SD-Net: Spatially-Disentangled Point Cloud Completion Network

Junxian Chen*
Hunan University
Changsha, China
cjxdsg@hnu.edu.cn

Ying Liu*
Hunan University
Changsha, China
liu_ying@hnu.edu.cn

Yiqi Liang
Hunan University
Changsha, China
yiqiliang@hnu.edu.cn

Dandan Long
Hunan University
Changsha, China
ddlong@hnu.edu.cn

Xiaolin He
Hunan University
Changsha, China
hxl4396@hnu.edu.cn

Ruihui Li†
Hunan University
Changsha, China
liruihui@hnu.edu.cn

ABSTRACT

Point clouds obtained from 3D scanning are typically incomplete, noisy, and sparse. Previous completion methods aim to generate complete point clouds, while taking into account the densification of point clouds, filling small holes, and proximity-to-surface, all through a single network. After revisiting the task, we propose SDNet, which disentangles the task based on the spatial characteristics of point clouds and formulates two sub-networks, a Dense Refiner and a Missing Generator. Given a partial input, the Dense Refiner produces a dense and clean point cloud, as a more reliable partial surface, which assists the Missing Generator to better infer the remaining point cloud structure. To promote the alignment and interaction across these two modules, we propose a Cross Fusion Unit with designed Non-Symmetrical Cross Transformers to capture geometric relationships between partial and missing regions, contributing to a complete, dense and well-aligned output. Extensive quantitative and qualitative results demonstrate that our method outperforms the state-of-the-art methods.

CCS CONCEPTS

• **Computing methodologies** → **Shape inference**; **Neural networks**; **Point-based models**.

KEYWORDS

Point Cloud Completion; Deep Neural Networks; Disentangle

ACM Reference Format:
Junxian Chen, Ying Liu, Yiqi Liang, Dandan Long, Xiaolin He, and Ruihui Li. 2023. SD-Net: Spatially-Disentangled Point Cloud Completion Network. In *Proceedings of the 31st ACM International Conference on Multimedia (MM '23), October 29-November 3, 2023, Ottawa, ON, Canada.* ACM, New York, NY, USA, 11 pages. https://doi.org/10.1145/3581783.3611716

*Both authors contributed equally to this research.
†Corresponding author.

MM '23, October 29-November 3, 2023, Ottawa, ON, Canada

ACM ISBN 979-8-4007-0108-5/23/10…$15.00
https://doi.org/10.1145/3581783.3611716

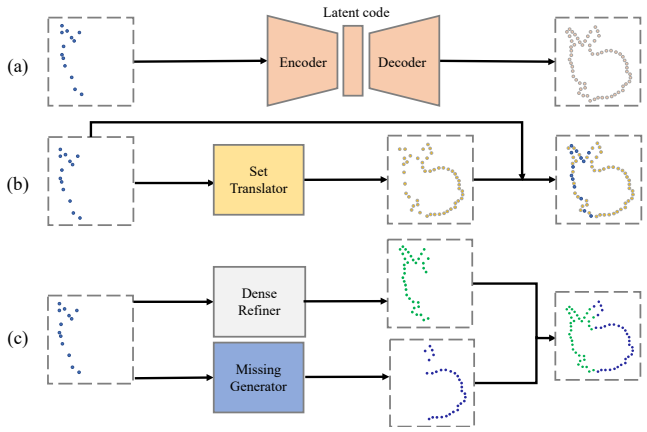

Figure 1: The overview of previous methods and ours. (a) is usually based on encoder-decoders, and it restores the complete point cloud based on the global features obtained by encoding the partial point cloud. (b) predicts only the missing point cloud, and concatenate the predicted point cloud with the partial point cloud to get the final complete point cloud. (c) is the method we propose in this paper. The Dense Refiner refines and upsamples the partial point cloud into a detailed and dense point cloud (green rabbit) while the Missing Generator predicts missing regions (violet rabbit). The final completion results are obtained by concatenating.

1 INTRODUCTION

In recent years, the development of 3D scanning technology has made it easier to acquire and process point cloud data, which has led to numerous downstream applications [4, 5, 14], e.g., autonomous driving, virtual reality, and robotics, etc. Nevertheless, it is common for the scanned point cloud to be sparse and incomplete due to noise, occlusion, or sensor limitations. Therefore, it is more desirable to complete the sparse raw data into complete and dense point clouds before further 3D analysis.

Point cloud completion aims to complete 3D models from partial inputs. However, point clouds acquired from scanning techniques are not only incomplete, but often sparse and noisy. Therefore, it is very critical that the generated complete point cloud satisfies the criteria of being dense, accurately positioned on the underlying surface, and free of holes and gaps. As a local-to-global inference task, these goals are extremely challenging since partial inputs provide limited and noisy information.

Over the years, with the help of data-driven machine learning and deep neural networks, several deep-learning-based methods

for point cloud completion have been proposed [1, 6, 9, 12, 19, 30, 36, 43], demonstrating superior performance over traditional methods. Inspired by the great success of PCN [37], most of the works [22, 23, 25, 30, 31, 39] adopt the encoder-decoder architecture to generate complete point clouds as shown in Fig. 1(a). Although this design is effective, it has inherent defects in that it relies on highly-concentrated global features to reconstruct the complete point cloud, inevitably losing the original geometric details of partial inputs. Another stream of methods [9, 36] treat point cloud completion as a set-to-set transformation problem and only predict the point cloud for the missing part, as shown in Fig. 1(b). Overall, existing methods with the paradigms in Fig. 1(a) and (b) take the completion process as a whole. In other words, the designed completion network is expected not only to infer point in the missing region, but also to generate more points near the partial inputs, in order to achieve a complete, dense and uniformly-distributed point cloud. However, it is very hard for a network to meet all goals at the same time. Thus, the complete results of existing methods tends to be noisy, as presented in Figs. 6 and 7. Essentially, generating more points around the partial input is a local-wise task, while inferring the missing points of a full shape is a global-wise task. Thus different network designs should be considered for them.

After revisiting the point cloud completion task, we propose a novel disentangled point cloud completion framework, called SDNet, which mainly consists of two sub-networks, a Dense Refiner and a Missing Generator. An illustration of our method can be seen in Fig. 1(c). Given a partial input, the Dense Refiner aims to produce a dense and clean point cloud, providing a more reliable partial surface representation. Benefited from that, the Missing Generator could mainly focus on inferring point clouds in missing regions. The complete shape is obtained by combining the outputs from these two branches. To better promote the geometric interaction across two branches, we further introduce a Cross Fusion Unit with designed Non-Symmetrical Cross Transformer via capturing geometric relationships between points in partial and missing regions. Such that, Missing Generator and Dense Refiner can complement with each other, contributing a complete, dense and well-aligned result.

In comparison to existing methods of completing point clouds, our spatially disentangled framework is able to generate dense and reliable partial point clouds and utilize these detailed point clouds to assist point cloud completion. Additionally, an off-line data preprocessing method is developed, which can increase the focus of the two sub-networks on the tasks they are responsible for. As a result, we are able to obtain dense complete point clouds with high detail recovery and smooth surfaces. Experimental results on ShapeNet-55/34 [36] and MVP [15] demonstrate that our method achieves the state-of-the-art performance.

2 RELATED WORK

Point Cloud Completion Point Cloud completion plays a key role in improving the computer's ability to understand point clouds. Traditional point cloud completion attempts usually rely on predefined smooth surfaces to reconstruct the full geometry [2, 8], or construct local priors from a large number of structured basic 3D shapes [20, 21]. However, this series of methods rely on prior structured data distribution, which is difficult to meet large-scale and complex application scenarios. Benefited from PointNet [16] and PointNet++ [17], many completion methods based on deep neural networks have been proposed in recent years. PCN [37] is one of the earliest learning-based methods for point cloud completion that directly operates on the original point cloud without introducing other structural assumptions. They adopt a coarse-to-fine completion strategy and use the encoder-decoder architecture, which has a profound impact on subsequent work [3, 13, 22, 24, 40]. GRNet [30] also adopted this architecture and learned the relationship between points by using a voxelized representation to aggregate features. ASFM-Net [28] is trained with an asymmetrical Siamese network architecture to learn more prior information in prediction by introducing global information in the complete point cloud reconstruction process. PMPNet [25] and PMPNet++ [26] regard point cloud completion as the movement of points, and the final complete point cloud is obtained by moving the points to the correct position. Thus, they do not need to predict the precise coordinates of the point cloud in 3D space, but only an offset, which greatly reduces the difficulty of completing point clouds. This strategy was adopted by Snowflakenet [29] and SeedFormer [43] as well, which completed the point cloud using three offset predictions. However, [29] utilizes a new Skip-Transformer to efficiently aggregate information from the previous step, and the addition of point clouds relies on deconvolution operations. [43] further introduces the uptransformer, allowing for more local information to be included in the upsampling process.

