Appendix for "ProxyFormer: Proxy Alignment Assisted Point Cloud Completion with Missing Part Sensitive Transformer"

Anonymous CVPR submission

Paper ID 10370

1. Overview

In this supplementary material, we provide additional information to complement the manuscript. First, we provide the details of missing part extractor in Sec. 2. Second, we present additional implementation details and experimental settings of *ProxyFormer* (Sec. 3). At last, we do more ablation studies and visualize more qualitative results of our method on the PCN dataset and ShapeNet-55/34. We also present detailed quantitative results on ShapeNet-55 and ShapeNet-34. (Sec. 4).

2. Missing Part Extractor

The given complete point cloud and incomplete point cloud of some point cloud completion common datasets such as PCN are not in the same scale of coordinate systems. So it is not possible to simply use the set difference operation to obtain the missing part. So we design a missing part extractor based on point-to-point and point-to-plane distances [4]. Fig. 1 is a detailed description of the missing part extractor.

Let $\mathbf{A}$ and $\mathbf{B}$ denote the Ground Truth (GT) and the incomplete point cloud, respectively. Firstly, we use the normal vector estimation method [5] to obtain the normal vector of each point in $\mathbf{B}$. For each point $\boldsymbol{p}_i$, we assume the covariance matrix $\mathcal{C}$ as follows:

$$\begin{aligned} \mathcal{C} &= \frac{1}{k}\sum_{i=1}^{k}\cdot(\boldsymbol{p}_i-\overline{\boldsymbol{p}})\cdot(\boldsymbol{p}_i-\overline{\boldsymbol{p}})^T\,, \\ \mathcal{C}\cdot\vec{\mathbf{v}}_j &= \lambda_j\cdot\vec{\mathbf{v}}_j, j\in\{0,1,2\}\,. \end{aligned} \tag{1}$$

Here k refers to the k points closest to $\boldsymbol{p}_i$. $\overline{\boldsymbol{p}}$ is the centroid of the nearest neighbor. λ_j is the j^{th} eigenvalue and $\vec{\mathbf{v}}_j$ is the j^{th} eigenvector. Then, the eigenvector corresponding to the largest eigenvalue is selected and normalized as the normal vector of the point.

After the calculation of the normal vector, the KNN algorithm is used to obtain the index of the corresponding adjacent points between the GT and the incomplete point cloud, which can be expressed as: $idx = KNN\left(\mathbf{A},\mathbf{B},\mathrm{K}=1\right)$. According to this index, we go to $\mathbf{B}$ to find the points $\boldsymbol{b}_i$ $(i\in idx)$ and normal vectors $\boldsymbol{n}_i$ $(i\in idx)$ of the adjacent points. Here we denote the point cloud composed of b_i as $\mathbf{R}$ (it contains N points and the value of each point comes from the incomplete input, but they are one-to-one correspondence with points in GT), and denote the set of normal vectors n_i corresponding to each point in the point cloud $\mathbf{R}$ as $\mathbb{N}$.

Then as shown in Fig. 2, we calculate the point-to-point distance and the point-to-plane distance as follows.

(1) For a point a_i in point cloud $\mathbf{A}$, *i.e.*, the blue point in the figure, a corresponding point r_j in the point cloud $\mathbf{R}$ can be found, *i.e.*, the orange point in the figure. Vice versa.

(2) Similarly, we can get the corresponding normal vector n_j from $\mathbb{N}$.

(3) We connect point r_j to point a_i to calculate an error vector whose length is the point-to-point distance, *i.e.*,

$$D^{c2c}_{a_i,r_j} = ||E\left(i,j\right)||_2\,. \tag{2}$$

(4) We project the error vector $E\left(i,j\right)$ along the direction of normal vector n_j to get another error vector $\hat{E}\left(i,j\right)$ whose length is the point-to-plane distance, *i.e.*,

$$D^{c2p}_{a_i,r_j} = ||\hat{E}(i,j)||_2 = \frac{E(i,j)\cdot n_j}{||n_j||_2} = ||E\left(i,j\right)\cdot n_j||_2\,. \tag{3}$$

(5) We use the weighted sum of the two distances as a comparison condition,

$$D = \alpha D^{c2c}_{a_i,r_j} + \beta D^{c2p}_{a_i,r_j}\,, \tag{4}$$

where $\alpha = 0.2$ and $\beta = 0.8$.

We perform the above operations on each point in the GT $\mathbf{A}$. If the calculated distance of the point is less than or equal to the preset threshold of 0.01, the point is stored in the set of existing parts, otherwise, it is stored in the set of missing parts. This process can be formulated as,

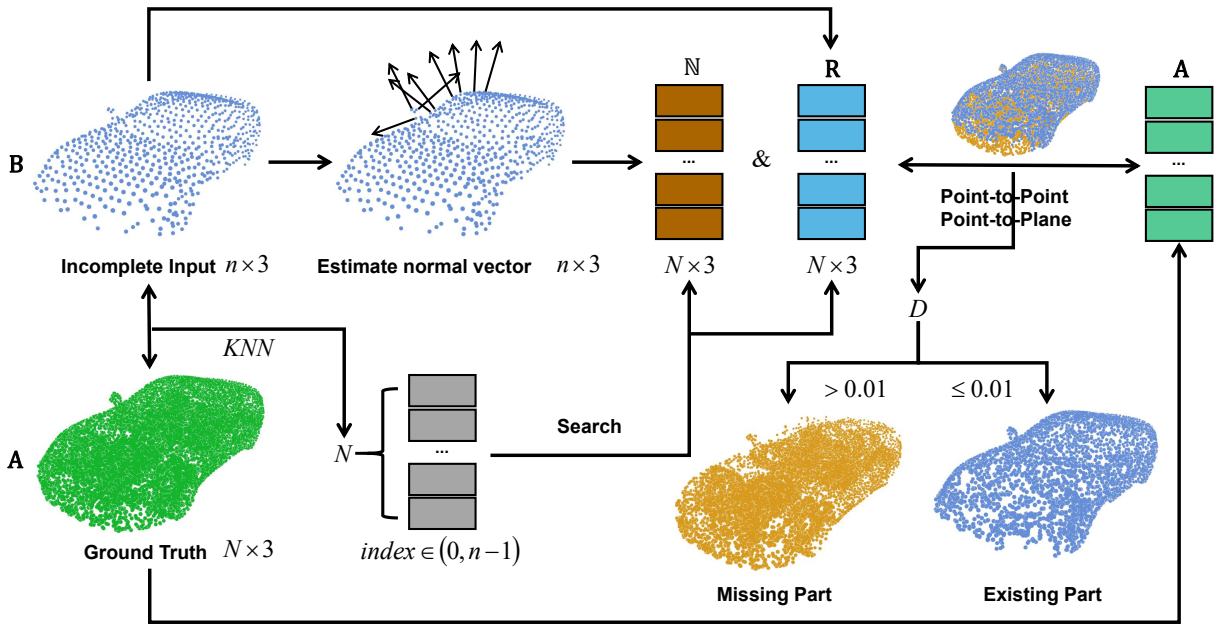


Figure 1. The pipeline of Missing Part Extractor. N is the count of points in ground truth and n is the count of points in incomplete input. Firstly, normal vectors are estimated for the incomplete input $\mathbf{B}$. Secondly, the KNN algorithm is used to obtain the index with the shortest distance in $\mathbf{B}$ corresponding to each point in the ground truth $\mathbf{A}$. A set of points $\mathbf{R}$ and a set of normal vectors $\mathbb{N}$ are obtained according to the index. Finally, we use $\mathbf{R}$, $\mathbb{N}$ and $\mathbf{A}$ to calculate the point-to-point distance and the point-to-plane distance. The weighted sum D of these two distances is used for comparison with a pre-defined threshold 0.01. If the distance is less than or equal to the threshold, the point belongs to the existing part, otherwise it belongs to the missing part.

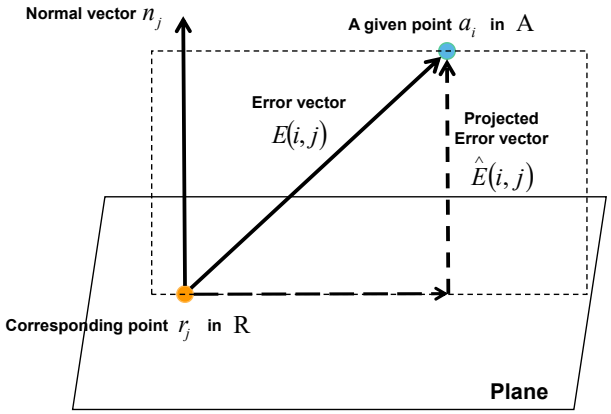


Figure 2. Point-to-point and point-to-plane distance. We use vector calculations to get the two distances. a_i is a point in point cloud $\mathbf{A}$ and r_j is the corresponding point in $\mathbf{R}$. n_j is the normal vector of point r_j. The error vector $E(i,j)$ is computed by connecting point r_j to point a_i. The projected error vector $\hat{E}(i,j)$ is the new error vector after $E(i,j)$ is projected along the normal vector n_j.

$$a_i \in \begin{cases} Existing\ part & if \quad D <= 0.01 \\ Missing\ part & else \end{cases}, \forall a_i \in \mathbf{A}\ . \tag{5}$$

The missing part is separated from the GT by the above method, and then downsampled to the same number as the incomplete input point for subsequent use. In Fig. 3, we show some results of missing part extractor.

