERIMENTS

We conduct experiments on both object point cloud completion and semantic scene completion tasks. In Section 5.1, we introduce the proposed benchmarks for diverse point cloud completion and the evaluation metric. Then, we show the results of both our method and several baseline methods on our proposed benchmarks. We also demonstrate the effectiveness of our model on the widely used PCN dataset, LiDAR-based KITTI benchmark, and MVP competition on ICCV 2021 Workshop. We perform the ablation studies to analyze each technical design in our AdaPoinTr. In Section 5.2, we detail the datasets and setups of scene-level

completion and report the experimental results for semantic scene completion on NYUCAD [14] and NYUV2 [47].

5.1 Object Point Cloud Completion

5.1.1 Benchmarks for Diverse Point Completion

We choose to generate the samples in our benchmarks based on the synthetic dataset, ShapeNet [65], because it contains the complete object models that cannot be obtained from real-world datasets like ScanNet [8] and S3DIS [2]. What makes our benchmarks distinct is that our benchmarks contain more object categories, more incomplete patterns, and more viewpoints. Besides, we pay more attention to the ability of networks to deal with the objects from novel categories that do not appear in the training set.

ShapeNet-55 Benchmark: In this benchmark, we use all the objects in ShapeNet from 55 categories. Most existing datasets for point cloud completion like PCN [72] only consider a relatively small number of categories (e.g., 8 categories in PCN). However, the incomplete point clouds from real-world scenarios are much more diverse. Therefore, we propose to evaluate the point cloud completion models on all 55 categories in ShapeNet to more comprehensively test the ability of models with a more diverse dataset. We split the original ShapeNet using the 80-20 strategy: we randomly sample 80% objects from each category to form the training set and use the rest for evaluation. As a result, we get 41,952 models for training and 10,518 models for testing. For each object, we randomly sample 8,192 points from the surface to obtain the point cloud.

ShapeNet-34 Benchmark: In this benchmark, we want to explore another important issue in point cloud completion: the performance on novel categories. We believe it is necessary to build a benchmark for this task to better evaluate the generalization performance of models. We first split the origin ShapeNet into two parts: 21 unseen categories and 34 seen categories. In the seen categories, we randomly sample 100 objects from each category to construct a test set of the seen categories (3,400 objects in total) and leave the rest as the training set, resulting in 46,765 object models for training. We also construct another test set consisting of 2,305 objects from 21 novel categories. We evaluate the performance on both the seen and unseen categories to show the generalization ability of models.

Training and Evaluation: In the *ShapeNet-55* and *ShapeNet-34* benchmarks, the partial point clouds for training are generated online. We sample 2048 points from the object as the input and 8192 points as the ground truth. In order to mimic the real-world situation, we first randomly select a viewpoint and then remove the n furthest points from the viewpoint to obtain a training partial point cloud. Our strategy for incomplete point clouds generation is more flexible and efficient for that we generate the incomplete point clouds in an online manner with little cost.. Besides, our strategy also ensures the diversity of our training samples in the aspect of viewpoints. During training, n is randomly chosen from 2048 to 6144 (25% to 75% of the complete point cloud), resulting in different levels of incompleteness. We then down-sample the remaining point clouds to 2048 points as the input data for models.

TABLE 1

Results on *ShapeNet-55*. We report the detailed results for each method on 10 categories and the overall results on 55 categories for three difficulty degrees. \dagger represents re-implemented results. We use CD-S, CD-M, and CD-H to represent the CD- ℓ_2 (multiplied by 1000) results under *Simple*, *Moderate*, and *Hard* settings. We also provided F-Score@1% averaged on three settings.

	Table	Chair	Airplane	Car	Sofa	Bird house	Bag	Remote	Key board	Rocket	CD-S	CD-M	CD-H	CD- ℓ_2 -Avg	F-Score@1%
FoldingNet \dagger [69]	2.53	2.81	1.43	1.98	2.48	4.71	2.79	1.44	1.24	1.48	2.67	2.66	4.05	3.12	0.082
PCN \dagger [72]	2.13	2.29	1.02	1.85	2.06	4.50	2.86	1.33	0.89	1.32	1.94	1.96	4.08	2.66	0.133
TopNet \dagger [54]	2.21	2.53	1.14	2.18	2.36	4.83	2.93	1.49	0.95	1.32	2.26	2.16	4.3	2.91	0.126
PFNet \dagger [24]	3.95	4.24	1.81	2.53	3.34	6.21	4.96	2.91	1.29	2.36	3.83	3.87	7.97	5.22	0.339
GRNet \dagger [67]	1.63	1.88	1.02	1.64	1.72	2.97	2.06	1.09	0.89	1.03	1.35	1.71	2.85	1.97	0.238
SnowflakeNet \dagger [66]	0.98	1.12	0.54	0.98	1.02	1.93	1.08	0.57	0.48	0.61	0.70	1.06	1.96	1.24	0.398
LAKeNet [52]	—	—	—	—	—	—	—	—	—	—	—	—	—	0.89	—
SeedFormer [78]	0.72	0.81	0.40	0.89	0.71	—	—	—	—	—	0.50	0.77	1.49	0.92	0.472
PoinTr [70]	0.81	0.95	0.44	0.91	0.79	1.86	0.93	0.53	0.38	0.57	0.58	0.88	1.79	1.09	0.464
AdaPoinTr	0.62	0.69	0.33	0.81	0.63	1.33	0.68	0.38	0.33	0.34	0.49	0.69	1.24	0.81	0.503

TABLE 2

Results on *ShapeNet-34*. We report the results of 34 seen categories and 21 unseen categories in three difficulty degrees. \dagger represents re-implemented results. We use CD-S, CD-M, and CD-H to represent the CD- ℓ_2 (multiplied by 1000) results under *Simple*, *Moderate*, and *Hard* settings. We also provided F-Score@1% averaged on three settings.

	34 seen categories					21 unseen categories				
	CD-S	CD-M	CD-H	CD- ℓ_2 -Avg	F-Score@1%	CD-S	CD-M	CD-H	CD- ℓ_2 -Avg	F-Score@1%
FoldingNet \dagger [69]	1.86	1.81	3.38	2.35	0.139	2.76	2.74	5.36	3.62	0.095
PCN \dagger [72]	1.87	1.81	2.97	2.22	0.154	3.17	3.08	5.29	3.85	0.101
TopNet \dagger [54]	1.77	1.61	3.54	2.31	0.171	2.62	2.43	5.44	3.50	0.121
PFNet \dagger [24]	3.16	3.19	7.71	4.68	0.347	5.29	5.87	13.33	8.16	0.322
GRNet \dagger [67]	1.26	1.39	2.57	1.74	0.251	1.85	2.25	4.87	2.99	0.216
SnowflakeNet \dagger [66]	0.60	0.86	1.50	0.99	0.422	0.88	1.46	2.92	1.75	0.388
SeedFormer [78]	0.48	0.70	1.30	0.83	0.452	0.61	1.07	2.35	1.34	0.402
PoinTr [70]	0.76	1.05	1.88	1.23	0.421	1.04	1.67	3.44	2.05	0.384
AdaPoinTr	0.48	0.63	1.07	0.73	0.469	0.61	0.96	2.11	1.23	0.416

During the evaluation, we fix 8 viewpoints and n is set to 2048, 4096 or 6144 (25%, 50% or 75% of the whole point cloud) for convenience. According to the value of n , we divide the test samples into three difficulty degrees, *simple*, *moderate*, and *hard* in our experiments. In the following experiments, we will report the performance for each method in *simple*, *moderate*, and *hard* to show the ability of each network to deal with tasks at difficulty levels. In addition, we use the average performance under three difficulty degrees to report the overall performance (*Avg*).

Projected-ShapeNet-55/34 Benchmark: Considering to further bridge the gaps between incomplete point clouds in benchmarks and real-world scenarios, we propose *Projected-ShapeNet-55* and *Projected-ShapeNet-34*, which are another versions of original ShapeNet-55 and ShapeNet-34 sharing the basic setting except for incomplete point cloud generation. In original ShapeNet-55 and ShapeNet-34, the incomplete point clouds are generated by directly cropping complete point clouds, while the incomplete point clouds in *Projected-ShapeNet-55* and *Projected-ShapeNet-34* are generated by noised back-projecting depth images from one viewpoint, as shown in the Fig. 3. We randomly choose 16 viewpoints for each sample to render the depth image and obtain 16 different incomplete-complete point cloud pairs by performing noised back-projecting.

5.1.2 Evaluation Metric

We follow the existing work [24], [54], [67], [72] to use the mean Chamfer Distance as the evaluation metric, which

can measure the distance between the prediction point cloud and ground-truth in set-level. For each prediction, the Chamfer Distance between the prediction point set \mathcal{P} and the ground-truth point set \mathcal{G} is calculated by:

$$d_{CD}(\mathcal{P}, \mathcal{G}) = \frac{1}{|\mathcal{P}|} \sum_{p \in \mathcal{P}} \min_{g \in \mathcal{G}} \|p - g\| + \frac{1}{|\mathcal{G}|} \sum_{g \in \mathcal{G}} \min_{p \in \mathcal{P}} \|g - p\|.$$

Following the previous methods, we use two versions of Chamfer distance as the evaluation metric to compare the performance with existing work. CD- ℓ_1 uses L1-norm to calculate the distance between two points, while CD- ℓ_2 uses L2-norm instead. We also follow [53] to adopt F-Score as another evaluation metric. We define the precision and recall for a point cloud completion result at the threshold d as:

$$P(d) = \frac{1}{|\mathcal{P}|} \sum_{p \in \mathcal{P}} [\min_{g \in \mathcal{G}} \|p - g\| < d],$$

$$R(d) = \frac{1}{|\mathcal{G}|} \sum_{g \in \mathcal{G}} [\min_{p \in \mathcal{P}} \|g - p\| < d].$$

Then we can calculate the F-Score@ d by:

$$\text{F-Score}(d) = \frac{2P(d)R(d)}{P(d) + R(d)},$$

where P(d) and R(d) denote the precision and recall. In our experiments, we set the threshold d to 1%.

5.1.3 Results on ShapeNet-55

We report the performance of our model PoinTr and AdaPoinTr in Table 1 and compare them with the existing

TABLE 3

Results on *Projected-ShapeNet-55*. We report the detailed results for each method on 10 categories and the overall results on 55 categories under $CD-\ell_1$ (multiplied by 1000). We also report F-Score@1% metric. \dagger represents re-implemented results.

	Table	Chair	Airplane	Car	Sofa	Bird house	Bag	Remote	Key board	Rocket	$CD-\ell_1$	F-Score@1%
PCN \dagger [72]	14.79	15.33	9.07	12.85	17.12	20.38	18.64	14.62	13.69	10.98	16.64	0.403
TopNet \dagger [54]	14.40	16.29	9.85	13.61	16.93	22.00	18.69	13.52	11.05	10.45	16.35	0.337
GRNet \dagger [67]	12.01	12.57	8.30	12.13	14.36	16.52	14.67	12.18	9.71	8.58	12.81	0.491
SnowflakeNet \dagger [66]	10.49	11.07	6.35	11.20	12.59	15.24	12.86	10.07	8.12	7.49	11.34	0.594
PoinTr [70]	9.97	10.43	6.02	10.58	12.11	14.60	12.15	9.55	7.61	6.86	10.68	0.615
AdaPoinTr	8.81	9.12	5.18	9.77	10.89	13.27	10.93	8.81	6.79	5.58	9.58	0.701

TABLE 4

Results on *Projected-ShapeNet-34*. We report the results under $CD-\ell_1$ (multiplied by 1000) of 34 seen categories and 21 unseen categories. We also report F-Score@1% metric for each method. \dagger represents re-implemented results.

	34 seen categories					21 unseen categories					$CD-\ell_1$	F-Score@1%
	Bin	Knife	Table	$CD-\ell_1$	F-Score@1%	Microphone	Skateboard	Earphone	$CD-\ell_1$			
PCN \dagger [72]	17.60	8.52	13.69	15.53	0.432	18.05	17.27	24.82	21.44	0.307		
TopNet \dagger [54]	14.90	7.58	11.18	12.96	0.464	14.34	12.59	19.34	15.98	0.358		
GRNet \dagger [67]	14.79	7.84	11.00	12.41	0.506	11.39	10.60	15.00	15.03	0.439		
SnowflakeNet \dagger [66]	13.21	5.80	9.46	10.69	0.616	10.10	9.58	15.19	12.82	0.551		
PoinTr [70]	12.36	5.64	8.97	10.21	0.634	9.34	8.98	14.23	12.43	0.555		
AdaPoinTr	11.45	4.95	7.95	9.12	0.721	7.96	8.34	12.30	11.37	0.642		

methods. We implement FoldingNet [69], PCN [72], TopNet [54], PFNet [24], GRNet [67] and SnowflakeNet [66] on our benchmark according to their open-source code, using the best hyper-parameters in their papers for fair comparisons. As shown in the table, our method can better cope with different situations with diverse viewpoints, diverse object categories, diverse incomplete patterns, and diverse incompleteness levels. We report the average chamfer distance on three settings for ten categories in the first ten columns of the table. Specifically, *Table*, *chair*, *Airplane*, *Car* and *Sofa* contain more than 2500 samples in the training set while *Birdhouse*, *Bag*, *Remote*, *Keyboard* and *Rocket* contain less than 80 samples. And we also provide detailed results for all 55 categories in our supplementary material. As shown in the last two columns of the table, our AdaPoinTr performs better than other existing work and establish a new state-of-the-art, which achieves 0.81 $CD-\ell_2$ and 0.503 F-Score on ShapeNet-55. These results clearly demonstrate the effectiveness of our AdaPoinTr even under such a diverse situation. Besides, AdaPoinTr achieves 0.09, 0.19 and 0.55 improvement in $CD-\ell_2$ (multiplied by 1000) under three settings (simple, moderate and hard) compared with PoinTr.

