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

Shanshan Li, Pan Gao*, Xiaoyang Tan, Mingqiang Wei
College of Computer Science and Technology, Nanjing University of Aeronautics and Astronautics
{markli, pan.gao, x.tan, mqwei}@nuaa.edu.cn

Abstract

Problems such as equipment defects or limited viewpoints will lead the captured point clouds to be incomplete. Therefore, recovering the complete point clouds from the partial ones plays an vital role in many practical tasks, and one of the keys lies in the prediction of the missing part. In this paper, we propose a novel point cloud completion approach namely ProxyFormer that divides point clouds into existing (input) and missing (to be predicted) parts and each part communicates information through its proxies. Specifically, we fuse information into point proxy via feature and position extractor, and generate features for missing point proxies from the features of existing point proxies. Then, in order to better perceive the position of missing points, we design a missing part sensitive transformer, which converts random normal distribution into reasonable position information, and uses proxy alignment to refine the missing proxies. It makes the predicted point proxies more sensitive to the features and positions of the missing part, and thus makes these proxies more suitable for subsequent coarse-to-fine processes. Experimental results show that our method outperforms state-of-the-art completion networks on several benchmark datasets and has the fastest inference speed. Code is available at https://github.com/I2-Multimedia-Lab/ProxyFormer.

1. Introduction

3D data is used in many different fields, including autonomous driving, robotics, remote sensing, and more [5, 12, 14, 17, 43]. Point cloud has a very uniform structure, which avoids the irregularity and complexity of composition. However, in practical applications, due to the occlusion of objects, the difference in the reflectivity of the target surface material, and the limitation of the resolution and viewing angle of the visual sensor, the collected point cloud data is often incomplete. The resultant missing geometric and semantic information will affect the subsequent 3D tasks [26]. Therefore, how to use a limited amount of incomplete data to complete point cloud and restore the original shape has become a hot research topic, and is of great significance to downstream tasks [3, 4, 10, 19, 34, 39].

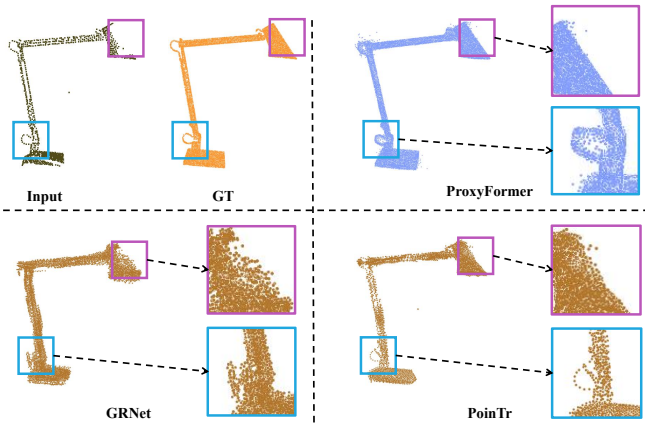


Figure 1. **Visual comparison of point cloud completion results.** Compared with GRNet [38] and PoinTr [41]. *ProxyFormer* completely retains the partial input (blue bounding box) and restores the missing part with details (purple bounding box).

With the tremendous success of PointNet [23] and PointNet++ [24], direct processing of 3D coordinates has become the mainstream of point cloud analysis. In recent years, there have been many point cloud completion methods [1, 11, 37, 38, 41, 42, 48], and the emergence of these networks has also greatly promoted the development of this area. Many methods [1, 38, 42] adopt the common encoder-decoder structure, which usually get global feature from the incomplete input by pooling operation and map this feature back to the point space to obtain a complete one. This kind of feature can predict the approximate shape of the complete point cloud. However, there are two drawbacks: (1) The global feature is extracted from partial input and thus lack the ability to represent the details of the missing part; (2) These methods discard the original incomplete point cloud and regenerate the complete shape after extracting features,

*Corresponding author.

which will cause the shape of the original part to deform to a certain extent. Methods like [11, 41] attempt to predict the missing part separately, but they do not consider the feature connection between the existing and the missing parts, which are still not ideal solutions to the first drawback. The results of GRNet [38] and PoinTr [41] in Fig. 1 illustrate the existence of these problems. GRNet failed to keep the ring on the light stand while PoinTr incorrectly predicted the straight edge of the lampshade as a curved edge. Besides, some methods [16, 37, 41, 48] are based on the transformer structure and use the attention mechanism for feature correlation calculation. However, this also brings up two other problems: (3) In addition to the feature, the position encoding also has a great influence on the effect of the transformer. Existing transformer-based methods, either directly using 3D coordinates [9, 48] or using MLP to upscale the coordinates [37, 41], the position information of the point cloud cannot be well represented; (4) It also leads to the problem of excessive parameters or calculation. Furthermore, we also note that most of the current supervised methods do not make full use of the known data. During the training process, the point cloud data we can obtain includes incomplete input and Ground Truth (GT). This pair of data can indeed undertake the point cloud completion task well, but in fact, we can obtain a new data through these two data, that is, the missing part of the point cloud, so as to increase our prior knowledge.

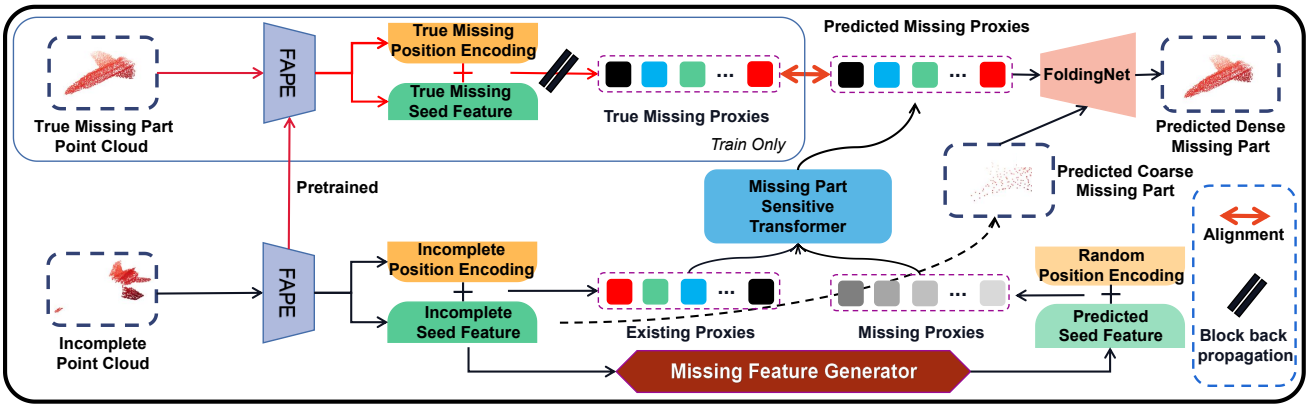
In order to solve the above-mentioned problems, we propose a novel point cloud completion network dubbed *ProxyFormer*, which completely preserves the incomplete input and has better detail recovery capability as shown in Fig. 1. Firstly, we design a feature and position extractor to convert the point cloud to proxies, with a particular attention to the representation of point position. Then, we let the proxies of the partial input interact with the generated missing part proxies through a newly proposed missing part sensitive transformer, instead of using the global feature extracted from incomplete input alone as in prior methods. After mapping proxies back to the point space, we splice it with the incomplete input points to 100% preserve the original data. During training, we use the true missing part of the point cloud to increase prior knowledge and for prediction error refinement. Overall, the main contributions of our work are as follows:

