Surface and Edge Detection for Primitive Fitting of Point Clouds

Yuanqi Li
State Key Lab for Novel Software Technology, Nanjing University
Nanjing, Jiangsu, China
yuanqili@smail.nju.edu.cn

Shun Liu
State Key Lab for Novel Software Technology, Nanjing University
Nanjing, Jiangsu, China
liushun@smail.nju.edu.cn

Xinran Yang
State Key Lab for Novel Software Technology, Nanjing University
Nanjing, Jiangsu, China
xryang@smail.nju.edu.cn

Jianwei Guo*
MAIS, Institute of Automation, Chinese Academy of Sciences
Beijing, China
jianwei.guo@nlpr.ia.ac.cn

Jie Guo
State Key Lab for Novel Software Technology, Nanjing University
Nanjing, Jiangsu, China
guojie@nju.edu.cn

Yanwen Guo*
State Key Lab for Novel Software Technology, Nanjing University
Nanjing, Jiangsu, China
ywguo@nju.edu.cn

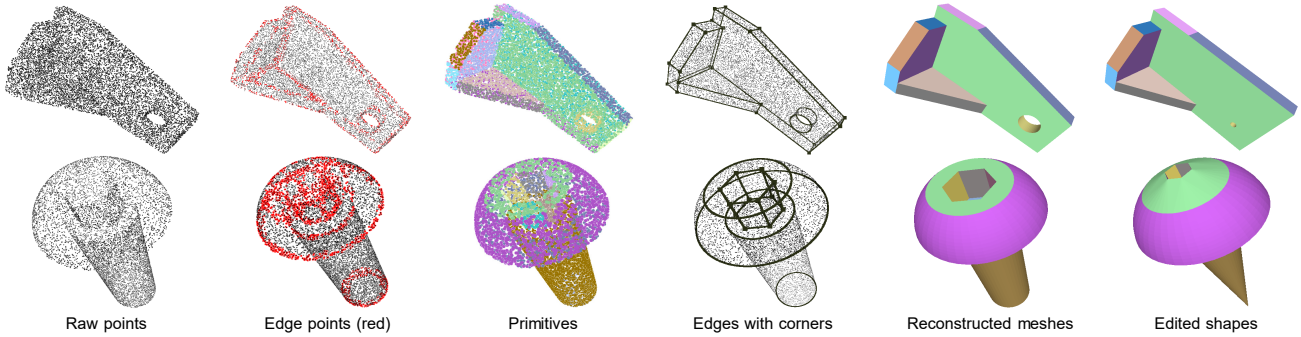

Figure 1: Given a raw point cloud, our pipeline fits it with a set of geometric primitives by recovering surfaces, edges, and corners simultaneously. The unified neural framework obtains a parametric and compact model representation, which enables us to reconstruct the CAD model accurately and allows the users to edit its shape easily.

ABSTRACT

Fitting primitives for point cloud data to obtain a structural representation has been widely adopted for reverse engineering and other graphics applications. Existing segmentation-based approaches only segment primitive patches but ignore edges that indicate boundaries of primitives, leading to inaccurate and incomplete reconstruction. To fill the gap, we present a novel surface and edge detection network (SED-Net) for accurate geometric primitive fitting of point clouds. The key idea is to learn parametric surfaces (including B-spline patches) and edges jointly that can be assembled into a regularized and seamless CAD model in one unified and efficient framework. SED-Net is equipped with a two-branch structure to extract type and edge features and geometry features of primitives. At the core of our network is a two-stage feature fusion mechanism to utilize the type, edge and geometry features fully. Precisely detected surface patches can be employed as contextual information to facilitate the detection of edges and corners. Benefiting from the simultaneous detection of surfaces and edges, we can obtain a parametric and compact model representation. This enables us to represent a CAD model with predefined primitive-specific meshes and also allows users to edit its shape easily. Extensive experiments and comparisons against previous methods demonstrate our effectiveness and superiority.

CCS CONCEPTS

• **Computing methodologies** → **Point-based models**; **Parametric curve and surface models**.

KEYWORDS

Primitive fitting, point cloud, shape reconstruction, deep neural network

ACM Reference Format:
Yuanqi Li, Shun Liu, Xinran Yang, Jianwei Guo, Jie Guo, and Yanwen Guo. 2023. Surface and Edge Detection for Primitive Fitting of Point Clouds. In *Special Interest Group on Computer Graphics and Interactive Techniques Conference Conference Proceedings (SIGGRAPH '23 Conference Proceedings), August 06–10, 2023, Los Angeles, CA, USA.* ACM, New York, NY, USA, 10 pages. https://doi.org/10.1145/3588432.3591522

*Jianwei Guo and Yanwen Guo are the co-corresponding authors.

SIGGRAPH '23 Conference Proceedings, August 06–10, 2023, Los Angeles, CA, USA

ACM ISBN 979-8-4007-0159-7/23/08…$15.00
https://doi.org/10.1145/3588432.3591522

1 INTRODUCTION

As the point cloud is one of the most convenient forms directly captured by 3D scanning devices, automatically reconstructing

CAD models from point cloud data can save a lot of labor for reverse engineering, digital twin, and other industrial applications. To solve this problem, previous works predict a set of bounding primitives, such as cubes and superquadrics, to roughly abstract input shapes [31, 36, 38, 39, 41, 60]. However, this straightforward solution tends to lose a lot of geometric details. The conversion of mapping a 3D point cloud to a collection of geometric primitives that best fit the underlying shape would undoubtedly benefit many downstream tasks. However, leading primitive fitting methods [20, 37, 53] only segment primitive patches, but ignore edges which are important visual features. As a result, the reconstructed model would inevitably suffer from visual artifacts, due to the existence of errors in surface fitting which could create gaps between patches. Guo et al. [7] detect primitive patches, edges, corners, and their topology relationship by using a complex network. Indeed, edge information is fully discovered by Guo et al. [7] but ignored in other methods [37, 53]. We argue that edges in fact indicate boundaries of primitive patches, and incorporating edge information would certainly facilitate the segmentation of primitive patches to achieve more accurate and seamless 3D shape reconstruction. However, fully discovering edges, *i.e.*, predicting edges and their relationships to patches, requires a complex network, which increases training difficulty. Incorrectly predicted relationships would also lead to failure, which injures the robustness. We propose a compromise between the two technical routes by leveraging edge information in an effective network.

Given a point cloud, our goal is to predict a set of geometric primitives for generating a seamless and high-quality mesh model, which is consistent with the underlying shape. What remarkably distinguishes our method from previous approaches lies in that (i) we utilize edge information to boost the performance of primitive segmentation, (ii) we implement a joint fitting strategy to obtain primitive patches and edges as the clear boundaries consecutively by using our precisely segmented primitives and edge points. To this end, we propose the *Surface and Edge Detection Network* (SED-Net), which achieves the detection of surface patches and edges to fit geometric primitives by exploring their relationship in an end-to-end deep learning-based framework.