5.1.4 Results on ShapeNet-34

On ShapeNet-34, we also conduct experiments for our method and existing methods. The results are shown in Table 2. For the 34 seen categories, we can see our method outperforms all the other methods. For the 21 unseen categories, we use the networks that are trained on the 34 seen categories to evaluate the performance on the novel objects from the other 21 categories that do not appear in the training phase. We see our method also achieves the best performance in this more challenging setting. Comparing with the results of seen categories, our AdaPoinTr obtains a more dominant performance over different difficulty de-

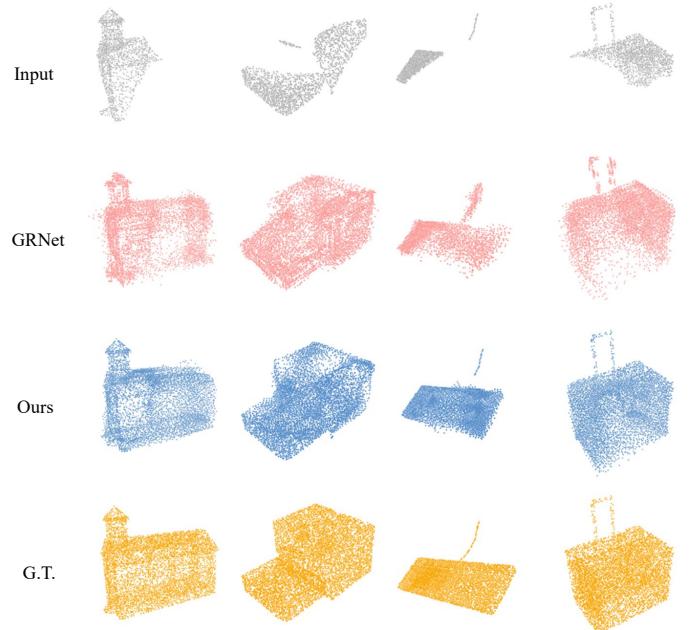


Fig. 8. Results on some objects from novel categories in ShapeNet-34. We show the input point cloud and the ground truth as well as the predictions of GRNet and our PoinTr.

grees, which demonstrates the superior transferability of our AdaPoinTr. We visualize the results in Fig. 8 to show the effectiveness of our method on the unseen categories.

5.1.5 Results on Projected-ShapeNet-55

We conduct experiments on Projected-ShapeNet-55, where the input point cloud is generated by the noised back-projecting method. In Projected-ShapeNet-55, there exists a noise in the input point cloud, making the dataset more

TABLE 5

Results on LiDAR scans from KITTI dataset under the Fidelity and MMD metrics. We follow the previous work to finetune the model on PCNCars.

CD- ℓ_2 ($\times 1000$)	AtlasNet [19]	PCN [72]	FoldingNet [69]	TopNet [54]	MSN [27]	NSFA [76]	PFNet [24]	CRN [61]	GRNet [67]	SeedFormer	PoinTr	AdaPoinTr
Fidelity ↓	1.759	2.235	7.467	5.354	0.434	1.281	1.137	1.023	0.816	0.151	0.000	0.237
MMD ↓	2.108	1.366	0.537	0.636	2.259	0.891	0.792	0.872	0.568	0.516	0.526	0.392

TABLE 6

Results on the PCN dataset. We use the CD- ℓ_1 (multiplied by 1000) and F-Score@1% to compare with other methods.

	Air	Cab	Car	Cha	Lam	Sof	Tab	Wat	Avg CD- ℓ_1	F-Score@1%
FoldingNet [69]	9.49	15.80	12.61	15.55	16.41	15.97	13.65	14.99	14.31	0.322
AtlasNet [19]	6.37	11.94	10.10	12.06	12.37	12.99	10.33	10.61	10.85	0.616
PCN [72]	5.50	22.70	10.63	8.70	11.00	11.34	11.68	8.59	9.64	0.695
TopNet [54]	7.61	13.31	10.90	13.82	14.44	14.78	11.22	11.12	12.15	0.503
MSN [27]	5.60	11.90	10.30	10.20	10.70	11.60	9.60	9.90	10.00	0.705
GRNet [67]	6.45	10.37	9.45	9.41	7.96	10.51	8.44	8.04	8.83	0.708
PMP-Net [64]	5.65	11.24	9.64	9.51	6.95	10.83	8.72	7.25	8.73	—
CRN [61]	4.79	9.97	8.31	9.49	8.94	10.69	7.81	8.05	8.51	—
NSFA [76]	4.76	10.18	8.63	8.53	7.03	10.53	7.35	7.48	8.06	—
SnowFlake [66]	4.29	9.16	8.08	7.89	6.07	9.23	6.55	6.40	7.21	—
LAKeNet [52]	4.17	9.78	8.56	7.45	5.88	9.39	6.43	5.98	7.23	—
SeedFormer [78]	3.85	9.05	8.06	7.06	5.21	8.85	6.05	5.85	6.74	—
PoinTr	4.75	10.47	8.68	9.39	7.75	10.93	7.78	7.29	8.38	0.745
AdaPoinTr	3.68	8.82	7.47	6.85	5.47	8.35	5.80	5.76	6.53	0.845

TABLE 7

Results on the MVP validation set. Inputs and outputs contain 2048 points. We report CD- ℓ_2 (multiplied by 10000) and F-Score@1% for each method.

Model	#Points	CD- ℓ_2	F-Score@1%
PCN [72]	2048	9.77	0.321
TopNet [54]	2048	10.11	0.308
ECG [36]	2048	7.25	0.434
CRN [61]	2048	6.64	0.476
VRCNet [37]	2048	5.96	0.499
PoinTr	2048	6.15	0.456
AdaPoinTr	2048	4.71	0.545

challenging and realistic. We report the results of our models and other existing methods. We report the class-wise CD and overall CD for all methods. Limited by the page size, we only pick 10 categories of all to show the detailed results. As shown in Table 3, our AdaPoinTr obtains the best performance on ten categories and overall CD, achieving 1.1 improvements on CD- ℓ_1 (multiplied by 1000) and 0.086 on F-Score@1% compared with PoinTr.

5.1.6 Results on Projected-ShapeNet-34

We also explore the performance of our AdaPoinTr and other existing methods on Project-ShapeNet-34, as shown in Table 4. On Project-ShapeNet-34, the model is trained on the training split of 34 seen categories, then tested on the test split of 34 seen categories and 21 unseen categories. AdaPoinTr outperforms SnowflakeNet [66] on both 34 seen categories and 21 unseen 21 categories.

5.1.7 Results on the Existing Benchmarks

Apart from the experiments on the newly proposed challenging benchmarks, we also conduct the experiments on the existing benchmarks including the PCN dataset [72],

LiDAR-based KITTI benchmark [16] and MVP Challenges [37].

Results on the PCN Dataset. The PCN dataset [72] is one of the most widely used benchmark datasets for the point cloud completion task. To verify the effectiveness of our method on existing benchmarks and compare it with more state-of-the-art methods, we conducted experiments on this dataset following the standard protocol and evaluation metric used in previous work [27], [61], [64], [66], [67], [72]. The results are shown in Table 6. We see our method largely improves the previous methods and establishes the new state-of-the-art on this dataset.

Results on the KITTI Benchmark. To show the performance of our method in real-world scenarios, we follow [67] to finetune our trained model on ShapeNetCars [72] and evaluate the performance of our model on KITTI dataset, which contains the incomplete point clouds of cars in the real-world scenes from LiDAR scans. We report the Fidelity and MMD metrics in Table 5 and show some reconstruction results in Fig. 10. Our method achieves better qualitative and quantitative performance. See supplementary for the description of Fidelity and MMD.

Results on the MVP Benchmark. The MVP benchmark evaluates the performance of generating complete 3D point clouds based on single-view partial point clouds. We conduct the experiments on its validation set and submit our model to MVP Challenges hosted in the ICCV 2021 Workshop for the online evaluation on the test set. We show the results on validation set in Table 7. Our method achieves the best performance among all the previous work, including the strong SOTA method VRCNet [37]. Besides, we ranked first on the online leaderboard of the MVP benchmark and won the first prize in the MVP Challenge [39].

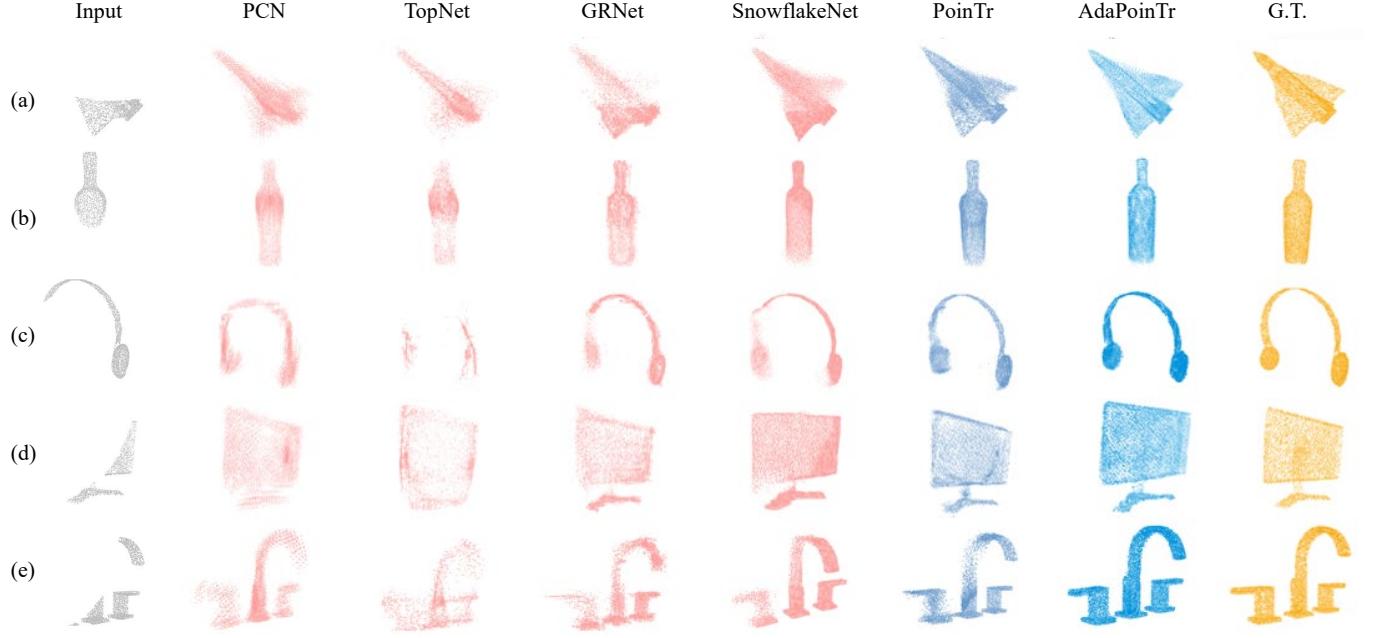


Fig. 9. Qualitative results on ShapeNet-55. All methods above take the point clouds in the first column as inputs and generate complete point clouds. Our methods can complete the point clouds with higher fidelity, which clearly shows the effectiveness of our method.

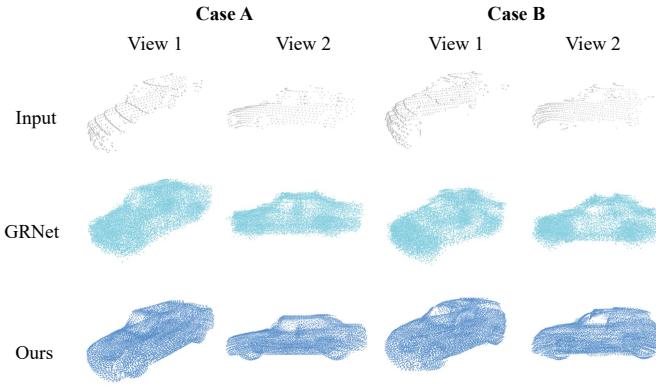


Fig. 10. Qualitative results on the KITTI. In order to better show the shape of the car, we provide two views of the same point cloud in each case. Our method can recover a car with more accurate boundaries and details (e.g. tires of cars).