PFNet [9] is the first work to only predict missing parts from partial point clouds. They utilize a feature-points-based multi-scale generative network combined with adversarial loss to generate more realistic missing regions. PoinTr [36] also adopts this approach and regard point cloud completion as a set-to-set transformation problem. They proposed a transformer-based encoder-decoder architecture, which only recovers the point clouds of missing regions. The closest to our work is NSFA [41], they also divide the point cloud completion task into two sub-networks, but they only decouple the two parts of the point cloud at the feature level, and our work further decouples the two parts in spatial space.

Point Cloud Upsampling Upsampling point clouds can be considered a form of point cloud completion, but upsampling is often used to fill small gaps, while completion is used to fill larger holes. Recently, some deep learning-based methods have been proposed [10, 18, 33–35], such as PU-Net [35], which expands point sets by extracting multi-scale features. PU-Gan [10] is the first work to introduce the adversarial loss in point cloud upsampling. Moreover, the Dis-PU [11] decouples the upsampling process, and regards upsampling as a coarse-to-fine process, which reduces the task difficulty of a single network.

After revisiting these methods mentioned above, we found that existing works almost rely on a single network to meet their specify task. They ignored the relationship between completion and upsampling. We often see such a phenomenon that it is often difficult for the completion network to preserve the geometric details of the object, and the upsampling network is often unable to restore large-area vacancies. The reason for this is that region refinement requires local geometric details, while missing inference requires global shape information. This means the features needed to achieve

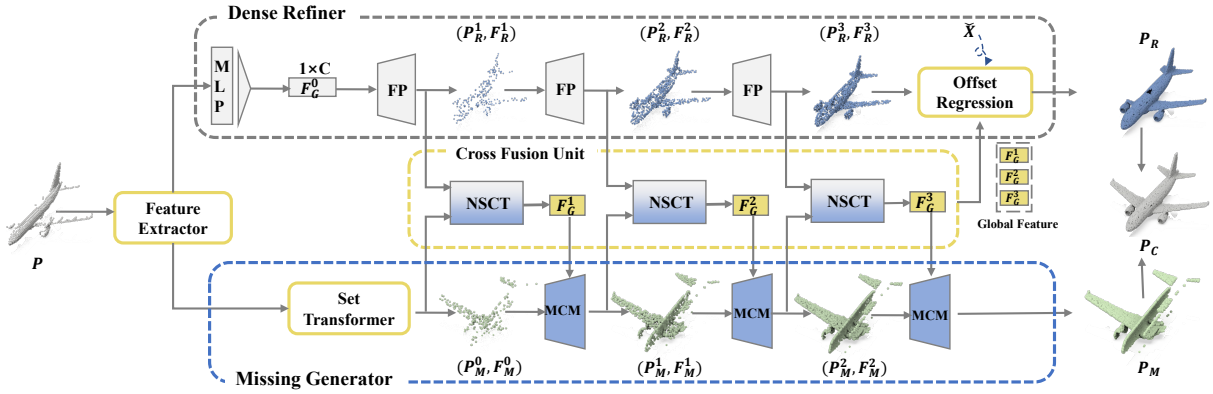

Figure 2: The overall architecture of our SDNet. Given an incomplete point cloud P, our network completes it via two branches. The top branch is composed of three Feature Propagation [17] (FP) layers and an Offset Regression unit. This branch aims to produce dense point clouds that fill small holes and restore broken links. The bottom branch is used to reconstruct missing regions, including a Set Transformer [36] and three Multi-scale Completion Modules (MCM). In order to make the two branches work together, we also employed a Cross Fusion Unit with three Non-Symmetrical Cross Transformers (NSCTs) to aggregate geometry features.

these two goals are in different distributions, and it is difficult for a single network to meet both objectives. Inspired by [11], We propose a novel approach to disentangle the task and achieve superior performance over prior works.

3 METHODS

3.1 Overview

Given an incomplete point cloud $P = \{p_i\}_{i=1}^{N}$ of N points which is typically sparse and noisy, the point cloud completion task aims to predict a complete point cloud $P_c = \{p_i\}_{i=1}^{N_c}$. Its goals are two-fold: (i) is to restore a completed point cloud P_c that represents the full structure of the object; (ii) is to produce a denser point cloud P_c while preserving the original geometric details of P. This completion task is very challenging, as we need to infer new knowledge from the sparse and partial input, where the original geometry does not fully presented.

Unlike existing approaches that try to meet all the completion goals via a single network, we propose to divide the completion task into two sub-goals. One is to restore the details and holes around the partial point cloud, which is similar to point cloud upsampling. The other is to predict large missing parts of the point cloud simultaneously. In order to better illustrate our method, we first discuss our key insights:

1) We propose an end-to-end disentangled point cloud completion framework with two sub-networks. The Dense Refiner refines and upsamples partial point clouds based on their structural information, resulting in dense point clouds with well-restored local geometric details. The Missing Generator infers missing parts from shape information predicted from partial point clouds. In this way, each sub-network can make better use of the local detail or global shape information.
2) Inferring the missing shape directly from partial point clouds often results in rough point clouds. Therefore, we designed a Cross Fusion Unit in order to assist the Missing Generator in capturing the multi-scale geometric details of partial regions. With NSCTs, features across distances can be captured, allowing the generated points to take advantage of the geometric structure information at hand.
3) We preprocessed the dataset and obtained the ground truth required for the Dense Refiner and Missing Generator, respectively. Thus, each sub-network can better focus on its specific sub-goal.

Fig 2 shows the overall pipeline of SDNet, which consists of four stages: Feature Extractor, Cross Fusion Unit, Dense Refiner and Missing Generator. Given a partial point cloud P, we first feed it into the Feature Extractor to extract features F_p of $N_p \times C_p$, which captures the structure feature of the partial point cloud. In order to aggregate multi-scale regional information and expand the receptive field, we adopt two layers of Set Abstraction from [17] and Point Transformer from [42]. After that, the extracted feature F_p and corresponding coordinates P_p are fed into the Dense Refiner and Missing Generator.

3.2 Dense Refiner

Inspired by the residual-learning strategy [7], the goal of Dense Refiner is to learn an offset to refine the partial point cloud P and generate a dense point cloud P_R. We first adopt the SA layer in [17] to gather features again to obtain the global features F_G^0 of the partial point cloud, and then use the Feature Propagation [17](FP) module of three consecutive layers to propagate the global feature F_G^0 to the entire partial point cloud space. Therefore, we will get three pairs of point cloud coordinates $P_R^i (i = 1, 2, 3)$ and the corresponding per point features $F_R^i (i = 1, 2, 3)$, which satisfy $P_R^1 \subset P_R^2 \subset P_R^3 \subseteq P$. In order to realize the refinement and upsampling of the partial point cloud, the usual follow step is to decode the features F_R^3 to obtain the offset. And then dense points are generated by adding offset to partial point cloud coordinates. However, there are two main problems with this approach. First of all, the distribution of points in the partial point cloud may deviate from the ground truth, necessitating the consideration of noise in point cloud completion. Furthermore, Dense Refiner only learns structural information, not

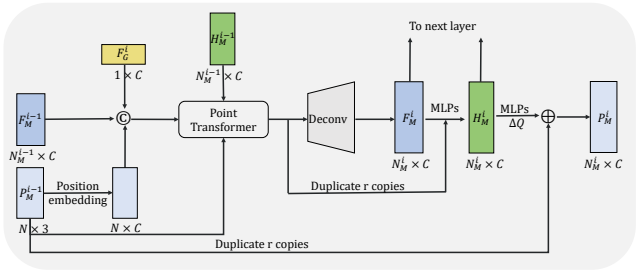

Figure 3: The detailed architecture of Multi-scale Completion Module (MCM), which takes the P_M^{i-1}, F_M^{i-1} and H_M^{i-1} from the previous layer and global feature F_G^i as input, and generates new point cloud P_M^i through upsampling and residual strategies.

a global picture of a shape as a whole. Hence, we designed a new offset regression module, by inputting the output of each level of feature propagation layer to the Cross Fusion Unit to obtain multi-scale global features $F_G^i (i = 1, 2, 3)$ to help the partial point cloud perceive the missing area and move towards it. In order to eliminate the influence of the input containing noise points, we add noise $\hat{x}$ when performing offset regression to train the denoising ability during decoding. And the introduction of noise also helps to encourage the module to explore a wider area when refining. The final offset can be expressed as:

$$\Delta p = \tanh\left(\mathrm{MLP}\left(\left[F_R^3 : P_G^1 : P_G^2 : P_G^3 : \hat{x}\right]\right)\right) \tag{1}$$

where ":" denotes the concatenation operation.