Our SED-Net contains two branches to separately extract type and edge features and local geometric features. Two individual point decoders are introduced to extract type and boundary information in the classification branch and geometry information from the instance segmentation branch. The core of our network is a *two-stage feature fusion mechanism* which utilizes the type, edge and geometric features fully. In the early fusion stage, type and edge features are embedded in the latent space of primitive features and added to primitive features to combine the type and boundary information with local geometric information. In the late fusion stage, predicted edges and types are concatenated and encoded into the instance segmentation branch. The primitive features are thus further aligned with type and edge features of the two learned maps. Taking advantage of features from edges and types, neighboring primitives with similar geometry features and obscure boundaries can be correctly distinguished. We design a hybrid loss function to regress primitives together with their types and edges.

Taking advantage of the detected edge points from our network, the primitive patches are fitted accurately by excluding the influence from the wrong segmentation near edges. We predefine a corresponding compact mesh representation for each geometric primitive. With the final fitted primitives, we are able to construct a compact and seamless model with such predefined primitive-specific meshes, as shown in Fig. 1. Furthermore, a parametric representation can be obtained, allowing the users to edit its shape easily.

To summarize, our work makes the following contributions:

- SED-Net, a novel neural network with a two-branch structure, which separately extracts the type and edge features and the geometric features, to boost the primitive segmentation performance on point cloud data.
- A two-stage feature fusion mechanism that fully utilizes classification information including types and edges, by fusing type and edge features to the instance segmentation branch.
- A joint fitting strategy based on our simultaneously detected primitives and edge points, which enables faithful reconstruction of a compact, seamless and editable shape with accurate patches, edges and corners.

2 RELATED WORK

Primitive fitting. Traditionally, RANSAC [6, 22, 34], parameter space [23, 32], primitive growing [2, 29], and variational surface fitting [5, 52, 58] are mainstream methodologies of approximating and abstracting 3D shapes. Kaiser et al. [13] provide a survey on primitive detection of 3D data. These methods require careful parameter tuning for each shape category but our learning-based approach is not user-aided.

With the development of deep learning, neural networks are used to deal with the unordered point cloud data [4, 21, 40, 47, 59] and inspire many applications [10, 26, 44, 45, 54]. Learning-based CAD reconstruction methods represent 3D shapes by either (i) using abstract representations or CAD instructions, or (ii) using parametric surfaces. In category (i), cuboids [39, 41, 60], rounded cuboids [38], superquadrics [31], configurations of planes [57], half-spaces [8], box sequences [24] are learned by networks to abstract 3D shapes. Furthermore, using constructive solid geometry (CSG) operations is also a popular solution [11, 12, 14, 17, 18, 33, 35, 36, 42, 48–51, 55]. In category (ii), Li et al. [20] propose SPFN, which segments point clouds and fits several primitive patches including plane, cylinder, cone and sphere. However, the structure of SPFN limits the number of segmented primitives not exceeding 20. Lê et al. [19] extend SPFN to a cascaded structure to deal with high-resolution point clouds. Sharma et al. [37] propose ParSeNet, which also fits B-spline patches. But ParSeNet does not fully utilize information of shape geometry. In contrast, our method exploits type and boundary information to promote the performance of primitive segmentation. Yan et al. [53] propose HPNet, which leverages hybrid representations to obtain primitives with smooth boundaries. However, their consistency adjacency matrix takes a lot of computation, which increases the inference time. Our network uses a novel two-stage feature fusion strategy to utilize type and edge features effectively and efficiently. Recently, Guo et al. [7] detected primitive patches,

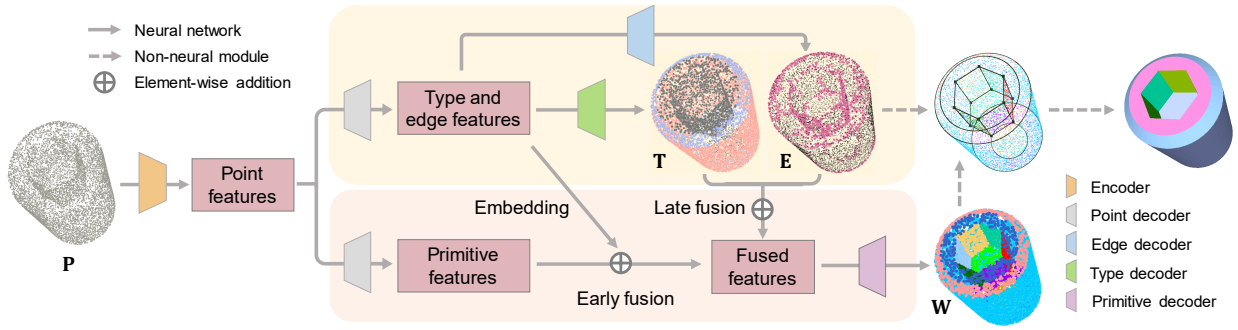

Figure 2: Illustration of our SED-Net with a two-branch structure, including a classification branch and a instance segmentation branch. After extracting point features by an encoder, we apply two point decoders to obtain type and edge features and primitive features, separately. A two-stage feature fusion mechanism is then proposed to utilize features from the two branches. Type and edge features are embedded in the primitive feature space and further fused by addition. The edge map E and types T are regressed by two decoders respectively. E and T are concatenated together and encoded to early fused features. We regress segmentation descriptors by a decoder from late fused features and segment points by mean-shift clustering. Finally, edges and corners are detected and the compact model is reconstructed by predefined primitive-specific meshes.

edges, corners and their topology relationships. They trained a complex network that consumes a lot of computational resources.

Edge detection and point cloud consolidation. As important geometric features, edges are critical to compact shape reconstruction and point cloud consolidation. Öztireli et al. [30] identify and preserve sharp features by using moving least squares. Huang et al. [9] propose EAR to compute reliable normals and progressively resample the point set for obtaining consolidated edges. Awrangjeb [1] and Lin et al. [25] extract line segments from outdoor large-scale point clouds. Yu et al. [56] propose EC-Net, which is a deep neural network, to upsample point cloud and consolidate edges. Wang et al. [46] propose the learning-based PIE-Net to infer parametric edges and corners of point clouds. Metzer et al. [28] propose a feature-aware point cloud consolidation network. Matveev et al. [27] propose DEF to predict the distance to edges of each point to reconstruct edges effectively. However, the distance field may be obscure between close feature curves. In summary, previous edge detection methods only detect edge points and their attributes, hence information from non-edge regions is not fully utilized. Our SED-Net treats edge detection and primitive segmentation as dual tasks, which can achieve better performance.