- We design a Missing Part Sensitive Transformer, which focuses on the geometry structure and details of the missing part. We also propose a new position encoding method that aggregates both the coordinates and features from neighboring points.
- We introduce Proxy Alignment into the training process. We convert the true missing part into proxies, which are used to enhance the prior knowledge while refining the predicted missing proxies.
- Our proposed method *ProxyFormer* discards the transformer decoder adopted in most transformer based completion methods such as PointTr, which achieves SOTA performance compared to various baselines while having considerably few parameters and the fastest inference speed in terms of GFLOPs.

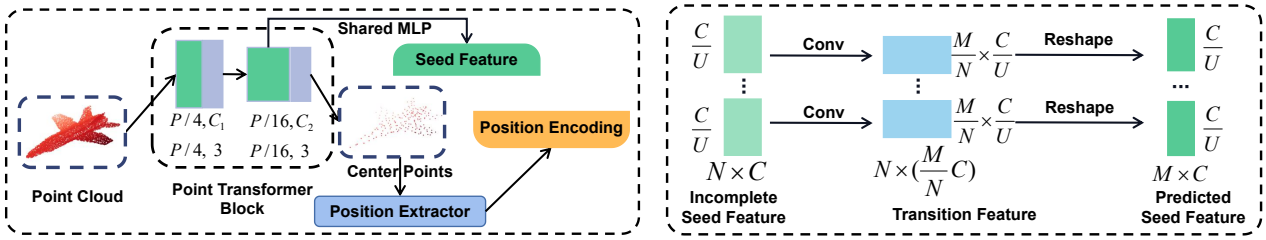
2. Related Work

3D shape completion. Traditional shape completion work mainly includes two categories: geometric rule completion [20, 22, 47] and template matching completion [13, 15, 21]. However, these methods require the input to be as complete as possible, and thus are not robust to new objects and environmental noise. VoxelNet [49] attempts to divide the point cloud into voxel grids and applies convolutional neural networks, but the voxelization will lose the details of the point cloud, and the increasing resolution of the voxel grid will significantly increase the memory consumption. Yuan *et al.* [42] designed PCN, which proposed a coarse-to-fine method based on the PointNet [23] and FoldingNet [40], but its decoder often fails to recover rare geometries of objects such as seat backs with gaps, *etc.* Therefore, after PCN, many other methods [11, 25, 28, 36] focus on multi-step point cloud generation, which is helpful to recover a final point cloud with fine-grained details. Furthermore, following DGCNN [29], some researchers developed graph-based methods [32, 33, 50] which consider regional geometric details. Although these methods provided better feature extractors and decoders, none of them considered the feature connection between the incomplete input and the missing part, which affects the quality of the completion result. Our proposed *ProxyFormer* is not limited to the partial input but also incorporates true missing points during training. We generate features separately for the missing part and explore the correlation with the features extracted from the partial input via self-attention.

Transformers. The transformer structure originated in the field of natural language processing, which is proposed by Vaswani *et al.* [27] and applied to machine translation tasks. Recently, this structure was introduced into point cloud processing tasks due to its advantage in extracting correlated features between points. Guo *et al.* [9] proposed PCT and optimized the self-attention module, making the transformer structure more suitable for point cloud learning, and achieved good performance in shape classification and part segmentation. Point Transformer [46] designs a vector attention for point cloud feature processing. PointTr [41] and SeedFormer [48] treat the point cloud completion as a set-to-set translation problem that share similar ideas as *ProxyFormer*. PoinTr designs a geometry-aware block that explicitly simulates local geometric relations to facilitate transformers to use better inductive bias. However, it adopts a transformer encoder-decoder structure for point



(a) Feature And Position Extractor (FAPE).



(b) Missing Feature Generator.

Figure 2. The pipeline of *ProxyFormer* is shown in the upper part. The completion of the point cloud is divided into two steps. First, we simply convert the incomplete seed feature into a predicted coarse missing part. Second, we send the predicted missing proxies and the coarse part into FoldingNet [40] to obtain the predicted dense missing part. True missing part is used for training only so that we block its back-propagation and directly employ the pretrained FAPE on the incomplete point cloud to generate the proxy. (a) The feature and position extractor is applied to obtain seed feature and position encoding which are combined to the so-called proxies. P represents the count of points in the input point cloud. C_1 and C_2 is the dimensions of point cloud features. (b) The missing feature generator is used to generate predicted seed feature from incomplete seed feature. N and M means the point count of the incomplete seed feature and predicted seed feature. C means the dimensions of the seed feature and is divided into U groups to speed up operations.

cloud completion, which results in a large amount of parameters. SeedFormer designs an upsample transformer by extending the transformer structure into a basic operation in point generators that effectively incorporates spatial and semantic relationships between neighboring points. However, the upsample transformer runs throughout its network, resulting in excessive computation. Differently, *ProxyFormer* discards the transformer decoder to reduce the number of parameters, and modifies the query of transformer to make it more suitable for the prediction of the missing part. In the coarse-to-fine process, we still adopt the Foldingnet [40], which greatly reduces the amount of calculation.

3. Method

The overall network structure of *ProxyFormer* is shown in Fig. 2. We will introduce our method in detail as follows.

3.1. Proxy Formation

Proxy introduction. A proxy represents a local region of the point clouds. All the proxies in this paper fuse two information: **feature** and **position**. The types of proxies are defined as follows:

- **Existing Proxies** (*EP*): It combines incomplete seed feature and incomplete position encoding. (obtained by FAPE).
- **Missing Proxies** (*MP*): It combines predicted seed feature and random position encoding. During the training process, *MP* is also divided into:
  - **Predicted Missing Proxies** (*pre-MP*): It is obtained by Missing Part Sensitive Transformer (Sec. 3.2).
  - **True Missing Proxies** (*true-MP*): It combines true missing seed feature and true missing position encoding. (obtained by pre-trained FAPE). It is only used for Proxy Alignment (Sec. 3.3).

For clarity, we next explain how to obtain the information these proxies need.

Feature and position extractor (FAPE). For **feature** extraction, as shown in Fig. 2a, the point cloud of dimension $(P, 3)$ is sent to point transformer block [46], and the center point cloud of $\left(\frac{P}{16}, 3\right)$ is obtained by farthest point sampling twice. The feature of $\left(\frac{P}{16}, C_2\right)$ is obtained through two vector attention calculations [46]. After that, we use a shared MLP to convert the feature to final seed feature.