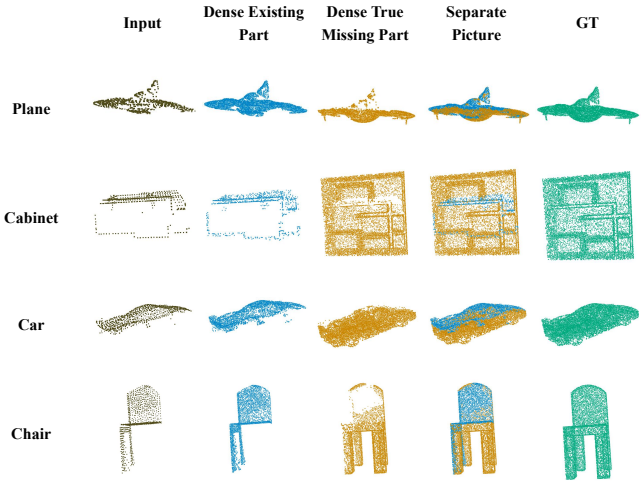


Figure 3. Some results of missing part extractor. From left to right are input, dense existing part, dense missing part, the splicing of the two and GT. From top to bottom are cases in four categories plane, cabinet, car and chair.

3. Implementation Details

In this section, we provide additional implementation details of the proposed *ProxyFormer*.

3.1. Basic experimental setup

We use Pytorch [2] for our implementation. All network are trained on a single NVIDIA GeForce RTX 3090 graphics card. AdamW optimizer [1] is uesd to train the network with initial learning rate as $5e-4$ and weight decay as $5e-4$. When training on the PCN dataset, we set the batch size to 64 and train the model for 400 epochs with the continuous learning rate decay of 0.95 for every 20 epochs. When training on ShapeNet-55/34, we set the batch size to 128 and train the model for 300 epochs with the continuous learning rate decay of 0.83 for every 20 epochs.

3.2. Four types of proxies

There are four types of proxies during training, existing proxies (*EP*), missing proxies (*MP*), true missing proxies (*true-MP*), predicted missing proxies (*pre-MP*). *true-MP* is not available during testing. When training on the PCN dataset [9], *EP* is a 128×384 matrix and *MP*, *true-MP* and *pre-MP* are both 224×384 matrices. When training on the ShapeNet-55/34 [8], *EP* is a 128×384 matrix and *MP*, *true-MP* and *pre-MP* are both 96×384 matrices.

3.3. Feature and Position Extractor

Feature Extraction. We use farthest point sampling [3] (FPS) to downsample the point cloud and use point transformer [10] (PT) to help exchange information between localized feature vectors. First we use an shared MLP to upscale the 3D coordinates of the point cloud to 32-dim as features. Suppose the set of input point cloud is P_i, at each downsampling stage: (1) perform farthest point sampling in P_i and form a new set P_o ($P_o \subset P_i$); (2) a kNN graph is performed on the P_i (we use $k = 16$ in our experiments); (3) local max pooling is used to aggregate the features of nearby k points to the current center point; (4) enhance features with point transformer block.

On both PCN and ShapeNet-55/34 datasets, the number of incomplete point cloud points input is $P = 2048$. Detailed network architecture is as follows:

incomplete input (2048×3) $\rightarrow$ shared-MLP (2048×3 to 2048×32) $\rightarrow$ FPS (2048×32 to 512×128) $\rightarrow$ PT (512×128 to 512×128) $\rightarrow$ FPS (512×128 to 128×384) $\rightarrow$ PT (128×384 to 128×384).

On PCN (**ShapeNet-55/34**) dataset, the number of missing part points input is $P = 3584$(**1536**). Bold numbers correspond to the settings on ShapNet-55/34. The process of FAPE is:

missing part (3584(**1536**) $\times$ 3) $\rightarrow$ shared-MLP (3584(**1536**) $\times$ 3 to 3584(**1536**) $\times$ 32) $\rightarrow$ FPS (3584(**1536**)$\times 32$ to 896(**384**)$\times 128$) $\rightarrow$ PT (896(**384**)$\times 128$ to 896(**384**)$\times 128$) $\rightarrow$ FPS (896(**384**)$\times 128$ to 224(**96**)$\times 384$) $\rightarrow$ PT (224(**96**) $\times$ 384 to 224(**96**) $\times$ 384).

Position Extraction. In order to keep the final position dimension consistent with the feature dimension extracted in the previous step, we set both C_2 and C_{out} in the main text to 384. Suppose the set of center points is P_c and the final feature of feature extraction is F_c. Perform k-neighbor (we use $k = 16$ in our experiments) subtraction (kNN-sub) and aggregation (kNN-agg) operations on P_c and F_c, respectively.

On both PCN and ShapeNet-55/34 datasets, the number of incomplete center points input is $n = 128$. Detailed network architecture is as follows:

P_c (128×3) $\rightarrow$ kNN-sub (128×3 to $128 \times 16 \times 3$) $\rightarrow$ kNN-agg ($128 \times 16 \times 3$ to 128×3) ,

F_c (128×384) $\rightarrow$ kNN-sub (128×384 to $128 \times 16 \times 384$) $\rightarrow$ kNN-agg ($128 \times 16 \times 384$ to 128×384) ,

[P_c, F_c] (128×387) $\rightarrow$ shared-MLP (128×387 to 128×384) .

On PCN (**ShapeNet-55/34**) dataset, the number of missing part center points input is $n = 224$(**96**). Bold numbers correspond to the settings on ShapNet-55/34. Detailed network architecture is as follows:

P_c (224(**96**) $\times$ 3) $\rightarrow$ kNN-sub (224(**96**) $\times$ 3 to 224(**96**) $\times$ 16×3) $\rightarrow$ kNN-agg (224(**96**) $\times 16 \times 3$ to 224(**96**) $\times$ 3) ,

F_c (224(**96**) $\times$ 384) $\rightarrow$ kNN-sub (224(**96**) $\times$ 384 to 224(**96**) $\times 16 \times 384$) $\rightarrow$ kNN-agg (224(**96**) $\times 16 \times 384$ to 224(**96**) $\times$ 384) ,

[P_c, F_c] (224(**96**) $\times$ 387) $\rightarrow$ shared-MLP (224(**96**) $\times$ 387 to 128×384) .

Subsequent attention score calculations do not affect the dimension of this feature.

3.4. Missing Part Prediction

In the main text we have mentioned using the incomplete seed feature to generate coarse missing part. When training on PCN dataset, the predicted coarse missing part contains 224 points and the predicted dense missing part contains 14336 points. When training on ShapeNet-55/34, the predicted coarse missing part contains 96 points and the predicted dense missing part contains 6144 points.

3.5. Missing Feature Generator

In Mssing Feature Generator, we set N to 128 and M to 224 (on PCN dataset) or 96 (on ShapeNet-55/34). $C = 384$ and we divide the feature dimension equally into $U = 16$ groups.

3.6. Missing Part Sensitive Transformer

Like other tansformer-based methods, we stack the missing part sensitive transformer and set its depth to 8.

Multi-Head Self-Attention. This structural design allows each attention mechanism to map to different spaces through Query, Key and Value to learn features. In this way, the different feature parts of each proxy are optimized, so as to make the proxies contain more diverse representations. In all our experiments, we set the number of multi-head attention heads to 8.

Feed-Forward Network (FFN). Referring to [6] and [8], we set up the feed-forward network as two linear layers with ReLU activation function and dropout.

3.7. The calculation of DCD

Wu *et al.* [7] study the limitations of CD, believe that CD is not the optimal indicator for evaluating the visual quality of point cloud completion tasks, and proposed the density-aware chamfering distance (DCD), which can retain the measurement ability similar to CD and can also better judge the visual effect of the point cloud completion result. The formula for DCD is as follows:

$$d_{DCD}\left(S_1, S_2\right) = \frac{1}{2}\left(\frac{1}{|S_1|}\sum_{x\in S_1}\left(1-\frac{1}{n_{\hat{y}}}e^{-\alpha\|x-\hat{y}\|_2}\right)\right) + \frac{1}{2}\left(\frac{1}{|S_2|}\sum_{y\in S_2}\left(1-\frac{1}{n_{\hat{x}}}e^{-\alpha\|y-\hat{x}\|_2}\right)\right), \tag{6}$$

where $\hat{y} = min_{y\in S_2}\|x-y\|_2, \hat{x} = min_{y\in S_1}\|y-x\|_2$, and α denotes a temperature scalar, which we set to 1000, just as the original text describes.

4. Addtional Ablation Studies and Experimental Results

In this section, we first conduct more ablation experiments on some of the components used in *ProxyFormer*. Then, we present more qualitative and quantitative experimental results to further demonstrate the effectiveness of our method.

4.1. More ablation sutdies and analysis

Feature Extractor. We respectively replace the point transformer with (1) a multilayer perceptron (MLP) composed of ordinary convolutional layers; (2) lightweight DGCNN; (3) traditional transformer using scalar attention. From Table 1, we can know that the point transformer can better capture the features of point clouds in the network structure proposed in this paper.

Missing Feature Generator. Many methods use the tiled copy of global feature as the feature of each point to generate a complete point cloud. In this experiment, we make two attempts to the network after removing the missing feature generator: (1) use random feature as the feature of *MP*; (2) use the global feature (incomplete seed feature) as the feature of *MP*. From Table 2, we can see that the effect of using random feature and global feature is not ideal. Using Missing Feature Generator, a more reasonable feature corresponding to the missing point can be generated from the feature of the existing points. Furthermore, experiments show that in the process of feature generation, evenly dividing the features into 16 groups can reduce the training time while reducing the CD of the final result to the lowest.

Table 1. Ablation study on Feature Extractor of FAPE Module.