3 METHOD

3.1 Problem Statement

Given a 3D point cloud with (or without) normals $\mathbf{P} = \{\mathbf{p}_i \in \mathbb{R}^3 \text{ or } \mathbb{R}^6, i = 1, 2, ..., N\}$, we aim to reconstruct a complete structured CAD model. The problem can be decomposed into two subproblems: (i) extract a collection of geometric primitives whose combination accurately fits the input shape, and (ii) predict a set of edges constituting clear boundaries of the primitives. Specifically, we segment the input shape into K primitives with primitive type $\mathbf{T} = \{\mathbf{t}_i \in [0, 1]^L\}$, where we use $\mathbf{W} = \{\mathbf{w}_i \in [0, 1]^K\}$ to represent the point-to-primitive membership. To accommodate the training datasets, we set $L = 6$ as we restrict to the following types of geometric primitives: plane, cylinder, cone, sphere, and open and closed B-spline patch. We set $K = 128$ following [37] and [53]. The edges are the intersection (*i.e.,* line, circle, ellipse, and other freeform curves) of the surface primitives. We also predict a binary edge classification $\mathbf{E} = \{\mathbf{e}_i \in [0, 1]^2\}$, which classifies a point is an edge point or not. Then the parameters of primitive patches (excluding B-splines) and parametric edges (excluding freeform curves), as well as the coordinates of corners can be directly fitted and calculated. To determine the parameters of B-splines, we reimplement the fitting approach introduced by SplineNet [37]. The freeform curves can be represented as a set of piecewise linear segments, which can be efficiently calculated by the intersection of triangular meshes of primitives. Finally, after learning the geometric elements with corresponding parameters, we can reconstruct a compact and seamless CAD model by using predefined primitive-specific meshes and fitted B-splines.

3.2 Learning to Instantiate Primitives

We exploit the assumption that accurate edge information would greatly promote the performance of surface primitive segmentation. Therefore, SED-Net is designed to learn an accurate primitive segmentation of the input shape by taking full advantage of the type and boundary information of primitives. Our network architecture is summarized in Fig. 2, comprising two novel components: a two-branch structure and a two-stage feature fusion mechanism. Given the input point cloud, we apply a graph CNN encoder equipped with three stacked edge convolution layers (EdgeConv) [47] to extract a point feature matrix $\mathbf{F} \in \mathbb{R}^{N\times 1280}$ including local and global features. Then, from the same point decoder, we obtain two feature matrices for the instance segmentation branch ($\mathbf{F_P} \in \mathbb{R}^{N\times 256}$), and classification branch ($\mathbf{F_C} \in \mathbb{R}^{N\times 256}$), respectively. Then, the type and edge features $\mathbf{F_C}$ in classification branch are decoded to obtain an edge map $\mathbf{E}$ and a type map $\mathbf{T}$ by supervision of our edge loss and type loss.

In order to make full use of primitive type and boundary information, we propose a novel two-stage feature fusion mechanism. In the first fusion stage, the output features of the classification branch

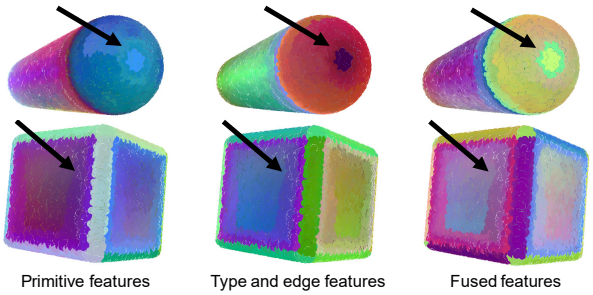

Figure 3: Visualization of the features extracted by the two branches of our network. Arrows point out key areas. The type and edge features (middle) are different in neighboring primitives with an obscure boundary (top) and different near and far from edges (bottom).

are embedded in the latent space of primitive features by implementing a fully connected (FC) layer with batch normalization and ReLU function. The embedded feature matrix is then added with the primitive features in an element-wise manner. In the second fusion stage, the edge map **E** and type map **T** are concatenated together and encoded into the instance segmentation branch by another FC layer and ReLU function. Thus the primitive features are further aligned with type and edge features, which contributes to distinguishing primitives with similar geometry features or complex edges. Finally, a primitive decoder is used to regress the primitive segmentation features, and a mean-shift clustering procedure is applied to the point-wise features to segment primitives, which is similar to [37].

Mathematically, the two-stage feature fusion mechanism can be formulated as:

$$\mathbf{F}_{early} = \mathbf{F}_{\mathbf{P}} + \mu\,\text{Embed}(\mathbf{F}_{\mathbf{C}}), \tag{1}$$

$$\mathbf{F}_{late} = \mathbf{F}_{early} + \nu\,\text{Encode}\Big(\text{Concat}(\mathbf{T},\mathbf{E})\Big), \tag{2}$$

where $\mathbf{F}$ denotes a feature matrix, $\mathbf{F}_{early} \in \mathbb{R}^{N\times 256}$, $\mathbf{F}_{late} \in \mathbb{R}^{N\times 256}$. In the implementation, we set $\mu = 0.2$, and $\nu = 0.2$.

Benefiting from our two-branch structure, the boundary and type information can be extracted by the classification branch. In contrast, local geometric features are extracted by the instance segmentation branch. Our two-stage feature fusion mechanism fully utilizes these features, by fusing type and edge information with local geometry information. Type information can distinguish neighboring primitives belonging to different types. Edges indicate the boundaries of each primitive. They are both important supplements for the instance segmentation branch. We visualize features in each branch before the early fusion in Fig. 3, by using t-SNE [43] to embed the feature matrices to 3-dimensional vectors. The top row of Fig. 3 shows that the classification branch can separate the neighboring primitives with a smooth boundary, which is hard to be learned by the instance segmentation branch individually due to the similar local geometric features. The bottom row of Fig. 3 reveals that the classification branch learns different features near and far from the edges. In contrast, the instance segmentation branch learns the geometric features of each surface primitive patch, which encodes information on patch types and edges for further improvement. Consequently, the fused features contribute to the differentiation of neighboring primitives and clear recognition of boundaries. Note that we do not extract type features and edge features individually, because the three-branch structure would increase the network complexity and do harm to the practicability of the network.

3.3 Loss Functions

To precisely detect primitives, we train SED-Net through the minimization of the sum of three objective terms:

$$\mathcal{L} = \alpha\mathcal{L}_{type} + \beta\mathcal{L}_{edge} + \mathcal{L}_{edge\_emb}, \tag{3}$$

where $\mathcal{L}_{type}$ and $\mathcal{L}_{edge}$ represent the type loss and edge loss used for training the type and edge classification, respectively. $\mathcal{L}_{edge\_emb}$ denotes the edge-enhanced embedding loss for supervising the segmentation of primitive patches. We set the weight parameters $\alpha = 1$ and $\beta = 1$ in default, which generally works well in our experiments.