3.3 Missing Generator

To reconstruct missing regions, it is essential to make full use of prior knowledge provided by existing points. The Missing Generator is also designed around this concept. The two strategies we have adopted for exploitation of existing points are as follows. One is similar to [25], which learns the context of generated point clouds. The second approach involves the Cross Fusion Unit capturing the geometric features of the partial point cloud. We first apply the Set Transformer [36] for set transformation, which converts the partial point cloud coordinates P_p and corresponding per-point features F_p into coordinates P_M^0 and features F_M^0 of key points in the missing regions. Considering the powerful capability of the SPD decoder [29], we adopt a similar structure in the subsequent reconstruction step, called Multi-scale Completion Module (MCM). Fig 3 shows the architecture of MCM in detail. MCM consumes the coordinates P_M^{i-1}, per-point features F_M^{i-1} and offset features H_M^{i-1} from the previous layer and the global features F_G^i from Cross Fusion Unit as input. F_G^i can be seen as a complement to the details. Firstly, we encode the P_M^{i-1} to obtain the position embedding, which is concatenated with F_M^{i-1} and F_G^i. Then, we employ a Point Transformer [42] to fuse current features F_c^i and H_M^{i-1} together. F_c^i can be expressed as:

$$F_c^i = \mathrm{MLP}\left(\left[\mathrm{MLP}\left(P_M^{i-1}\right) : F_M^{i-1} : F_G^i\right]\right) \tag{2}$$

Next we apply an upsampler from [29] to upsample the fused feature from N_M^{i-1} to N_M^i, where N_M^i denotes the number of points P_M^i. Finally we regress per-point displacements as the same as [29].

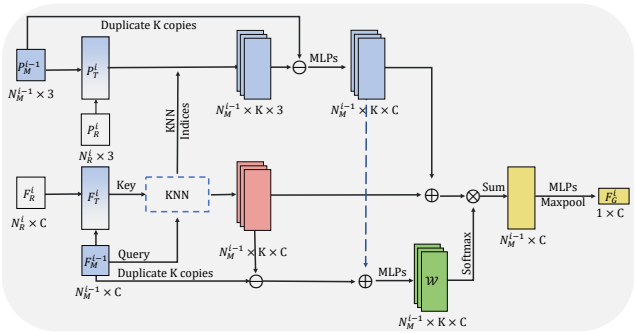

Figure 4: The detailed architecture of Non-Symmetrical Cross Transformer (NSCT). Instead of equally treating all of the features, we only consider the relation among K feature neighbors of F_M^{i-1}. And with this feature KNN strategy, NSCT can aggregate the similar features across spatial distances.

3.4 Cross Fusion Unit

It is often difficult to recover the complete point cloud only by extracting global features from the partial point cloud, as these global features are only incomplete information. Unfortunately, many works [23, 37, 41] in the past have adopted this scheme to obtain dense point clouds. Considering this issue, we propose a Cross Fusion Unit to extract global features during the point cloud step-by-step completion process. Cross Fusion Units take the multi-scale point cloud and features as inputs, and output the global features of the corresponding scales to guide the two sub-networks. In addition, during experiments we noticed that feature aggregation with emphasis is more effective than fully impartial global feature extraction. For specific analysis, please refer to Sec 4.3. Therefore, we designed three similar units to aggregate features, called Non-Symmetrical Cross Transformer (NSCT). Fig 4 shows the details of NSCT. Specifically, we first concatenate two sets of point cloud coordinates and features to obtain the total coordinates P_T^i and features F_T^i. Then we use the point-wise features F_M^{i-1} of missing regions as queries to search k nearest neighbors in F_T^i and group these points and features with the closest features to obtain the stacked $N_M^{i-1} \times K \times 3$ point volume and $N_M^{i-1} \times K \times C$ feature volume. We duplicate k copies of the coordinates and features at the same time and then subtract them from the grouped coordinates and features to obtain the spatial relationships and feature maps of the feature approximation points. Then we apply MLPs on the spatial relationships to get positional embeddings, which will be added into feature maps and feature groups. In order to make the feature maps have error correction ability, we apply MLPs followed by a softmax to further obtain the feature weight matrix W. We dot product the feature groups with the W followed by a summation along the K-dimension to obtain the aggregated point-wise feature. Finally, we apply MLPs and Maxpool on the new point-wise feature as the final aggregated global feature of the i-th layer.

3.5 Data Preprocessing and Training Loss

Data Preprocessing. The goal of data preprocessing is to completely disentangle the outputs of the two sub-networks in space. The process can refer to Fig 5(For the best rendering effect, a two-dimensional plane is used here to display). We first grid the partial point cloud to get Fig 5(a), and the grid containing the points is

Table 1: Shape completion results (CD loss multiplied by 10^4) on multi-view partial point cloud dataset (8,192 points). The proposed SDNet achieves the lowest reconstruction errors in 12 categories. The best results are highlighted in bold.

Method	Avg.	airplane	cabinet	car	chair	lamp	sofa	table	watercraft	bed	bench	bookshelf	bus	guitar	motorbike	pistol	skateboard
PCN [37]	9.28	3.78	6.53	4.43	12.71	20.38	8.22	12.44	9.41	20.15	9.73	13.70	3.86	1.34	4.68	5.20	4.55
GRNet [30]	7.53	3.34	6.77	4.52	10.21	16.08	7.71	9.37	5.70	14.46	7.36	10.21	4.14	2.18	3.74	3.22	4.38
TopNet [23]	7.54	3.13	5.65	3.97	10.40	17.21	6.99	9.71	6.85	16.48	7.22	10.40	3.46	1.12	3.96	4.74	2.30
VRCNet [15]	5.66	2.34	4.72	3.33	6.78	12.23	5.62	7.34	4.99	13.74	5.03	6.82	2.82	1.09	2.78	3.90	2.85
PoinTr [36]	4.83	1.79	5.15	3.32	5.71	8.51	5.45	6.36	3.79	11.06	4.29	6.15	2.69	1.56	**2.37**	2.51	3.62
SnowflakeNet [29]	4.62	**1.58**	5.12	3.33	5.64	7.75	4.98	6.23	4.14	9.55	**4.12**	6.19	2.72	**0.62**	2.48	2.65	3.07
Ours	**4.26**	1.90	**4.48**	**3.31**	**5.53**	**6.98**	**4.77**	**5.46**	**3.33**	**8.95**	4.26	**5.86**	**2.56**	0.82	2.52	**1.86**	**2.29**

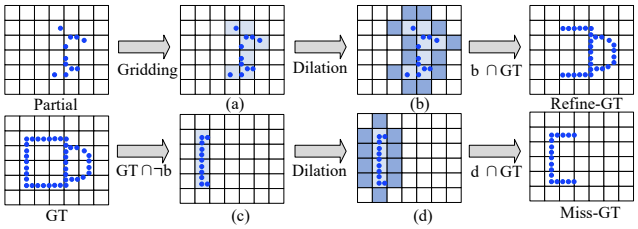

Figure 5: The data preprocessing process. Point clouds are represented by blue points, grids are represented by light blue, and dilation grids are represented by navy blue.

marked in light blue, and then we use a morphological dilation on these grids to obtain Fig 5(b). The dilation of A can be defined as:

$$A \oplus B = \left\{ z \in \mathbb{E} \mid \left(B^s\right)_z \cap A \neq \varnothing \right\}, \quad B^s = \{ x \in \mathbb{E} \mid -x \in B \} \tag{3}$$

Here, B denotes the structuring element and $\mathbb{E}$ is the integer grid $\mathbb{Z}^d$. The new grids resulting from the dilation we color navy blue, from where we can see that through the dilation we close the small holes and gaps. We then intersect these grids with the rasterized ground truth to obtain the ground truth required for refinement, which is denser and smoother than a partial point cloud. We invert the expanded partial grids and then intersect with the ground truth to obtain the ground truth for the missing regions. In order to avoid the ground truth of two parts appearing thin seams or holes at the junction, we also perform the dilation operation on the grid of the missing area. This creates an overlapping region between the two parts of the point cloud, which facilitates the feature fusion of the two sub-networks.