5.1.8 Model Design Analysis

To examine the effectiveness of our designs, we conduct a detailed ablation study on the key components of AdaPoinTr. The results are summarized in Table 8. The baseline model A is the vanilla Transformer model for point cloud completion, which uses the encoder-decoder architecture with the standard Transformer blocks. In this model, we form the point proxies directly from the point cloud using a single-layer DGCNN model. We then add the query generator between the encoder and decoder (model B). We see the query generator improve the baseline by 0.34 in Chamfer distance. When using DGCNN to extract features from the input point cloud (model C), we observe a significant improvement to 8.69. By adding the geometric block to all the Transformer blocks (model D), we see the performance can be further improved, which clearly demonstrates the effectiveness of the geometric structures learned by the

TABLE 8
Ablation study on the PCN dataset. We investigate different designs including query generator (Query), DGCNN feature extractor (DGCNN), Geometry-aware Blocks (Geometry), Adaptive Query Generation (Ada. Query), Query Selection (Selection) and Denoising task (Denoising). We report $CD-\ell_1$ (multiplied by 1000) and F-Score@1%.

Model	Query	DGCNN	Geometry	$CD-\ell_1$	F-Score@1%
				$CD-\ell_1$	F-Score@1%
A				9.43	0.678
B	✓			9.09	0.713
C	✓	✓		8.69	0.736
D	✓	✓	✓	8.44	0.741
PoinTr	✓	✓	all	8.38	0.745
			1 st		
Model	Ada. Query	Selection	Denoising	$CD-\ell_1$	F-Score@1%
PoinTr				8.38	0.745
F	✓			7.06	0.797
G	✓	✓		6.92	0.810
AdaPoinTr	✓	✓	✓	6.53	0.844

block. We find that only adding the geometric block to the first Transformer block in both encoder and decoder can lead to a slightly better performance (PoinTr), which indicates the role of the geometric block is to introduce the inductive bias and a single layer is sufficient while adding more blocks may result in over-fitting. We then adopt the adaptive query generation mechanism on PoinTr, which achieves an improvement about 1.28 (model F). By adding the dynamic selection, the model can be more flexible when dealing with diverse categories and situations, which brings 0.14 improvement (model G). Finally, by introducing the auxiliary denoising task, our AdaPoinTr achieves the-state-of-art performance.

5.1.9 Complexity Analysis

Our method achieves the best performance on both our newly proposed diverse benchmarks and the existing

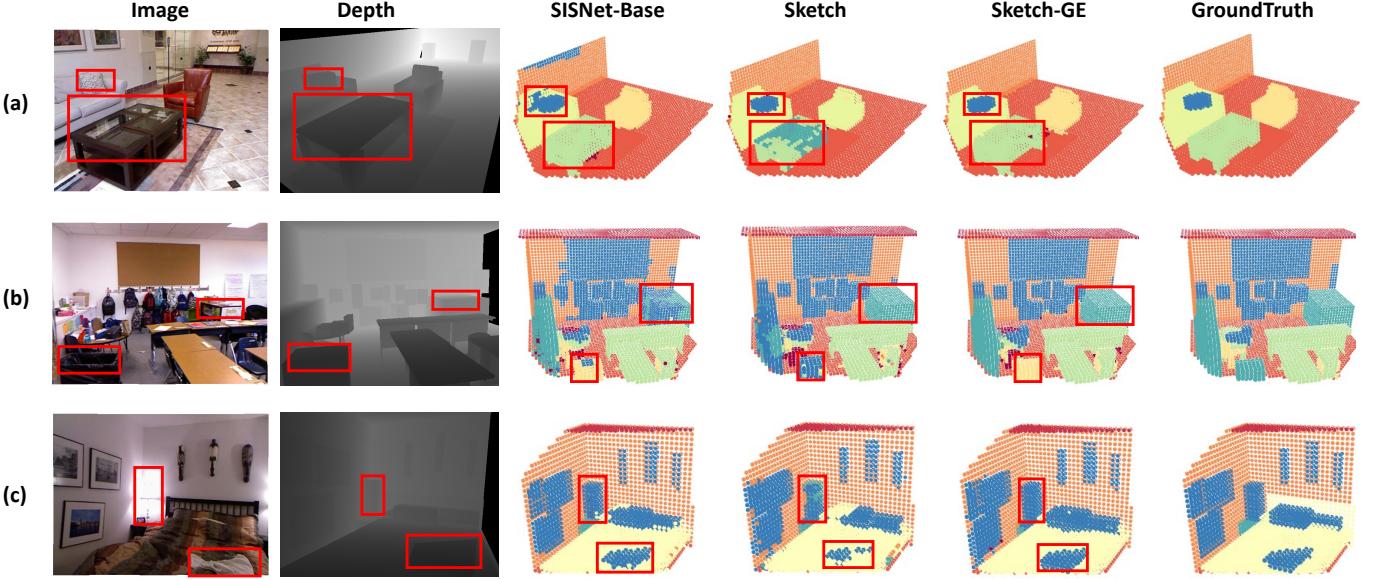


Fig. 11. Qualitative results on NYUCAD. All methods above are evaluated on both the observed and occluded voxels in the view frustum. We highlight some regions with *red* bounding box, which clearly show the effectiveness of our proposed block.

TABLE 9

Complexity analysis. We report theoretical computation cost (FLOPs) and throughput (T.put) of our method and existing methods. We also provide Chamfer distances metric of all categories in ShapeNet-55 and projected-ShapeNet-55 (CD_{55} and CD_{p55}), unseen categories in ShapeNet34 and projected-ShapeNet-34 (CD_{34} and CD_{p34}) and PCN benchmark as references.

Models	FLOPs	T.put	CD_{55}	CD_{34}	CD_{p55}	CD_{p34}	CD_{PCN}
FoldingNet [69]	27.58 G	158 pc/s	3.12	3.62	—	—	14.31
PCN [72]	15.25 G	292 pc/s	2.66	3.85	16.64	21.44	9.64
TopNet [54]	6.72 G	248 pc/s	2.91	3.50	16.35	15.98	12.15
GRNet [67]	40.44 G	65 pc/s	1.97	2.99	12.81	15.03	8.83
SnowflakeNet [66]	17.16 G	72 pc/s	1.24	1.75	11.34	12.82	7.21
LAKeNet [52]	24.48 G	3 pc/s	0.89	—	—	—	7.23
SeedFormer [78]	13.97 G	21 pc/s	0.92	1.34	—	—	6.74
PoinTr	10.41 G	62 pc/s	1.09	2.05	10.68	12.43	8.38
AdaPoinTr	12.43 G	61 pc/s	0.81	1.23	9.58	11.37	6.53

benchmarks. We provide the detailed complexity analysis of our method in Table 9. We report theoretical computation cost (FLOPs) and Throughput (Point Clouds Per Second) of our method and other methods. We also provide the results on ShapeNet-55/Projected-ShapeNet-55, unseen categories in ShapeNet-34/Projected-ShapeNet-34 and PCN benchmark as references. We observe that many recent advanced methods sacrifice efficiency in pursuit of higher performance while our method achieves the best performance on those benchmarks with less FLOPs and can process more than 60 point clouds within one second (using a batch size of 1). We argue that Throughputs should be paid more attention since it determines whether the model can be applied in some real-time situations.

5.1.10 Qualitative Results

In Fig. 9, we show some completion results for all methods and find our method performs better. For example, the input data in (a) nearly lose all the geometric information and can be hardly recognized as an airplane. In this

case, other methods can only roughly complete the shape with unsatisfactory geometry details (clear contour of the wings), while our method can still complete the point cloud with higher fidelity. These results show our method has a stronger ability to recover details and is more robust to various incomplete patterns.

5.2 Semantic Scene Completion

5.2.1 Benckmarks for Semantic Scene Completion

In our experiments, we evaluate our method on two real-world datasets. These two datasets are the popular NYU Depth V2 [47] (denoted as NYUV2 in the following part) and NYUCAD [14]. They both consist of 1449 indoor scenes and we follow the previous work to divide the datasets into training and test split, containing 795 and 654 scenes respectively. We follow the same experiment setting as [48].

5.2.2 Evaluation Metric

In our experiments, we follow [48] to consider two sets of evaluation metrics: evaluation metric for scene completion (SC) and for semantic scene completion (SSC). Following [14], we do not consider the voxels outside the view or the room while evaluating.

The evaluation metrics for SC. We focus on the occupancy prediction results for each voxel in the scene. We follow the previous work to perform a binary prediction, *i.e.*, empty or non-empty. We use *recall*, *precision* and voxel-wise intersection over union (*IoU*) as the evaluation metrics to evaluate the performance on occluded voxels in the view frustum.

The evaluation metrics for SSC. We focus on the semantic understanding of the entire scene. We regard all the empty voxels as one additional category and evaluate the intersection over union (*IoU*) for each category on both the observed and occluded voxels in the view frustum.

TABLE 10

Results on NYUV2 dataset. We compare our method with other end-to-end and iterative methods. We evaluate the models using *Precision*, *Recall* and *IoU* for Scene Completion, detailed *IoU* and *mIoU* for Semantic Scene Completion. PoinTr only takes depth maps as the input, while other methods take additional RGB images as another input.

Type	Scene Completion				Semantic Scene Completion											
	Prec.	Recall	IoU		ceil	floor	wall	win.	chair	bed	sofa	table	tvs	furn	objs	mIoU
SISNet [4]	Iterative	90.7	84.6	77.8	53.9	93.2	51.3	38.0	38.7	65.0	56.3	37.8	25.9	51.3	36.0	49.8
PoinTr [70]	Depth Only	71.2	91.3	62.9	—	—	—	—	—	—	—	—	—	—	—	—
SSCNet [48]	End-to-end	57.0	94.5	55.1	15.1	94.7	24.4	0.0	12.6	32.1	35.0	13.0	7.8	27.1	10.1	24.7
AICNet [25]	End-to-end	62.4	91.8	59.2	23.2	90.8	32.3	14.8	18.2	51.1	44.8	15.2	22.4	38.3	15.7	33.3
TS3D [15]	End-to-end	—	—	60.0	9.7	93.4	25.5	21.0	17.4	55.9	49.2	17.0	27.5	39.4	19.3	34.1
CCPNet [75]	End-to-end	74.2	90.8	63.5	23.5	96.3	35.7	20.2	25.8	61.4	56.1	18.1	28.1	37.8	20.1	38.5
SISNet-Base [4]	End-to-end	87.6	78.9	71.0	46.9	93.3	41.3	26.7	30.8	58.4	49.5	27.2	22.1	42.2	28.7	42.5
DDRNet [26]	End-to-end	71.5	80.8	61.0	21.1	92.2	33.5	6.8	14.8	48.3	42.3	13.2	13.9	35.3	13.2	30.4
Sketch [6]	End-to-end	85.0	81.6	71.3	43.1	93.6	40.5	24.3	30.0	57.1	49.3	29.2	14.3	42.5	28.6	41.1
DDRNet-GE	End-to-end	76.2	81.9	65.2	24.6	92.4	36.4	17.1	18.7	49.1	41.9	15.9	27.6	37.1	15.6	34.2
Sketch-GE	End-to-end	90.1	82.9	74.6	45.6	93.7	41.5	29.5	35.4	56.3	48.1	26.9	32.8	45.1	30.3	44.1

TABLE 11

Results on NYUCAD dataset. We compare our method with other end-to-end and iterative methods. We evaluate the models using *Precision*, *Recall* and *IoU* for Scene Completion, detailed *IoU* and *mIoU* for Semantic Scene Completion. PoinTr only takes depth maps as the input, while other methods take additional RGB images as another input.