Method	CD-Avg
w/ MLP	8.08
w/ DGCNN	7.18
w/ Traditional Transformer (scalar attention)	7.53
w/ Point Transformer (vector attention)	**6.77**

Table 2. Ablation study of Missing Feature Generator

Methods	Attempts	CD-Avg
w/o Missing Feature Generator	random feature	7.90
	copy global feature	8.49
w/ Missing Feature Generator	num of feature patch = 1	6.83
	num of feature patch = 8	6.80
	num of feature patch =16	**6.77**
	num of feature patch = 32	6.85

Coarse missing part prediction. We try to use the predicted missing seed feature to generate coarse missing part and did ablation experiments with this. The quantitative results on PCN dataset [9] are listed in Tables 3 and 4. Through this experiment, it can be proved that the features extracted from the partial input can better predict the approximate position of the missing part (that is, the position of the center point of the coarse missing part). However, as analyzed in the main text, it is not enough to use only the features extracted from the partial input for the prediction of the missing details. Using Missing Feature Generator can better generates features that incorporate the missing details.

Model performance Analysis. In the main text, we have compared the complexity of our method with other methods on the PCN dataset, and here we show the evaluation time of each method on ShapeNet-55/34 (all methods are tested on the same device, *i.e.* NVIDIA GeForce RTX 3090 graphics card). The ShapeNet-55 dataset contains 10518 test models. The ShapeNet-34 contains 3400 models in 34 visible categories and 2305 models in 21 novel categories. Since 8 viewpoints are fixed for each model during the testing process, and the average of the results of the 8 viewpoints is used as the final CD value, the testing process is also time consuming. In Fig. 4, we show the comparison of *ProxyFormer* with other methods on ShapeNet-55/34 with

Table 3. Ablation study on feature used for generating predicted coarse missing part. For CD, lower is better.

Methods	Chamfer Distance(10^{-3})								
	Air	Cab	Car	Cha	Lam	Sof	Tab	Ves	Ave
ProxyFormer(using predicted missing seed feature)	4.13	9.22	8.01	7.57	5.60	9.11	6.43	6.28	7.04
ProxyFormer(using incomplete seed feature)	**4.01**	**9.01**	**7.88**	**7.11**	**5.35**	**8.77**	**6.03**	**5.98**	**6.77**

Table 4. Ablation study on feature used for generating predicted coarse missing part. For DCD, lower is better.

Methods	Density-aware Chamfer Distance								
	Air	Cab	Car	Cha	Lam	Sof	Tab	Ves	Ave
ProxyFormer(using predicted missing seed feature)	0.572	0.604	0.611	0.597	0.609	0.641	0.530	0.579	0.593
ProxyFormer(using incomplete seed feature)	**0.555**	**0.590**	**0.597**	**0.571**	**0.562**	**0.626**	**0.518**	**0.507**	**0.577**

evaluation time (Eval. time) *vs.* CD, DCD and F1-Score. For Eval. time *vs.* CD and DCD, the closer the value is to the origin of the coordinates, the better the model performance (that is, the smaller CD and DCD can be obtained while the inference speed is fast). For Eval. time *vs.* F1-Score, the closer the value is to the coordinates $(0, 1)$, the better the model performance (that is, the higher the F1-Score can be obtained while the inference speed is fast). From the pictures, we can observe that our proposed *ProxyFormer* has the fastest inference speed while achieving the smallest DCD and the highest F1-Score on ShapeNet-55/34.

4.2. More experimental results

Three-views drawings. In Figs. 5 and 6, we have drawn three views of *ProxyFormer* on the PCN dataset, and we can see that proxy alignment plays a significant role in the point cloud completion task.

More qualitative results on PCN dataset. We show more visualization results of *ProxyFormer* on PCN dataset in Fig. 7. From this, we can find that our method is more sensitive to the concentrated distribution of missing part, which makes it impossible to easily distinguish the shape of objects from the existing point clouds, such as the second cabinet and the first car. But our method can complete its shape well, and the final result is basically consistent with GT.

More qualitative results on ShapeNet-55/34. In Figure 8, we show the visualization results of point cloud completion on ShapNet-55 by PoinTr [8], SeedFormer [11], and *ProxyFormer*. We can intuitively see that *ProxyFormer* can better complete the point cloud completion task. For example, for the table in the first row, although the resulting shape of PoinTr is complete, it fails to generate the details of the bottom of the table well, and SeedFormer introduces some noise points in the completion process, which affects the final result. But our method is able to take both shape and detail into account. Another example is the car in the fourth row. The point cloud contains not only a car, but also a person. There is a problem with the results of PoinTr and SeedFormer completion, that is, the distribution of point clouds is uneven, while *ProxyFormer* can better perceive the distribution of missing points, thereby generating a more reasonable complete point cloud. In Fig. 9, we also show more visualization results of *ProxyFormer* on the ShapNet-55. To better demonstrate the point cloud completion capability of our method, for each object, we show two different parts missing. For example, for the second bird house, we show both cases where the bottom and top are missing, and our method can easily get complete point clouds with local details.

Detailed quantitative results on ShapeNet-55/34. We report complete results of our method on ShapeNet-55 in Tables 5 and 6 and results of novel categories on ShapeNet-34 in Tables 7 and 8. The models are tested under three difficulty levels: simple, moderate and hard. We can see that *ProxyFormer* achieves lowest DCD on almost all categories on the three settings.

References

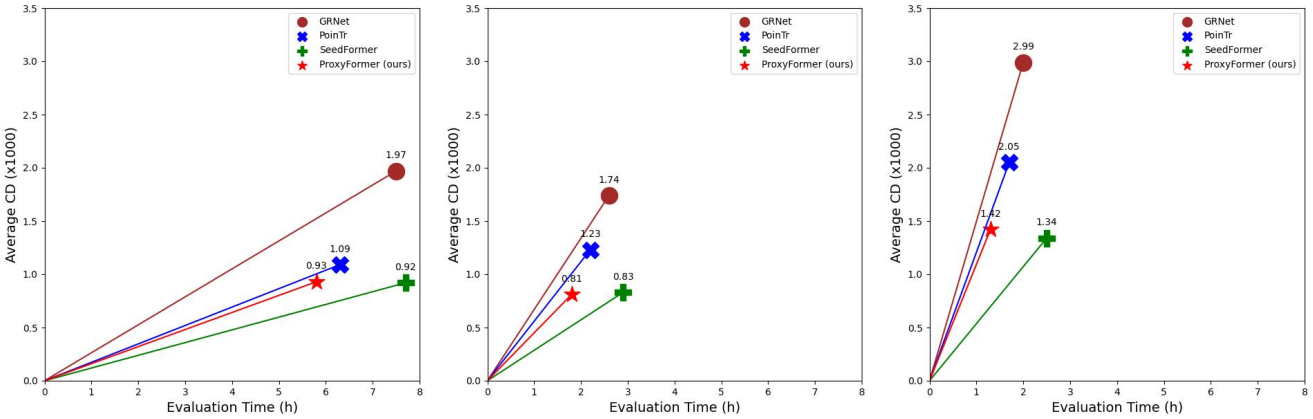
[1] Ilya Loshchilov and Frank Hutter. Fixing weight decay regularization in adam. 2018. 3

[2] Adam Paszke, Sam Gross, Francisco Massa, Adam Lerer, James Bradbury, Gregory Chanan, Trevor Killeen, Zeming Lin, Natalia Gimelshein, Luca Antiga, et al. Pytorch: An imperative style, high-performance deep learning library. *Advances in neural information processing systems*, 32, 2019. 3

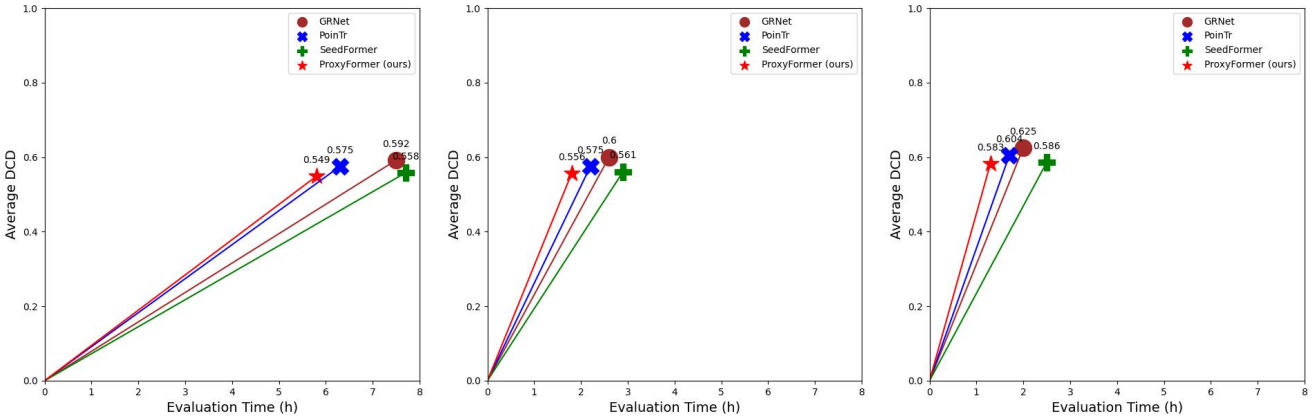
[3] Charles Ruizhongtai Qi, Li Yi, Hao Su, and Leonidas J Guibas. Pointnet++: Deep hierarchical feature learning on point sets in a metric space. *Advances in neural information processing systems*, 30, 2017. 3

[4] Dong Tian, Hideaki Ochimizu, Chen Feng, Robert Cohen, and Anthony Vetro. Geometric distortion metrics for point cloud compression. In *2017 IEEE International Conference on Image Processing (ICIP)*, pages 3460–3464. IEEE, 2017. 1