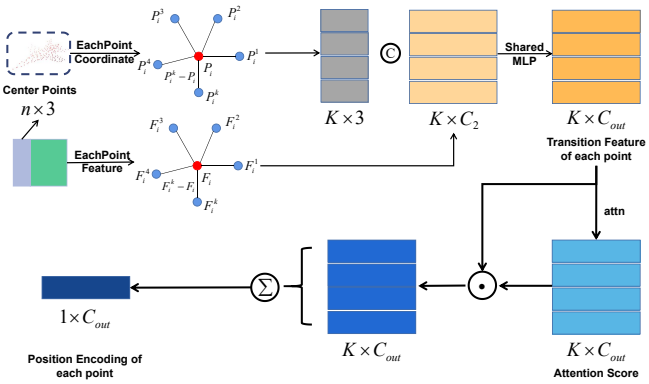


Figure 3. Details of position extractor. $n = \frac{P}{16}$. K is the count of neighbor points. C_2 and C_{out} are the dimension of point features. $Shared\ MLP$ means shared multi-layer perceptron and $attn$ means attention score calculation.

For **position** encoding, we found that directly concatenating the point coordinates with extracted features [24, 29, 45] or simply using MLP to upgrade the three-dimensional coordinates [41] are ineffective. So we design a new position extractor (as shown in Fig. 3) to improve this. The coordinates and features of the center points after feature extraction are used as input. For each point, we take its adjacent K points, and use $\tilde{p}_i^k = p_i^k - p_i$ to calculate the relative position of the point. $p_i^k \in \mathbb{P} = \left\{p_i^1, p_i^2, ..., p_i^K\right\}$, which means the neighbor point coordinates of p_i. We also perform neighbor points subtraction in the feature dimension using $\tilde{f}_i^k = |f_i^k - f_i|$. Similarly, $f_i^k \in \mathbb{F} = \left\{f_i^1, f_i^2, ..., f_i^K\right\}$, which means the neighbor point features of p_i. After that, we get the coordinate information of $K \times 3$ and the feature information of $K \times C_2$. Then we concatenate them and transform feature from $K \times (3 + C_2)$ to $K \times C_{out}$ (the transition feature TF_i^k for each neighboring point) using a shared MLP. After obtaining TF_i^k, we use attention mechanism to learn a unique attention score for each channel of point features and then aggregate them. The attention score is calculated and channel-wise multiplied with the feature and summed to obtain the final PE (*i.e.* position encoding) of each point. This process can be represented by Eq. (1).

$$PE = \sum_{k=1}^{K} \left(attn\left(\left\{TF_i^k\right\}\right) \cdot \left\{TF_i^k\right\} \right), \qquad (1)$$

where $\left\{TF_i^k\right\}$ is the set of transition feature of K-neighbor points and $attn()$ is a shared function (per-point MLPs) with learnable weights W_{attn}.

Incomplete point cloud and true missing part point cloud are sent into FAPE to get *EP* and *true-MP*, and the imcomplete seed feature of *EP* is used to predict coarse missing part at the same time.

3.2. Missing Part Sensitive Transformer

Usually, query, key and value come from the same input. Many methods [37,41,48] attempt to modify the source of value to adapt to the specific tasks (the left part of Fig. 4). Differently, we change the query source, taking the *MP* with random position encoding as query conditions, to maximally mine the representation of the missing part from the features and positions of existing proxies via self-attention mechanism.

In order to change the query source to *MP*, we propose a Missing Feature Generator, which is specially used to learn the missing part features from the existing features. The generation process is shown in Fig. 2b. Specifically, incomplete seed feature of $N \times C$ is used as input, and the C-dimensional channels are divided into U equal length groups. Then, the change dimension of the convolution is determined by the point number M of the predicted coarse missing part, which means that we convert each $\frac{C}{U}$ to $\frac{M}{N} \times \frac{C}{U}$. Lastly, we transform the transition feature into predicted seed feature of $M \times C$ through the reshape operation. All channel groups use convolutional layers with shared parameters, reducing the amount of parameters and computation. Predicted missing feature is added with random position encoding to get *MP*.

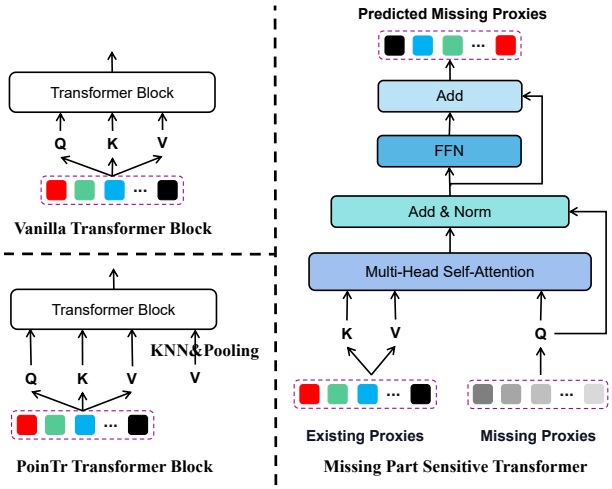


Figure 4. Details of Missing Part Sensitive Transformer. Compared with vanilla transformer block and PoinTr transformer block, we change the source of query to make it more suitable for the prediction of the missing part.

In Sec. 3.1, we have obtained *EP* and *true-MP*, and through the missing feature generator described above, we have obtained *MP*. Then we design a Missing Part Sensitive Transformer to further explore their relationship and learn the representation of the missing points for subsequent completion work. Its structure is illustrated in Fig. 4, which receives *EP* and *MP* as input and outputs *pre-MP*. *EP* is a

matrix $\mathbb{E}$ of $N \times C$, and *MP* is a matrix $\mathbb{M}$ of $M \times C$. Output *pre-MP* is a matrix $\mathbb{P}$ of $M \times C$.

We use multi-head self-attention mechanism and add residual connections to obtain *pre-MP*:

$$\begin{aligned} T &= \mu\left(Q + \xi(Q, K, V)\right) , \\ \textit{pre-MP} &= \sigma(T) + T , \end{aligned} \tag{2}$$

where $Q = \mathbb{M} \times W^Q$, $K = \mathbb{E} \times W^{KV}$, $V = \mathbb{E} \times W^{KV}$. μ means layer normalization. ξ means multi-head attention calculation. σ means feed forward network. The attention of each head is calculated with Q_i, K_i, V_i on head i:

$$attn_i = softmax\left(\frac{Q_i \left(K_i\right)^T}{\sqrt{d_k}}\right) V_i . \tag{3}$$

This method predicts the proxies of the missing points, which not only discovers feature associations between missing and existing parts, but also converts random positions into meaningful position information. The *pre-MP* is next used for proxy alignment with *true-MP*.