Type	Scene Completion				Semantic Scene Completion											
	Prec.	Recall	IoU		ceil	floor	wall	win.	chair	bed	sofa	table	tvs	furn	objs	mIoU
SISNet [4]	Iterative	94.2	91.3	86.5	65.6	94.4	67.1	45.2	57.2	75.5	66.4	50.9	31.1	62.5	42.9	59.9
PoinTr [70]	Depth Only	89.8	91.7	80.9	—	—	—	—	—	—	—	—	—	—	—	—
SSCNet [48]	End-to-end	75.4	96.3	73.2	32.5	92.6	40.2	8.9	33.9	57.0	59.5	28.3	8.1	44.8	25.1	40.0
AICNet [25]	End-to-end	88.2	90.3	80.5	53.0	91.2	57.2	20.2	44.6	58.4	56.2	36.2	9.7	47.1	30.4	45.8
TS3D [15]	End-to-end	—	—	76.1	25.9	93.8	48.9	33.4	31.2	66.1	56.4	31.6	38.5	51.4	30.8	46.2
CCPNet [75]	End-to-end	91.3	92.6	82.4	56.2	94.6	58.7	35.1	44.8	68.6	65.3	37.6	35.5	53.1	35.2	53.2
SISNet-Base [4]	End-to-end	—	—	82.8	—	—	—	—	—	—	—	—	—	—	—	53.6
DDRNet [26]	End-to-end	88.7	88.5	79.4	54.1	91.5	56.4	14.9	37.0	55.7	51.0	28.8	9.2	44.1	27.8	42.8
Sketch [6]	End-to-end	90.6	92.2	84.2	59.7	94.3	64.3	32.6	51.7	72.0	68.7	45.9	19.0	60.5	38.5	55.2
DDRNet-GE	End-to-end	89.4	90.2	81.2	56.7	91.9	57.7	25.1	36.5	54.1	49.3	29.1	21.8	43.2	26.9	44.7
Sketch-GE	End-to-end	94.4	90.64	86.1	65.1	94.4	64.9	40.9	50.3	69.4	58.5	38.5	40.8	57.7	37.2	56.1

5.2.3 Results on NYUV2 and NYUCAD

Results on NYUV2 and NYUCAD. On NYUV2 and NYUCAD, we implement the proposed block to the existing SOTA model, donated as DDRNet-GE and Sketch-GE. As shown in the Tabel 10, we compare our methods with other methods under SC and SSC tasks. The proposed block improves the performance of DDRNet [26] and Sketch and establishes a new SOTA for the end-to-end SSC model. Furthermore, We convert the depth map of a scene into a point cloud, and use PoinTr with Point-to-Voxel Translation (see Sec. 4.3) to treat it as a larger-scale completion task, we show the result in the table.

5.2.4 Qualitative Results

In Fig. 11, we compare our method with other end-to-end methods for SSC and show some completion results from SISNet-Base [4] and Sketch [6]. By comparing Sketch-GE and Sketch [6], we can find that our proposed block can enhance the geometric information for instances in the scenes. For example, our block helps the model to correctly recognize the table in (a), chair in (b) and object in (c). The good performance for our method is largely based on the geometrical knowledge and point-wise interactions learned

by Transformers. By leveraging the geometric relations with the proposed block, models are encouraged to be aware of the existence of objects in scenes and effectively propagate and integrate the information from objects and scenes.

6 CONCLUSION

In this paper, we have proposed a new architecture, PoinTr, to convert the point cloud completion task into a set-to-set translation task. With several technical innovations, we successfully applied the Transformer model to this task and achieved state-of-the-art performance. Moreover, we proposed four more challenging benchmarks for more diverse object point cloud completion. We also verify that PoinTr is helpful to scene-level tasks. We expect introductions of PoinTr can provide some inspiration for the researchers in this area and we think extending our Transformer architecture to more 3D tasks can an interesting future direction.

ACKNOWLEDGEMENT

This work was supported in part by the National Key Research and Development Program of China under Grant 2017YFA0700802, in part by the National Natural Science Foundation of China under Grant 62125603, Grant

61822603, Grant U1813218, Grant U1713214, in part by Beijing Academy of Artificial Intelligence (BAAI), and in part by a grant from the Institute for Guo Qiang, Tsinghua University.

REFERENCES

- [1] P. Achlioptas, O. Diamanti, I. Mitliagkas, and L. J. Guibas, "Learning representations and generative models for 3d point clouds," in *ICLR*, 2018. 1
- [2] I. Armeni, S. Sax, A. R. Zamir, and S. Savarese, "Joint 2d-3d semantic data for indoor scene understanding," *arXiv preprint arXiv:1702.01105*, 2017. 8
- [3] I. Bello, B. Zoph, A. Vaswani, J. Shlens, and Q. V. Le, "Attention augmented convolutional networks," in *ICCV*, 2019. 5
- [4] Y. Cai, X. Chen, C. Zhang, K.-Y. Lin, X. Wang, and H. Li, "Semantic scene completion via integrating instances and scene in-the-loop," in *CVPR*, 2021. 4, 7, 14
- [5] N. Carion, F. Massa, G. Synnaeve, N. Usunier, A. Kirillov, and S. Zagoruyko, "End-to-end object detection with transformers," in *ECCV*. Springer, 2020. 4
- [6] X. Chen, K.-Y. Lin, C. Qian, G. Zeng, and H. Li, "3d sketch-aware semantic scene completion via semi-supervised structure prior," in *CVPR*, 2020. 3, 7, 14
- [7] C. B. Choy, D. Xu, J. Gwak, K. Chen, and S. Savarese, "3d-r2n2: A unified approach for single and multi-view 3d object reconstruction," in *ECCV*, B. Leibe, J. Matas, N. Sebe, and M. Welling, Eds., 2016, pp. 628–644. 3
- [8] A. Dai, A. X. Chang, M. Savva, M. Halber, T. Funkhouser, and M. Nießner, "Scannet: Richly-annotated 3d reconstructions of indoor scenes," in *CVPR*, 2017. 8
- [9] A. Dai, C. R. Qi, and M. Nießner, "Shape completion using 3d encoder-predictor cnns and shape synthesis," in *CVPR*, 2017. 1, 3
- [10] J. Devlin, M.-W. Chang, K. Lee, and K. Toutanova, "Bert: Pre-training of deep bidirectional transformers for language understanding," *arXiv preprint arXiv:1810.04805*, 2018. 4, 5
- [11] J. Devlin, M. Chang, K. Lee, and K. Toutanova, "BERT: pre-training of deep bidirectional transformers for language understanding," in *NAACL*, J. Burstein, C. Doran, and T. Solorio, Eds., 2019. 4
- [12] A. Dosovitskiy, L. Beyer, A. Kolesnikov, D. Weissenborn, X. Zhai, T. Unterthiner, M. Dehghani, M. Minderer, G. Heigold, S. Gelly *et al.*, "An image is worth 16x16 words: Transformers for image recognition at scale," *arXiv preprint arXiv:2010.11929*, 2020. 4
- [13] H. Fan, H. Su, and L. J. Guibas, "A point set generation network for 3d object reconstruction from a single image," in *CVPR*, 2017. 7
- [14] M. Firman, O. Mac Aodha, S. Julier, and G. J. Brostow, "Structured prediction of unobserved voxels from a single depth image," in *CVPR*, 2016. 3, 8, 13, 17
- [15] M. Garbade, Y.-T. Chen, J. Sawatzky, and J. Gall, "Two stream 3d semantic scene completion," in *CVPRW*, 2019. 3, 7, 14
- [16] A. Geiger, P. Lenz, C. Stiller, and R. Urtasun, "Vision meets robotics: The kitti dataset. international journal of robotics research (ijrr)," 2013. 3, 11
- [17] R. Girdhar, D. F. Fouhey, M. Rodriguez, and A. Gupta, "Learning a predictable and generative vector representation for objects," in *ECCV*, B. Leibe, J. Matas, N. Sebe, and M. Welling, Eds., 2016. 3
- [18] B. Graham, M. Engelcke, and L. van der Maaten, "3d semantic segmentation with submanifold sparse convolutional networks," in *CVPR*, 2018. 3
- [19] T. Groueix, M. Fisher, V. G. Kim, B. C. Russell, and M. Aubry, "Atlasnet: A papier-mâché approach to learning 3d surface generation," *CoRR*, 2018. 11
- [20] ——, "A papier-mâché approach to learning 3d surface generation," in *CVPR*, 2018. 1
- [21] M.-H. Guo, J.-X. Cai, Z.-N. Liu, T.-J. Mu, R. R. Martin, and S.-M. Hu, "Pct: Point cloud transformer," *Computational Visual Media*, 2021. 4
- [22] Y.-X. Guo and X. Tong, "View-volume network for semantic scene completion from a single depth image," *arXiv preprint arXiv:1806.05361*, 2018. 3
- [23] X. Han, Z. Li, H. Huang, E. Kalogerakis, and Y. Yu, "High-resolution shape completion using deep neural networks for global structure and local geometry inference," in *ICCV*, 2017. 1, 3
- [24] Z. Huang, Y. Yu, J. Xu, F. Ni, and X. Le, "Pf-net: Point fractal network for 3d point cloud completion," in *CVPR*, 2020. 1, 3, 5, 9, 10, 11, 18, 20, 21
- [25] J. Li, K. Han, P. Wang, Y. Liu, and X. Yuan, "Anisotropic convolutional networks for 3d semantic scene completion," in *CVPR*, 2020. 3, 14
- [26] J. Li, Y. Liu, D. Gong, Q. Shi, X. Yuan, C. Zhao, and I. Reid, "Rgbd based dimensional decomposition residual network for 3d semantic scene completion," in *CVPR*, 2019. 3, 7, 14
- [27] M. Liu, L. Sheng, S. Yang, J. Shao, and S.-M. Hu, "Morphing and sampling network for dense point cloud completion," *arXiv preprint arXiv:1912.00280*, 2019. 11
- [28] ——, "Morphing and sampling network for dense point cloud completion," in *AAAI*, 2020. 3
- [29] S. Liu, Y. Hu, Y. Zeng, Q. Tang, B. Jin, Y. Han, and X. Li, "See and think: Disentangling semantic scene completion," *NeurIPS*, 2018. 3, 7
- [30] Y. Liu, B. Fan, S. Xiang, and C. Pan, "Relation-shape convolutional neural network for point cloud analysis," in *CVPR*, 2019. 1
- [31] Z. Liu, H. Tang, Y. Lin, and S. Han, "Point-voxel CNN for efficient 3d deep learning," in *NeurIPS*, H. M. Wallach, H. Larochelle, A. Beygelzimer, F. d'Alché-Buc, E. B. Fox, and R. Garnett, Eds., 2019. 1
- [32] I. Loshchilov and F. Hutter, "Fixing weight decay regularization in adam," 2018. 17
- [33] C. Lüscher, E. Beck, K. Irie, M. Kitza, W. Michel, A. Zeyer, R. Schlüter, and H. Ney, "RWTH ASR systems for librispeech: Hybrid vs attention - w/o data augmentation," *CoRR*, vol. abs/1905.03072, 2019. 4
- [34] P. Mandikal and R. V. Babu, "Dense 3d point cloud reconstruction using a deep pyramid network," *CoRR*, vol. abs/1901.08906, 2019. 1
- [35] D. T. Nguyen, B. Hua, M. Tran, Q. Pham, and S. Yeung, "A field model for repairing 3d shapes," in *CVPR*, 2016. 1
- [36] L. Pan, "Ecg: Edge-aware point cloud completion with graph convolution," *IEEE Robotics and Automation Letters*, 2020. 11
- [37] L. Pan, X. Chen, Z. Cai, J. Zhang, H. Zhao, S. Yi, and Z. Liu, "Variational relational point completion network," *CVPR*, 2021. 2, 3, 11
- [38] L. Pan, T. Wu, Z. Cai, Z. Liu, X. Yu, Y. Rao, J. Lu, J. Zhou, M. Xu, X. Luo *et al.*, "Multi-view partial (mvp) point cloud challenge 2021 on completion and registration: Methods and results," *arXiv e-prints*, pp. ArXiv-2112, 2021. 3
- [39] ——, "Multi-view partial (mvp) point cloud challenge 2021 on completion and registration: Methods and results," *arXiv preprint arXiv:2112.12053*, 2021. 3, 11
- [40] N. Parmar, A. Vaswani, J. Uszkoreit, L. Kaiser, N. Shazeer, and A. Ku, "Image transformer," *CoRR*, vol. abs/1802.05751, 2018. 4
- [41] C. R. Qi, H. Su, K. Mo, and L. J. Guibas, "Pointnet: Deep learning on point sets for 3d classification and segmentation," in *CVPR*, 2017. 1, 3
- [42] C. R. Qi, L. Yi, H. Su, and L. J. Guibas, "Pointnet++: Deep hierarchical feature learning on point sets in a metric space," in *NeurIPS*, I. Guyon, U. von Luxburg, S. Bengio, H. M. Wallach, R. Fergus, S. V. N. Vishwanathan, and R. Garnett, Eds., 2017. 1, 3, 5
- [43] L. Roldao, R. De Charette, and A. Verroust-Blondet, "3d semantic scene completion: a survey," *arXiv preprint arXiv:2103.07466*, 2021. 3
- [44] S. Ruder, "An overview of gradient descent optimization algorithms," *arXiv preprint arXiv:1609.04747*, 2016. 17
- [45] M. Sarmad, H. J. Lee, and Y. M. Kim, "Rl-gan-net: A reinforcement learning agent controlled gan network for real-time point cloud shape completion," in *CVPR*, 2019. 1
- [46] A. Sharma, O. Grau, and M. Fritz, "Vconv-dae: Deep volumetric shape learning without object labels," *CoRR*, vol. abs/1604.03755, 2016. 1
- [47] N. Silberman, D. Hoiem, P. Kohli, and R. Fergus, "Indoor segmentation and support inference from rgbd images," in *ECCV*. Springer, 2012. 3, 8, 13
- [48] S. Song, F. Yu, A. Zeng, A. X. Chang, M. Savva, and T. Funkhouser, "Semantic scene completion from a single depth image," in *CVPR*, 2017. 3, 7, 8, 13, 14
- [49] D. Stutz and A. Geiger, "Learning 3d shape completion from laser scan data with weak supervision," in *CVPR*, 2018. 1, 3