Type loss is calculated in a point-wise manner. A cross-entropy H_{CE} is employed to measure the loss between the predicted point-wise primitive type t_i and the ground truth label $\hat{t}_i$:

$$\mathcal{L}_{type} = \frac{1}{N}\sum_{i=1}^{N} H_{CE}(t_i, \hat{t}_i). \tag{4}$$

Edge loss is used to classify whether or not a point is an edge point. We denote $\mathbf{p}_i$ as an edge point if its nearest distance to the edges of the ground truth shape is less than a threshold ϵ ($\epsilon = 7\times 10^{-3}$ in default). We employ a weighted cross-entropy loss to define the edge loss:

$$\mathcal{L}_{edge} = \frac{1}{N}\Big(\sum_{\mathbf{p}_i\in\Phi_e} H_{CE}(e_i,\hat{e}_i) + \frac{|\Phi_e|}{|\mathbf{P}|-|\Phi_e|}\sum_{\mathbf{p}_j\notin\Phi_e} H_{CE}(e_j,\hat{e}_j)\Big), \tag{5}$$

where $\mathbf{p}_i,\ \mathbf{p}_j \in \mathcal{P}$, $e_i \in [0,1]$ indicates whether $\mathbf{p}_i$ is an edge point, Φ_e is the ground truth edge point set, $|\cdot|$ denotes point number.

Edge enhanced embedding loss can pull the primitive segmentation descriptors of points close to each other in the same primitive patch, and push the descriptors of different patches farther apart. Edge points contribute larger value in this term, thereby increasing accuracy near edges:

$$\begin{aligned}\mathcal{L}_{edge\_emb} = \frac{1}{K}\sum_{k=1}^{K}\frac{1}{|\hat{\mathcal{P}}_k|}\sum_{\mathbf{p}_i\in\hat{\mathcal{P}}_k}\lambda_i \max(\|\mathbf{d}_i - \mathbf{d}_{\hat{\mathcal{P}}_k}\| - \delta_1, 0) \\ + \frac{1}{K(K-1)}\sum_{k<k'}\max(\delta_2 - \|\mathbf{d}_{\hat{\mathcal{P}}_k} - \mathbf{d}_{\hat{\mathcal{P}}_{k'}}\|, 0),\end{aligned} \tag{6}$$

where $\hat{\mathcal{P}}_k$ represents k-th ground truth primitive patch, $\mathbf{d}_i$ denotes the primitive segmentation descriptor of $\mathbf{p}_i$, $\mathbf{d}_{\hat{\mathcal{P}}_k} = \sum_{\mathbf{p}_i\in\hat{\mathcal{P}}_k}\mathbf{d}_i/|\hat{\mathcal{P}}_k|$. λ_i takes different values for an edge point or not, when $e_i = 1$, $\lambda_i = 1.2$, when $e_i = 0$, $\lambda_i = 1$. We set $\delta_1 = 0.5$, $\delta_2 = 1.5$.

3.4 Primitive Fitting

Our network learns the edge map, type map, and primitive segmentation from the raw point cloud. We then fit each primitive with the assistance of the edge map and type map. For a primitive patch $\mathcal{P}$, its type $\mathcal{T}(\mathcal{P})$ is determined by the mode of $t(\mathbf{p})$, where $\mathbf{p}\in\mathcal{P}$, $t(\cdot)$ denotes predicted type of $\mathbf{p}$. To eliminate the negative influence from wrongly segmented points near edges, we fit each basic

Figure 4: Parametric edges can be calculated as the intersection of primitives (*e.g.*, plane, cylinder, cone, sphere).

primitive by applying the least-squares fitting to those primitives excluding edge points:

$$\mathcal{A}_i^* = \arg\min_{\mathcal{A}_i} \sum_{\mathbf{p}\in\mathcal{P}_i \& \mathbf{p}\notin\mathcal{E}} (\text{dist}(\mathbf{p}, S(\mathcal{A}_i))), \tag{7}$$

where $\mathcal{A}_i$ and $\mathcal{P}_i$ represent the parameters and the point set of i-th primitive patch respectively, $\mathcal{E}$ is the edge point set classified by SED-Net, and $\text{dist}(\cdot)$ denotes the point-to-surface distance between a point $\mathbf{p}$ and a surface patch $S(\cdot)$ represented by $\mathcal{A}_i$. For open and closed B-spline patches, we follow SplineNet [37] to predict B-spline control points, from which a discrete mesh can be generated. Note that the predicted edge points in the B-spline patches should also be input into SplineNet.

Precisely segmented and fitted primitive patches can be used as contextual information to obtain accurate parametric edges. Lines, circles and ellipses are calculated as intersections of planes, cylinders, cones and spheres, which is illustrated in Fig. 4. Intersection points of parametric edges, which are corners of the shape, can also be obtained. Then the predefined meshes of each primitive type can be used to reconstruct a compact shape. However, freeform intersections are difficult to be represented as a parametric style, for example, a cylinder intersecting with another cylinder or a B-spline intersecting with other basic primitives. To represent such freeform edges, we generate explicit geometry by using piecewise line segments to approximate them. Specifically, according to the fitted parameters of primitive patches and the corresponding parametric edges, predefined meshes which are restricted by calculated parametric edges and the bounding box can be generated. Two triangles from two patches have an intersection line, thus piecewise line segments are obtained to represent freeform edges. Since a tiny fitting error may lead to no intersection of two primitive patches, we implement a jitter of which the amplitude is set to 0.01 on the primitive patches for obtaining edges. With the freeform edges calculated by the intersection of two triangulated mesh primitives, we cut the mesh into two parts and the part attached with original points is retained. Fig. 5 displays two examples of freeform edges which are intersections of cylinders or cones. Fig. 6 shows predefined meshes that are cut by piecewise linear segments.

4 EXPERIMENTS

4.1 Implementation Details

We evaluate the performance of SED-Net on the widely-used ABC-Parts dataset [37] (a portion of ABC dataset [16]) which provides a large source of 3D CAD models with labeled surface primitive patches and edge information. Following previous works [37] [53] [7], we use $24K$, $4K$, $4K$ models for training, validating and testing respectively, where each model contains 10000 points.