Trainning Loss. In order to maximize the effectiveness of network training, we supervise the results from Dense Refiner and Missing Generator of each step simultaneously. For performance considerations, we use the most widely used Chamfer Distance (CD) as the loss function to measure the difference between two point sets, which is defined as follows:

$$\mathcal{L}_{CD}(P, G) = \frac{1}{|P|} \sum_{x \in P} \min_{y \in G} \|x - y\|_2 + \frac{1}{|G|} \sum_{y \in G} \min_{x \in P} \|y - x\|_2.$$

For Dense Refiner, we note that the point clouds generated by each step of FP are a subset of the partial point clouds. Therefore, only the final output P_R is used as the calculation loss, and the loss function is defined as:

$$\mathcal{L}_{Refine} = \mathcal{L}_{CD}\left(P_R, G_{Refine}\right) \tag{4}$$

where G_{Refine} is the ground truth of the refinement obtained from the data preprocessing. And each step in Missing Generator means the generation of a new point cloud. Therefore, for the output of each MCM in Missing Generator, we adopt the preprocessed ground truth G_{Miss} of the missing region for supervision, and its loss function can be written as:

$$\mathcal{L}_{Miss} = \sum_{i=0}^{T} \mathcal{L}_{CD}\left(P_M^i, G_{Miss}\right) \tag{5}$$

Here, the P_M^i is the i-th step output of MCM. In addition, we concatenate the output of the two subnetworks as the final output P_{Final}, compare it with the complete point clouds G, and its loss function is as follows:

$$\mathcal{L}_{Final} = \mathcal{L}_{CD}\left(P_{Final}, G\right) \tag{6}$$

The total training loss can be written as:

$$\mathcal{L} = \alpha \mathcal{L}_{Refine} + \beta \mathcal{L}_{Miss} + \gamma \mathcal{L}_{Final} \tag{7}$$

The α, β and γ is the hyperparameter.

4 EXPERIMENTS

In this section, we will reveal the test results of SD-Net on two high-difficulty benchmarks: MVP [15] and ShapeNet-55/34 [36]. Experimental results show that our proposed network surprisingly achieves state-of-the-art on point cloud completion tasks. In addition, to demonstrate the effectiveness of our structure, we also conduct ablation experiments for further validation.

4.1 Experiments on MVP Dataset

The MVP dataset [15] is a high-quality multi-view partial point cloud dataset that contains a total of 4,000 CAD models (2,400 for training and 1,600 for testing) and 16 categories. For each CAD model, there are 26 corresponding partial point clouds from different camera perspectives. In addition, the MVP dataset also provides complete point cloud ground truth with different resolutions (including 2,048, 4,096, 8,192 and 16,384). In our experiments, we adopt the ground truth with 8,192 points for training and testing, and we apply L2 version of Chamfer distance as our evaluation indicators.

Table 1 shows the quantitative results of our SD-Net and previous completion methods on MVP dataset, where we can find that SDNet has achieved the best performance in 12 categories over the others. Specifically, compared to the two closest works to our network module, PoinTr [36] and SnowflakeNet [29], SDNet reduces the average CD by more than 8%, and reduces the CD by 2.11

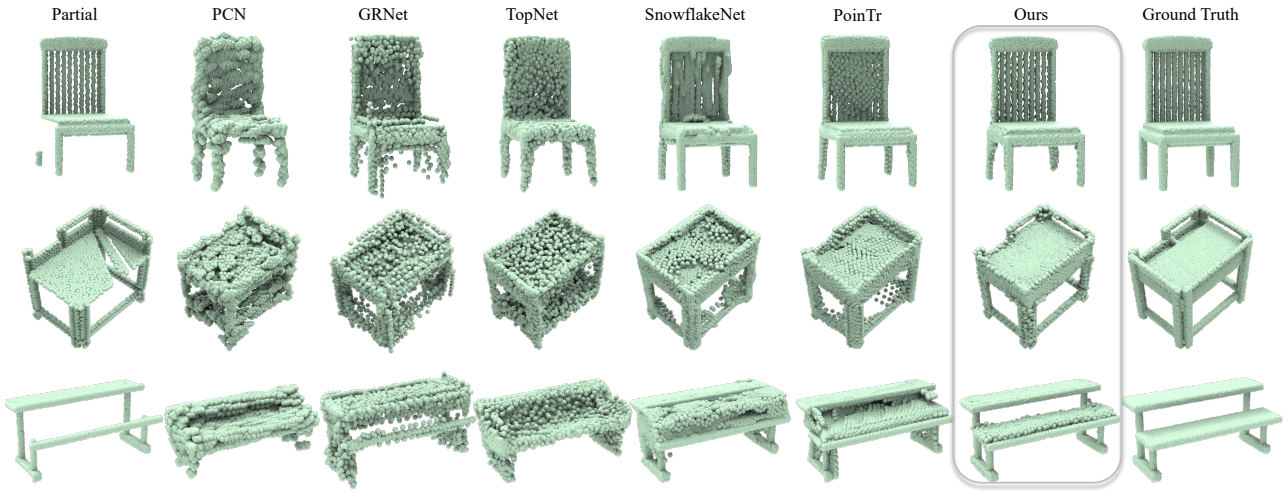

Figure 6: Qualitative results on the MVP dataset. Our SDNet is able to preserve more partial details and also outperforms the other state-of-the-art point cloud completion methods in recovering missing structures.

Table 2: Completion results on ShapeNet-55 evaluated as CD-ℓ_1, CD-ℓ_2 $\times 10^3$ (lower is better) and F-Score@1% (higher is better).

Methods	F-Average	F-S	F-M	F-H	CD-ℓ_1-Average	CD-ℓ_1-S	CD-ℓ_1-M	CD-ℓ_1-H	CD-ℓ_2-Average	CD-ℓ_2-S	CD-ℓ_2-M	CD-ℓ_2-H
PCN [37]	0.164	0.175	0.169	0.149	25.21	23.18	24.32	28.13	2.37	1.85	2.14	3.14
FoldingNet [32]	0.082	-	-	-	-	-	-	-	3.13	2.68	2.66	4.06
TopNet [23]	0.177	0.191	0.181	0.159	23.41	21.66	22.80	26.16	1.94	1.52	1.76	2.54
PFNet [38]	0.339	-	-	-	-	-	-	-	5.22	3.84	3.88	8.03
GRNet [30]	0.248	0.256	0.252	0.236	21.09	19.02	20.46	23.79	1.66	1.13	1.47	2.37
SnowflakeNet [29]	0.413	0.451	0.421	0.366	15.20	12.95	14.62	18.02	1.03	0.61	0.87	1.60
PoinTr [36]	0.469	0.497	0.485	0.424	13.95	11.62	13.08	17.14	1.05	0.58	0.88	1.70
SeedFormer [43]	0.444	0.486	0.455	0.391	13.98	11.66	13.33	16.96	0.92	0.49	0.77	1.50
Ours	**0.532**	**0.558**	**0.557**	**0.481**	**11.76**	**9.94**	**10.78**	**14.56**	**0.85**	**0.48**	**0.69**	**1.39**

Table 3: Completion results on ShapeNet-34 dataset evaluated as CD-ℓ_2 $\times 1000$ (lower is better) and F-Score@1% (higher is better).

	34 seen categories					21 unseen categories				
	CD-Avg	F-Avg	CD-S	CD-M	CD-H	CD-Avg	F-Avg	CD-S	CD-M	CD-H
FoldingNet [32]	2.35	0.139	1.86	1.81	3.38	3.62	0.095	2.76	2.74	5.36
PCN [37]	2.22	0.154	1.87	1.81	2.97	3.85	0.101	3.17	3.08	5.29
TopNet [23]	2.31	0.171	1.77	1.61	3.54	3.50	0.121	2.62	2.43	5.44
PFNet [38]	4.68	0.347	3.16	3.19	7.71	8.16	0.322	5.29	5.87	13.33
GRNet [30]	1.74	0.251	1.26	1.39	2.57	2.99	0.216	1.85	2.25	4.87
PoinTr [36]	1.23	0.421	0.76	1.05	1.88	1.67	3.44	1.04	1.67	3.44
SnowflakeNet [29]	0.99	0.422	0.60	0.86	1.50	1.75	0.388	0.88	1.46	2.92
SeedFormer [43]	0.83	0.452	**0.48**	0.70	1.30	1.34	0.402	**0.61**	1.07	2.35
Ours	**0.81**	**0.535**	0.49	**0.67**	**1.27**	**1.33**	**0.503**	0.64	**1.02**	**2.32**

on the bed category, which is lower 19% than the PoinTr's result. Therefore, these improvements should be attributed to our network architecture,which has a great effect on improving the performance of point cloud completion. We also compare with previous methods on qualitative results, as shown in Fig 6. It shows that SDNet can generate obviously better results with more faithful geometric details and less noise.

