Parametric CAD Scan Completion

Minji Yang, Zongtai Li

IIS, Tsinghua University

Abstract. Computer-aided design (CAD) is widely used and understanding CAD models is quite essential for high-accuracy operations in industry. We propose a brand-new framework completing a partial scan of some CAD model from a depth camera and as a by-product, estimating the parametric representation of its construction process. Precisely, our approach first segments the partial scan into different extrusion cylinders and completes each cylinder with the assistance of visibility information. Compared with previous methods, our approach gets rid of the category-specific limitation and alleviate the blurring problem.

Keywords: point cloud completion, computer-aided design, parameterized

1 Introduction

Computer-aided design, also known as CAD, is widely used in the production of everyday objects. It is to use computers to aid in the creation, modification, or optimization of a design. When modeling with CAD, we usually draw 2d sketches first, and then use the extrusion operation to turn them into cylinders. We can also combine different cylinders through Boolean operations.

In daily production, we often need to do some careful operations with mechanical arms. In the era of artificial intelligence, the mechanical arms will no longer operate according to the preset program, but will be able to think and operate by itself. Under this setting, the mechanical arm needs to have an overall understanding of the object to be operated. However, we usually cannot obtain the complete scan of the object to be operated in advance. How to quickly get a complete understanding of a new object in such a limited situation is an urgent problem to be solved.

In our problem setting, we only use a depth camera to capture an incomplete point cloud as the input and the task is to complete the partial scan. Over these years, researchers have made many attempts to the problem of point cloud completion. [5] [13][1] [4] However, these existing methods cannot work well on our task. Firstly, most of these methods are category-based, trained on several fixed categories such as airplanes and bicycles. [16][18] However, in our CAD completion task, we are dealing with lots of different small objects and shapes, which cannot be easily divided into a few classes. Secondly, most CAD models have sharp edges and angles, but using the methods above will blur out these sharp edges and get only a fuzzy cloud.

The reason of the above problems is that these methods treat the partial point cloud as an entire body. However, the particularity of CAD models is that they are combination of different cylinders, which is also the reason why they have so many sharp edges. So we would like to restore the whole construction process and do the completion. To be more specific, given a partial scan of an object, we first do segmentation which means we segment the object into several parts, and each part corresponds to one extrusion cylinder. Then we fit the parameters including the extrusion direction and range for each cylinder. Then we map the cylinders to 2D according to the predicted extrusion direction, do the completion on 2D scale, and lift it back to 3D. Combining each completed parts, we get a coarse completion of the object. Finally, we combine the ground truth partial scan and the coarse completion to do a refinement.

Since the project is inherited from last year, the work of the predecessors has provided us a good foundation. [20] They have proposed an elementary pipeline for scan completion of CAD models. They also have done the segmentation and fitting part, which is to segment the object into different extrusion cylinders and predict the parameters for each cylinder. We mainly focus on the completion of each cylinder and reconstruction of the whole object.

2 Related Work

Point cloud completion

Point cloud completion methods can be roughly divided into two categories: traditional methods and deep learning based methods. The development of deep learning based methods have greatly boosted the research of point cloud completion. TopNet [11] modeled the point cloud generation process as the growth of a rooted tree, where one parent point feature is projected into several child point features. PCN [19] used a two-stage generation framework, where a coarse and low resolution point cloud is first generated and then a lifting module is used to improve the completion quality. Following the similar practice, CDN [13] and PF-Net [5] increased the number of generation stages and achieved better result. SA-Net [14] proposed a skip-attention mechanism to effectively exploit the local structure details of incomplete point clouds during the inference of missing parts. SnowflakeNet [16] modeled the completion process as the snowflake-like growth of points in 3D space and designed Snowflake Point Deconvolution (SPD) to generate the complete point clouds. However, as described before, most of these methods suffer from very strong category restrictions but our data cannot be easily categorized. There is also a blurring problem. A lot of these existing methods do the completion by generating points around existing points. So their output tends to be fuzzy and fleshy.

Reverse engineering of CAD models

Reverse engineering is when a product or system is deconstructed in order to figure out how it was built. In CAD communities, reverse engineering mainly refers to extracting the parameters of each step in the construction process according to the complete CAD model. However, most people do this manually and there is not much research on automatic methods in this area. Point2Cyl [12] introduced a neural network which decomposed the whole model into cylinders in a geometry-grounded way by first learning underlying geometric proxies. However, it started from a complete scan of a CAD model, which is hard to get in our daily life.