3.3. Proxy Alignment

In this subsection, we describe the Proxy Alignment strategy and how this operation assists us in the point cloud completion task.

The detailed computational graph of *ProxyFormer* is plotted in Fig. 5. Therefore, *pre-MP* and *true-MP* can be formulated as:

$$\begin{aligned} \textit{pre-MP} &= T\left(PE_R \oplus \theta\left(\omega\left(C_i\right)\right)\right) \\ \textit{true-MP} &= PE_T \oplus \omega\left(C_m\right) \end{aligned} . \tag{4}$$

In order to refine the prediction error in *pre-MP*, the proxy alignment constraint is imposed on the model, which can be formulated as:

$$l_p = MSE\left(\textit{pre-MP}, \textit{true-MP}\right) , \tag{5}$$

where l_p means the alignment loss that we will apply to our training loss (Sec. 3.4) and MSE means the mean squared error. After correcting the *pre-MP*, it is used as a feature and sent to FoldingNet [40] for coarse-to-fine conversion, and then combined with the previously predicted coarse missing part to obtain the dense missing part.

3.4. Training Loss

Chamfer Distance. We use the average Chamfer Distance(CD) [6] as the first type of our completion loss.

proxy alignment Loss. We use the MSE loss between *pre-MP* and *true-MP* as the second type of loss.

To sum up, as shown in Fig. 5, the loss used in this paper consists three parts: (1). l_{c1}, the CD between the predicted coarse missing part C_{pcm} and the true center point C_{tcm} of the missing part; (2). l_{c2}, the CD between the predicted complete point cloud C_{pc} and the GT C_{gt}; (3) l_p, the alignment loss between *pre-MP* and *ture-MP*. We use the weighted sum of these three terms for network training (we set γ to 1.5 in experiments):

$$L = l_{c1} + l_{c2} + \gamma l_p . \tag{6}$$

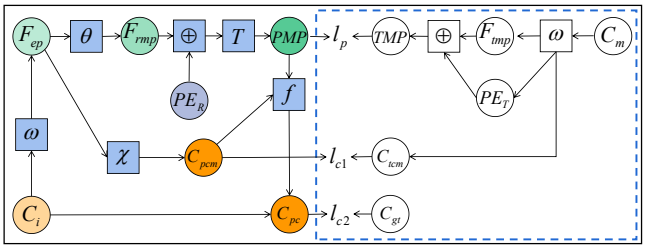


Figure 5. The computational graph for *ProxyFormer*. The part framed by the blue dotted line is used for training only. For the left part, we input incomplete point cloud C_i and use *FAPE* (ω) to get feature F_{ep} in *EP*. F_{ep} is not only sent to a linear projection layer χ to generate predicted coarse missing part C_{pcm}, but also sent to missing feature generator (θ) to generate feature F_{rmp}. F_{rmp} is added ($\oplus$) with random position distribution PE_R and sent to missing part sensitive transformer (T) to get *pre-MP*. Then *pre-MP* and C_{pcm} are sent to FoldingNet (f), and the result is spliced with input C_i to form predicted complete point cloud C_{pc}. For the right part, we input true missing part point cloud C_m and use the same *FAPE* (ω) to get feature F_{tmp} and position PE_T in *true-MP*. *pre-MP* is aligned with *true-MP* for correcting deviation values. We also retain the center point C_{tcm} obtained after C_m is downsampled by *FAPE*. l_p, l_{c1} and l_{c2} are the losses we use, which will be detailed in the next subsection.

4. Experiments

In this section, we use *ProxyFormer* for two common point cloud completion benchmarks PCN [42] and KITTI [7] to evaluate the completion ability of the network, and then we also train and test on two other datasets, ShapeNet-55 and ShapeNet-34 proposed by PoinTr [41]. Finally, through ablation experiments, we demonstrate the effectiveness of each module in the proposed *ProxyFormer*.

4.1. Point Cloud Completeion on PCN Dataset

Dataset and evaluation metric. The PCN dataset [42] is created from ShapeNet dataset [2], including eight categories with a total of 30974 CAD models. When preparing the data, we use missing part extractor to extract the missing part of the point cloud from the complete point cloud and then downsample it to 3584 points as the true missing part (This process is described in detail in the supplementary material).

We use the L1 CD to evaluate the results of each methods. In addition, We also use density-aware chamfer dis-

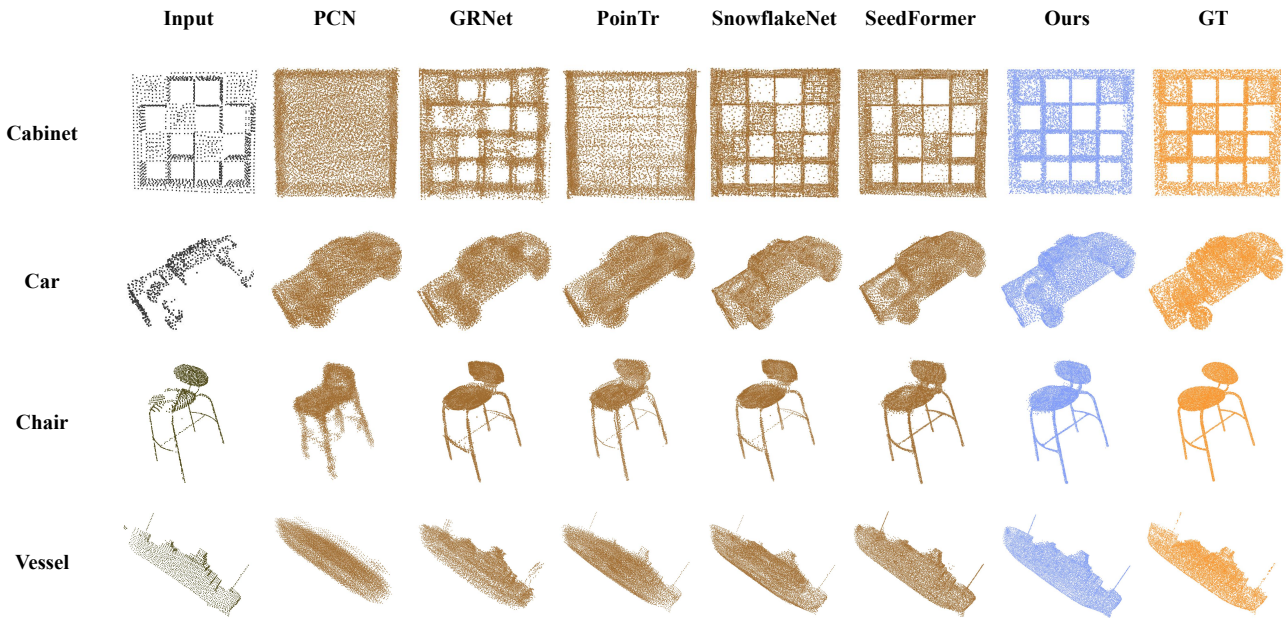


Figure 6. The visualization results of each method on the PCN dataset, showing Cabinet, Car, Chair and Vessel from top to bottom.

tance (DCD) [35] as a quantitative evaluation criterion, which can retain the measurement ability similar to CD but also better judge the visual effect of the result.