4.2 Experiments on ShapeNet-55/34 Dataset

The ShapeNet-55 dataset [36] is composed of artificially synthesized ShapeNet datasets [27], which were first proposed in PoinTr [36]. ShapeNet-55 contains a total of 55 categories with 41,952 models for training and 10,518 models for testing, which is currently the point cloud completion dataset with the most categories and the most difficulties. The ShapeNet-34 dataset is also a subset of ShapeNet, divided into two parts: 21 unseen categories for testing with 2,305 models and 34 seen categories with 46,765 models for training and 3,400 for testing. Both of the datasets sample 2,048 points as input and 8,192 points as ground truth. In the evaluation stage, we adopt the same strategy as PoinTr, select 8 fixed viewing angles to generate partial point cloud by sampling 2,048, 4,096 or 6,144 points

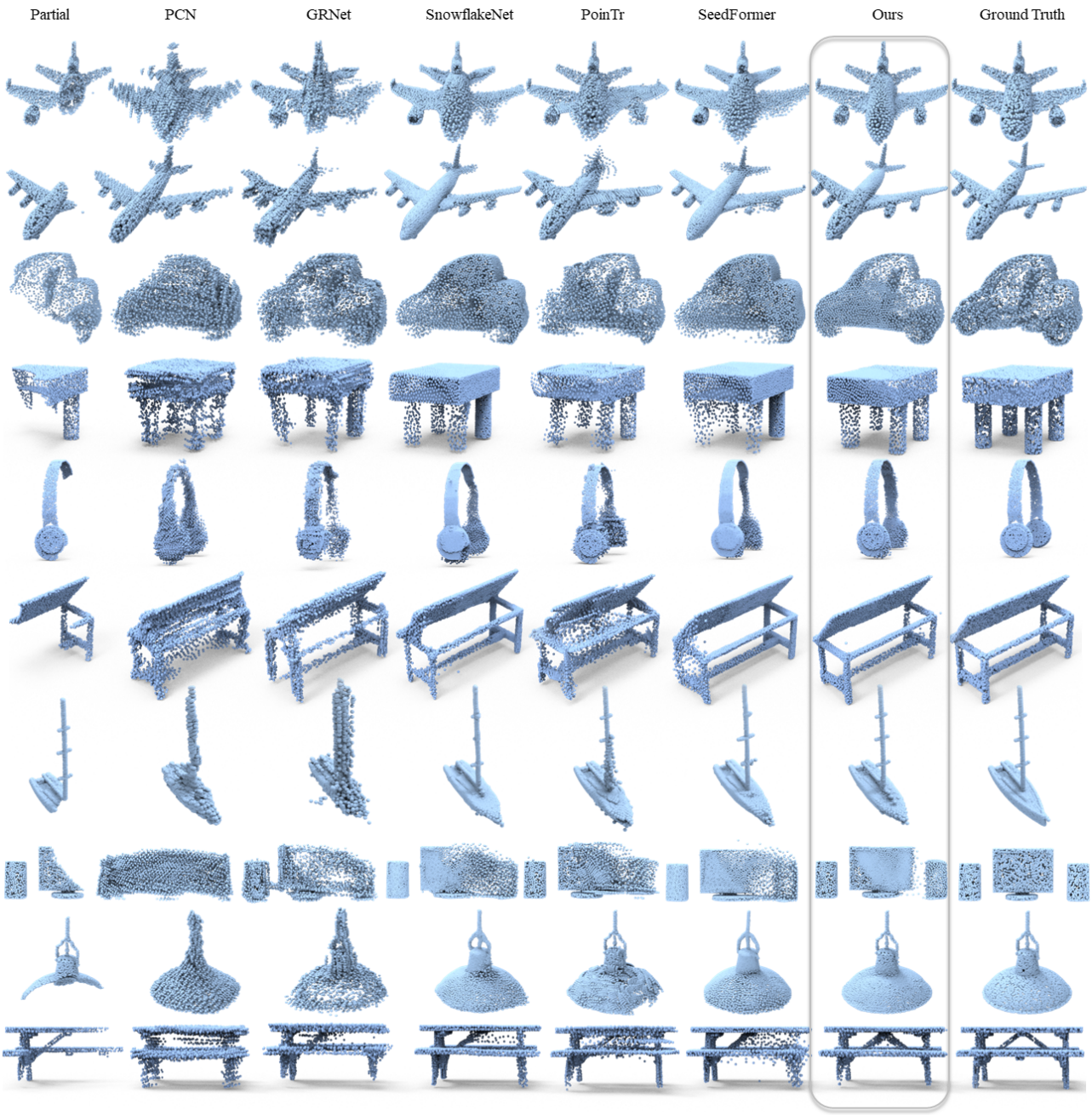

Figure 7: Qualitative results on the ShapeNet-55 dataset. Our SDNet is able to preserve more partial details and also outperforms the other state-of-the-art point cloud completion methods in recovering missing structures.

(25%, 50% or 75% of the complete point cloud) which corresponds to three different levels of difficulty.

We tested all 55 categories under three levels of difficulty, using CD-ℓ_1, CD-ℓ_2 and F-Score as our evaluation indicators, and the test results are shown in table 2. It can be seen that SDNet exceeds the state-of-the-art model in any evaluation indicators under the three difficulties. On easy difficulty, our CD-ℓ_2 metric is close to that of SeedFormer [43], but on hard difficulty, our CD-ℓ_2 metric has improved significantly, which shows that SDNet has better robustness.

It is also worth mentioning that our F-Score and CD-ℓ_1 have greatly improved compared to SeedFormer [43] and PoinTr [36], which shows that the point cloud generated by SDNet better follows the distribution of ground truth. This will be more evident in the qualitative experimental results in Fig 7. From these results, we can clearly see that our results are closer to the complete point cloud, especially in the details of missing regions, e.g., the jet port of the aircraft, audios on both sides of the computer, etc. Furthermore, we provide error maps as shown in Fig 8, where the colors indicate

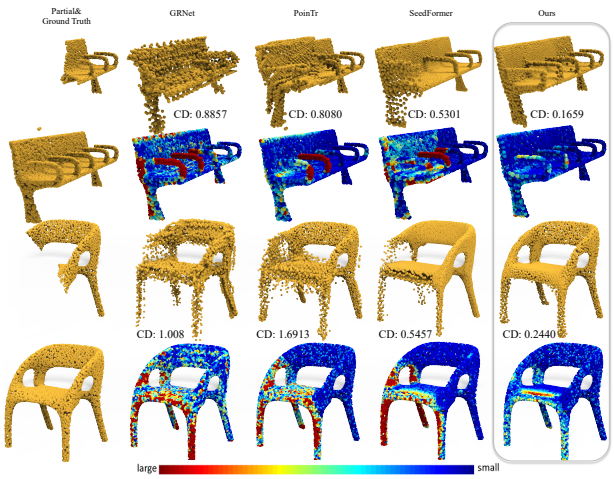

Figure 8: Comparing point cloud completion results from partial inputs using different methods. We also show the associated error maps, where the colors reveal the nearest distance for each target point to the predicted point set generated by each method.

Table 4: Ablation study on the MVP dataset. We investigate different designs including Dense Refiner, Non-Symmetrical Cross Transformer (NSCT) and Data Preprocess.

Model	Missing Generator	Dense Refiner	NSCT	Data Preprocess	CD-ℓ_2
A	✓				5.50
B	✓	✓		✓	4.50
C	✓	✓	✓		4.83
Full	✓	✓	✓	✓	**4.26**

the nearest distance for each point in the target point set to the predicted point set. We can see that the errors of our completed results are the lowest (i.e., most points are blue and least points are red), which is also verified by CD values. More comparison results can be found in the supplemental material.

We also conduct experiments for SDNet and previous state-of-the-art methods on ShapeNet-34. The quantitative experimental results are shown in Table 3, from where we can see that our results are significantly better than PoinTr in both 34 seen categories and 21 unseen categories. Compared with the state-of-the-art SeedFormer, our CD-ℓ_2 is basically similar to their results, but we achieved 0.535 F-score on 34 seen categories, while SeedFormer only obtained 0.452 F-score. Furthermore, in the 21 unseen categories, we have improved the F-score by about 0.1 compared with the SeedFormer.

4.3 Ablation Study

To demonstrate the effectiveness of the major components in our SDNet, we conducted a detailed ablation study on the MVP dataset with the resolution of 8192 points by simplifying the total framework in the flowing three cases: (A) removing the Dense Refiner and Cross Fusion Unit and only keeping the Missing Generator; (B) removing the whole Non-Symmetrical Cross Transformer in Cross Fusion Unit and replace it with MLPs; (C) removing the ground truth used for supervision obtained by our data preprocessing. As shown in Table 4, we compared each case's CD-ℓ_2 value to our full pipeline (bottom row). We can clearly observe that when only the Missing Generator is used for point cloud completion (case A), the performance of the network has declined sharply, which shows that the ability of Dense Refiner to aggregate local geometric features is crucial for point cloud completion. In case B, we adopted MLPs to fairly aggregate the point-wise features of the two sub-networks, which resulted in the aggregated feature being the average feature of the two sub-networks, whereas NSCT tends to be more inclined to the features of the missing region points when aggregating, which is beneficial to maintain the feature distribution learned by Missing Generator, and helps Dense Refiner to perceive the information of the missing area. By comparing case3 with the full model, we can conclude that such data processing can provide better supervision for point cloud completion. Clearly, our full pipeline performs the best with the lowest CD value, and removing any component will decrease overall quality, meaning that each component in our framework contributes.