- [50] H. Su, V. Jampani, D. Sun, S. Maji, E. Kalogerakis, M. Yang, and J. Kautz, "Splatnet: Sparse lattice networks for point cloud processing," in *CVPR*, 2018. 3
- [51] G. Synnaeve, Q. Xu, J. Kahn, T. Likhomanenko, E. Grave, V. Pratap, A. Sriram, V. Liptchinsky, and R. Collobert, "End-to-end ASR: from Supervised to Semi-Supervised Learning with Modern Architectures," *arXiv e-prints*, 2019. 4
- [52] J. Tang, Z. Gong, R. Yi, Y. Xie, and L. Ma, "Lake-net: Topology-aware point cloud completion by localizing aligned keypoints," in *CVPR*, 2022. 3, 9, 11, 13
- [53] M. Tatarchenko, S. R. Richter, R. Ranftl, Z. Li, V. Koltun, and T. Brox, "What do single-view 3d reconstruction networks learn?" *CoRR*, 2019. 9
- [54] L. P. Tchapmi, V. Kosaraju, H. Rezatofighi, I. D. Reid, and S. Savarese, "Topnet: Structural point cloud decoder," in *CVPR*, 2019. 1, 2, 3, 9, 10, 11, 13, 18, 20, 21, 22
- [55] H. Touvron, M. Cord, M. Douze, F. Massa, A. Sablayrolles, and H. Jégou, "Training data-efficient image transformers & distillation through attention," in *ICML*. PMLR, 2021. 4
- [56] J. Varley, C. DeChant, A. Richardson, J. Ruales, and P. K. Allen, "Shape completion enabled robotic grasping," in *IROS*, 2017. 1
- [57] A. Vaswani, N. Shazeer, N. Parmar, J. Uszkoreit, L. Jones, A. N. Gomez, L. Kaiser, and I. Polosukhin, "Attention is all you need," in *NeurIPS*, 2017. 1, 4, 5, 16
- [58] O. Vinyals, S. Bengio, and M. Kudlur, "Order matters: Sequence to sequence for sets," *arXiv preprint arXiv:1511.06391*, 2015. 4
- [59] P. Wang, Y. Liu, Y. Guo, C. Sun, and X. Tong, "O-CNN: octree-based convolutional neural networks for 3d shape analysis," *ACM Trans. Graph.*, 2017. 3
- [60] W. Wang, Q. Huang, S. You, C. Yang, and U. Neumann, "Shape inpainting using 3d generative adversarial network and recurrent convolutional networks," in *ICCV*, 2017. 1
- [61] X. Wang, M. H. Ang Jr, and G. H. Lee, "Cascaded refinement network for point cloud completion," in *CVPR*, 2020. 6, 11
- [62] Y. Wang, D. J. Tan, N. Navab, and F. Tombari, "Forknet: Multi-branch volumetric semantic completion from a single depth image," in *ICCV*, 2019. 3
- [63] Y. Wang, Y. Sun, Z. Liu, S. E. Sarma, M. M. Bronstein, and J. M. Solomon, "Dynamic graph cnn for learning on point clouds," *TOG*, 2019. 4, 5, 16
- [64] X. Wen, P. Xiang, Z. Han, Y.-P. Cao, P. Wan, W. Zheng, and Y.-S. Liu, "Pmp-net: Point cloud completion by learning multi-step point moving paths," in *CVPR*, 2021. 11
- [65] Z. Wu, S. Song, A. Khosla, F. Yu, L. Zhang, X. Tang, and J. Xiao, "3d shapenets: A deep representation for volumetric shapes," in *CVPR*, 2015. 3, 8
- [66] P. Xiang, X. Wen, Y.-S. Liu, Y.-P. Cao, P. Wan, W. Zheng, and Z. Han, "Snowflakenet: Point cloud completion by snowflake point deconvolution with skip-transformer," in *ICCV*, 2021. 3, 9, 10, 11, 13, 18, 20, 21, 22
- [67] H. Xie, H. Yao, S. Zhou, J. Mao, S. Zhang, and W. Sun, "Grnet: Gridding residual network for dense point cloud completion," in *ECCV*, A. Vedaldi, H. Bischof, T. Brox, and J. Frahm, Eds., 2020. 3, 9, 10, 11, 13, 18, 20, 21, 22
- [68] B. Yang, H. Wen, S. Wang, R. Clark, A. Markham, and N. Trigoni, "3d object reconstruction from a single depth view with adversarial learning," in *ICCVW*, 2017. 1
- [69] Y. Yang, C. Feng, Y. Shen, and D. Tian, "Foldingnet: Interpretable unsupervised learning on 3d point clouds," *CoRR*, vol. abs/1712.07262, 2017. 5, 9, 10, 11, 13, 18, 20, 21
- [70] X. Yu, Y. Rao, Z. Wang, Z. Liu, J. Lu, and J. Zhou, "Pointr: Diverse point cloud completion with geometry-aware transformers," in *ICCV*, 2021. 3, 9, 10, 14, 22
- [71] X. Yu, L. Tang, Y. Rao, T. Huang, J. Zhou, and J. Lu, "Pointbert: Pre-training 3d point cloud transformers with masked point modeling," in *Proceedings of the IEEE/CVF Conference on Computer Vision and Pattern Recognition*, 2022, pp. 19 313–19 322. 4
- [72] W. Yuan, T. Khot, D. Held, C. Mertz, and M. Hebert, "PCN: point completion network," in *3DV*, 2018. 1, 2, 3, 5, 8, 9, 10, 11, 13, 18, 20, 21, 22
- [73] H. Zhang, F. Li, S. Liu, L. Zhang, H. Su, J. Zhu, L. M. Ni, and H.-Y. Shum, "Dino: Detr with improved denoising anchor boxes for end-to-end object detection," 2022. 4
- [74] J. Zhang, H. Zhao, A. Yao, Y. Chen, L. Zhang, and H. Liao, "Efficient semantic scene completion network with spatial group convolution," in *ECCV*, 2018. 3
- [75] P. Zhang, W. Liu, Y. Lei, H. Lu, and X. Yang, "Cascaded context pyramid for full-resolution 3d semantic scene completion," in *ICCV*, 2019. 3, 14
- [76] W. Zhang, Q. Yan, and C. Xiao, "Detail preserved point cloud completion via separated feature aggregation," in *ECCV*, 2020. 11
- [77] H. Zhao, L. Jiang, J. Jia, P. H. Torr, and V. Koltun, "Point transformer," in *ICCV*, 2021. 4
- [78] H. Zhou, Y. Cao, W. Chu, J. Zhu, T. Lu, Y. Tai, and C. Wang, "Seedformer: Patch seeds based point cloud completion with upsample transformer," in *ECCV*. Springer, 2022. 3, 9, 11, 13

APPENDIX A

TECHNICAL DETAILS ON TRANSFORMERS

Encoder-Decoder Architecture. The overall architecture of the Transformer encoder-decoder networks is illustrated in Fig. 12. The point proxies are passed through the Transformer encoder with N multi-head self-attention layers and feed-forward network layers. Then, the decoder receives the generated query embeddings and encoder memory, and produces the final set of predicted point proxies that represents the missing part of the point cloud through N multi-head self-attention layers, decoder-encoder attention layers and feed-forward network layers. We set N to 6 in all our experiments following the common practice [57].

Multi-head Attention. Multi-head attention mechanism allows the network to jointly attend to information from different representation subspaces at different positions [57]. Specifically, given the input values V , keys K and queries Q , the multi-head attention is computed by:

$$\text{MultiHead}(Q, K, V) = W^O \text{Concat}(\text{head}_1, \dots, \text{head}_h),$$

where W^O the weights of the output linear layer and each head feature can be obtained by:

$$\text{head}_i = \text{softmax}\left(\frac{QW_i^Q(KW_i^K)^T}{\sqrt{d_k}}\right)VW_i^V$$

where W_i^Q , W_i^K , and W_i^V are the linear layers that project the inputs to different subspaces and d_k is the dimension of the input features.

Feed-forward network (FFN). Following [57], we use two linear layers with ReLU activations and dropout as the feed-forward network.

APPENDIX B

IMPLEMENTATION DETAILS

Point Proxies: In our experiments, we convert the point cloud into a set of point proxies and employ a lightweight DGCNN [63] model to extract the point proxy features. To reduce the computational cost, we hierarchically downsample the original input point cloud to N center points and use several DGCNN layers to capture local geometric relationships. The detailed network architecture is: $\text{Linear}(C_{in} = 3, C_{out} = 8) \rightarrow \text{DGCNN}(C_{in} = 8, C_{out} = 32, K = 8, N_{out} = 2048) \rightarrow \text{DGCNN}(C_{in} = 32, C_{out} = 64, K = 8, N_{out} = 512) \rightarrow \text{DGCNN}(C_{in} = 64, C_{out} = 64, K = 8, N_{out} = 512) \rightarrow \text{DGCNN}(C_{in} = 64, C_{out} = 128, K = 8, N_{out} = N)$, where C_{in} and C_{out} are the numbers of channels of input and output features, N_{out} is the number of points after FPS. We

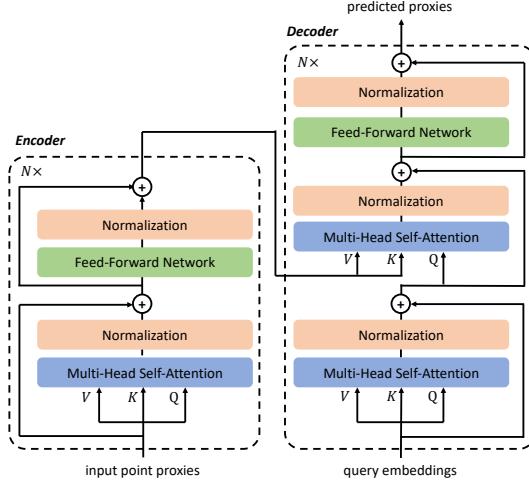


Fig. 12. The overall architecture of the Transformer encoder-decoder networks.

set N to 256 for experiments point cloud completion and semantic scene completion, respectively.

Object Point Cloud Completion: We utilize AdamW optimizer [32] to train the network with initial learning rate as 0.0001 and weight decay as 0.0005. In all of our experiments, we set the depth of the encoder and decoder in our Transformer to 6 and 8 and set k of kNN operation to 16 and 8 for the DGCNN feature extractor and the geometry-aware block respectively. We use 6 head attention for all Transformer blocks and set their hidden dimensions to 384. On the PCN dataset, the network takes 2048 points as inputs and is required to complete the other 14336 points. We set the batch size to 48 and train the model for 300 epochs with the continuous learning rate decay of 0.9 for every 20 epochs. We set N to 256 and M to 512 (including 256 queries from input point proxies). We add 64 *Denoise Queries* during training phase. On ShapeNet-55/34 and Projected-ShapeNet-55/34, the model takes 2048 points as inputs and is required to complete the other 6144 points. We set the batch size to 64 for ShapeNet-55/34 and Projected-ShapeNet-55/34. We train the model for 300 epochs with the continuous learning rate decay of 0.76 for every 20 epochs. The number of *Dynamic Queries* M is set to 256 and the number of *Denoise Queries* is set to 64. During the inference, we skip the denoise process and only focus on completing point cloud from the generated queries.

Semantic Scene Completion: We follow the previous work to utilize SGD optimizer [44] during the training phase. The learning rate and the weight decay are set to 0.1 and 0.0005, respectively. We adjust the learning rate from 0.1 to 0.00001 with a cosine schedule and set the batch size to 4. In all of our experiments, we set the depth of the encoder and decoder in our Transformer to 3 and 3, and k of kNN operation to 16 and 8 for the DGCNN feature extractor and the geometry-aware block respectively. We use 6 head attention for all Transformer blocks and set their hidden dimensions to 384. The number of *Voxel Queries* for the Transformer decoder is set to 75.

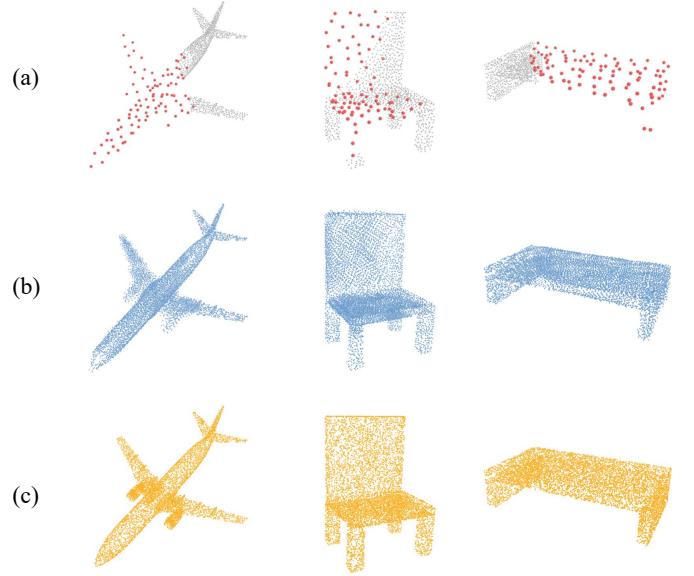


Fig. 13. Visualization of predicted points proxies. In Line (a), we show the input partial point clouds and the predicted centers. Based on predicted point proxies, we can easily predict the accurate point centers and then complete the point clouds, as shown in Line (b). We show the ground-truth point cloud in Line (c) for comparisons.