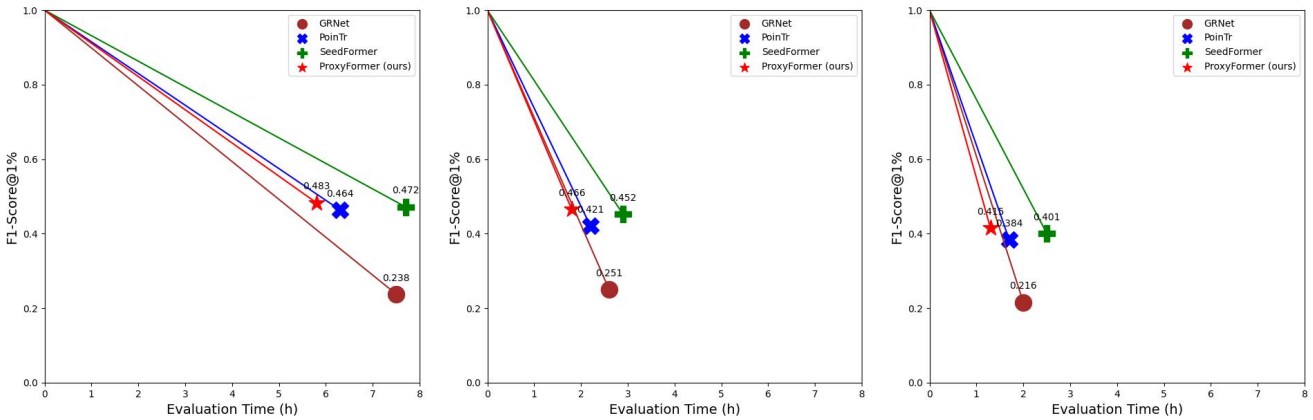
[5] Federico Tombari, Samuele Salti, and Luigi Di Stefano. Unique signatures of histograms for local surface description. In *European conference on computer vision*, pages 356–369. Springer, 2010. 1



(a) CD vs. Eval. time on ShapeNet-55. (b) CD vs. Eval. time on 34 seen categories. (c) CD vs. Eval. time on 21 novel categories.



(d) DCD vs. Eval. time on ShapeNet-55. (e) DCD vs. Eval. time on 34 seen categories. (f) DCD vs. Eval. time on 21 novel categories.



(g) F1 vs. Eval. time on ShapeNet-55. (h) F1 vs. Eval. time on 34 seen categories. (i) F1 vs. Eval. time on 21 novel categories.

Figure 4. Performance comparison. The first column represents the comparison results on ShapeNet-55. The second column represents the comparison results on 34 seen categories in ShapeNet-34, and the third column represents the comparison results on 21 novel categories in ShapeNet-34. The first row represents Eval. time *vs.* CD. The second row represents Eval. time *vs.* DCD. The third row represents Eval. time *vs.* F1-Score@1%. In order to observe the distance between the values and the coordinates (0, 0) or (0, 1) more clearly, we connect them, so the lines in the figure have no practical meaning and are only used for comparison.

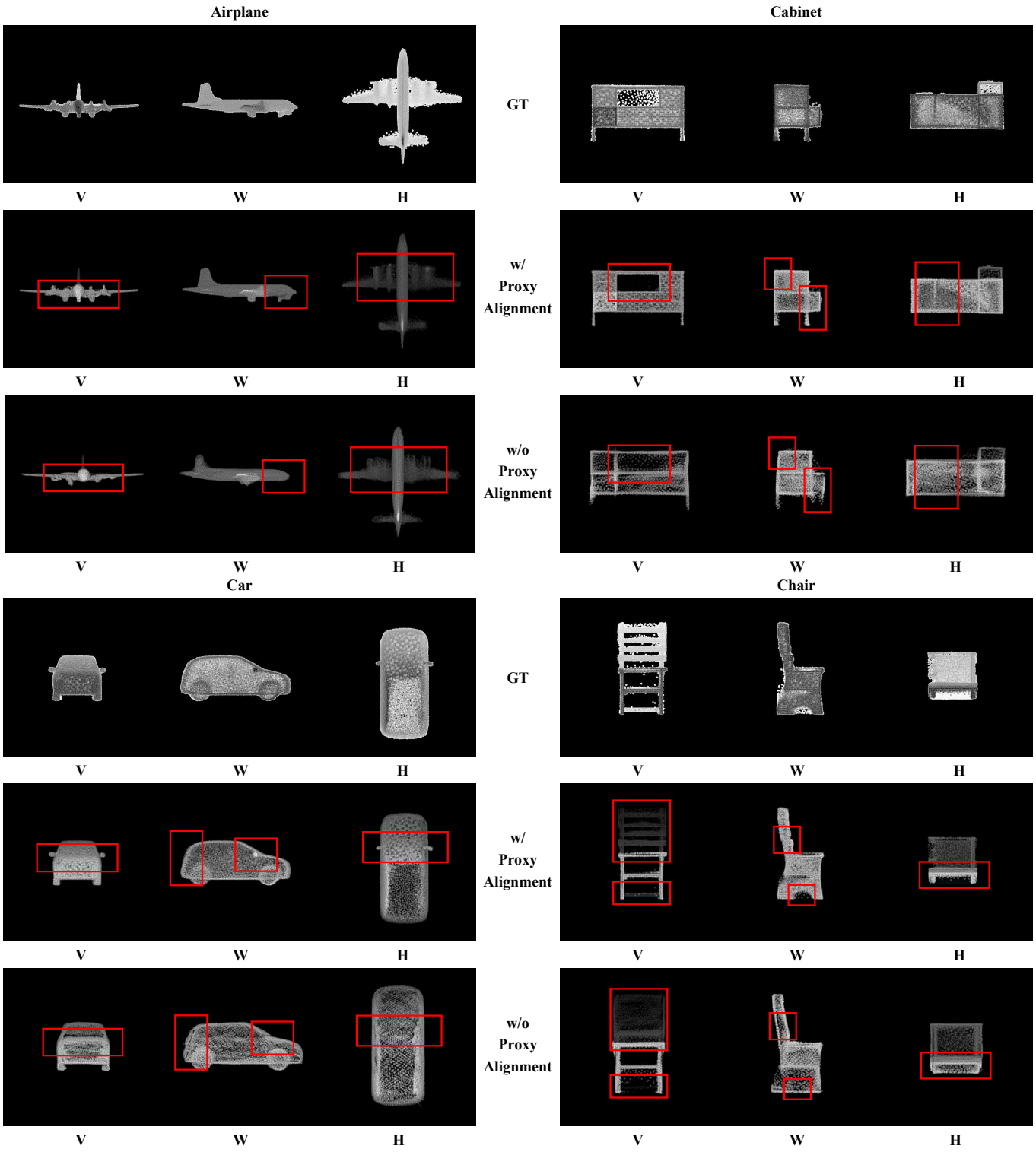


Figure 5. Three-view drawings of GT, results with proxy alignment and results without proxy alignment. Each figure represents vertical plane (V), width plane (W) and horizontal plane (H) from left to right. The result six categories of objects in the PCN are listed separately, and the red box indicates where Proxy Alignment contributes the most.

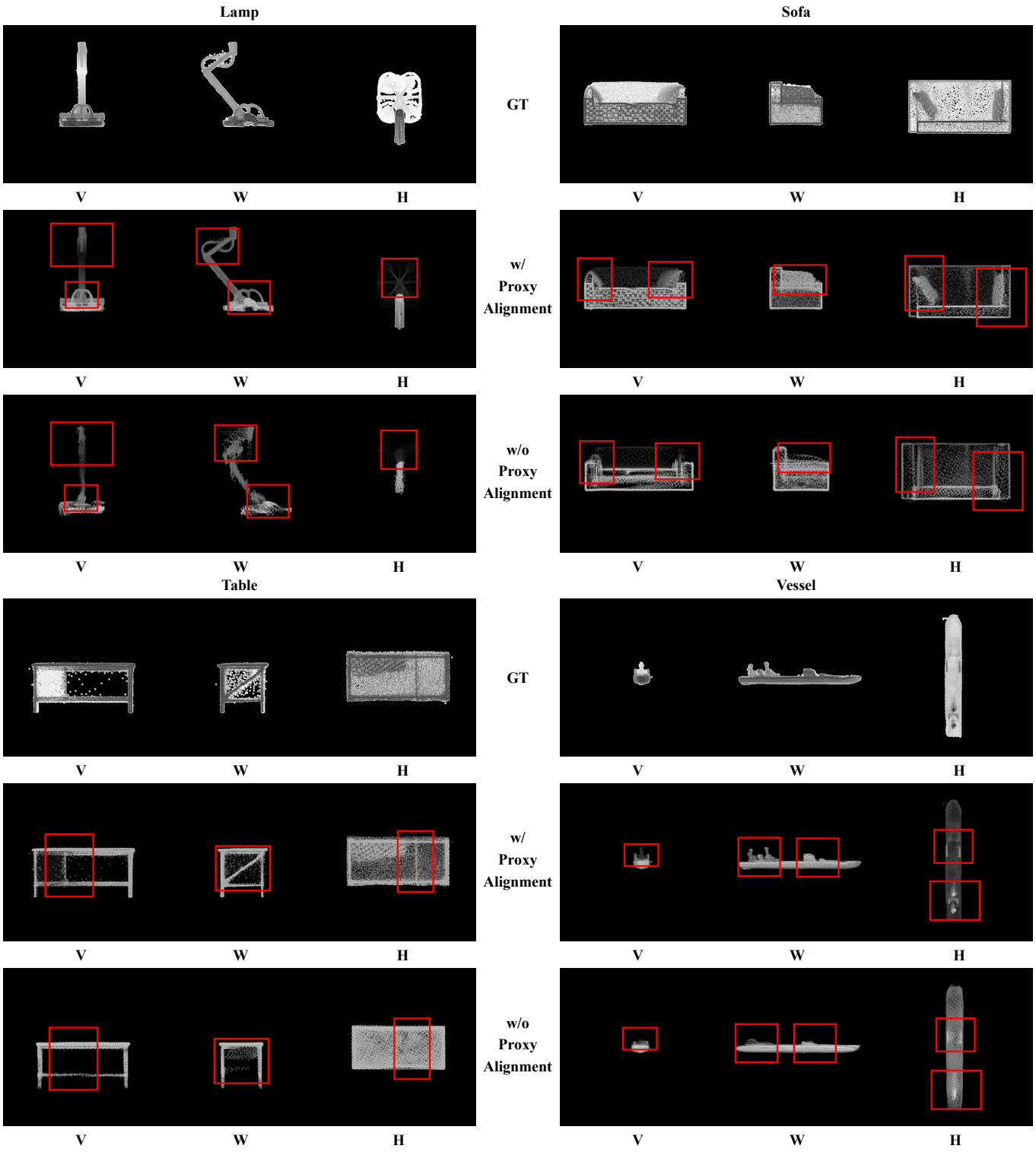


Figure 6. Three-view drawings of GT, results with proxy alignment and results without proxy alignment. Each figure represents vertical plane (V), width plane (W) and horizontal plane (H) from left to right. The result six categories of objects in the PCN are listed separately, and the red box indicates where Proxy Alignment contributes the most.