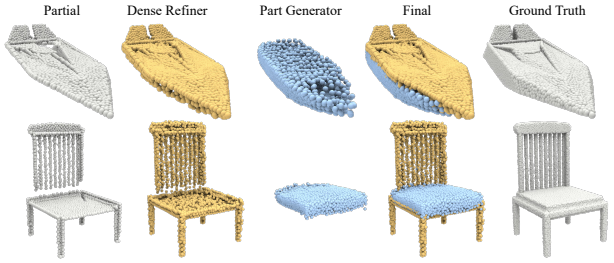

Figure 9: Visualization of the intermediate results of SDNet. The objects in green are outputs of Dense Refiner while the objects in yellow are Missing Generator's results. The last column is the final output of SDNet.

4.4 Visualization of SDNet

In Fig 9, we visualize the process of point cloud completion. We can see that Dense Refiner refines the partial and completes the area close to the partial, while the Missing Generator focuses on the completion of the missing part. In addition, we can also see that the output of the two sub-networks has some overlapping parts, which is conducive to the final output having a smoother surface in the connection of the two parts.

5 CONCLUSION

In this paper, we present a novel point cloud completion framework, which disentangled the completion task into two sub-goals, partial refinement and missing region generation. To this end, we formulate an end-to-end disentangled completion structure with two sub-networks:a Dense Refiner and a Missing Generator. Additionally, we designed a Fusion Unit to aggregate point relationships between refine and missing regions. With the help of NSCTs, these features can provide accurate detail information for missing region inference. By cascading, these multi-scale features can provide global structural information for region refinement. And we processed the data so that each sub-network could focus on its own tasks. Experimental results demonstrate the superiority of our method over others.

6 ACKNOWLEDGEMENT

This research was supported by National Natural Science Foundation of China (No.62202151).

REFERENCES

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

[2] Matthew Berger, Andrea Tagliasacchi, Lee Seversky, Pierre Alliez, Joshua Levine, Andrei Sharf, and Claudio Silva. 2014. State of the art in surface reconstruction from point clouds. *Eurographics 2014-State of the Art Reports* 1, 1 (2014), 161–185.

[3] Yakun Chang, Cheolkon Jung, and Yuanquan Xu. 2021. FinerPCN: High fidelity point cloud completion network using pointwise convolution. *Neurocomputing* 460 (2021), 266–276.

[4] Xiaozhi Chen, Huimin Ma, Ji Wan, Bo Li, and Tian Xia. 2017. Multi-view 3d object detection network for autonomous driving. In *Proceedings of the IEEE conference on Computer Vision and Pattern Recognition*. 1907–1915.

[5] David M Cole and Paul M Newman. 2006. Using laser range data for 3D SLAM in outdoor environments. In *Proceedings 2006 IEEE International Conference on Robotics and Automation, 2006. ICRA 2006.* IEEE, 1556–1563.

[6] Thibault Groueix, Matthew Fisher, Vladimir G Kim, Bryan C Russell, and Mathieu Aubry. 2018. A papier-mâché approach to learning 3d surface generation. In *Proceedings of the IEEE conference on computer vision and pattern recognition*. 216–224.

[7] Kaiming He, Xiangyu Zhang, Shaoqing Ren, and Jian Sun. 2016. Deep residual learning for image recognition. In *Proceedings of the IEEE conference on computer vision and pattern recognition*. 770–778.

[8] Wei Hu, Zeqing Fu, and Zongming Guo. 2019. Local frequency interpretation and non-local self-similarity on graph for point cloud inpainting. *IEEE Transactions on Image Processing* 28, 8 (2019), 4087–4100.

[9] Zitian Huang, Yikuan Yu, Jiawen Xu, Feng Ni, and Xinyi Le. 2020. Pf-net: Point fractal network for 3d point cloud completion. In *Proceedings of the IEEE/CVF conference on computer vision and pattern recognition*. 7662–7670.

[10] Ruihui Li, Xianzhi Li, Chi-Wing Fu, Daniel Cohen-Or, and Pheng-Ann Heng. 2019. PU-GAN: a point cloud upsampling adversarial network. In *CVPR*. 7203–7212.

[11] Ruihui Li, Xianzhi Li, Pheng-Ann Heng, and Chi-Wing Fu. 2021. Point cloud upsampling via disentangled refinement. In *Proceedings of the IEEE/CVF conference on computer vision and pattern recognition*. 344–353.

[12] Priyanka Mandikal and Venkatesh Babu Radhakrishnan. 2019. Dense 3d point cloud reconstruction using a deep pyramid network. In *2019 IEEE Winter Conference on Applications of Computer Vision (WACV)*. IEEE, 1052–1060.

[13] Yinyu Nie, Yiqun Lin, Xiaoguang Han, Shihui Guo, Jian Chang, Shuguang Cui, Jian Zhang, et al. 2020. Skeleton-bridged point completion: From global inference to local adjustment. *Advances in Neural Information Processing Systems* 33 (2020), 16119–16130.

[14] Sergio Orts-Escolano, Christoph Rhemann, Sean Fanello, Wayne Chang, Adarsh Kowdle, Yury Degtyarev, David Kim, Philip L Davidson, Sameh Khamis, Mingsong Dou, et al. 2016. Holoportation: Virtual 3d teleportation in real-time. In *Proceedings of the 29th annual symposium on user interface software and technology*. 741–754.

[15] Liang Pan, Xinyi Chen, Zhongang Cai, Junzhe Zhang, Haiyu Zhao, Shuai Yi, and Ziwei Liu. 2021. Variational relational point completion network. In *Proceedings of the IEEE/CVF conference on computer vision and pattern recognition*. 8524–8533.

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

[17] Charles Ruizhongtai Qi, Li Yi, Hao Su, and Leonidas J Guibas. 2017. Pointnet++: Deep hierarchical feature learning on point sets in a metric space. *Advances in neural information processing systems* 30 (2017).

[18] Yue Qian, Junhui Hou, Sam Kwong, and Ying He. 2020. PUGeo-Net: A geometry-centric network for 3D point cloud upsampling. In *Computer Vision–ECCV 2020: 16th European Conference, Glasgow, UK, August 23–28, 2020, Proceedings, Part XIX*. Springer, 752–769.

[19] Muhammad Sarmad, Hyunjoo Jenny Lee, and Young Min Kim. 2019. Rl-gan-net: A reinforcement learning agent controlled gan network for real-time point cloud shape completion. In *Proceedings of the IEEE/CVF Conference on Computer Vision and Pattern Recognition*. 5898–5907.

[20] Chao-Hui Shen, Hongbo Fu, Kang Chen, and Shi-Min Hu. 2012. Structure recovery by part assembly. *ACM Transactions on Graphics (TOG)* 31, 6 (2012), 1–11.

[21] Minhyuk Sung, Vladimir G Kim, Roland Angst, and Leonidas Guibas. 2015. Data-driven structural priors for shape completion. *ACM Transactions on Graphics (TOG)* 34, 6 (2015), 1–11.

[22] Junshu Tang, Zhijun Gong, Ran Yi, Yuan Xie, and Lizhuang Ma. 2022. LAKe-Net: topology-aware point cloud completion by localizing aligned keypoints. In *Proceedings of the IEEE/CVF conference on computer vision and pattern recognition*. 1726–1735.

[23] Lyne P Tchapmi, Vineet Kosaraju, Hamid Rezatofighi, Ian Reid, and Silvio Savarese. 2019. Topnet: Structural point cloud decoder. In *Proceedings of the IEEE/CVF Conference on Computer Vision and Pattern Recognition*. 383–392.

[24] Yida Wang, David Joseph Tan, Nassir Navab, and Federico Tombari. 2022. Learning local displacements for point cloud completion. In *Proceedings of the IEEE/CVF Conference on Computer Vision and Pattern Recognition*. 1568–1577.

[25] Xin Wen, Peng Xiang, Zhizhong Han, Yan-Pei Cao, Pengfei Wan, Wen Zheng, and Yu-Shen Liu. 2021. Pmp-net: Point cloud completion by learning multi-step point moving paths. In *Proceedings of the IEEE/CVF Conference on Computer Vision and Pattern Recognition*. 7443–7452.