Quantiative comparison. According to the results in Table 1, on the PCN dataset, our method has substantially surpassed PoinTr [41] and reaches lowest CD in the cabinet, car, sofa and table categories. Further, as can be seen from the DCD shown in Table 2, our method outperforms state-of-the-art in all the categories, which means that our method is more able to take into account the rationality of shape and distribution density while complementing the object.

Table 1. Quantitative comparison of PCN dataset. Point resolutions for the output and GT are 16384. For CD, lower is better.

Methods	Chamfer Distance(10^{-3})								
	Air	Cab	Car	Cha	Lam	Sof	Tab	Ves	Ave
FoldingNet [40]	9.49	15.80	12.61	15.55	16.41	15.97	13.65	14.99	14.31
TopNet [25]	7.61	13.31	10.90	13.82	14.44	14.78	11.22	11.12	12.15
AtlasNet [8]	6.37	11.94	10.10	12.06	12.37	12.99	10.33	10.61	10.85
PCN [42]	5.50	22.70	10.63	8.70	11.00	11.34	11.68	8.59	9.64
GRNet [38]	6.45	10.37	9.45	9.41	7.96	10.51	8.44	8.04	8.83
CRN [28]	4.79	9.97	8.31	9.49	8.94	10.69	7.81	8.05	8.51
NSFA [44]	4.76	10.18	8.63	8.53	7.03	10.53	7.35	7.48	8.06
PMP-Net [30]	5.65	11.24	9.64	9.51	6.95	10.83	8.72	7.25	8.73
PoinTr [41]	4.75	10.47	8.68	9.39	7.75	10.93	7.78	7.29	8.38
PMP-Net++ [31]	4.39	9.96	8.53	8.09	6.06	9.82	7.17	6.52	7.56
SnowflakeNet [37]	4.29	9.16	8.08	7.89	6.07	9.23	6.55	6.40	7.21
SeedFormer [48]	**3.85**	9.05	8.06	**7.06**	**5.21**	8.85	6.05	**5.85**	**6.74**
ProxyFormer(Ours)	4.01	**9.01**	**7.88**	7.11	5.35	**8.77**	**6.03**	5.98	6.77

Qualitative comparison. In Fig. 6, we visualize the completion results of different methods on the PCN dataset. Compared with other methods, the results show that our method can perceive the position of the missing points while completing, and reduce the noisy points in the process of refinement. For example, as can be seen from the chair in the third row, except that the chair generated by PCN [42] has been deformed to a large extent, the other methods have successfully recovered the complete chair, but there are many noisy points around it. The chair completed by our method is more visually plausible. In addition, it can be evident from the chair leg and back that the chair completed by our method are more prominent in detail.

Table 2. Quantitative comparison of PCN dataset. Point resolutions for the output and GT are 16384. For DCD, lower is better.

Methods	Density-aware Chamfer Distance								
	Air	Cab	Car	Cha	Lam	Sof	Tab	Ves	Ave
GRNet [38]	0.688	0.582	0.610	0.607	0.644	0.622	0.578	0.642	0.622
PoinTr [41]	0.574	0.611	0.630	0.603	0.628	0.669	0.556	0.614	0.611
SnowflakeNet [37]	0.560	0.597	0.603	0.582	0.598	0.633	0.521	0.583	0.585
SeedFormer [48]	0.557	0.592	0.598	0.579	0.585	**0.626**	0.520	0.605	0.583
ProxyFormer(Ours)	**0.555**	**0.590**	**0.597**	**0.571**	**0.562**	**0.626**	**0.518**	**0.507**	**0.577**

4.2. Point Cloud Completion on ShapeNet-55/34

Dataset and evaluation metric. We also evaluate our model on two other datasets, ShapeNet-55 and ShapeNet-34, proposed in PoinTr [41]. In the two datasets, the input incomplete point cloud has 2048 points, and the complete point cloud contains 8192 points. Like [41], we randomly select a viewpoint during training, and select a value from 2048 to 6144 to delete the corresponding points (25% to 75% of the complete point cloud), and then downsample the remaining points to 2048, as the input for model training. For the deleted part, we downsample it to 1536 points, as the true missing part. During testing, we choose 8 fixed viewpoints, and set the count of incomplete points to 2048, 4096 or 6144 (25%, 50% or 75% of the complete point cloud),

Table 3. Quantitative comparison on ShapeNet-55. For L2 CD $\times 1000$ and DCD, lower is better. For F1-Score@1%, higher is better.

Methods	Table	Chair	Plane	Car	Sofa	CD-S	CD-M	CD-H	DCD-S	DCD-M	DCD-H	CD-Avg	DCD-Avg	F1
FoldingNet [40]	2.53	2.81	1.43	1.98	2.48	2.67	2.66	4.05	-	-	-	3.12	-	0.082
PCN [42]	2.13	2.29	1.02	1.85	2.06	1.94	1.96	4.08	0.570	0.609	0.676	2.66	0.618	0.133
TopNet [25]	2.21	2.53	1.14	2.18	2.36	2.26	2.16	4.30	-	-	-	2.91	-	0.126
PFNet [11]	3.95	4.24	1.81	2.53	3.34	3.83	3.87	7.97	-	-	-	5.22	-	0.339
GRNet [38]	1.63	1.88	1.02	1.64	1.72	1.35	1.71	2.85	0.545	0.581	0.650	1.97	0.592	0.238
PoinTr [41]	0.81	0.95	0.44	0.91	0.79	0.58	0.88	1.79	0.525	0.562	0.637	1.09	0.575	0.464
SeedFormer [48]	0.72	**0.81**	0.40	0.89	0.71	0.50	0.77	**1.49**	0.513	0.549	0.612	**0.92**	0.558	0.472
Ours	**0.70**	0.83	**0.34**	**0.78**	**0.69**	**0.49**	**0.75**	1.55	**0.502**	**0.536**	**0.608**	0.93	**0.549**	**0.483**

Table 4. Quantitative comparison on ShapeNet-34. For L2 CD $\times 1000$ and DCD, lower is better. For F1-Score@1%, higher is better.