We stack three EdgeConv layers of $[3, 64, 64, 128]$ to extract local features and concatenate them to a feature matrix of $N \times 256$. A fully connected layer and a max-pooling layer are used to extract a global feature vector of 1×1024. Then, we tile global features and concatenate them with local features to yield a matrix of $N \times (256 + 1024)$, which is our point feature. The point decoder, edge decoder, type decoder, and primitive decoder are shared MLP layers of size $[1280, 256]$, $[256, 2]$, $[256, 6]$ and $[256, 128]$, respectively. The primitive segmentation descriptors are used to obtain point-to-primitive membership $\mathbf{W}$ by a mean-shift clustering procedure. SED-Net is trained with batchsize=8 for 100 epochs, which takes 20 hours on two NVIDIA-2080Ti GPUs.

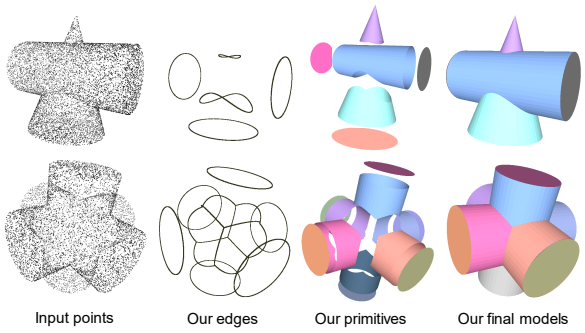

Figure 5: The intersection of cylinders or cones is hard to be represented by parametric curves. We use piece-wise linear segments to represent them.

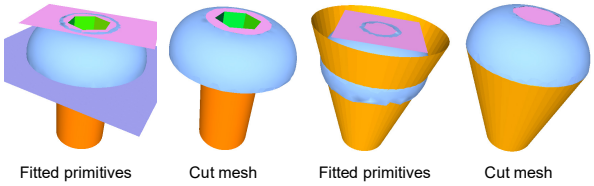

Figure 6: Meshes of B-splines fitted by a network and the primitive meshes are cut by the edges. Blue patches are B-splines.

4.2 Comparisons

To demonstrate the effectiveness of our SED-Net, we compare it against state-of-the-art methods on primitive fitting for point clouds, including ParSeNet [37], HPNet [53], ComplexGen [7], as well as an edge detection method, PIE-Net [46].

Qualitative comparison. Fig. 7 displays the qualitative comparison results of primitive segmentation. It can be seen that the competitors may not distinguish primitives which have similar geometric and type features. For example, in the first column, all previous methods detect the two separate circles as one primitive. Taking the nail model in the second column and the handle of the hammer model in the third column as another two examples, cylinders with similar geometric and type features are detected as one primitive by previous methods. Furthermore, Fig. 7 obviously shows that previous methods over-segment some large primitives. By contrast, our SED-Net, which takes advantage of fused type information and boundary information from a two-branch structure, can precisely detect primitives with similar features and segment points robustly.

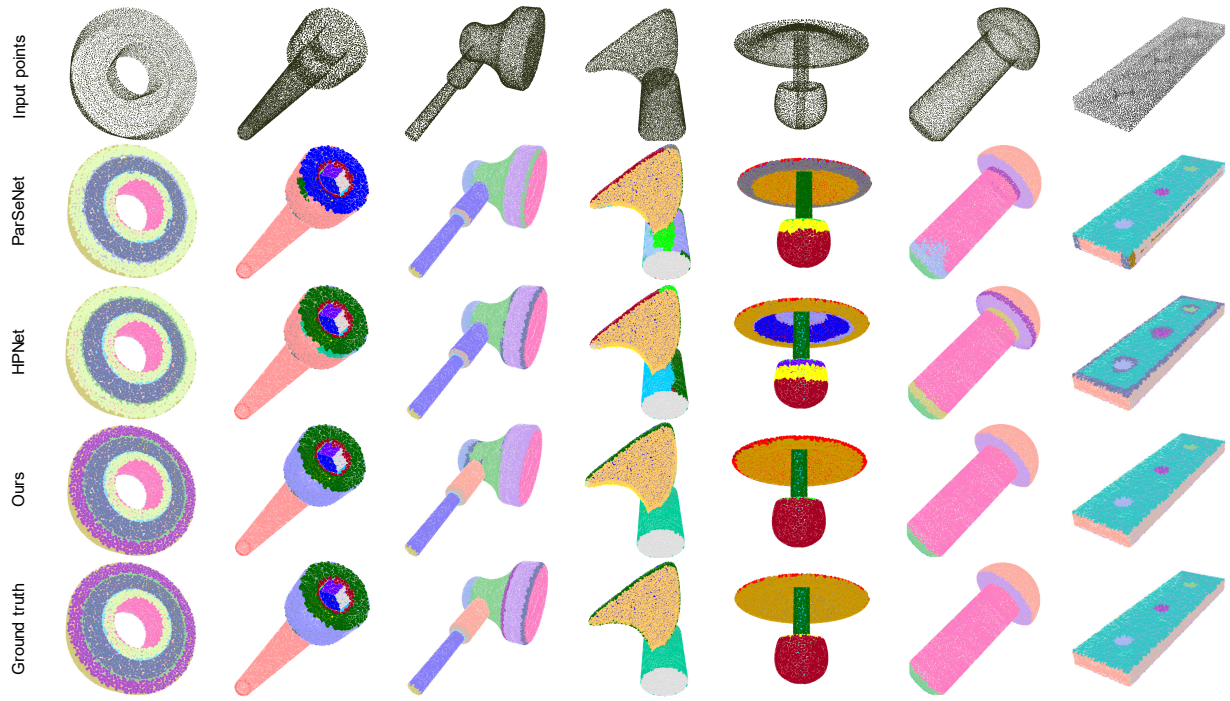

Figure 7: Qualitative comparison with state-of-the-art methods, including ParSeNet [37], and HPNet [53]. Our SED-Net can accurately segment both large and small primitive patches and distinguish primitives with similar geometric and type features robustly.