3 Problem Setting

We have mentioned it many times that we need to restore the whole construction process in a parametric way, but we haven't indicate what parameters we need to solve the completion task. Before introducing our methods, we first define the extrusion cylinders and the corresponding parameters. (Part of the definitions are borrowed from Point2Cyl. [12])

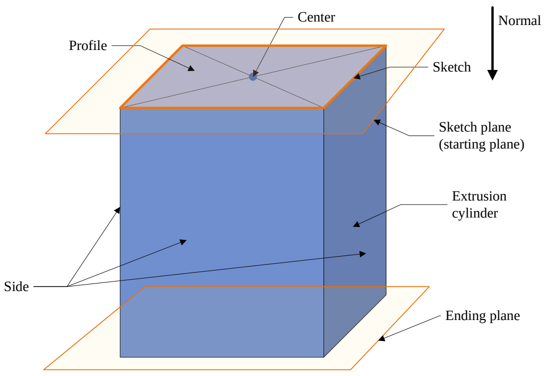

Fig. 1. An example of an extrusion cylinder

Definition 1 (Sketch and profile). *We consider a non-self-intersecting, finite area, closed loop and normalized 2D* ***sketch***

$$\tilde{\mathbf{S}} = \left\{ p(q(t)) \in \mathbb{R}^2 \mid t \in [0,1], p(q(0)) = p(q(1)) \right\}$$

for continuous function $q : [0,1] \to \mathbb{R}$ and $p : \mathbb{R} \to \mathbb{R}^2$.
The area enclosed by $\tilde{S}$ is called a ***profile****.*

Definition 2 (Sketch plane). *We define the plane containing $\tilde{S}$ as the* ***sketch plane****. It can be parameterized by a* ***center*** *$c \in \mathbb{R}^3$ and a* ***normal*** *$e \in \mathbb{S}^2$.*

The profiles can be then extruded and form extrusion cylinders.

Definition 3 (Extrusion cylinders). ***Extrusion*** *is to push the material forward along a fixed cross-sectional profile to a desired height. We define the* ***extrusion cylinder*** *as the part of 3D space that the profile passes through in this process. Each extrusion cylinder can be parameterized by an* ***normal axis*** $e \in \mathbb{S}^2$, *a* ***center*** $c \in \mathbb{R}^3$, *and the* ***extents*** $(r_{min}, r_{max}) \in (\mathbb{R} \times \mathbb{R})$. *Thus, an extrusion cylinder* $E = (e, c, \tilde{S}, r_{min}, r_{max})$. *The starting plane and the ending plane of the cylinder are defined as* ***starting plane*** *and* ***ending plane****, and the side of the cylinder is defined as* ***side****.*

Now we have defined basic extrusion cylinders. Users of CAD software need to combine different cylinders with Boolean operation to get what they want. To simplify the process, we only consider two Boolean operations: add and subtract. We assume that all the operations are done sequentially and no parallel processing is allowed. We also assume that **all adding operations come before subtracting operations**, which does not influence the generation result and can largely simplify the problem.

Definition 4 (Types of extrusion cylinders). *The building process of a CAD model can be divided into several extrusion steps. Each step we either add a cylinder to or subtract a cylinder from our current structure. If a cylinder is added to the current structure, it is an* ***adding extrusion cylinder****. Otherwise it is a* ***subtracting extrusion cylinder****.*

4 Methods

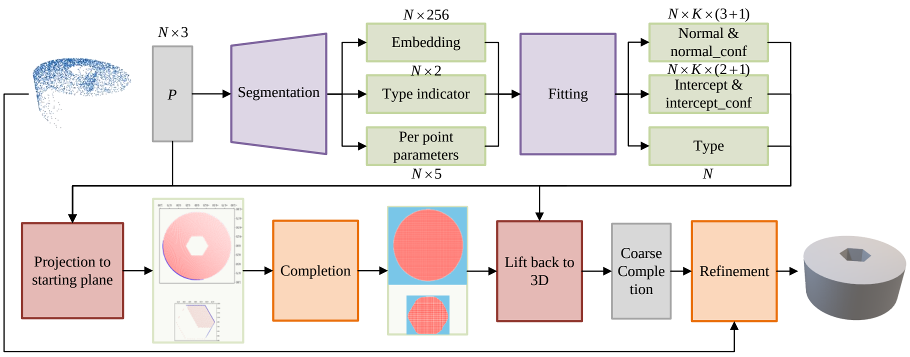

Fig. 2. The whole pipeline of our model

The overall architecture of our method is shown in Fig. 2, which consists four main parts: segmentation, fitting, completion and refinement. We will briefly introduce the first two parts, since they are mainly completed last year, and we will detail the completion and refinement parts in the following.