Methods	34 seen categories									21 unseen categories								
	CD-S	CD-M	CD-H	DCD-S	DCD-M	DCD-H	CD-Avg	DCD-Avg	F1	CD-S	CD-M	CD-H	DCD-S	DCD-M	DCD-H	CD-Avg	DCD-Avg	F1
FoldingNet	1.86	1.81	3.38	-	-	-	2.35	-	0.139	2.76	2.74	5.36	-	-	-	3.62	-	0.095
PCN	1.87	1.81	2.97	0.571	0.617	0.683	2.22	0.624	0.150	3.17	3.08	5.29	0.601	0.638	0.692	3.85	0.644	0.101
TopNet	1.77	1.61	3.54	-	-	-	2.31	-	0.171	2.62	2.43	5.44	-	-	-	3.50	-	0.121
PFNet	3.16	3.19	7.71	-	-	-	4.68	-	0.347	5.29	5.87	13.33	-	-	-	8.16	-	0.322
GRNet	1.26	1.39	2.57	0.550	0.594	0.656	1.74	0.600	0.251	1.85	2.25	4.87	0.583	0.623	0.670	2.99	0.625	0.216
PoinTr	0.76	1.05	1.88	0.533	0.570	0.622	1.23	0.575	0.421	1.04	1.67	3.44	0.558	0.608	0.647	2.05	0.604	0.384
SeedFormer	0.48	0.70	**1.30**	0.513	0.561	0.608	0.83	0.561	0.452	0.61	**1.07**	**2.35**	0.541	0.587	0.629	**1.34**	0.586	0.402
Ours	**0.44**	**0.67**	1.33	**0.506**	**0.557**	**0.606**	**0.81**	**0.556**	**0.466**	**0.60**	1.13	2.54	**0.538**	**0.584**	**0.627**	1.42	**0.583**	**0.415**

corresponding to three difficulty levels (simple, moderate and hard) during testing.

We use L2 CD, DCD and F-Score as evaluation metrics. **Quantitative comparison.** We list the quantitative performance of several methods on ShapeNet-55 and ShapeNet-34 in Tables 3 and 4, respectively. We use CD-S, CD-M and CD-H to represent the CD results under the simple, moderate and hard settings. The same goes for DCD (the short line in the table indicates that the DCD value of this method is not competitive). It can be seen from Table 3 that *ProxyFormer* achieves the best performance on most listed categories and make lowest CD on simple and moderate settings. As for DCD, our method has the lowest values on all the three settings, which proves that the objects completed by *ProxyFormer* have the closest density distribution to GT. In terms of F1-Score, our method improved by 2.3% compared with SeedFormer, reaching the highest value. Similarly, in Table 4, we can also see that the three indicators of CD, DCD and F1-Score of *ProxyFormer* in the 34 visible categories have greatly exceeded PoinTr [41], and in all the three settings, we still get the lowest DCD. Among 21 unseen categories, *ProxyFormer* also make the lowest DCD and the highest F1-Score, which demonstrates the generalization performance of *ProxyFormer*. (More results will be presented in the supplementary material)

4.3. Point Cloud Completion on KITTI

Dataset and evaluation metric. To further evaluate our proposed model, we test it on the real-scanned dataset KITTI [7], which have no GT values as a reference, and some of the data are very sparse.

We use Fidelity Distance and Minimal Matching Distance (MMD) as evaluation metric.

Quantitative and Qualitative comparison. Following GRNet [38], we fine-tune our pretrained model on ShapeNetCars (cars in the ShapeNet dataset) and evaluate it on the KITTI dataset, and the evaluation results are shown in Table 5. From this we can see that our method achieves the state-of-the-art on MMD (since both our method and PoinTr [41] merge input into the final result, the Fidelity Distance is both 0). As shown in Fig. 7, our method performs well on such real scan data, and even if the input point cloud is very sparse, our method can restore its shape well, and by comparing with the results of GRNet, it can be seen that the point cloud generated by *ProxyFormer* is softer, with less noisy points, and is more ornamental.

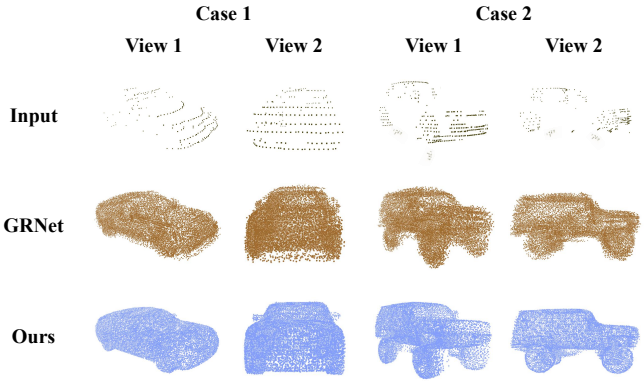


Figure 7. The visualization results on KITTI dataset. To better show the effect of completion, we provide two views for each car.

4.4. Ablation Studies

In this subsection, we conduct ablation experiments for *ProxyFormer* on the PCN dataset [42] to demonstrate the effectiveness of our proposed components.

Table 5. Quantitative comparison on KITTI dataset. For Fidelity Distance and Minimal Matching Distance (MMD), lower is better.

	AtlasNet [8]	PCN [42]	FoldingNet [40]	TopNet [25]	MSN [18]	NSFA [44]	CRN [28]	GRNet [38]	PoinTr [41]	SeedFormer [48]	Ours
Fidelity	1.759	2.235	7.467	5.354	0.434	1.281	1.023	0.816	**0.000**	0.151	**0.000**
MMD	2.108	1.366	0.537	0.636	2.259	0.891	0.872	0.568	0.526	0.516	**0.508**

Model Design Analysis. The results of removing each component are listed in Table 6. The baseline model A only uses Point Transformer for feature extraction, and then send this feature into vanilla transformer encoder to get the feature for FoldingNet. We then add the position extractor to extract the position encoding for each point (Model B). It can be seen that the position extractor we designed reduces the CD of the baseline by 1.06. After using missing part sensitive transformer for missing proxies prediction (Model C), we can observe that the CD drops significantly to 7.74. When proxy alignment comes into play, the CD value drops a further 0.97.

Table 6. Ablation study of each component. We add components including Position Extractor (PE), Missing Part Sensitive Transformer (Sensitive) and Proxy Alignment (PA) step by step.

Model	PE	Sensitive	PA	CD
A				11.08
B	✓			10.02
C	✓	✓		7.74
D	✓	✓	✓	6.77

After conducting ablation experiments on the proposed module, we further demonstrate the irreplaceability of position extractor through one more ablation experiments.

Position Extractor. Our position extractor can synthesize the coordinates and feature information of the point cloud to more accurately represent the correlation and similarity between points. In this experiment, we compare our proposed position encoding method with two method: (1) directly use 3D coordinates as position encoding; (2) MLP-style position encoding method. which performs a simple upscaling operation on the 3D coordinates of the point cloud to form position encoding. The results in Table 7 show that the direct use of 3D coordinates provides very limited position information and MLP cannot extract the positional of the point cloud well. Our proposed position encoding method can perceive the geometric structure of the point cloud well, and in this process, it is optimal to fuse the coordinates and feature information of 16 nearby points.

More ablation experiments and analysis will be given in the supplementary material.