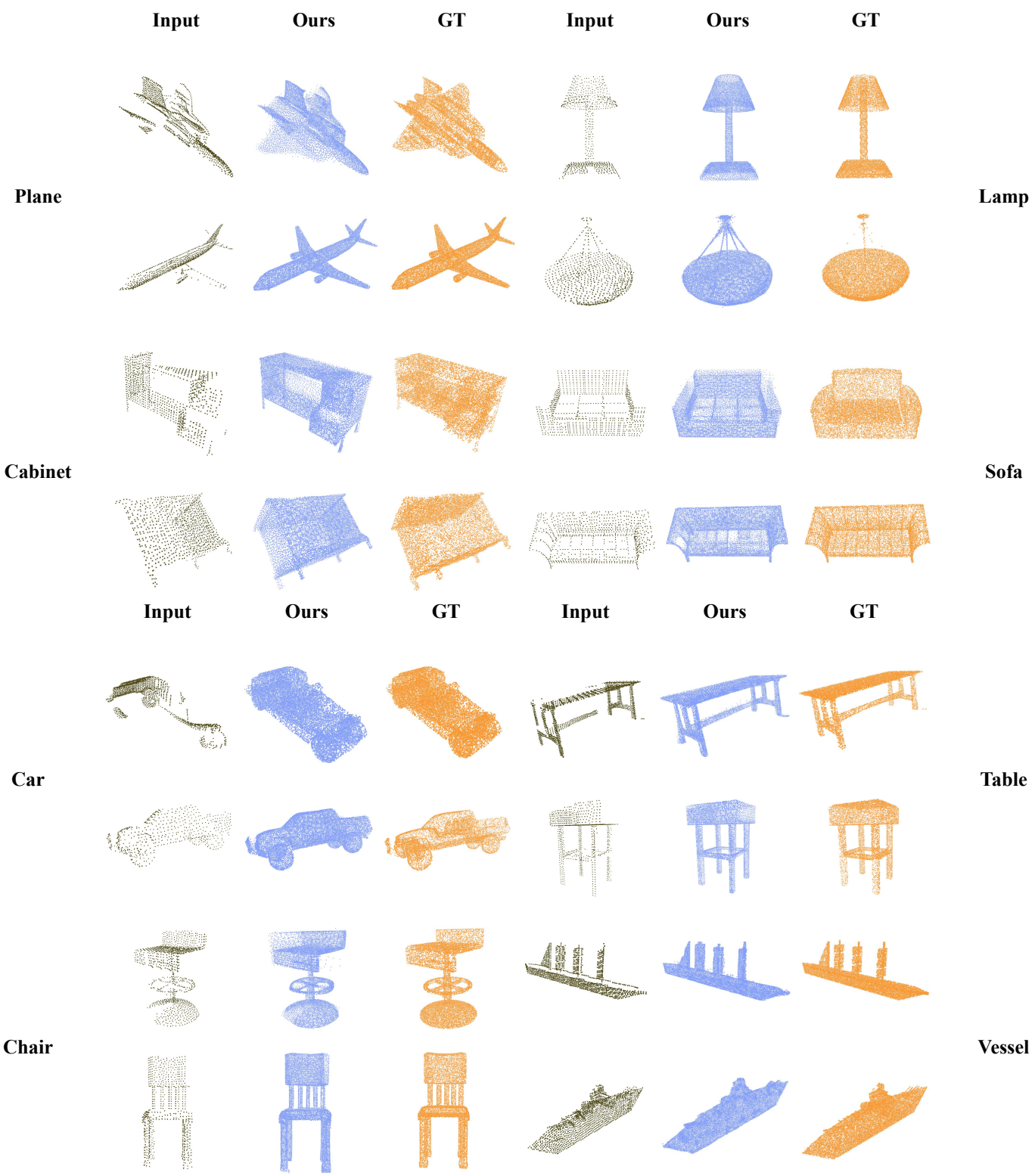


Figure 7. More visual completion results of *ProxyFormer* on PCN dataset. For each category in PCN, we show two results.

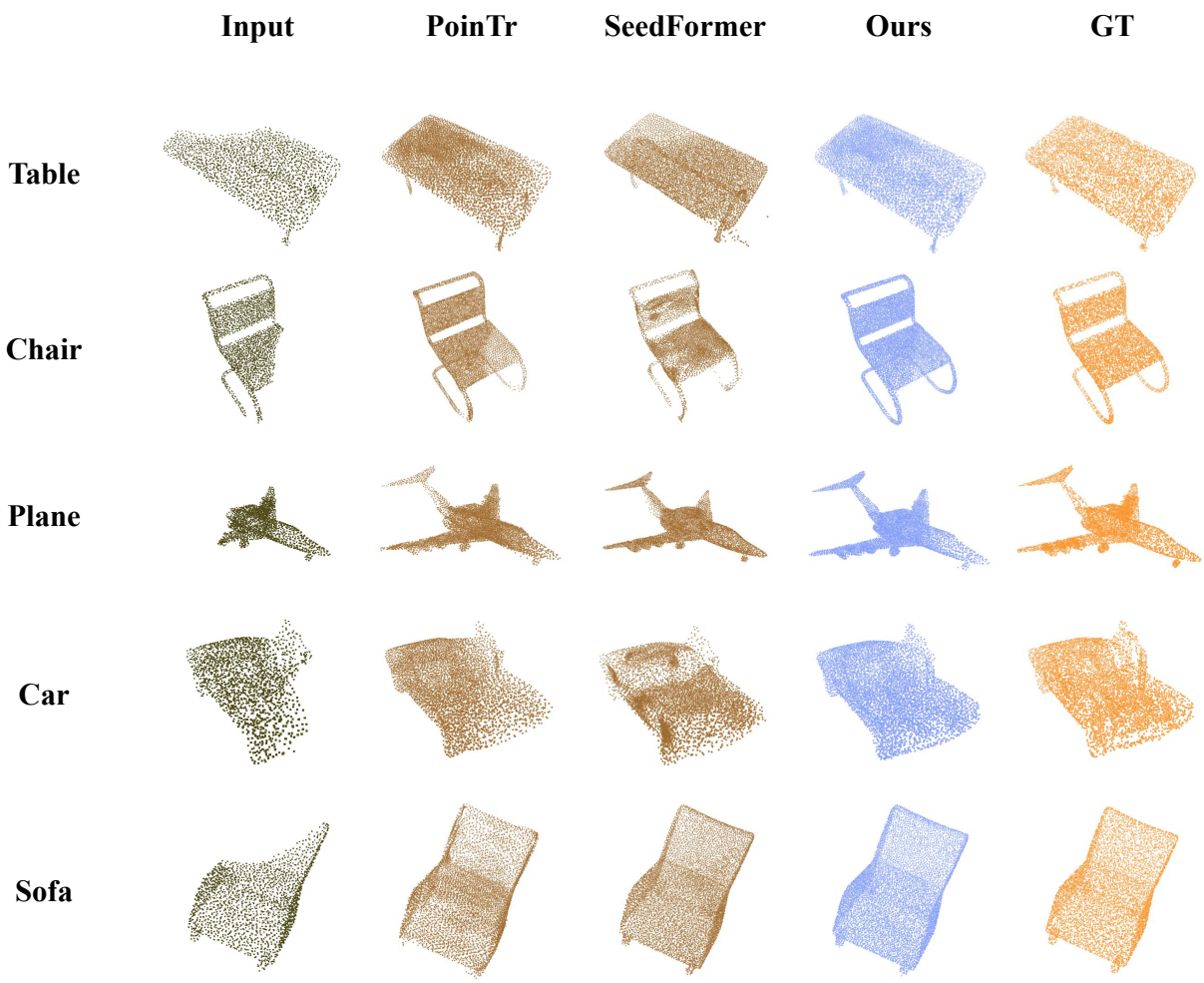


Figure 8. The visualization results of each method on ShapeNet-55, showing Table, Chair, Plane, Car and Sofa from top to bottom.