APPENDIX C QUALITATIVE RESULTS

We provide more qualitative results for PoinTr on point cloud completion and semantic scene completion in this part.

Predicted Centers We visualize the local center prediction results on ShapeNet-55. We adopt a coarse-to-fine strategy to recover the point cloud. Our method starts with the prediction of local centers, then we can obtain the final results by adding the points around the centers. As shown in Fig. 13, Line (a) shows the input partial point cloud and the predicted point centers. Line (b) is the predicted point cloud. We see the predicted point proxies can successfully represent the overall structure of the point cloud and the details then are added in the final predictions.

Point Cloud Completion: In Fig. 14, we show more completion results on ShapeNet-55. We see that our PoinTr has a stronger ability to recover details and is more robust to various incomplete patterns.

Semantic Scene Completion: In Fig. 15, we show more completion results on NYUCAD [14]. We see our Geometry-Enhanced block can help to keep more geometry information and guide the final prediction.

APPENDIX D MORE EXPERIMENTAL RESULTS

Ablation on the number of point proxies and the number of queries: We perform the ablation studies about the number of input point proxies N and the number of generated queries q on widely-used PCN benchmark as below. We can see that, increasing the number of queries q or the number

CD- ℓ_1	N=128	N=256	N=512
$q = 128$	6.82	6.78	6.75
$q = 256$	6.68	6.65	6.63
$q = 512$	6.57	6.51	6.53

of input point proxies N can both bring improvement. However, we can not increase the number infinitely due to the quadratic complexity of attention mechanism, which will bring an unbearable computational cost. Moreover, we observe a saturation trend when $N > 256$, while the situation is different for q , which is because there are only 2048 points in the input point cloud.

Ablation on formulation of point proxies: As for the presentation of point proxies, we have conducted ablation studies to investigate the gathering operation and the feature extractor. We build the point proxies as $F_i = F'_i + \varphi(p_i)$,

	CD- ℓ_1
AdaPoinTr	6.51
- Abandoning Positional Embedding	6.89 (-0.48)
- Replace mini-DGCNN with PointNet	7.12 (-0.23)

where F'_i is the local structure feature around the point p_i and $\varphi(p_i)$ is the positional embedding. We first abandon the positional embedding and build the point proxies as $F_i = F'_i$, the performance drops 0.48. Then, we replace the feature extractor from mini-DGCNN to PointNet, and the performance drops 0.23.

APPENDIX E ADDITIONAL DESCRIPTION TO KITTI METRIC

We adopt the same definition for those metric on KITTI benchmark as PCN [72]

MMD: Minimal Matching Distance (MMD), which is the Chamfer Distance (CD) between the output and the car point cloud from PCN [72] that is closest to the output point cloud in terms of CD. This measures how much the output resembles a typical car;

Fidelity: Fidelity is the average distance from each point in the input to its nearest neighbour in the output. This measures how well the input is preserved;

APPENDIX F DETAILED EXPERIMENTAL RESULTS

Detailed results on ShapeNet-34: In Table 12, we report the detailed results of FoldingNet [69], PCN [72], TopNet [54], PFNet [24], GRNet [67], SnowflakeNet [66] and the proposed method for the novel objects from 21 categories in ShapeNet-34. Each row in the table stands for a category of objects. We test each method under the three settings: simple, moderate and hard and use CD- ℓ_2 as the evaluation metric.

Detailed results on Projected-ShapeNet-34: In Table 13, we report the detailed results for the novel objects from 21 categories in Projected-ShapeNet-34. Each row in the table stands for a category of objects. We report CD- ℓ_1 and F-score@1% for each method.

Detailed results on ShapeNet-55: In Table 14, we report the detailed results of each method on ShapeNet-55. Each row in the table stands for a category of objects. We test each method under three settings: simple, moderate and hard and use CD- ℓ_2 as the evaluation metric.

Detailed results on Projected-ShapeNet-55: In Table 15, we report the detailed results of each method on Projected-ShapeNet-55. Each row in the table stands for a category of objects. We report CD- ℓ_1 and F-score@1% for each method.

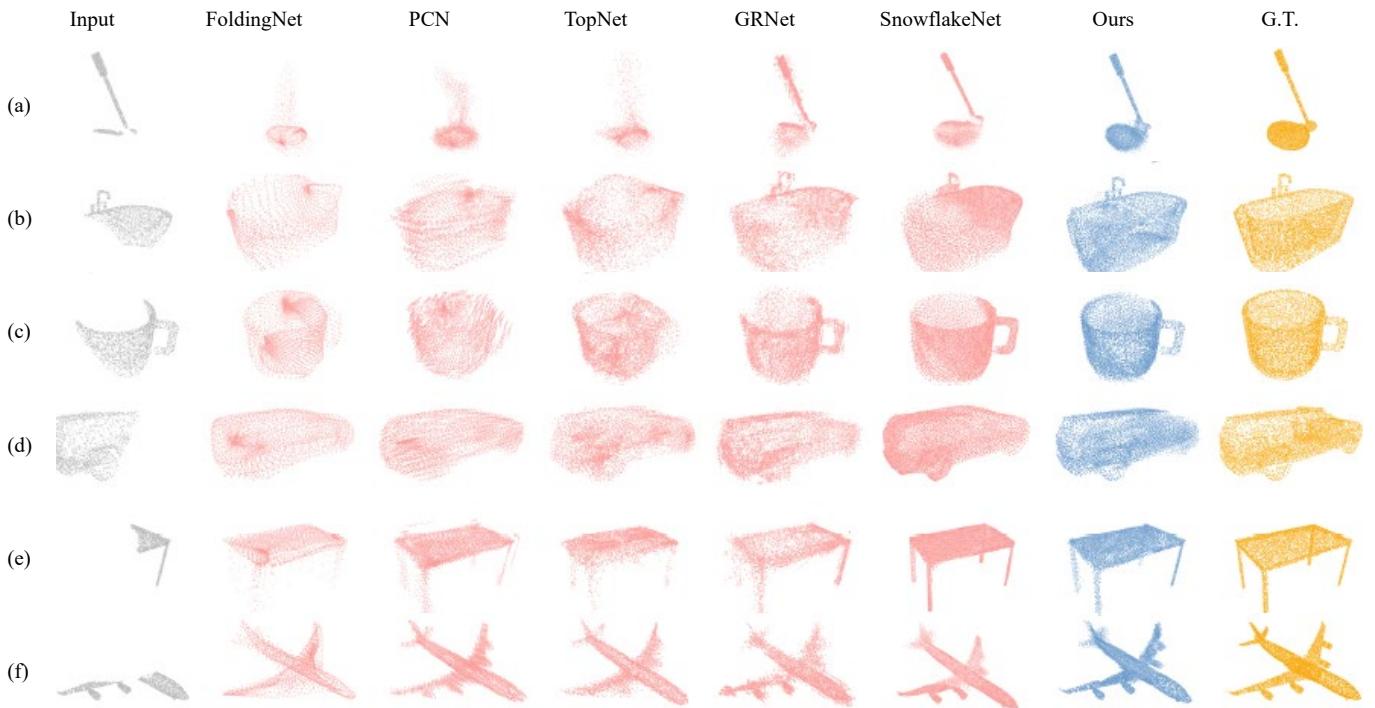


Fig. 14. More qualitative results on ShapeNet-55.

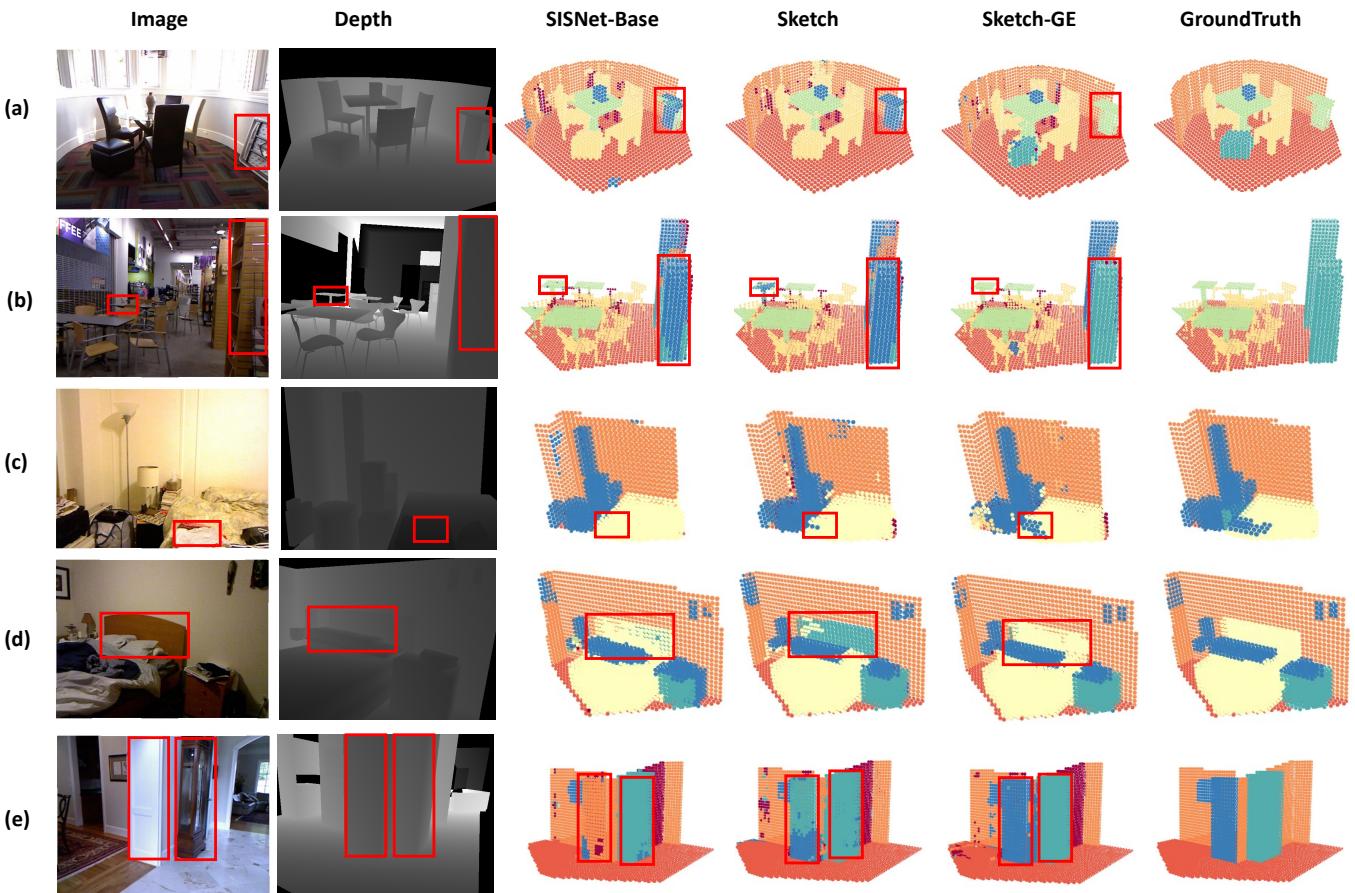


Fig. 15. More qualitative results on NYUCAD.

TABLE 12

Detailed results under $CD-\ell_2$ (multiplied by 1000) for the novel objects on ShapeNet-34. *S.*, *M.* and *H.* stand for the simple, moderate and hard settings.