4.5. Complexity Analysis

Our method achieves the best performance on many metrics on PCN dataset, ShapeNet-55, ShapeNet-34 and KITTI datasets. In Table 8, we list the number of parameters (Params), theoretical computation cost (FLOPs), the average chamfer distances (CD-Avg) and the average density-aware chamfer distances (DCD-Avg) of our method and other six methods. It can be seen that our method can obtain the lowest DCD-Avg while having the smallest FLOPs, and it is the second best only a litter inferior to SeedFormer [48] in terms of CD. Since the transformer decoder part was no longer needed in *ProxyFormer*, the number of parameters is also greatly reduced compared to PoinTr [41], which also shows that our method can better balance computational cost and performance.

Table 7. Ablation study of Position Extractor of FAPE Module.

Methods	Attempts	CD-Avg
w/o Position Extractor	w/ 3D coordinates	9.63
	w/ MLP	7.83
w/ Position Extractor	num of neighbor = 8	6.86
	num of neighbor = 16	**6.77**
	num of neighbor = 32	6.92

Table 8. Complexity analysis. We show the the number of parameter (Params) and FLOPs of our method and six existing methods. We also provide the distance metrics CD-Avg and DCD-Avg on PCN dataset.

Methods	Params	FLOPs	CD-Avg	DCD-Avg
FoldingNet [40]	**2.41M**	27.65G	14.31	0.688
PCN [42]	6.84M	14.69G	9.64	0.651
GRNet [38]	76.71M	25.88G	8.83	0.622
PoinTr [41]	30.9M	10.41G	8.38	0.611
SnowflakeNet [37]	19.32M	10.32G	7.21	0.585
SeedFormer [48]	3.20M	29.61G	**6.74**	0.583
Ours	12.16M	**9.88G**	6.77	**0.577**

5. Conclusion

In this paper, we propose a new point cloud completion framework named *ProxyFormer*, which designs a missing part sensitive transformer to generate missing proxies. We extract feature and position for the missing points and form point proxies. We regularize the distribution of predicted point proxies through proxy alignment, so as to better complete the input partial point clouds. Experiments also show that our method achieves state-of-the-art performance on multiple metrics on several challenging benchmark datasets, and has the fastest inference speed.

Acknowledgement

This work was supported in part by the National Key Research and Development Program of China under Grant 2021ZD0113203 and the Natural Science Foundation of China under Grant 62272227.

References

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

[2] Angel X Chang, Thomas Funkhouser, Leonidas Guibas, Pat Hanrahan, Qixing Huang, Zimo Li, Silvio Savarese, Manolis Savva, Shuran Song, Hao Su, et al. Shapenet: An information-rich 3d model repository. *arXiv preprint arXiv:1512.03012*, 2015. 5

[3] Honghua Chen, Zeyong Wei, Yabin Xu, Mingqiang Wei, and Jun Wang. Imlovenet: Misaligned image-supported registration network for low-overlap point cloud pairs. In *ACM SIGGRAPH 2022 Conference Proceedings*, pages 1–9, 2022. 1

[4] Jun-Kun Chen and Yu-Xiong Wang. Pointtree: Transformation-robust point cloud encoder with relaxed kd trees. *arXiv preprint arXiv:2208.05962*, 2022. 1

[5] Yi-Nan Chen, Hang Dai, and Yong Ding. Pseudo-stereo for monocular 3d object detection in autonomous driving. In *Proceedings of the IEEE/CVF Conference on Computer Vision and Pattern Recognition*, pages 887–897, 2022. 1

[6] Haoqiang Fan, Hao Su, and Leonidas J Guibas. A point set generation network for 3d object reconstruction from a single image. In *Proceedings of the IEEE conference on computer vision and pattern recognition*, pages 605–613, 2017. 5

[7] Andreas Geiger, Philip Lenz, Christoph Stiller, and Raquel Urtasun. Vision meets robotics: The kitti dataset. *The International Journal of Robotics Research*, 32(11):1231–1237, 2013. 5, 7

[8] Thibault Groueix, Matthew Fisher, Vladimir G Kim, Bryan C Russell, and Mathieu Aubry. A papier-mâché approach to learning 3d surface generation. In *Proceedings of the IEEE conference on computer vision and pattern recognition*, pages 216–224, 2018. 6, 8

[9] Meng-Hao Guo, Jun-Xiong Cai, Zheng-Ning Liu, Tai-Jiang Mu, Ralph R Martin, and Shi-Min Hu. Pct: Point cloud transformer. *Computational Visual Media*, 7(2):187–199, 2021. 2

[10] Zhixing Hou, Yan Yan, Chengzhong Xu, and Hui Kong. Hitpr: Hierarchical transformer for place recognition in point cloud. *arXiv preprint arXiv:2204.05481*, 2022. 1

[11] Zitian Huang, Yikuan Yu, Jiawen Xu, Feng Ni, and Xinyi Le. Pf-net: Point fractal network for 3d point cloud completion. In *Proceedings of the IEEE/CVF conference on computer vision and pattern recognition*, pages 7662–7670, 2020. 1, 2, 7

[12] Stephen James, Kentaro Wada, Tristan Laidlow, and Andrew J Davison. Coarse-to-fine q-attention: Efficient learning for visual robotic manipulation via discretisation. In *Proceedings of the IEEE/CVF Conference on Computer Vision and Pattern Recognition*, pages 13739–13748, 2022. 1

[13] Vladimir G Kim, Wilmot Li, Niloy J Mitra, Siddhartha Chaudhuri, Stephen DiVerdi, and Thomas Funkhouser. Learning part-based templates from large collections of 3d shapes. *ACM Transactions on Graphics (TOG)*, 32(4):1–12, 2013. 2

[14] Peizhao Li, Pu Wang, Karl Berntorp, and Hongfu Liu. Exploiting temporal relations on radar perception for autonomous driving. In *Proceedings of the IEEE/CVF Conference on Computer Vision and Pattern Recognition*, pages 17071–17080, 2022. 1

[15] Yangyan Li, Angela Dai, Leonidas Guibas, and Matthias Nießner. Database-assisted object retrieval for real-time 3d reconstruction. In *Computer graphics forum*, volume 34, pages 435–446. Wiley Online Library, 2015. 2

[16] Jianjie Lin, Markus Rickert, Alexander Perzylo, and Alois Knoll. Pctma-net: Point cloud transformer with morphing atlas-based point generation network for dense point cloud completion. In *2021 IEEE/RSJ International Conference on Intelligent Robots and Systems (IROS)*, pages 5657–5663. IEEE, 2021. 2

[17] Kunliang Liu, Ouk Choi, Jianming Wang, and Wonjun Hwang. Cdgnet: Class distribution guided network for human parsing. In *Proceedings of the IEEE/CVF Conference on Computer Vision and Pattern Recognition*, pages 4473–4482, 2022. 1

[18] Minghua Liu, Lu Sheng, Sheng Yang, Jing Shao, and Shi-Min Hu. Morphing and sampling network for dense point cloud completion. In *Proceedings of the AAAI conference on artificial intelligence*, volume 34, pages 11596–11603, 2020. 8