[6] Ashish Vaswani, Noam Shazeer, Niki Parmar, Jakob Uszkoreit, Llion Jones, Aidan N Gomez, Łukasz Kaiser, and Illia Polosukhin. Attention is all you need. *Advances in neural information processing systems*, 30, 2017. 4

[7] Tong Wu, Liang Pan, Junzhe Zhang, Tai Wang, Ziwei Liu, and Dahua Lin. Density-aware chamfer distance as a comprehensive metric for point cloud completion. *arXiv preprint arXiv:2111.12702*, 2021. 4

[8] Xumin Yu, Yongming Rao, Ziyi Wang, Zuyan Liu, Jiwen Lu, and Jie Zhou. Pointr: Diverse point cloud completion with geometry-aware transformers. In *Proceedings of the IEEE/CVF international conference on computer vision*, pages 12498–12507, 2021. 3, 4, 5

[9] Wentao Yuan, Tejas Khot, David Held, Christoph Mertz, and Martial Hebert. Pcn: Point completion network. In *2018 International Conference on 3D Vision (3DV)*, pages 728–737. IEEE, 2018. 3, 4

[10] Hengshuang Zhao, Li Jiang, Jiaya Jia, Philip HS Torr, and Vladlen Koltun. Point transformer. In *Proceedings of the IEEE/CVF International Conference on Computer Vision*, pages 16259–16268, 2021. 3

[11] Haoran Zhou, Yun Cao, Wenqing Chu, Junwei Zhu, Tong Lu, Ying Tai, and Chengjie Wang. Seedformer: Patch seeds based point cloud completion with upsample transformer. *arXiv preprint arXiv:2207.10315*, 2022. 5

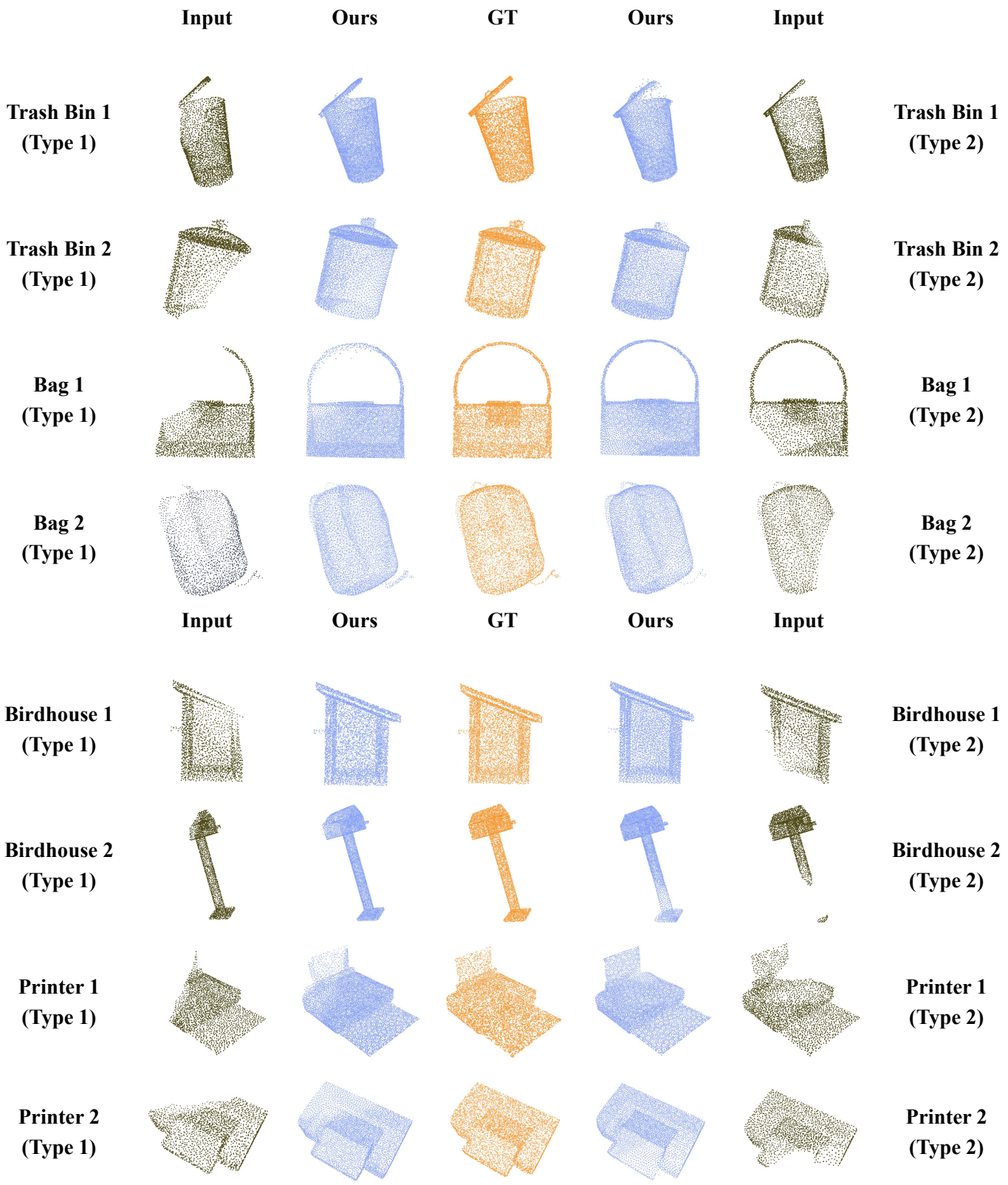


Figure 9. More visual completion results of *ProxyFormer* on ShapeNet-55. Each object contains two missing angles.

Table 5. Detailed results on the ShapeNet-55 dataset, including Simple (S), Moderate (M) and Hard (H) three difficulties. For CD-l_2 ($\times 1000$), lower is better.

	GRNet			PoinTr			SeedFormer			Ours		
	CD-S	CD-M	CD-H	CD-S	CD-M	CD-H	CD-S	CD-M	CD-H	CD-S	CD-M	CD-H
airplane	0.87	0.87	1.27	0.27	0.38	0.69	0.23	0.35	0.61	**0.21**	**0.28**	**0.54**
trash bin	1.69	2.01	3.48	0.80	1.15	2.15	0.73	1.08	**1.94**	**0.70**	**1.04**	1.98
bag	1.41	1.70	2.97	0.53	0.74	1.51	0.43	0.67	**1.28**	**0.39**	**0.65**	1.34
basket	1.65	1.84	3.15	0.73	0.88	1.82	0.65	0.83	1.54	**0.64**	**0.74**	**1.40**
bathtub	1.46	1.73	2.73	0.64	0.94	1.68	**0.52**	**0.82**	**1.45**	**0.52**	0.85	1.47
bed	1.64	2.03	3.70	0.76	1.10	2.26	0.63	**0.91**	**1.89**	**0.62**	**0.91**	2.04
bench	1.03	1.09	1.71	0.38	0.52	0.94	0.32	0.42	0.84	**0.30**	**0.39**	**0.80**
birdhouse	1.87	2.40	4.71	0.98	1.49	3.13	**0.76**	1.30	**2.46**	0.83	**1.22**	2.67
bookshelf	1.42	1.71	2.78	0.71	1.06	1.93	0.57	**0.84**	**1.57**	**0.55**	0.92	1.73
bottle	1.05	1.44	2.67	0.37	0.74	1.50	**0.31**	0.63	**1.21**	**0.31**	**0.60**	1.27
bowl	1.60	1.77	2.99	0.68	0.78	1.44	0.56	0.65	1.18	**0.54**	**0.62**	**1.14**
bus	1.06	1.16	1.48	0.42	0.55	0.79	0.42	0.55	0.73	**0.40**	**0.46**	**0.59**
cabinet	1.27	1.41	2.09	0.55	0.66	1.16	**0.57**	0.69	1.05	0.64	**0.59**	**1.00**
camera	2.14	3.15	6.09	1.10	2.03	4.34	0.83	**1.68**	**3.45**	**0.82**	1.77	4.04
can	1.58	2.11	3.81	0.68	1.19	2.14	0.58	1.03	**1.79**	**0.55**	**1.02**	1.86
cap	1.17	1.37	3.05	0.46	0.62	1.64	0.33	**0.45**	**1.18**	**0.31**	0.51	1.28
car	1.29	1.48	2.14	0.64	0.86	1.25	0.65	0.86	1.17	**0.61**	**0.69**	**1.03**
cellphone	0.82	0.91	1.18	0.32	0.39	0.60	0.31	0.40	0.54	**0.26**	**0.38**	**0.45**
chair	1.24	1.56	2.73	0.49	0.74	1.63	0.41	0.65	**1.38**	**0.40**	**0.61**	1.48
clock	1.46	1.66	2.67	0.62	0.84	1.65	0.53	0.74	**1.35**	**0.50**	**0.67**	1.45
keyboard	0.74	0.81	1.09	0.30	0.39	0.45	0.28	0.36	0.45	**0.27**	**0.33**	**0.43**
dishwasher	1.43	1.59	2.53	**0.55**	0.69	1.42	0.56	0.69	1.30	0.57	**0.61**	**1.21**
display	1.13	1.38	2.29	0.48	0.67	1.33	**0.39**	**0.59**	**1.10**	0.46	0.62	1.18
earphone	1.78	2.18	5.33	0.81	1.38	3.78	0.64	**1.04**	**2.75**	**0.62**	1.12	3.36
faucet	1.81	2.32	4.91	0.71	1.42	3.49	0.55	**1.15**	**2.63**	**0.49**	1.16	3.01
filecabinet	1.46	1.71	2.89	0.63	0.84	1.69	**0.63**	0.84	1.49	0.74	**0.79**	**1.45**
guitar	0.44	0.48	0.76	0.14	0.21	0.42	**0.13**	**0.19**	0.32	0.14	0.20	**0.28**
helmet	2.33	3.18	6.03	0.99	1.93	4.22	0.79	**1.52**	**3.61**	**0.76**	1.60	3.91
jar	1.72	2.37	4.37	0.77	1.33	2.87	0.63	**1.13**	**2.36**	**0.62**	**1.13**	2.69
knife	0.72	0.66	0.96	0.20	0.33	0.56	**0.15**	0.28	0.45	**0.15**	**0.21**	**0.43**
lamp	1.68	2.43	5.17	0.64	1.40	3.58	0.45	1.06	**2.67**	**0.42**	**1.05**	3.03
laptop	0.83	0.87	1.28	0.32	0.34	0.60	0.32	0.37	0.55	**0.30**	**0.31**	**0.42**
loudspeaker	1.75	2.08	3.45	0.78	1.16	2.17	0.67	**1.01**	**1.80**	**0.66**	1.03	1.89
mailbox	1.15	1.59	3.42	0.39	0.78	2.56	**0.30**	0.67	**2.04**	**0.30**	**0.65**	2.17
microphone	2.09	2.76	5.70	0.70	1.66	4.48	0.62	1.61	**3.66**	**0.59**	**1.58**	3.98
microwaves	1.51	1.72	2.76	0.67	0.83	1.82	0.63	0.79	1.47	**0.61**	**0.69**	**1.35**
motorbike	1.38	1.52	2.26	0.75	1.10	1.92	0.68	**0.96**	**1.44**	**0.67**	**0.96**	1.49
mug	1.75	2.16	3.79	0.91	1.17	2.35	0.79	1.03	2.06	**0.75**	**0.92**	**2.00**
piano	1.53	1.82	3.21	0.76	1.06	2.23	0.62	0.87	1.79	**0.55**	**0.85**	**1.73**
pillow	1.42	1.67	3.04	0.61	0.82	1.56	**0.48**	0.75	1.41	0.49	**0.71**	**1.35**
pistol	1.11	1.06	1.76	0.43	0.66	1.30	0.37	0.56	0.96	**0.34**	**0.52**	**0.90**
flowerpot	2.02	2.48	4.19	1.01	1.51	2.77	0.93	**1.30**	**2.32**	**0.89**	1.39	2.52
printer	1.56	2.38	4.24	0.73	1.21	2.47	0.58	1.11	2.13	**0.53**	**1.09**	**2.08**
remote	0.89	1.05	1.29	0.36	0.53	0.71	0.29	0.46	0.62	**0.20**	**0.33**	**0.54**
rifle	0.83	0.77	1.16	0.30	0.45	0.79	0.27	0.41	0.66	**0.21**	**0.32**	**0.50**
rocket	0.78	0.92	1.44	0.23	0.48	0.99	0.21	0.46	0.83	**0.21**	**0.38**	**0.80**
skateboard	0.82	0.87	1.24	0.28	0.38	0.62	0.23	0.32	0.62	**0.19**	**0.28**	**0.56**
sofa	1.35	1.45	2.32	0.56	0.67	1.14	0.50	0.62	1.02	**0.49**	**0.57**	**1.01**
stove	1.46	1.72	3.22	0.63	0.92	1.73	**0.59**	**0.87**	**1.49**	0.68	0.88	1.67
table	1.15	1.33	2.33	0.46	0.64	1.31	0.41	0.58	1.18	**0.39**	**0.48**	**1.06**
telephone	0.81	0.89	1.18	0.31	0.38	0.59	0.31	0.39	0.55	**0.28**	**0.32**	**0.48**
tower	1.26	1.69	3.06	0.55	0.90	1.95	0.47	0.84	**1.65**	**0.46**	**0.82**	1.67
train	1.09	1.14	1.61	0.50	0.70	1.12	0.51	0.66	1.01	**0.49**	**0.61**	**0.97**
watercraft	1.09	1.12	1.65	0.41	0.62	1.07	**0.35**	**0.56**	**0.92**	0.44	0.62	1.04
washer	1.72	2.05	4.19	0.75	1.06	2.44	0.64	**0.91**	**2.04**	**0.63**	0.94	2.26
mean	1.35	1.63	2.86	0.58	0.88	1.80	0.50	0.77	**1.49**	**0.49**	**0.75**	1.55