	FoldingNet [69]			PCN [72]			TopNet [54]			PFNet [24]			GRNet [67]			SnowflakeNet [66]			AdaPoinTr		
	S.	M.	H.	S.	M.	H.	S.	M.	H.	S.	M.	H.	S.	M.	H.	S.	M.	H.	S.	M.	H.
bag	2.15	2.27	3.99	2.48	2.46	3.94	2.08	1.95	4.36	3.88	4.42	9.67	1.47	1.88	3.45	0.67	1.08	1.82	0.54	0.75	1.23
basket	2.37	2.2	4.87	2.79	2.51	4.78	2.46	2.11	5.18	4.47	4.55	14.46	1.78	1.94	4.18	0.78	1.16	2.48	0.67	0.82	1.57
birdhouse	3.27	3.15	5.62	3.53	3.47	5.31	3.17	2.97	5.89	3.9	4.65	9.88	1.89	2.34	5.16	0.95	1.46	2.78	0.75	1.06	1.96
bowl	2.61	2.3	4.55	2.66	2.35	3.97	2.46	2.16	4.84	4.35	5.0	14.59	1.77	1.97	3.9	0.77	1.15	2.03	0.66	0.77	1.13
camera	4.4	4.78	7.85	4.84	5.3	8.03	4.24	4.43	8.11	6.78	8.04	13.91	2.31	3.38	7.2	1.27	2.30	4.31	0.85	1.56	3.33
can	1.95	1.73	5.86	1.95	1.89	5.21	2.02	1.7	5.82	2.95	3.47	23.02	1.53	1.8	3.08	0.62	0.94	1.73	0.64	0.88	1.48
cap	6.07	5.98	11.49	7.21	7.14	10.94	4.68	4.23	9.17	14.11	14.86	28.23	3.29	4.87	13.02	1.29	3.10	7.37	0.47	0.98	3.70
keyboard	0.98	0.96	1.35	1.07	1.0	1.23	0.79	0.77	1.55	1.13	1.16	2.58	0.73	0.77	1.11	0.38	0.46	0.63	0.32	0.36	0.45
dishwasher	2.09	1.8	4.55	2.45	2.09	3.53	2.51	1.77	4.72	3.44	3.78	9.31	1.79	1.7	3.27	0.67	0.89	1.66	0.73	0.83	1.28
earphone	6.86	6.96	12.77	7.88	6.59	16.53	5.33	4.83	11.67	20.31	23.21	39.49	4.29	4.16	10.3	2.29	4.01	9.61	1.17	2.24	7.50
helmet	4.86	5.04	8.86	6.15	6.41	9.16	4.89	4.86	8.73	8.78	10.07	21.2	3.06	4.38	10.27	1.74	2.96	5.87	1.05	2.12	4.74
mailbox	2.2	2.29	4.49	2.74	2.68	4.31	2.35	2.2	4.91	5.2	5.33	10.94	1.52	1.9	4.33	0.83	1.36	2.43	0.43	0.78	1.93
microphone	2.92	3.27	8.54	4.36	4.65	8.46	3.03	3.2	7.15	6.39	7.99	19.41	2.29	3.23	8.41	1.63	2.40	5.18	0.69	1.54	3.95
microwaves	2.29	2.12	5.17	2.59	2.35	4.47	2.67	2.12	5.41	3.89	4.08	9.01	1.74	1.81	3.82	0.72	0.96	1.78	0.74	0.85	1.51
pillow	2.07	2.11	3.73	2.09	2.16	3.54	2.08	2.05	4.01	4.15	4.29	12.01	1.43	1.69	3.43	0.55	0.86	1.75	0.45	0.56	1.06
printer	3.02	3.23	5.53	3.28	3.6	5.56	2.9	2.96	6.07	5.38	5.94	10.29	1.82	2.41	5.09	0.87	1.65	2.72	0.69	1.08	2.09
remote	0.89	0.92	1.85	0.95	1.08	1.58	0.89	0.89	2.28	1.51	1.75	6.0	0.82	1.02	1.29	0.32	0.47	0.67	0.29	0.38	0.48
rocket	1.28	1.09	2.0	1.39	1.22	2.01	1.14	0.96	2.03	1.84	1.51	4.01	0.97	0.79	1.6	0.36	0.57	1.05	0.25	0.50	0.95
skateboard	1.53	1.42	1.99	1.97	1.78	2.45	1.23	1.2	2.01	2.43	2.53	4.25	0.93	1.07	1.83	0.48	0.77	1.21	0.29	0.49	0.73
tower	2.25	2.25	4.74	2.37	2.4	4.35	2.2	2.17	5.47	3.38	4.15	13.11	1.35	1.8	3.85	0.67	1.12	2.20	0.48	0.83	1.65
washer	2.58	2.34	5.5	2.77	2.52	4.64	2.63	2.14	6.57	4.53	4.27	9.23	1.83	1.97	5.28	0.75	1.07	2.23	0.70	0.86	1.67
mean	2.79	2.77	5.49	3.22	3.13	5.43	2.65	2.46	5.52	5.37	5.95	13.55	1.84	2.23	4.95	0.88	1.46	2.92	0.61	0.96	2.11

TABLE 13

Detailed results $CD-\ell_1$ (multiplied by 1000) and F-Score@1% on Projected-ShapeNet-34.

	PCN [72]		TopNet [54]		GRNet [67]		SnowflakeNet [67]		PoinTr [66]		AdaPoinTr	
	CD	F-Score	CD	F-Score	CD	F-Score	CD	F-Score	CD	F-Score	CD	F-Score
bag	20.61	0.269	15.96	0.330	15.25	0.407	12.90	0.518	12.87	0.520	11.87	0.59
basket	19.54	0.278	14.70	0.373	15.12	0.380	12.55	0.517	11.98	0.550	11.22	0.62
birdhouse	20.75	0.259	18.06	0.246	16.21	0.373	14.55	0.450	13.83	0.459	12.78	0.54
bowl	15.66	0.406	12.74	0.470	13.25	0.478	10.92	0.622	10.45	0.663	9.68	0.74
camera	24.58	0.204	19.40	0.216	16.70	0.369	14.48	0.458	14.22	0.461	13.22	0.53
can	20.16	0.253	15.45	0.356	15.61	0.363	12.98	0.496	12.24	0.546	11.02	0.64
cap	25.86	0.231	17.90	0.314	16.01	0.467	13.54	0.601	12.75	0.599	11.30	0.71
dishwasher	23.16	0.208	17.19	0.270	17.39	0.290	15.21	0.416	14.40	0.421	13.39	0.50
earphone	24.82	0.287	19.34	0.297	15.00	0.485	15.19	0.532	14.23	0.505	12.30	0.61
helmet	23.46	0.252	19.04	0.261	16.71	0.396	14.65	0.490	14.27	0.497	13.50	0.56
keyboard	14.59	0.529	10.48	0.589	10.12	0.621	8.78	0.710	8.87	0.710	8.00	0.77
mailbox	22.34	0.292	16.93	0.316	14.30	0.469	12.53	0.580	12.00	0.583	10.51	0.70
microphone	19.05	0.375	14.34	0.387	11.39	0.577	10.10	0.679	9.34	0.672	7.96	0.79
microwaves	39.08	0.148	19.41	0.231	19.59	0.270	15.41	0.396	15.76	0.385	15.39	0.47
pillow	23.45	0.230	17.55	0.336	18.64	0.373	15.35	0.494	14.82	0.502	14.19	0.58
printer	27.89	0.194	18.86	0.244	18.00	0.331	14.74	0.442	14.94	0.437	13.72	0.52
remote	13.73	0.463	11.36	0.547	12.19	0.513	9.92	0.686	9.54	0.701	8.09	0.80
rocket	11.27	0.584	10.07	0.621	9.57	0.656	7.90	0.774	7.97	0.719	6.84	0.84
skateboard	17.27	0.419	12.59	0.517	10.60	0.666	9.58	0.740	8.98	0.737	8.34	0.80
tower	17.42	0.384	15.39	0.372	14.52	0.473	12.50	0.565	12.18	0.582	10.95	0.67
washer	25.62	0.176	18.71	0.234	19.45	0.270	15.44	0.401	15.43	0.406	14.49	0.50
mean	21.44	0.307	15.98	0.358	15.03	0.439	12.82	0.551	12.43	0.555	11.37	0.64

TABLE 14

Detailed results under $CD-\ell_2$ (multiplied by 1000) on ShapeNet-55. S., M. and H. stand for the simple, moderate and hard settings.

	FoldingNet [69]			PCN [72]			TopNet [54]			PFNet [24]			GRNet [67]			SnowflakeNet [66]			AdaPoinTr		
	S.	M.	H.	S.	M.	H.	S.	M.	H.	S.	M.	H.	S.	M.	H.	S.	M.	H.	S.	M.	H.
airplane	1.36	1.28	1.7	0.9	0.89	1.32	1.02	0.99	1.48	1.35	1.44	2.69	0.87	0.87	1.27	0.36	0.50	0.76	0.22	0.30	0.49
trash bin	2.93	2.9	5.03	2.16	2.18	5.15	2.51	2.32	5.03	4.03	3.39	9.63	1.69	2.01	3.48	0.94	1.32	2.51	0.75	0.98	1.61
bag	2.31	2.38	3.67	2.11	2.04	4.44	2.36	2.23	4.21	3.63	3.66	7.6	1.41	1.7	2.97	0.62	0.93	1.70	0.45	0.62	1.00
basket	2.98	2.77	4.8	2.21	2.1	4.55	2.62	2.43	5.71	4.74	3.88	8.47	1.65	1.84	3.15	0.81	1.05	2.13	0.68	0.78	1.28
bathtub	2.68	2.66	4.0	2.11	2.09	3.94	2.49	2.25	4.33	3.64	3.5	5.74	1.46	1.73	2.73	0.71	1.11	1.81	0.53	0.72	1.18
bed	4.24	4.08	5.65	2.86	3.07	5.54	3.13	3.1	5.71	4.44	5.36	9.14	1.64	2.03	3.7	0.92	1.35	2.53	0.63	0.86	1.64
bench	1.94	1.77	2.36	1.31	1.24	2.14	1.56	1.39	2.4	2.17	2.16	4.11	1.03	1.09	1.71	0.48	0.65	1.15	0.31	0.37	0.64
birdhouse	4.06	4.18	5.88	3.29	3.53	6.69	3.73	3.98	6.8	3.96	5.0	9.66	1.87	2.4	4.71	1.06	1.69	3.04	0.81	1.18	2.08
bookshelf	3.04	3.03	3.91	2.7	2.7	4.61	3.11	2.87	4.87	3.19	3.47	5.72	1.42	1.71	2.78	0.78	1.17	1.94	0.61	0.79	1.38
bottle	1.7	1.91	4.02	1.25	1.43	4.61	1.56	1.66	4.02	2.37	2.89	10.03	1.05	1.44	2.67	0.44	0.80	1.54	0.31	0.55	1.07
bowl	2.79	2.6	4.23	2.05	1.83	3.66	2.33	1.98	4.82	4.3	3.97	8.76	1.6	1.77	2.99	0.76	0.96	1.81	0.60	0.63	0.92
bus	1.47	1.42	2.0	1.2	1.14	2.08	1.32	1.21	2.29	2.06	1.88	3.75	1.06	1.16	1.48	0.50	0.64	0.87	0.43	0.52	0.65
cabinet	2.0	1.86	2.79	1.6	1.49	3.47	1.91	1.65	3.36	2.72	2.37	4.73	1.27	1.41	2.09	0.66	0.83	1.27	0.59	0.65	0.92
camera	5.5	6.04	8.87	4.05	4.54	8.27	4.75	4.98	9.24	6.57	8.04	13.11	2.14	3.15	6.09	1.27	2.36	4.30	0.80	1.55	3.15
can	2.84	2.68	5.71	2.02	2.28	6.48	2.67	2.4	5.5	5.65	4.05	16.29	1.58	2.11	3.81	0.63	1.23	2.23	0.61	0.94	1.68
cap	4.1	4.04	5.87	1.82	1.76	4.2	3.0	2.69	5.59	10.92	9.04	20.3	1.17	1.37	3.05	0.55	0.94	1.98	0.37	0.43	0.69
car	1.81	1.81	2.31	1.48	1.47	2.6	1.71	1.65	3.17	2.06	2.1	3.43	1.29	1.48	2.14	0.81	1.01	1.32	0.65	0.79	1.00
cellphon	1.04	1.06	1.87	0.8	0.79	1.71	1.01	0.96	1.8	1.25	1.37	3.65	0.82	0.91	1.18	0.37	0.49	0.71	0.33	0.36	0.44
chair	2.37	2.46	3.62	1.7	1.81	3.34	1.97	2.04	3.59	2.94	3.48	6.34	1.24	1.56	2.73	0.62	0.94	1.79	0.41	0.56	1.12
clock	2.56	2.41	3.46	2.1	2.01	3.98	2.48	2.16	4.03	3.15	3.27	6.03	1.46	1.66	2.67	0.74	1.00	1.65	0.53	0.68	1.18
keyboard	1.21	1.18	1.32	0.82	0.82	1.04	0.88	0.83	1.15	0.83	1.06	1.97	0.74	0.81	1.09	0.36	0.45	0.63	0.28	0.32	0.37
dishwasher	2.6	2.17	3.5	1.93	1.66	4.39	2.43	1.74	4.64	4.57	3.23	6.39	1.43	1.59	2.53	0.63	0.82	1.69	0.60	0.66	1.16
display	2.15	2.24	3.25	1.56	1.66	3.26	1.84	1.85	3.48	2.27	2.83	5.52	1.13	1.38	2.29	0.57	0.90	1.57	0.41	0.53	0.89
earphone	6.37	6.48	9.14	3.13	2.94	7.56	4.36	4.47	8.36	15.07	17.5	33.37	1.78	2.18	5.33	0.94	1.55	4.15	0.60	0.83	1.98
faucet	4.46	4.39	7.2	3.21	3.48	7.52	3.61	3.59	7.25	5.68	6.79	14.29	1.81	2.32	4.91	1.04	1.83	3.83	0.47	0.93	2.14
filecabinet	2.59	2.48	3.76	2.02	1.97	4.14	2.41	2.12	4.12	3.72	3.57	7.13	1.46	1.71	2.89	0.78	1.06	1.83	0.65	0.78	1.29
guitar	0.65	0.6	1.25	0.42	0.38	1.23	0.57	0.47	1.42	0.74	0.89	5.41	0.44	0.48	0.76	0.18	0.27	0.47	0.11	0.17	0.28
helmet	5.39	5.37	7.96	3.76	4.18	7.53	4.36	4.55	7.73	9.55	8.41	15.44	2.33	3.18	6.03	1.37	2.40	4.68	0.76	1.24	3.03
jar	3.65	3.87	6.51	2.57	2.82	6.0	3.03	3.17	7.03	5.44	5.56	11.87	1.72	2.37	4.37	0.97	1.52	3.03	0.65	0.98	2.01
knife	1.29	0.87	1.21	0.94	0.62	1.37	0.84	0.68	1.44	2.11	1.53	3.89	0.72	0.66	0.96	0.23	0.36	0.62	0.13	0.23	0.42
lamp	3.93	4.23	6.87	3.1	3.45	7.02	3.03	3.39	8.15	6.82	7.61	14.22	1.68	2.43	5.17	0.88	1.70	3.88	0.41	0.94	2.31
laptop	1.02	1.04	1.96	0.75	0.79	1.59	0.8	0.85	1.66	1.04	1.21	2.46	0.83	0.87	1.28	0.39	0.49	0.86	0.33	0.34	0.44
loudspeaker	3.21	3.15	4.55	2.5	2.45	5.08	3.1	2.76	5.32	4.32	4.19	7.6	1.75	2.08	3.45	0.90	1.34	2.32	0.69	0.93	1.60
mailbox	2.44	2.61	4.98	1.66	1.74	5.18	2.16	2.1	5.1	3.82	4.2	10.51	1.15	1.59	3.42	0.49	0.91	2.45	0.28	0.52	1.62
microphone	4.42	5.06	7.04	3.44	3.9	8.52	2.83	3.49	6.87	6.58	7.56	16.74	2.09	2.76	5.7	1.52	2.66	5.51	0.45	1.24	2.95
microwaves	2.67	2.48	4.43	2.2	2.01	4.65	2.65	2.15	5.07	4.63	3.94	6.52	1.51	1.72	2.76	0.73	0.96	1.68	0.68	0.77	1.28
motorbike	2.63	2.55	3.52	2.03	2.01	3.13	2.29	2.25	3.54	2.17	2.48	5.09	1.38	1.52	2.26	0.96	1.20	1.70	0.64	0.89	1.33
mug	3.66	3.67	5.7	2.45	2.48	5.17	2.89	2.56	5.43	4.76	4.3	8.37	1.75	2.16	3.79	0.98	1.41	2.73	0.83	1.01	1.72
piano	3.86	4.04	6.04	2.64	2.74	4.83	2.99	2.89	5.64	4.57	5.26	9.26	1.53	1.82	3.21	0.95	1.34	2.69	0.61	0.72	1.33
pillow	2.33	2.38	3.87	1.85	1.81	3.68	2.31	2.26	4.19	4.21	3.82	7.89	1.42	1.67	3.04	0.64	0.94	1.73	0.44	0.53	0.89
pistol	1.92	1.62	2.52	1.25	1.17	2.65	1.5	1.3	2.62	2.27	2.09	7.2	1.11	1.06	1.76	0.57	0.78	1.23	0.36	0.51	0.81
flowerpot	4.53	4.68	6.46	3.32	3.39	6.04	3.61	3.45	6.28	4.83	5.51	10.68	2.02	2.48	4.19	1.32	1.80	3.10	0.89	1.16	2.01
printer	3.66	4.01	5.34	2.9	3.19	5.84	3.04	3.19	5.84	5.56	6.06	9.29	1.56	2.38	4.24	0.83	1.63	2.79	0.62	0.93	1.83
remote	1.14	1.2	1.98	0.99	0.97	2.04	1.14	1.17	2.16	1.74	2.37	4.61	0.89	1.05	1.29	0.39	0.56	0.77	0.29	0.40	0.48
rifle	1.27	1.02	1.37	0.98	0.8	1.31	0.98	0.86	1.46	1.72	1.45	3.02	0.83	0.77	1.16	0.36	0.50	0.77	0.25	0.36	0.58
rocket	1.37	1.18	1.88	1.05	1.04	1.87	1.04	1.0	1.93	1.65	1.61	3.82	0.78	0.92	1.44	0.29	0.58	0.96	0.18	0.34	0.77
skateboard	1.58	1.58	2.07	1.04	0.94	1.68	1.08	1.05	1.84	1.43	1.6	3.09	0.82	0.87	1.24	0.38	0.53	0.74	0.22	0.29	0.40
sofa	2.22	2.09	3.14	1.65	1.61	2.92	1.93	1.76	3.39	2.65	2.53	4.84	1.35	1.45	2.32	0.65	0.85	1.39	0.52	0.58	0.84
stove	2.69	2.63	3.99	2.07	2.02	4.72	2.44	2.16	4.84	4.03	3.71	7.15	1.46	1.72	3.22	0.75	1.07	1.91	0.62	0.80	1.28
table	2.23	2.15	3.21	1.56	1.5	3.36	1.78	1.65	3.21	3.03	3.11										