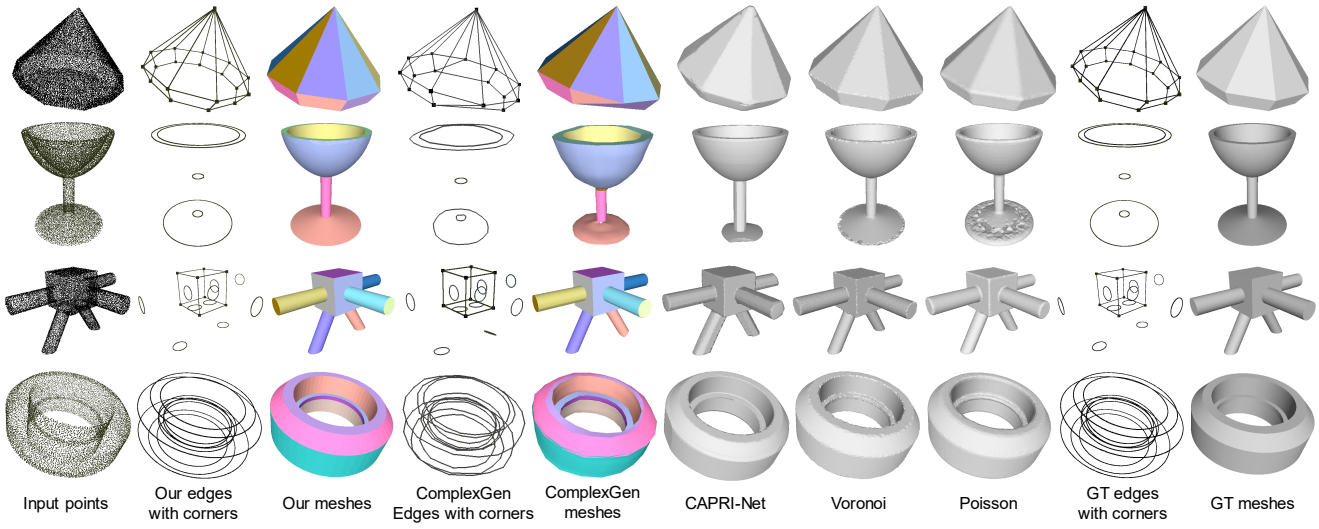

Figure 8: Primitive patches, edges and corners are used to reconstruct compact shapes. We compare our reconstructed meshes with two learning-based CAD reconstruction methods (ComplexGen [7], CAPRI-Net [55]) and two popular surface reconstruction methods, including screened Poisson reconstruction [15] and the Voronoi-based approach [3].

Fig. 8 compares our method with two learning-based CAD reconstruction methods (ComplexGen [7], CAPRI-Net [55]) and two popular surface reconstruction methods, including the screened Poisson reconstruction [15] and an efficient Voronoi-based method [3], to verify the effectiveness of our solution on primitive fitting. As the direct reconstruction methods are vulnerable to noise, we use clean data for this experiment. Previous methods which directly reconstruct meshes from point clouds may generate artifacts due to the sparseness and non-uniformity of input point clouds. Few points can be exactly sampled on edges, which leads to residual errors on sharp edges for direct reconstruction methods. Fig. 8 also shows that wrong topological relations (top row) or wrong patch types (cones

Table 1: Quantitative comparison of different methods. The best results are shown in bold.

Methods	P-mIoU ↑	T-mIoU ↑	Recall ↑	Recall* ↑	Residual ↓
ComplexGen	\	\	87.90	\	0.0196
ParSeNet	82.12	88.46	80.02	88.91	0.0131
HPNet	85.05	90.35	82.55	92.52	0.0109
Ours	**88.70**	**91.06**	**89.91**	**97.07**	**0.0074**

Table 2: Quantitative comparison of different surface reconstruction methods in terms of reconstruction fidelity and storage.

Methods	$D_c(10^{-3})\downarrow$	$D_h(10^{-2})\downarrow$	Model Size (KB)
Poisson reconstruction	12.1	2.33	856
Voronoi-based reconstruction	9.97	2.58	674
Ours	**9.41**	**1.71**	**57**

and cylinders are classified as B-splines by mistake in the bottom row) lead to imprecise results of ComplexGen. It is obvious that our method can generate mesh models with sharper edges and clearer corners. To further validate our superior performance on edge detection, we compare our method with PIE-Net [46] and DEF [27], which are learning-based edge detection approaches. Fig. 9 shows that our precisely segmented primitive patches can effectively promote the correct detection of edges and corners since the mutual assistance of surface and edge detection plays an important role in this task.

Quantitative comparison. For quantitative evaluations, we follow the metrics that are commonly used in previous methods [7, 53], including primitive mean IoU (P-mIoU), type mean IoU (T-mIoU), primitive recall rate, and point residual error. The P-mIoU and T-mIoU are mean mIoU between the ground truth and primitive segmentation results or type prediction results, respectively. Since ComplexGen [7] does not segment points, mIoU numbers are not reported for this method. The point residual error is the mean distance from points of the ground truth patch to the predicted patch. The statistics on the test dataset of ABCParts are reported in Table 1. Note that we retrain all of these methods under the same experimental conditions. To compute the recall number, we follow the setting of [7], where a patch is regarded as a recalled one if the chamfer distance between the matched ground truth patch and the detected one is less than 0.1. The recall* is calculated from the matched patches without the restriction of chamfer distance. Without this restriction, ComplexGen would detect patches with large errors, which causes failure of reconstruction, so this term of ComplexGen is not listed.

To quantitatively compare our SED-Net with other surface reconstruction methods, we introduce mesh distance as a quantitative metric to measure the approximation error between the results and the ground truth model. We randomly sample dense point clouds $\mathbf{P}_D$ and $\mathbf{Q}_D$ (both containing $M = 100K$ points) from the ground truth model and our reconstructed model. The mesh distance between $\mathbf{P}_D$ and $\mathbf{Q}_D$ includes two terms, the chamfer distance (D_c) and the Hausdorff distance (D_h). Statistics listed in Table 2 show that our SED-Net achieves the best performance for maintaining the geometric fidelity of the shape and occupies the least storage.

Robustness and generalization. To further evaluate our robustness, following [7, 37], we generate noisy and partial data respectively, and retrain our network on the two datasets for evaluation. **Noisy:** for the coordinates, we add random noise in the range $[-4\%, +4\%]$ of the radius of the bounding sphere in the normal direction. For the normals, we add random noise in the range $[-5^\circ, +5^\circ]$ to the original direction. Table 3 shows that the negative impact of noise on our network is smaller than that of other competitors. **Partial:** we use partial point clouds which are produced by $3-4$ scan views on the bounding box corners. Our results on noisy and partial data are visually shown in Fig. 10. It can be seen that for a partial patch, our approach could reconstruct its complete form by the fitting operation. Taking the box in the third row as an example, a part of each face is retained in the input point cloud, thus the segmented and fitted patches can be used to calculate the complete edges and reconstruct the missing corner.

Table 3: Comparison of robustness on noisy data.

Methods	P-mIoU↑	T-mIoU ↑	Residual↓
ParSeNet	74.65	84.44	0.036
HPNet	79.25	86.74	0.032
Ours	**85.90**	**88.62**	**0.019**

Table 4: Ablation study. Statistics of our network without each module are listed.

Modules (without)	P-mIoU ↑	T-mIoU ↑	$D_c(10^{-3})\downarrow$
Two-branch structure	-17.2	\	\
Late fusion	-5.36	-0.61	+0.82
Early fusion	-4.12	-1.46	+0.24
λ_i in $\mathcal{L}_{edge\_emb}$	-0.90	-0.11	+0.11
Fitting only non-edge points	\	\	+1.57

We further test our pipeline on two real-scanned point clouds obtained from AIM@SHAPE-VISIONAIR repository. The network trained with synthetic noisy data is adopted. Results on the Doorknob and Mechanical Pin in Fig. 11 reveal that our method performs well on real-scanned data with noise and non-uniform distribution.