4.1 segmentation

For each point in the partial point cloud, we will predict its parent extrusion cylinder which is defined as the last extrusion cylinder whose surface contains this point. According to this, we can separate the partial scan into different extrusion cylinders. We will also predict whether this point belongs to a starting or ending plane or belongs to the side. This will help us in the completion part.

Inspired by HPNet [17], we train a DGCNN-based network. [8] For each point, we will output a 263-dimension vector containing the following parts:

- the semantic embedding $d \in \mathbb{R}^{256}$, used to identify the parent extrusion cylinder;
- start/end/side indicator $t \in [0, 1]$, $t \leq 0.5$ means this point is on either the starting or ending plane and $t > 0.5$ means this point is on the side of its parent extrusion cylinder;
- extrusion type indicator $o \in [0, 1]$, indicates whether its parent extrusion cylinder is an adding one or a subtracting one;
- parameters per point, including normal direction and extents.

The loss terms will not be introduced here.

At inference time, we apply mean-shift clustering [2] to the semantic embedding get the segmentation result.

4.2 fitting

In this part, we also train a DGCNN-based network. [8] The difference is that we have segmented the partial scan, and we will do feature extraction to each segment by adding a mask to the input. We also introduce a hyper-parameter *num_experts*. We make *num_experts* predictions for each point. We will use K to denote *num_experts* and K is set to 4 by default. For each point, we will predict the following:

- normal, $[K, 3]$, K predictions each with 3 dimensions;
- normal_conf, $[K]$, denoting the confidence of each prediction;
- intercept, $[K, 2]$, denoting the predictions of r_{min} and r_{max};
- intercept_conf, $[K]$, also denoting the confidence;
- type $\in$ [adding, subtracting], predicted for each extrusion cylinder.

The loss terms will not be introduced here.

4.3 completion

After segmentation and fitting, we have divided the partial scan into extrusion cylinders and get the normal axis and the extents for each extrusion. Now we need to complete each extrusion cylinder and combine them to get the final completion result.

Now let's consider about only one partial extrusion cylinder E. As defined before, $E = (e, c, \tilde{S}, r_{min}, r_{max})$ and now we only have to figure out $\tilde{S}$. We will first map all the points to the starting plane, and do the 2D completion by DeepSDF. [7]

DeepSDF We first embed a shape s into a low dimensional latent code z by a PointNet [9][10] encoder. We want to train $f_\theta(z_i, x) \in \{0,1\}$ such that given the shape code z_i and a point x, $f_\theta(z_i, x)$ denotes whether x is inside or outside the shape. (For those points on the boundary of the shape, assigning their SDF values to 0 or 1 does not affect the shape.)

Original SDF (signed distance function) outputs the signed distance of the query point to the shape. Here we simplify it to only $\{0,1\}$ in order to make the boundaries of the shape sharper and reduce blurring.

During training, we will feed complete shapes to the model to let it "remember" these shapes. We project the complete point cloud to the starting plane and use the complete shapes for training. Given a shape s_i indexed with i, we will first sample K points and their simplified signed distance values:

$$X_i = \left\{(x_j, t_j) : t_j = SDF^i(x_j) \in \{0,1\}\right\}.$$

Then we just want to maximize:

$$p_\theta\left(\boldsymbol{s}_j \mid z_i; \boldsymbol{x}_j\right) = \exp\left(-\mathcal{L}\left(f_\theta\left(\boldsymbol{z}_i, \boldsymbol{x}_j\right), s_j\right)\right).$$

At inference time, given a partial shape X', we fix the parameters θ, and the optimal shape code z can be estimated via MAP estimation:

$$\hat{\boldsymbol{z}} = \arg\min_{\boldsymbol{z}} \sum_{(\boldsymbol{x}_j, \boldsymbol{s}_j) \in X'} \mathcal{L}\left(f_\theta\left(\boldsymbol{z}, \boldsymbol{x}_j\right), s_j\right) + \frac{1}{\sigma^2}\|\boldsymbol{z}\|_2^2.$$

Visibility information We also add visibility information during inference time. Since our data are scanned from a certain viewpoint, we are sure that some part of the starting plane is empty and some part are occluded. We can only do completion in those occluded area. An example can be shown in Fig. 3.