Table 6. Detailed results on the ShapeNet-55 dataset, including Simple (S), Moderate (M) and Hard (H) three difficulties, For DCD, lower is better.

	GRNet			PoinTr			SeedFormer			Ours		
	DCD-S	DCD-M	DCD-H	DCD-S	DCD-M	DCD-H	DCD-S	DCD-M	DCD-H	DCD-S	DCD-M	DCD-H
airplane	0.520	0.559	0.614	0.487	0.524	0.608	0.478	0.505	0.574	**0.475**	**0.496**	**0.565**
trash bin	0.586	0.616	0.705	0.557	0.599	0.679	0.547	0.588	**0.648**	**0.544**	**0.578**	0.653
bag	0.518	0.563	0.653	0.510	0.549	0.628	0.503	0.546	0.590	**0.483**	**0.515**	**0.588**
basket	0.559	0.593	0.648	0.533	0.573	0.630	0.530	0.558	**0.609**	**0.522**	**0.548**	0.616
bathtub	0.503	0.553	0.646	0.499	0.544	0.619	0.498	0.523	0.609	**0.492**	**0.519**	**0.589**
bed	0.553	0.620	0.694	0.544	0.589	0.661	0.537	0.583	0.637	**0.528**	**0.554**	**0.622**
bench	0.528	0.546	0.607	0.507	0.531	0.589	0.506	0.504	0.543	**0.483**	**0.494**	**0.539**
birdhouse	0.576	0.617	0.722	0.562	0.597	0.697	0.554	0.585	**0.650**	**0.538**	**0.579**	0.658
bookshelf	0.563	0.589	0.678	0.541	0.569	0.655	0.523	0.562	**0.623**	**0.519**	**0.554**	0.631
bottle	0.505	0.546	0.635	0.485	0.533	0.628	0.458	0.516	0.606	**0.455**	**0.508**	**0.605**
bowl	0.545	0.591	0.676	0.531	0.560	0.641	0.524	0.529	0.613	**0.507**	**0.529**	**0.602**
bus	0.533	0.559	0.606	0.525	0.547	0.599	0.521	0.525	0.584	**0.499**	**0.523**	**0.570**
cabinet	0.590	0.593	0.643	0.557	0.577	0.626	0.532	0.548	**0.595**	**0.521**	**0.544**	0.596
camera	0.583	0.647	0.728	0.558	0.615	0.701	0.547	0.608	**0.670**	**0.535**	**0.584**	0.671
can	0.549	0.599	0.651	0.527	0.571	0.649	0.511	0.550	0.621	**0.500**	**0.542**	**0.616**
cap	0.528	0.544	0.640	0.495	0.541	0.638	0.491	0.539	**0.609**	**0.474**	**0.510**	0.612
car	0.614	0.660	0.683	0.581	0.631	0.673	0.566	0.607	0.661	**0.565**	**0.592**	**0.644**
cellphone	0.508	0.509	0.568	0.483	0.506	0.541	0.482	0.497	**0.520**	**0.458**	**0.480**	0.521
chair	0.526	0.559	0.623	0.505	0.555	0.620	0.501	0.527	0.609	**0.488**	**0.523**	**0.606**
clock	0.543	0.578	0.657	0.528	0.561	0.634	0.515	0.554	**0.594**	**0.502**	**0.531**	0.595
keyboard	0.515	0.528	0.553	0.499	0.511	0.541	0.477	0.498	**0.510**	**0.473**	**0.483**	0.515
dishwasher	0.566	0.600	0.643	0.540	0.568	0.629	0.514	0.536	**0.593**	**0.508**	**0.535**	0.596
display	0.532	0.544	0.595	0.517	0.544	0.587	0.483	0.519	**0.554**	**0.483**	**0.508**	0.560
earphone	0.579	0.609	0.708	0.555	0.605	0.694	0.539	0.603	0.669	**0.529**	**0.571**	**0.675**
faucet	0.512	0.602	0.681	0.502	0.573	0.687	0.495	0.565	**0.663**	**0.483**	**0.548**	0.665
filecabinet	0.580	0.615	0.662	0.559	0.579	0.644	0.529	0.562	0.619	**0.529**	**0.556**	**0.616**
guitar	0.499	0.538	0.616	0.488	0.514	0.602	0.459	0.495	**0.571**	**0.456**	**0.493**	0.578
helmet	0.591	0.613	0.699	0.567	0.608	0.695	0.551	0.597	**0.668**	**0.538**	**0.586**	0.676
jar	0.574	0.598	0.679	0.541	0.590	0.669	0.516	0.567	**0.650**	**0.512**	**0.565**	0.660
knife	0.486	0.571	0.651	0.470	0.544	0.643	0.464	0.533	**0.606**	**0.461**	**0.518**	0.613
lamp	0.526	0.587	0.717	0.520	0.581	0.689	0.510	0.578	0.677	**0.488**	**0.553**	**0.657**
laptop	0.501	0.538	0.558	0.492	0.510	0.557	0.488	0.506	0.548	**0.473**	**0.488**	**0.532**
loudspeaker	0.561	0.589	0.669	0.548	0.583	0.649	0.544	0.566	0.622	**0.526**	**0.556**	**0.620**
mailbox	0.499	0.543	0.674	0.477	0.540	0.667	0.465	0.520	**0.620**	**0.458**	**0.518**	0.628
microphone	0.546	0.622	0.709	0.511	0.587	0.701	0.503	0.579	0.681	**0.490**	**0.560**	**0.674**
microwaves	0.559	0.580	0.634	0.546	0.566	0.621	0.541	0.544	0.614	**0.522**	**0.544**	**0.597**
motorbike	0.653	0.690	0.714	0.623	0.657	0.709	0.614	0.652	**0.687**	**0.596**	**0.627**	0.695
mug	0.582	0.618	0.706	0.560	0.588	0.678	0.536	0.585	0.657	**0.535**	**0.569**	**0.644**
piano	0.550	0.585	0.658	0.538	0.573	0.632	0.537	0.552	0.611	**0.519**	**0.544**	**0.602**
pillow	0.495	0.529	0.665	0.494	0.529	0.633	0.485	0.518	0.613	**0.474**	**0.510**	**0.595**
pistol	0.557	0.586	0.668	0.529	0.571	0.655	0.509	0.570	**0.620**	**0.506**	**0.545**	0.627
flowerpot	0.608	0.626	0.701	0.596	0.622	0.697	0.591	0.609	**0.667**	**0.570**	**0.601**	0.671
printer	0.560	0.595	0.648	0.543	0.578	0.640	0.529	0.561	0.619	**0.513**	**0.544**	**0.617**
remote	0.510	0.538	0.564	0.479	0.506	0.559	0.468	0.483	0.538	**0.454**	**0.483**	**0.536**
rifle	0.534	0.572	0.626	0.519	0.562	0.646	0.500	0.551	0.632	**0.497**	**0.539**	**0.616**
rocket	0.512	0.567	0.670	0.511	0.553	0.650	0.503	0.546	**0.610**	**0.484**	**0.524**	0.616
skateboard	0.517	0.525	0.586	0.484	0.507	0.589	0.461	0.503	0.580	**0.461**	**0.483**	**0.566**
sofa	0.554	0.588	0.603	0.532	0.559	0.596	0.525	0.541	**0.563**	**0.507**	**0.523**	0.567
stove	0.562	0.577	0.650	0.528	0.554	0.634	0.521	0.550	**0.601**	**0.516**	**0.546**	0.606
table	0.513	0.529	0.588	0.499	0.529	0.579	0.491	0.517	0.576	**0.481**	**0.502**	**0.553**
telephone	0.506	0.532	0.564	0.483	0.498	0.554	0.467	0.495	0.529	**0.457**	**0.476**	**0.516**
tower	0.571	0.622	0.676	0.536	0.593	0.684	0.513	0.567	0.659	**0.512**	**0.562**	**0.649**
train	0.530	0.589	0.660	0.529	0.565	0.629	0.520	0.563	0.618	**0.518**	**0.541**	**0.599**
watercraft	0.525	0.551	0.652	0.507	0.547	0.641	0.496	0.524	0.605	**0.488**	**0.524**	**0.599**
washer	0.553	0.591	0.649	0.540	0.570	0.647	0.533	0.567	0.621	**0.522**	**0.551**	**0.612**
mean	0.545	0.581	0.650	0.525	0.562	0.637	0.513	0.549	0.612	**0.502**	**0.536**	**0.608**