4.3 Ablation Study

Finally, we conduct ablation studies to evaluate the effectiveness of our proposed two-stage feature fusion mechanism, edge-enhanced embedding loss function, and fitting strategy which excludes edge points. Table 4 lists evaluation results, which are tested by using points with normals. Only the decrease or increase amplitudes relative to the full pipeline are listed. Note that we use a baseline network with only one branch to extract primitive features and segment the point cloud without our two-branch structure. Thus the fusion mechanism is not used in this network. Features extracted from only one branch have insufficient ability to represent geometry, type and edge information of the input point cloud, which is revealed by the performance of the network without a two-branch structure. Without the two-stage feature fusion mechanism, the network can not fully utilize type and boundary information, which is also shown in Table 4. By comparison, using the two-branch structure, both the two-stage feature fusion mechanism and the edge-enhanced embedding loss could contribute to the performance improvement of primitive segmentation. Furthermore, our fitting strategy excluding edge points detected by our network avoids the negative impact of incorrectly segmented points near edges.

4.4 Limitations

Since we fit each primitive patch by using the least-square method, small holes or gaps would not damage our results. However, significant missing regions with several neighboring primitives may lead to the failure of our method. Fig. 12 shows an example with a large data missing region (*e.g.*, the bottom cylinder is totally missed). First, the missing primitives cannot be detected. Second, the circle edge of the plane patch (in green color) cannot be calculated due to missing of the cylinder, thus the plane mesh is restricted by a bounding box, which leads to failure.

5 CONCLUSIONS AND FUTURE WORK

We have presented a novel pipeline to detect and fit primitive patches and edges from point cloud data. At the core of our method is the SED-Net, which has a two-branch structure to simultaneously extract geometric, type, and edge features. These features are fully utilized by our novel two-stage feature fusion mechanism to precisely detect the primitives. Taking advantage of the simultaneous detection of surfaces and edges, we are able to obtain a parametric, compact and seamless model representation. This enables us to represent the model with predefined primitive-specific meshes and allows the user editing to create variations. Since other attributes (*e.g.*, color) of point clouds also provide structure information of shapes, we would like to train our network with these data type to further improve segmentation accuracy.

ACKNOWLEDGMENTS

Real scans of Fig. 11 are provided courtesy of INRIA by the AIM@SHAPE-VISIONAIR Shape Repository. This work was supported in part by the National Natural Science Foundation of China under Grant numbers 62032011, 61972194, U22B2034, and 62172416.