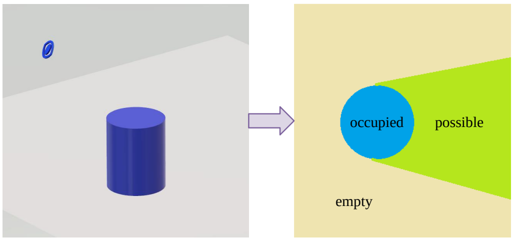

Fig. 3. Example of visibility information. There is a cylinder on the plane and we look at it from the position shown in the figure. Then we can divide the start plane into three areas. The blue one is occupied by the cylinder. The green one is where we can do completion since the cylinder blocked our view. The yellow one should be empty and we cannot do completion in this area.

For one extrusion step, let X denote the point samples of the partial scan and S denote the starting plane we are working on. After normalization, we are now only considering the square $[-1,1] \times [-1,1]$. We first divide the square into small girds: $g_{i,j} = [-1+ki, -1+k(i+1)] \times [-1+kj, -1+k(j+1)]$ where k denote the side length of one small gird. Let $c_{i,j}$ be the center of grid $g_{i,j}$.

We want to know for each grid whether it is occluded by something or empty. Consider each point in the partial scan. For one such point p, connect the camera position it and find the intersection q with the starting plane S. Then, we just compare the depth of p with that of q and we know whether q is blocked by p or not. If q is blocked, we find the grid $g_{i,j}$ which contains q and mark it. Finally, we find all the unmarked grids $g_{i,j}$ and add $(c_{i,j}, 1)$ to X, which means we have already know this gird is outside the shape from the visibility information.

Fig. 4 shows one example of X before and after adding visibility information. It works like a constraint to the completion process and also help deal with the blurring problem.

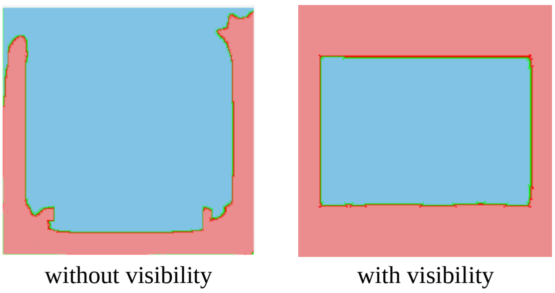

Fig. 4. Comparison between adding & not adding visibility information.

4.4 refinement

Now we have predicted all the parameters used in the construction process. We can simply combine all the extrusion cylinders to get the final shape. Since we use a code z to describe the shape, we cannot get explicit curves. But we can use methods like Marching Cubes [6] to reconstruct the completion result.

However, our completion is easily influenced by the quality of predicted parameters. Fig. 5 shows that for a long narrow cylinder, a little aberration in the prediction of extrusion direction makes a big difference. Fig. 6 shows that a little noise in segmentation results in quite different completion result.

In order to fix this, we try to reuse the original partial scan to refine our completion. Referring to SPAGHETTI [3], we use attention mechanism to do the refinement.

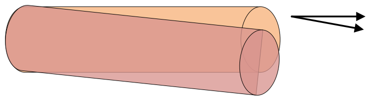

Fig. 5. A small error in direction makes a big difference.

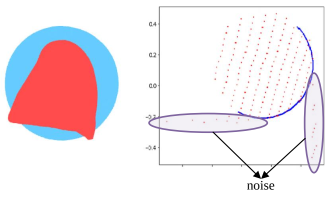

Fig. 6. Small noise in segmentation largely influence the completion result.

Let P denote the original partial scan and Q denotes the coarse completion. We first use PointNet encoder to extract features from each point cloud:

$$L = f_1(P) \in \mathbb{R}^{n*d}, F = f_2(Q) \in \mathbb{R}^{m*d}.$$

Then we apply a cross attention:

$$\begin{aligned} & q = FW^q, k = LW^k, v = LW^v, \\ & W^q \in \mathbb{R}^{d\times d_1}, W^k \in \mathbb{R}^{d\times d_1}, W^v \in \mathbb{R}^{d\times d_2}, \\ & \mathrm{Z} = \text{ cross-attention } (\mathrm{L}, \mathrm{F}) \in \mathbb{R}^{m\times d_2}. \end{aligned}$$