[26] Xin Wen, Peng Xiang, Zhizhong Han, Yan-Pei Cao, Pengfei Wan, Wen Zheng, and Yu-Shen Liu. 2022. PMP-Net++: Point cloud completion by transformer-enhanced multi-step point moving paths. *IEEE Transactions on Pattern Analysis and Machine Intelligence* 45, 1 (2022), 852–867.

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

[28] Yaqi Xia, Yan Xia, Wei Li, Rui Song, Kailang Cao, and Uwe Stilla. 2021. Asfm-net: Asymmetrical siamese feature matching network for point completion. In *Proceedings of the 29th ACM International Conference on Multimedia*. 1938–1947.

[29] Peng Xiang, Xin Wen, Yu-Shen Liu, Yan-Pei Cao, Pengfei Wan, Wen Zheng, and Zhizhong Han. 2021. Snowflakenet: Point cloud completion by snowflake point deconvolution with skip-transformer. In *Proceedings of the IEEE/CVF international conference on computer vision*. 5499–5509.

[30] Haozhe Xie, Hongxun Yao, Shangchen Zhou, Jiageng Mao, Shengping Zhang, and Wenxiu Sun. 2020. Grnet: Gridding residual network for dense point cloud completion. In *Computer Vision–ECCV 2020: 16th European Conference, Glasgow, UK, August 23–28, 2020, Proceedings, Part IX*. Springer, 365–381.

[31] Xuejun Yan, Hongyu Yan, Jingjing Wang, Hang Du, Zhihong Wu, Di Xie, Shiliang Pu, and Li Lu. 2022. Fbnet: Feedback network for point cloud completion. In *Computer Vision–ECCV 2022: 17th European Conference, Tel Aviv, Israel, October 23–27, 2022, Proceedings, Part II*. Springer, 676–693.

[32] Yaoqing Yang, Chen Feng, Yiru Shen, and Dong Tian. 2018. Foldingnet: Point cloud auto-encoder via deep grid deformation. In *Proceedings of the IEEE conference on computer vision and pattern recognition*. 206–215.

[33] Wang Yifan, Shihao Wu, Hui Huang, Daniel Cohen-Or, and Olga Sorkine-Hornung. 2019. Patch-based progressive 3d point set upsampling. In *Proceedings of the IEEE/CVF Conference on Computer Vision and Pattern Recognition*. 5958–5967.

[34] Lequan Yu, Xianzhi Li, Chi-Wing Fu, Daniel Cohen-Or, and Pheng-Ann Heng. 2018. Ec-net: an edge-aware point set consolidation network. In *Proceedings of the European conference on computer vision (ECCV)*. 386–402.

[35] Lequan Yu, Xianzhi Li, Chi-Wing Fu, Daniel Cohen-Or, and Pheng-Ann Heng. 2018. Pu-net: Point cloud upsampling network. In *Proceedings of the IEEE conference on computer vision and pattern recognition*. 2790–2799.

[36] Xumin Yu, Yongming Rao, Ziyi Wang, Zuyan Liu, Jiwen Lu, and Jie Zhou. 2021. Pointr: Diverse point cloud completion with geometry-aware transformers. In *Proceedings of the IEEE/CVF international conference on computer vision*. 12498–12507.

[37] Wentao Yuan, Tejas Khot, David Held, Christoph Mertz, and Martial Hebert. 2018. Pcn: Point completion network. In *2018 International Conference on 3D Vision (3DV)*. IEEE, 728–737.

[38] Junchao Zhang, Jianbo Shao, Jianlai Chen, Degui Yang, Buge Liang, and Rongguang Liang. 2020. PFNet: an unsupervised deep network for polarization image fusion. *Optics letters* 45, 6 (2020), 1507–1510.

[39] Wenxiao Zhang, Zhen Dong, Jun Liu, Qingan Yan, Chunxia Xiao, et al. 2022. Point cloud completion via skeleton-detail transformer. *IEEE Transactions on Visualization and Computer Graphics* (2022).

[40] Wenxiao Zhang, Chengjiang Long, Qingan Yan, Alix LH Chow, and Chunxia Xiao. 2020. Multi-stage point completion network with critical set supervision. *Computer Aided Geometric Design* 82 (2020), 101925.

[41] Wenxiao Zhang, Qingan Yan, and Chunxia Xiao. 2020. Detail preserved point cloud completion via separated feature aggregation. In *Computer Vision–ECCV 2020: 16th European Conference, Glasgow, UK, August 23–28, 2020, Proceedings, Part XXV 16*. Springer, 512–528.

[42] Hengshuang Zhao, Li Jiang, Jiaya Jia, Philip HS Torr, and Vladlen Koltun. 2021. Point transformer. In *Proceedings of the IEEE/CVF international conference on computer vision*. 16259–16268.

[43] Haoran Zhou, Yun Cao, Wenqing Chu, Junwei Zhu, Tong Lu, Ying Tai, and Chengjie Wang. 2022. Seedformer: Patch seeds based point cloud completion with upsample transformer. In *Computer Vision–ECCV 2022: 17th European Conference, Tel Aviv, Israel, October 23–27, 2022, Proceedings, Part III*. Springer, 416–432.

7 SUPPLEMENTARY

7.1 Implementation Details

In this section, we will give the specific implementation details of the SDNet. We train our network on the PyTorch platform. For ShapeNet-55/34 dataset, the network is set to train with a batchsize of 48 for 200 epochs and the AdamW optimizer is used with the learning rate of 0.005, which is continuously decreased by a decay rate of 0.76 for every 20 epochs. The hyperparameter α, β and γ are all set to 1, which means that the model training will pay more attention to the generation of missing regions. If the input point cloud in the dataset has large noise, the values of α and γ can be increased appropriately. On the MVP dataset [15], we set the batch size as before but set the epoch to 100 and adopt the same learning rate and decay rate as VRCNet[15].

Feature Extractor. We use two cascaded set abstraction (SA) [17] and point transformer (PT) [42] to extract features from the partial point cloud. In order to save computational cost, we progressively use Farthest Point Sampling(FPS) to downsample the original partial point cloud (2,048 points) to 256 center points. The detailed network architecture is: $SA\,(C_{in} = 3, C_{out} = 128, N_{out} = 512) \rightarrow PT\,(C_{in} = 128, C_{out} = 128) \rightarrow SA(C_{in} = 128, C_{out} = 256, N_{out} = 256) \rightarrow PT(C_{in} = 256, C_{out} = 256)$, where C_{in} and C_{out} are the numbers of feature channels of the input and output point clouds, and N_{out} is the number of points after FPS.

Set Transformer. We employ a lightweight Geometry-aware Transformer [36] as our Set Transformer to convert the center point of the partial point cloud to the keypoint of the missing part. Due to the strong strength of MCM in point cloud generation, we set the numbers of blocks of the encoder and decoder in Set Transformer to 1 and 2 and the hidden dimensions are set to 256.

Table 5: Complexity analysis. We report the the number of parameter (Params) and theoretical computation cost (FLOPs) of our method and eight existing methods. We also provide the average Chamfer distances of all categories in ShapeNet-55 (CD_{55}) and unseen categories in ShapeNet-34 (CD_{34}) as references.

Models	Params	FLOPs	CD_{55}	CD_{34}
FoldingNet [32]	2.30 M	27.58 G	3.12	3.62
PCN [37]	5.04 M	15.25 G	2.66	3.85
TopNet [23]	5.76 M	6.72 G	2.91	3.50
PFNet [38]	73.05 M	4.96 G	5.22	8.16
GRNet [30]	73.15 M	40.44 G	1.97	2.99
PoinTr [36]	30.9 M	10.41 G	1.07	2.05
SnowflakeNet [29]	19.30 M	9.17 G	1.03	1.75
SeedFormer [43]	3.24 M	20.70 G	0.92	1.34
Ours	14.32 M	10.83 G	**0.85**	**1.33**

7.2 Complexity Analysis

We provide a detailed complexity analysis in Table 5, including the number of parameters and theoretical computation cost (FLOPs) of different models under the ShapeNet-55 dataset. We also provide the overall Chamfer Distance of all categories in ShapeNet-55 and unseen 21 categories in ShapeNet-34 as references. Recent state-of-the-art methods have shown impressive performance gains, but often at the cost of increased time and space requirements. In contrast, our method achieves superior performance while maintaining low time and space overhead. Our approach strikes a balance between efficiency and effectiveness, with a focus on achieving the best possible performance without sacrificing practical considerations.

7.3 Dataset Preprocessing Details

The purpose of data preprocessing is to provide strong and effective supervision for the training process of SDNet. This allows the point clouds generated by the two sub-networks to be better disentangled spatially. In this section, we give detailed data preprocessing details, as shown in Algorithm 1.