TABLE 15
Detailed results $CD-\ell_1$ (multiplied by 1000) and F-Score@1% on Projected-ShapeNet-55.

	PCN [72]		TopNet [54]		GRNet [67]		SnowflakeNet [66]		PoinTr [70]		AdaPoinTr	
	CD	F-Score	CD	F-Score	CD	F-Score	CD	F-Score	CD	F-Score	CD	F-Score
airplane	9.07	0.703	9.85	0.621	8.30	0.726	6.35	0.862	6.02	0.830	5.18	0.93
bag	18.64	0.309	18.69	0.235	14.67	0.428	12.86	0.535	12.15	0.555	10.93	0.63
basket	18.03	0.302	18.19	0.252	14.60	0.383	13.19	0.478	12.00	0.538	10.18	0.64
bathtub	20.96	0.320	17.19	0.292	13.90	0.426	12.16	0.543	11.49	0.576	10.05	0.68
bed	20.54	0.266	20.36	0.214	15.32	0.376	13.94	0.462	13.48	0.465	12.24	0.55
bench	13.01	0.531	13.01	0.474	10.55	0.605	9.04	0.714	8.49	0.739	7.34	0.83
birdhouse	20.38	0.253	22.00	0.165	16.52	0.347	15.24	0.428	14.60	0.432	13.27	0.51
bookshelf	18.32	0.313	18.66	0.261	14.10	0.416	12.80	0.504	12.59	0.508	11.42	0.60
bottle	15.21	0.430	16.33	0.301	13.00	0.488	11.12	0.601	9.83	0.680	8.67	0.77
bowl	13.96	0.448	15.04	0.356	12.05	0.514	10.74	0.626	9.64	0.692	8.76	0.76
bus	12.73	0.482	13.11	0.422	11.83	0.504	9.99	0.638	9.65	0.661	8.76	0.74
cabinet	17.86	0.302	17.13	0.259	14.57	0.366	12.86	0.469	12.48	0.490	11.31	0.57
camera	22.83	0.228	21.96	0.164	15.47	0.391	13.68	0.474	13.00	0.493	11.57	0.57
can	17.12	0.315	16.49	0.285	14.01	0.392	11.98	0.517	11.09	0.583	9.57	0.70
cap	18.20	0.326	19.74	0.191	12.50	0.520	12.07	0.631	10.08	0.706	8.51	0.84
car	12.85	0.443	13.61	0.377	12.13	0.465	11.20	0.526	10.58	0.568	9.77	0.63
cellphone	12.36	0.502	12.56	0.438	11.30	0.515	9.18	0.695	8.83	0.706	7.61	0.81
chair	15.33	0.399	16.29	0.316	12.57	0.491	11.07	0.591	10.43	0.616	9.12	0.72
clock	15.85	0.414	15.58	0.345	13.01	0.476	11.23	0.606	10.63	0.621	9.60	0.70
dishwasher	19.61	0.252	19.10	0.212	15.19	0.316	13.29	0.457	13.18	0.470	11.94	0.53
display	16.40	0.378	16.10	0.318	12.74	0.467	11.54	0.571	10.78	0.601	9.37	0.70
earphone	19.78	0.360	19.75	0.257	13.42	0.506	14.37	0.534	11.97	0.575	10.23	0.67
faucet	16.98	0.396	17.74	0.243	10.81	0.592	9.81	0.669	9.10	0.662	7.23	0.83
file cabinet	18.28	0.287	17.88	0.249	15.08	0.353	13.07	0.461	12.64	0.492	11.55	0.56
flowerpot	17.59	0.309	17.75	0.245	13.73	0.426	12.84	0.489	11.80	0.535	10.97	0.61
guitar	7.96	0.758	9.05	0.668	7.56	0.775	6.13	0.881	5.74	0.901	4.91	0.95
helmet	20.56	0.277	20.11	0.215	14.63	0.424	13.66	0.496	12.39	0.547	11.61	0.62
jar	17.82	0.339	18.36	0.251	14.08	0.442	12.83	0.523	11.56	0.604	10.49	0.68
keyboard	13.69	0.557	11.05	0.586	9.71	0.664	8.12	0.766	7.61	0.797	6.79	0.85
knife	8.04	0.759	8.75	0.704	7.52	0.778	5.77	0.890	5.66	0.873	4.98	0.93
lamp	17.50	0.394	17.52	0.314	12.73	0.555	10.76	0.663	10.21	0.679	9.16	0.79
laptop	16.05	0.429	13.52	0.395	10.83	0.533	9.79	0.648	8.93	0.689	8.11	0.78
loudspeaker	21.21	0.262	19.72	0.211	16.12	0.354	13.95	0.455	13.61	0.475	12.60	0.55
mailbox	17.97	0.350	16.82	0.283	12.52	0.507	10.61	0.637	9.84	0.654	8.64	0.78
microphone	17.75	0.413	17.05	0.281	10.45	0.630	10.37	0.697	8.64	0.691	7.60	0.79
microwaves	39.54	0.153	26.64	0.170	18.35	0.281	16.92	0.398	17.06	0.406	17.43	0.49
motorbike	12.66	0.505	14.33	0.364	10.68	0.561	10.13	0.605	9.83	0.622	8.99	0.69
mug	18.12	0.287	20.56	0.179	14.60	0.372	14.16	0.428	13.10	0.448	11.39	0.54
piano	19.15	0.291	19.91	0.230	14.46	0.414	13.47	0.498	12.42	0.520	10.83	0.61
pillow	20.04	0.291	19.57	0.238	15.15	0.421	14.02	0.496	12.57	0.563	11.82	0.66
pistol	10.93	0.587	12.43	0.442	9.69	0.635	8.40	0.729	8.25	0.733	7.27	0.81
printer	26.86	0.217	20.74	0.192	16.17	0.353	13.75	0.475	13.53	0.466	12.46	0.56
remote	14.62	0.447	13.52	0.436	12.18	0.511	10.07	0.702	9.55	0.711	8.81	0.77
rifle	8.79	0.722	9.11	0.673	7.30	0.792	6.10	0.873	5.98	0.863	5.21	0.93
rocket	10.98	0.585	10.45	0.610	8.58	0.707	7.49	0.793	6.86	0.782	5.58	0.90
skateboard	9.65	0.662	11.01	0.568	8.82	0.704	7.39	0.801	7.24	0.775	6.16	0.89
sofa	17.12	0.297	16.93	0.257	14.36	0.373	12.59	0.477	12.11	0.502	10.89	0.59
stove	22.04	0.270	20.33	0.231	16.64	0.355	13.77	0.463	13.65	0.474	12.62	0.55
table	14.79	0.469	14.40	0.433	12.01	0.537	10.49	0.645	9.97	0.663	8.81	0.75
telephone	12.02	0.517	12.18	0.457	10.95	0.532	8.82	0.720	8.62	0.720	7.51	0.81
tower	15.39	0.432	16.94	0.306	13.26	0.493	11.96	0.577	11.06	0.600	10.12	0.68
train	11.93	0.541	12.56	0.462	10.64	0.579	9.47	0.665	9.22	0.673	8.20	0.76
trash bin	15.39	0.343	16.38	0.270	14.01	0.380	12.62	0.466	11.88	0.518	10.83	0.61
washer	22.39	0.208	21.90	0.162	18.50	0.280	15.16	0.401	15.21	0.410	14.19	0.51
watercraft	12.61	0.527	13.16	0.439	10.58	0.593	9.17	0.692	8.85	0.692	7.90	0.78
mean	16.64	0.403	16.35	0.337	12.81	0.491	11.34	0.594	10.68	0.615	9.58	0.70