We let all the points in the coarse completion attend to the ground truth partial scan and aggregate the information. We get the final result still in an implicit way. Given an input $x \in [-1,1]^3$, we first do a positional encoding $x_0 = \mathrm{PE}(x) \in \mathbb{R}^{d_{pe}}$ using a learned positional encoding layer:

$$\mathrm{PE}(x) = \sin\left(a\left(W_{pe}x + B_{pe}\right)\right),$$

where W_{pe} and B_{pe} are learned parameters and a is a fixed scalar. Then we use a sequence of T cross attention layers to calculate the refined occupancy of x. Layer t outputs the embedding x^{t+1}, calculated by the cross attention of x^t with Z:

$$\begin{gathered} q = x_t W^{q'}, \qquad k = ZW^{k'}, \qquad v = ZW^{v'}, \\ W^{q'} \in \mathbb{R}^{d_{pe}\times d_k}, W^{k'} \in \mathbb{R}^{d_2\times d_k}, W^{v'} \in \mathbb{R}^{d_k\times d_{pe}}. \end{gathered}$$

Let $\hat{x}$ denote the final output $x_T \in \mathbb{R}^{d_{pe}}$. Then we use a MLP which decodes $\hat{x}$ to an occupancy indicator $\hat{y}$. Given a complete point cloud, we sample 40000 points near the surface and 10000 points randomly. These samples and their corresponding ground truth occpancy form the training data X. Here the occupancy loss is given by the binary cross entropy loss (BCE):

$$\mathcal{L}_{occ} = \frac{1}{|X|} \sum_{(x,y)\in X} BCE(y, \hat{y}).$$

5 Experiments

5.1 Dataset

We use Fusion 360 Gallery reconstruction Dataset [15]. This is a CAD dataset containing 8625 human designed CAD programs, expressed in sketch and extrude, which exactly matches our needs.

5.2 Evaluation

We choose Chamfer Distance as our evaluation metric. For a prediction $\mathcal{P}$, the Chamfer Distance between it and the ground truth point cloud $\mathcal{G}$ is calculated by:

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

It represents the "relative distance" or the similarity between the two point clouds.

5.3 Results

Our model can be divided into four stages as stated above. We did not make much modifications on the first two stages segmentation and fitting and the results of these two stages are presented in last year's report.

We mainly focus on the third part: completion. The results are shown below:

	Training	**Validation**
PoinTr	13.6270	13.4749
Ours (without visibility)	13.3402	15.3301
Ours (with visibility)	9.4716	11.5760

Table 1. CD distance (L1) * 1000

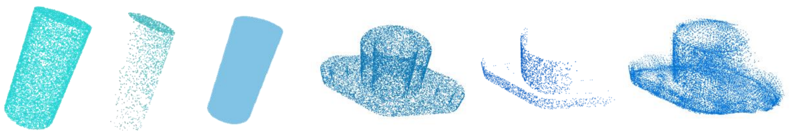

Fig. 7. Some visualized results.

Our model works well on training set primarily because it is not sensitive to categories. With the latent code z, it can generalize to all types of models.

Visibility information is also quite important. Since most of our data are quite simple, with only one or two extrusion steps, adding visibility information can directly determine the real results.

We haven't done experiments with the refinement part yet. We will continue working on this project after the AI+X course.

Some visualized results are shown in Fig. 7.

6 Analysis & Conclusion

In this project, we continue the previous work on CAD-specific scan completion. Besides completing their framework, we add visibility information to improve completion quality and introduce cross attention to refine the result. This task is quite important in our daily life. We can simply take a photo of some furniture and let the model to automatically generate a CAD model. In factories, as mentioned in the introduction part, this model can also help mechanical arms to better understand the objects they are working on and operate more accurately. However, to complete this project, we still have a long way to go.

The biggest problem is about the noise. We haven't experiment with our refinement part yet. But we believe even if with this part, we still cannot completely eliminate the impact brought by errors in segmentation and fitting part. We certainly need to improve the performance of the first two parts, but these two parts will always have errors no matter how well we do. Thus, find a better way to do refinement is a key problem.

What's more, we still need to cope with the blurring problem. We cannot always turn a rectangle into a rounded rectangle. Now we just change the signed distance values into occupancy, but we expect a more direct and explicit solution to this problem. We think it is better to directly add this problem to the loss function as a penalty, but we haven't found a good way yet.

In the future, we would like to cope with the problems above and do more experiments. We believe we could make a big step on it!!!

Contribution

Contributions are listed below.