[19] Zhijian Liu, Haotian Tang, Shengyu Zhao, Kevin Shao, and Song Han. Pvnas: 3d neural architecture search with point-voxel convolution. *arXiv preprint arXiv:2204.11797*, 2022. 1

[20] Niloy J Mitra, Mark Pauly, Michael Wand, and Duygu Ceylan. Symmetry in 3d geometry: Extraction and applications. In *Computer Graphics Forum*, volume 32, pages 1–23. Wiley Online Library, 2013. 2

[21] Liangliang Nan, Ke Xie, and Andrei Sharf. A search-classify approach for cluttered indoor scene understanding. *ACM Transactions on Graphics (TOG)*, 31(6):1–10, 2012. 2

[22] Mark Pauly, Niloy J Mitra, Johannes Wallner, Helmut Pottmann, and Leonidas J Guibas. Discovering structural regularity in 3d geometry. In *ACM SIGGRAPH 2008 papers*, pages 1–11. 2008. 2

[23] Charles R Qi, Hao Su, Kaichun Mo, and Leonidas J Guibas. Pointnet: Deep learning on point sets for 3d classification and segmentation. In *Proceedings of the IEEE conference on computer vision and pattern recognition*, pages 652–660, 2017. 1, 2

[24] 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. 1, 4

[25] Lyne P Tchapmi, Vineet Kosaraju, Hamid Rezatofighi, Ian Reid, and Silvio Savarese. Topnet: Structural point cloud decoder. In *Proceedings of the IEEE/CVF Conference on Computer Vision and Pattern Recognition*, pages 383–392, 2019. 2, 6, 7, 8

[26] Kutub Uddin, Tae Hyun Jeong, and Byung Tae Oh. Incomplete region estimation and restoration of 3d point cloud human face datasets. *Sensors*, 22(3):723, 2022. 1

[27] 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. 2

[28] Xiaogang Wang, Marcelo H Ang Jr, and Gim Hee Lee. Cascaded refinement network for point cloud completion. In *Proceedings of the IEEE/CVF Conference on Computer Vision and Pattern Recognition*, pages 790–799, 2020. 2, 6, 8

[29] Yue Wang, Yongbin Sun, Ziwei Liu, Sanjay E Sarma, Michael M Bronstein, and Justin M Solomon. Dynamic graph cnn for learning on point clouds. *Acm Transactions On Graphics (tog)*, 38(5):1–12, 2019. 2, 4

[30] Xin Wen, Peng Xiang, Zhizhong Han, Yan-Pei Cao, Pengfei Wan, Wen Zheng, and Yu-Shen Liu. 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*, pages 7443–7452, 2021. 6

[31] Xin Wen, Peng Xiang, Zhizhong Han, Yan-Pei Cao, Pengfei Wan, Wen Zheng, and Yu-Shen Liu. Pmp-net++: Point cloud completion by transformer-enhanced multi-step point moving paths. *IEEE Transactions on Pattern Analysis and Machine Intelligence*, 2022. 6

[32] Hang Wu and Yubin Miao. Lra-net: local region attention network for 3d point cloud completion. In *Thirteenth International Conference on Machine Vision*, volume 11605, pages 357–367. SPIE, 2021. 2

[33] Hang Wu, Yubin Miao, and Ruochong Fu. Point cloud completion using multiscale feature fusion and cross-regional attention. *Image and Vision Computing*, 111:104193, 2021. 2

[34] Rundi Wu, Xuelin Chen, Yixin Zhuang, and Baoquan Chen. Multimodal shape completion via conditional generative adversarial networks. In *European Conference on Computer Vision*, pages 281–296. Springer, 2020. 1

[35] 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. 6

[36] Yaqi Xia, Yan Xia, Wei Li, Rui Song, Kailang Cao, and Uwe Stilla. Asfm-net: Asymmetrical siamese feature matching network for point completion. In *Proceedings of the 29th ACM International Conference on Multimedia*, pages 1938–1947, 2021. 2

[37] Peng Xiang, Xin Wen, Yu-Shen Liu, Yan-Pei Cao, Pengfei Wan, Wen Zheng, and Zhizhong Han. Snowflakenet: Point cloud completion by snowflake point deconvolution with skip-transformer. In *Proceedings of the IEEE/CVF international conference on computer vision*, pages 5499–5509, 2021. 1, 2, 4, 6, 8

[38] Haozhe Xie, Hongxun Yao, Shangchen Zhou, Jiageng Mao, Shengping Zhang, and Wenxiu Sun. Grnet: Gridding residual network for dense point cloud completion. In *European Conference on Computer Vision*, pages 365–381. Springer, 2020. 1, 2, 6, 7, 8

[39] Zelin Xu, Yichen Zhang, Ke Chen, and Kui Jia. Bico-net: Regress globally, match locally for robust 6d pose estimation. *arXiv preprint arXiv:2205.03536*, 2022. 1

[40] Yaoqing Yang, Chen Feng, Yiru Shen, and Dong Tian. Foldingnet: Point cloud auto-encoder via deep grid deformation. In *Proceedings of the IEEE conference on computer vision and pattern recognition*, pages 206–215, 2018. 2, 3, 5, 6, 7, 8

[41] 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. 1, 2, 4, 5, 6, 7, 8

[42] 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. 1, 2, 5, 6, 7, 8

[43] Qingzhao Zhang, Shengtuo Hu, Jiachen Sun, Qi Alfred Chen, and Z Morley Mao. On adversarial robustness of trajectory prediction for autonomous vehicles. In *Proceedings of the IEEE/CVF Conference on Computer Vision and Pattern Recognition*, pages 15159–15168, 2022. 1

[44] Wenxiao Zhang, Qingan Yan, and Chunxia Xiao. Detail preserved point cloud completion via separated feature aggregation. In *European Conference on Computer Vision*, pages 512–528. Springer, 2020. 6, 8

[45] Hengshuang Zhao, Li Jiang, Chi-Wing Fu, and Jiaya Jia. Pointweb: Enhancing local neighborhood features for point cloud processing. In *Proceedings of the IEEE/CVF conference on computer vision and pattern recognition*, pages 5565–5573, 2019. 4

[46] 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. 2, 3

[47] Wei Zhao, Shuming Gao, and Hongwei Lin. A robust hole-filling algorithm for triangular mesh. *The Visual Computer*, 23(12):987–997, 2007. 2

[48] 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. 1, 2, 4, 6, 7, 8

[49] Yin Zhou and Oncel Tuzel. Voxelnet: End-to-end learning for point cloud based 3d object detection. In *Proceedings of the IEEE conference on computer vision and pattern recognition*, pages 4490–4499, 2018. 2

[50] Liping Zhu, Bingyao Wang, Gangyi Tian, Wenjie Wang, and Chengyang Li. Towards point cloud completion: Point rank sampling and cross-cascade graph cnn. *Neurocomputing*, 461:1–16, 2021. 2