Table 7. Detailed results on the 21 unseen categories of ShapeNet-34 dataset, including Simple (S), Moderate (M) and Hard (H) three difficulties. For CD-l_2 ($\times$1000), lower is better.

	GRNet			PoinTr			SeedFormer			Ours		
	CD-S	CD-M	CD-H	CD-S	CD-M	CD-H	CD-S	CD-M	CD-H	CD-S	CD-M	CD-H
bag	1.47	1.88	3.45	0.96	1.34	2.08	0.49	0.82	1.45	**0.48**	**0.80**	**1.43**
basket	1.78	1.94	4.18	1.04	1.40	2.90	0.60	**0.85**	**1.98**	**0.57**	0.91	2.08
birdhouse	1.89	2.34	5.16	1.22	1.79	3.45	0.72	1.19	2.31	**0.51**	**1.03**	**2.00**
bowl	1.77	1.97	3.90	1.05	1.32	2.40	**0.60**	**0.77**	**1.50**	0.63	0.96	1.79
camera	2.31	3.38	7.20	1.63	2.67	4.97	0.89	**1.77**	**3.75**	**0.88**	1.94	4.04
can	1.53	1.80	3.08	0.80	1.17	2.85	0.56	0.89	**1.57**	**0.50**	**0.64**	1.73
cap	3.29	4.87	13.02	1.40	2.74	8.35	**0.50**	**1.34**	**5.19**	0.51	1.57	6.38
keyboard	0.73	0.77	1.11	0.43	0.45	0.63	0.32	0.41	0.60	**0.29**	**0.34**	**0.59**
dishwasher	1.79	1.70	3.27	0.93	1.05	2.04	**0.63**	**0.78**	**1.44**	0.72	0.90	1.49
earphone	4.29	4.16	10.30	2.03	5.10	10.69	1.18	**2.78**	**6.71**	**1.09**	2.96	8.98
helmet	3.06	4.38	10.27	1.86	3.30	6.96	**1.10**	**2.27**	**4.78**	1.32	2.44	5.74
mailbox	1.52	1.90	4.33	1.03	1.47	3.34	**0.56**	**0.99**	**2.06**	0.74	1.09	2.14
microphone	2.29	3.23	8.41	1.25	2.27	5.47	0.80	**1.61**	4.21	**0.73**	1.73	**3.70**
microwaves	1.74	1.81	3.82	1.01	1.18	2.14	0.64	**0.83**	1.69	**0.60**	0.90	**1.58**
pillow	1.43	1.69	3.43	0.92	1.24	2.39	**0.43**	**0.66**	1.45	**0.43**	0.73	**1.07**
printer	1.82	2.41	5.09	1.18	1.76	3.10	0.69	**1.25**	**2.33**	**0.59**	1.40	2.56
remote	0.82	1.02	1.29	0.44	0.58	0.78	**0.27**	**0.42**	**0.61**	0.29	0.51	0.67
rocket	0.97	0.79	1.60	0.39	0.72	1.39	0.28	0.51	1.02	**0.26**	**0.46**	**0.82**
skateboard	0.93	1.07	1.83	0.52	0.80	1.31	**0.35**	**0.56**	0.92	**0.35**	0.61	**0.78**
tower	1.35	1.80	3.85	0.82	1.35	2.48	**0.51**	0.92	1.87	0.59	**0.83**	**1.79**
washer	1.83	1.97	5.28	1.04	1.39	2.73	**0.61**	**0.87**	1.94	0.62	0.89	**1.90**
mean	1.84	2.23	4.95	1.05	1.67	3.45	0.61	**1.07**	**2.35**	**0.60**	1.13	2.54

Table 8. Detailed results on the 21 unseen categories of ShapeNet-34 dataset, including Simple (S), Moderate (M) and Hard (H) three difficulties. For DCD, lower is better.

	GRNet			PoinTr			SeedFormer			Ours		
	DCD-S	DCD-M	DCD-H	DCD-S	DCD-M	DCD-H	DCD-S	DCD-M	DCD-H	DCD-S	DCD-M	DCD-H
bag	0.521	0.597	0.617	0.506	0.573	0.593	0.489	**0.556**	**0.572**	**0.492**	**0.556**	0.573
basket	0.599	0.631	0.633	0.574	0.621	0.626	0.555	**0.605**	**0.613**	**0.548**	0.608	0.614
birdhouse	0.610	0.678	0.708	0.580	0.658	0.680	0.561	0.632	0.659	**0.554**	**0.615**	**0.641**
bowl	0.603	0.637	0.659	0.581	0.624	0.631	0.558	**0.595**	**0.614**	**0.553**	0.597	0.616
camera	0.624	0.640	0.708	0.594	0.629	0.689	**0.574**	**0.618**	0.666	0.587	0.619	**0.665**
can	0.570	0.628	0.637	0.550	0.617	0.582	0.525	0.587	**0.568**	**0.518**	**0.571**	0.569
cap	0.632	0.737	0.783	0.598	0.715	0.757	0.568	**0.689**	0.743	**0.560**	0.692	**0.742**
keyboard	0.484	0.513	0.535	0.461	0.501	0.519	0.450	0.484	0.505	**0.446**	**0.477**	**0.493**
dishwasher	0.597	0.631	0.701	0.564	0.609	0.681	**0.554**	**0.590**	0.658	0.566	0.591	**0.656**
earphone	0.695	0.687	0.803	0.666	0.671	0.773	**0.655**	**0.645**	**0.739**	0.660	0.648	0.751
helmet	0.669	0.730	0.781	0.639	0.717	0.761	**0.627**	0.706	**0.738**	0.632	**0.699**	**0.738**
mailbox	0.563	0.571	0.627	0.547	0.552	0.605	0.526	**0.536**	0.582	**0.518**	0.541	**0.581**
microphone	0.618	0.717	0.764	0.585	0.700	0.746	0.573	0.672	**0.732**	**0.566**	**0.666**	0.734
microwaves	0.609	0.622	0.660	0.575	0.604	0.634	0.557	**0.582**	0.623	**0.549**	0.584	**0.622**
pillow	0.595	0.620	0.626	0.575	0.606	0.603	0.562	**0.577**	0.578	**0.556**	0.578	**0.568**
printer	0.631	0.675	0.725	0.607	0.665	0.700	0.593	**0.650**	0.687	**0.588**	0.651	**0.683**
remote	0.496	0.516	0.542	0.480	0.494	0.526	0.462	0.480	0.508	**0.456**	**0.470**	**0.507**
rocket	0.499	0.510	0.560	0.476	0.501	0.542	0.457	0.488	0.518	**0.449**	**0.469**	**0.516**
skateboard	0.493	0.539	0.607	0.474	0.527	0.580	0.463	**0.511**	0.567	**0.456**	0.514	**0.566**
tower	0.545	0.602	0.685	0.526	0.590	0.667	0.500	0.563	**0.655**	**0.494**	**0.544**	0.656
washer	0.586	0.612	0.717	0.562	0.591	0.698	0.551	**0.566**	**0.686**	**0.544**	0.569	**0.686**
mean	0.583	0.623	0.670	0.558	0.608	0.647	0.541	0.587	0.629	**0.538**	**0.584**	**0.627**