Zongtai Li:

- data processing
  - re-generate data and align camera parameters
  - write interface to facilitate calling data
- segmentation and fitting part
  - code reading and review papers
  - visualize and analyze problems
  - improve performance and efficiency
- completion and refinement part
  - code reading and review papers
  - design the completion model
  - design the refinement
  - write all codes, do all experiments and visualizations
- others
  - prepare the opening and mid-term report
  - prepare the final report, make PPT and write speech draft
  - write the final paper

Minji Yang:

- completion and refinement part
  - code reading and review papers
  - design the refinement
- others
  - write discussion report

Thank Prof. Li Yi and Dr. Rui Chen for instruction.
Special thank to Xiaodi Yuan and Yushuo Chen for their gratuitous help.

References

1. Chen, X., Chen, B., Mitra, N.J.: Unpaired point cloud completion on real scans using adversarial training. Learning (2020)
2. Cheng, Y.: Mean shift, mode seeking, and clustering. IEEE Transactions on Pattern Analysis and Machine Intelligence (1995)
3. Hertz, A., Perel, O., Giryes, R., Sorkine-Hornung, O., Cohen-Or, D.: Spaghetti: Editing implicit shapes through part aware generation
4. Hu, T., Han, Z., Shrivastava, A., Zwicker, M.: Render4completion: Synthesizing multi-view depth maps for 3d shape completion (2019)
5. Huang, Z., Yu, Y., Xu, J., Ni, F., Le, X.: Pf-net: Point fractal network for 3d point cloud completion. computer vision and pattern recognition (2020)
6. Lorensen, W.E., Cline, H.E.: Marching cubes: A high resolution 3d surface construction algorithm. international conference on computer graphics and interactive techniques (1987)

7. Park, J.J., Florence, P.R., Straub, J., Newcombe, R., Lovegrove, S.: Deepsdf: Learning continuous signed distance functions for shape representation. computer vision and pattern recognition (2019)
8. Phan, A.V., Nguyen, M.L., Nguyen, Y.L.H., Bui, L.T.: Dgcnn: A convolutional neural network over large-scale labeled graphs. Neural Networks (2018)
9. Qi, C.R., Su, H., Mo, K., Guibas, L.J.: Pointnet: Deep learning on point sets for 3d classification and segmentation. computer vision and pattern recognition (2016)
10. Qi, C.R., Yi, L., Su, H., Guibas, L.J.: Pointnet++: Deep hierarchical feature learning on point sets in a metric space. neural information processing systems (2017)
11. Tchapmi, L.P., Kosaraju, V., Rezatofighi, H., Reid, I., Savarese, S.: Topnet: Structural point cloud decoder. computer vision and pattern recognition (2019)
12. Uy, M.A., yu Chang, Y., Sung, M., Goel, P., Lambourne, J., Birdal, T., Guibas, L.: Point2cyl: Reverse engineering 3d objects from point clouds to extrusion cylinders
13. Wang, X., Ang, M.H., Lee, G.H.: Cascaded refinement network for point cloud completion. computer vision and pattern recognition (2020)
14. Wen, X., Li, T., Han, Z., Liu, Y.S.: Point cloud completion by skip-attention network with hierarchical folding. arXiv: Computer Vision and Pattern Recognition (2020)
15. Willis, K.D., Pu, Y., Luo, J., Chu, H., Du, T., Lambourne, J.G., Solar-Lezama, A., Matusik, W.: Fusion 360 gallery: A dataset and environment for programmatic cad construction from human design sequences. ACM Transactions on Graphics (2020)
16. Xiang, P., Wen, X., Liu, Y.S., Cao, Y.P., Wan, P., Zheng, W., Han, Z.: Snowflakenet: Point cloud completion by snowflake point deconvolution with skip-transformer. international conference on computer vision (2021)
17. Yan, S., Yang, Z., Ma, C., Huang, H., Vouga, E., Huang, Q.: Hpnet: Deep primitive segmentation using hybrid representations. arXiv: Computer Vision and Pattern Recognition (2021)
18. Yu, X., Rao, Y., Wang, Z., Liu, Z., Lu, J., Zhou, J.: Pointr: Diverse point cloud completion with geometry-aware transformers. international conference on computer vision (2021)
19. Yuan, W., Khot, T., Held, D., Mertz, C., Hebert, M.: Pcn: Point completion network. international conference on 3d vision (2018)
20. Yuan, X., Chen, Y.: Parametric cad scan completion with extrusion cylinders (2022)