Algorithm 1 Point Cloud Spatial Disentangle Algorithm

Input: Partial point cloud $P = \{p_i\}_{i=1}^{N}$, complete point cloud $P_c = \{p_i\}_{i=1}^{N_c}$, resolution σ
Output: Ground truth for refining $P_r = \{p_i\}_{i=1}^{N_r}$, ground truth for missing $P_m = \{p_i\}_{i=1}^{N_m}$
Initialization: $N_r = N_m = 0$, the dilation kernel $D_r = D_m = (2, 2, 2)$
Voxelize the P and P_c with the same resolution r to obtain V_p^σ and V_c^σ
for $N_r < \frac{N_c}{2}$ **do**
  Increase the size of the dilation kernel: $D_r = D_r + 1$
  Perform a dilation operation on V_p^σ: $V_p^{\sigma\prime} = dilation(V_p^\sigma, D_r)$
  $P_r = \left(V_p^{\sigma\prime} \cap V_c^\sigma\right) \cap P_c$, $N_r = num\,(P_r)$
  $V_m^\sigma = \neg V_p^\sigma \cap V_c^\sigma$
  for $N_m < \frac{N_c}{2}$ **do**
    Increase the size of the dilation kernel: $D_m = D_m + 1$
    Perform a dilation operation on V_m^σ: $V_m^{\sigma\prime} = dilation(V_m^\sigma, D_m)$
    $P_m = \left(V_m^{\sigma\prime} \cap V_c^\sigma\right) \cap P_c$, $N_m = num\,(P_m)$
  end for
end for
Grid sampling for data alignment: $P_r = GS\left(P_r, \frac{N_c}{2}\right)$
Grid sampling for data alignment: $P_m = GS\left(P_m, \frac{N_c}{2}\right)$

Specifically, for the ShapeNet-34/55 dataset, we set the ground truth points of the missing region and refine region to be half of the total points in the complete point clouds, i.e., 4096 points. This results in better data alignment and concatenation. Furthermore, to determine the most suitable voxelization resolution, we also performed an analysis on the dataset and the results are shown in Fig. 10. The blue column indicates the number of samples included in this category. The polylines of different colors represent the number of sampling points at different resolutions. The closer these polylines are to 4096, the better the precision. As points exceed 4096, downsampling is required. This process will inevitably lose the structure of existing points, which is not conducive to refinement. There will be more cracks at the connection point of the final completion result when the number of points is less than 4096, since a larger dilation kernel will increase the connection range. Additionally, we use Grid Sampling(GS) rather than FPS in order to avoid destroying the overall structure of the point cloud.

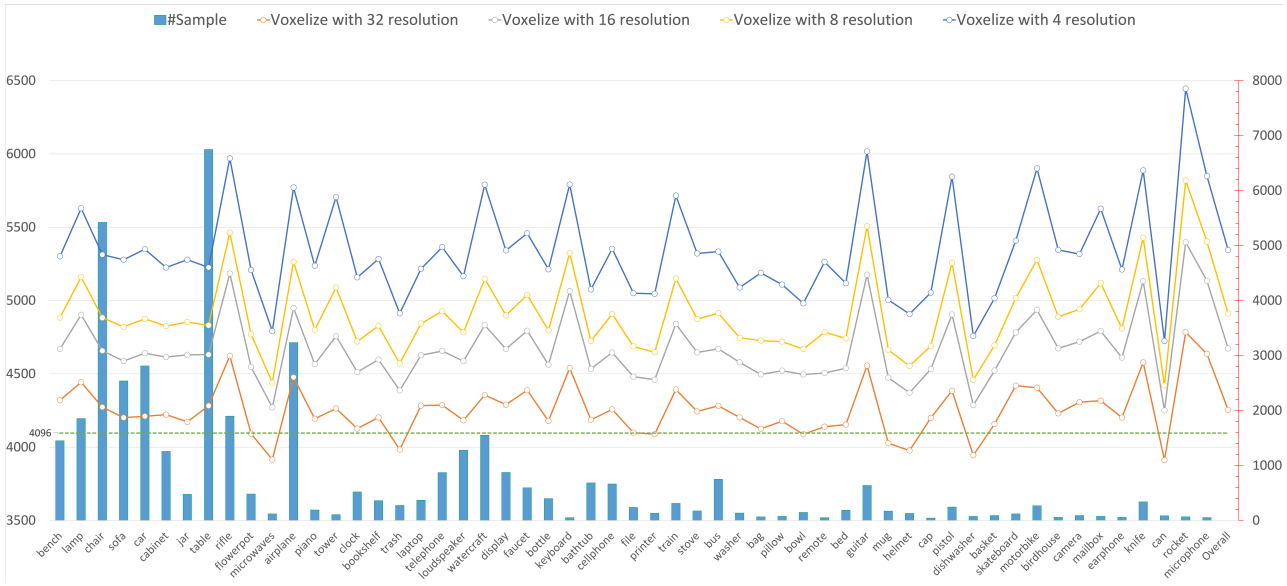

Figure 10: Specific ShapeNet-55 processing and analysis results. We provide the number of points obtained by voxelizing the point cloud at different resolutions and the number of samples included in the different categories.

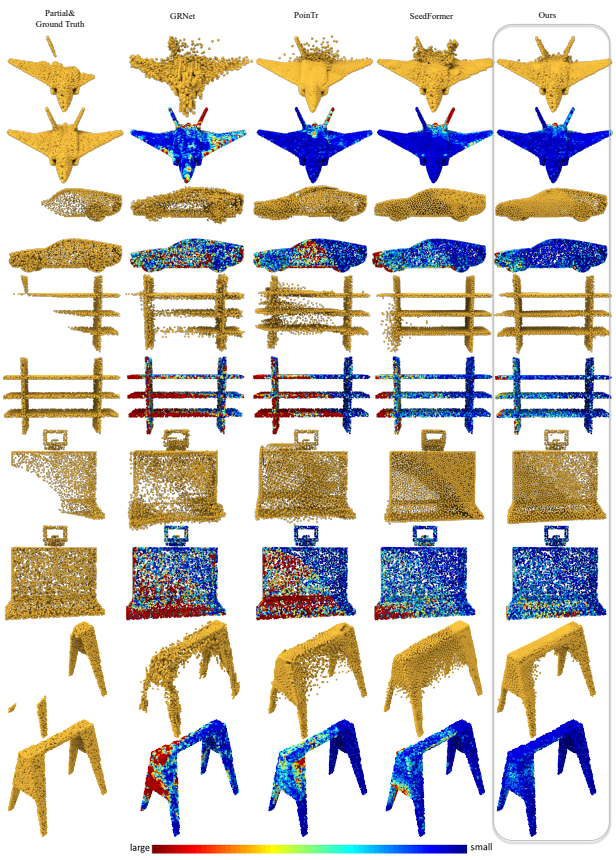

Figure 11: Comparing point cloud completion results using different methods. We also show the associated error maps, where the colors reveal the nearest distance for each target point to the point set generated by each method.

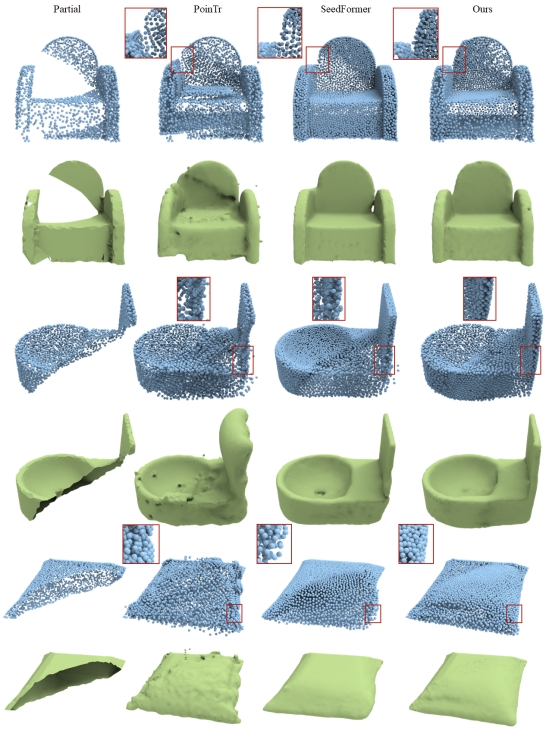

Figure 12: Comparing point cloud completion results and reconstructed 3D meshes using different methods.

7.4 More Qualitative Results

In Fig. 11 and Fig. 12, we provide more qualitative results of point cloud completion. It can be seen that our results can better infer missing structures and have smoother surfaces compared to other methods.