REFERENCES

[1] M Awrangjeb. 2016. Using point cloud data to identify, trace, and regularize the outlines of buildings. *International Journal of Remote Sensing* 37, 3 (2016), 551–579.
[2] Jean-Philippe Bauchet and Florent Lafarge. 2020. Kinetic shape reconstruction. *ACM Transactions on Graphics (TOG)* 39, 5 (2020), 1–14.
[3] Dobrina Boltcheva and Bruno Lévy. 2017. Surface reconstruction by computing restricted Voronoi cells in parallel. *Computer-Aided Design* 90 (2017), 123–134. SI:SPM2017.
[4] R. Q. Charles, H. Su, M. Kaichun, and L. J. Guibas. 2017. PointNet: Deep Learning on Point Sets for 3D Classification and Segmentation. In *IEEE Conference on Computer Vision and Pattern Recognition (CVPR)*. 77–85.
[5] David Cohen-Steiner, Pierre Alliez, and Mathieu Desbrun. 2004. Variational shape approximation. In *ACM SIGGRAPH*. 905–914.
[6] Martin A. Fischler and Robert C. Bolles. 1981. Random Sample Consensus: A Paradigm for Model Fitting with Applications to Image Analysis and Automated Cartography. *Commun. ACM* 24, 6 (1981), 381–395.
[7] Haoxiang Guo, Shilin Liu, Hao Pan, Yang Liu, Xin Tong, and Baining Guo. 2022. ComplexGen: CAD Reconstruction by B-Rep Chain Complex Generation. *ACM Trans. Graph. (SIGGRAPH)* 41, 4, Article 129 (2022), 18 pages.
[8] Hao-Xiang Guo, Yang Liu, Hao Pan, and Baining Guo. 2022. Implicit Conversion of Manifold B-Rep Solids by Neural Halfspace Representation. *ACM Transactions on Graphics (TOG)* 41, 6 (2022), 1–15.
[9] Hui Huang, Shihao Wu, Minglun Gong, Daniel Cohen-Or, Uri Ascher, and Hao (Richard) Zhang. 2013. Edge-Aware Point Set Resampling. *ACM Trans. Graph.* 32, 1, Article 9 (2013), 12 pages.
[10] Jingwei Huang, Yanfeng Zhang, and Mingwei Sun. 2021. PrimitiveNet: Primitive Instance Segmentation with Local Primitive Embedding under Adversarial Metric. In *Proceedings of the IEEE/CVF International Conference on Computer Vision*. 15343–15353.
[11] Pradeep Kumar Jayaraman, Joseph G Lambourne, Nishkrit Desai, Karl DD Willis, Aditya Sanghi, and Nigel JW Morris. 2022. Solidgen: An autoregressive model for direct b-rep synthesis. *arXiv preprint arXiv:2203.13944* (2022).
[12] Benjamin T Jones, Michael Hu, Vladimir G Kim, and Adriana Schulz. 2022. Self-Supervised Representation Learning for CAD. *arXiv preprint arXiv:2210.10807* (2022).
[13] Adrien Kaiser, Jose Alonso Ybanez Zepeda, and Tamy Boubekeur. 2019. A survey of simple geometric primitives detection methods for captured 3D data. In *Computer Graphics Forum*, Vol. 38. 167–196.
[14] Kacper Kania, Maciej Zieba, and Tomasz Kajdanowicz. 2020. UCSG-NET-unsupervised discovering of constructive solid geometry tree. *Advances in Neural Information Processing Systems* 33 (2020), 8776–8786.
[15] Michael Kazhdan and Hugues Hoppe. 2013. Screened Poisson Surface Reconstruction. *ACM Trans. Graph.* 32, 3, Article 29 (2013), 13 pages.
[16] Sebastian Koch, Albert Matveev, Zhongshi Jiang, Francis Williams, Alexey Artemov, Evgeny Burnaev, Marc Alexa, Denis Zorin, and Daniele Panozzo. 2019. Abc: A big cad model dataset for geometric deep learning. In *Proceedings of the IEEE/CVF Conference on Computer Vision and Pattern Recognition*. 9601–9611.
[17] Joseph George Lambourne, Karl Willis, Pradeep Kumar Jayaraman, Longfei Zhang, Aditya Sanghi, and Kamal Rahimi Malekshan. 2022. Reconstructing editable prismatic CAD from rounded voxel models. In *SIGGRAPH Asia 2022 Conference Papers*. 1–9.
[18] Joseph G Lambourne, Karl DD Willis, Pradeep Kumar Jayaraman, Aditya Sanghi, Peter Meltzer, and Hooman Shayani. 2021. Brepnet: A topological message passing system for solid models. In *Proceedings of the IEEE/CVF Conference on Computer Vision and Pattern Recognition*. 12773–12782.
[19] Eric-Tuan Lê, Minhyuk Sung, Duygu Ceylan, Radomir Mech, Tamy Boubekeur, and Niloy J. Mitra. 2021. CPFN: Cascaded Primitive Fitting Networks for High-Resolution Point Clouds. In *Proceedings of the IEEE/CVF International Conference on Computer Vision (ICCV)*. 7457–7466.
[20] Lingxiao Li, Minhyuk Sung, Anastasia Dubrovina, Li Yi, and Leonidas J. Guibas. 2019. Supervised Fitting of Geometric Primitives to 3D Point Clouds. In *Proceedings of the IEEE/CVF Conference on Computer Vision and Pattern Recognition (CVPR)*.
[21] Yangyan Li, Rui Bu, Mingchao Sun, Wei Wu, Xinhan Di, and Baoquan Chen. 2018. PointCNN: Convolution On X-Transformed Points. In *Advances in Neural Information Processing Systems*, Vol. 31.
[22] Yangyan Li, Xiaokun Wu, Yiorgos Chrysathou, Andrei Sharf, Daniel Cohen-Or, and Niloy J. Mitra. 2011. GlobFit: Consistently Fitting Primitives by Discovering Global Relations. *ACM Trans. Graph.* 30, 4, Article 52 (2011), 12 pages.
[23] Frederico A. Limberger and Manuel M. Oliveira. 2015. Real-time detection of planar regions in unorganized point clouds. *Pattern Recognition* 48, 6 (2015), 2043–2053.
[24] Cheng Lin, Tingxiang Fan, Wenping Wang, and Matthias Nießner. 2020. Modeling 3d shapes by reinforcement learning. In *European Conference on Computer Vision*. 545–561.
[25] Yangbin Lin, Cheng Wang, Bili Chen, Dawei Zai, and Jonathan Li. 2017. Facet Segmentation-Based Line Segment Extraction for Large-Scale Point Clouds. *IEEE Transactions on Geoscience and Remote Sensing* 55, 9 (2017), 4839–4854.
[26] Changfeng Ma, Yang Yang, Jie Guo, Fei Pan, Chongjun Wang, and Yanwen Guo. 2022. Unsupervised Point Cloud Completion and Segmentation by Generative Adversarial Autoencoding Network. In *NeurIPS*.
[27] Albert Matveev, Ruslan Rakhimov, Alexey Artemov, Gleb Bobrovskikh, Vage Egiazarian, Emil Bogomolov, Daniele Panozzo, Denis Zorin, and Evgeny Burnaev. 2022. Def: Deep estimation of sharp geometric features in 3D shapes. *ACM Transactions on Graphics (TOG)* 41, 4 (2022), 1–22.
[28] Gal Metzer, Rana Hanocka, Raja Giryes, and Daniel Cohen-Or. 2021. Self-Sampling for Neural Point Cloud Consolidation. *ACM Trans. Graph.* 40, 5, Article 191 (2021), 14 pages.
[29] Sven Oesau, Florent Lafarge, and Pierre Alliez. 2016. Planar Shape Detection and Regularization in Tandem. *Comput. Graph. Forum* 35, 1 (2016), 203–215.
[30] A Cengiz Öztireli, Gael Guennebaud, and Markus Gross. 2009. Feature preserving point set surfaces based on non-linear kernel regression. In *Computer graphics forum*, Vol. 28. 493–501.
[31] Despoina Paschalidou, Ali Osman Ulusoy, and Andreas Geiger. 2019. Superquadrics Revisited: Learning 3D Shape Parsing Beyond Cuboids. In *Proceedings of the IEEE/CVF Conference on Computer Vision and Pattern Recognition (CVPR)*.
[32] Tahir Rabbani, Sander Dijkman, Frank van den Heuvel, and George Vosselman. 2007. An integrated approach for modelling and global registration of point clouds. *ISPRS Journal of Photogrammetry and Remote Sensing* 61, 6 (2007), 355–370.
[33] Daxuan Ren, Jianmin Zheng, Jianfei Cai, Jiatong Li, Haiyong Jiang, Zhongang Cai, Junzhe Zhang, Liang Pan, Mingyuan Zhang, Haiyu Zhao, et al. 2021. Csg-stump: A learning friendly csg-like representation for interpretable shape parsing. In *Proceedings of the IEEE/CVF International Conference on Computer Vision*. 12478–12487.
[34] Ruwen Schnabel, Roland Wahl, and Reinhard Klein. 2007. Efficient RANSAC for point-cloud shape detection. In *Computer Graphics Forum*, Vol. 26. 214–226.
[35] Gopal Sharma, Rishabh Goyal, Difan Liu, Evangelos Kalogerakis, and Subhransu Maji. 2018. Csgnet: Neural shape parser for constructive solid geometry. In

Proceedings of the IEEE Conference on Computer Vision and Pattern Recognition. 5515–5523.

[36] Gopal Sharma, Rishabh Goyal, Difan Liu, Evangelos Kalogerakis, and Subhransu Maji. 2022. Neural Shape Parsers for Constructive Solid Geometry. *IEEE Transactions on Pattern Analysis and Machine Intelligence* 44, 5 (2022), 2628–2640.

[37] Gopal Sharma, Difan Liu, Subhransu Maji, Evangelos Kalogerakis, Siddhartha Chaudhuri, and Radomír Měch. 2020. Parsenet: A parametric surface fitting network for 3d point clouds. In *European Conference on Computer Vision*. Springer, 261–276.

[38] Dmitriy Smirnov, Matthew Fisher, Vladimir G. Kim, Richard Zhang, and Justin Solomon. 2020. Deep Parametric Shape Predictions Using Distance Fields. In *Proceedings of the IEEE/CVF Conference on Computer Vision and Pattern Recognition (CVPR)*.

[39] Chun-Yu Sun, Qian-Fang Zou, Xin Tong, and Yang Liu. 2019. Learning Adaptive Hierarchical Cuboid Abstractions of 3D Shape Collections. *ACM Trans. Graph.* 38, 6, Article 241 (2019), 13 pages.

[40] Hugues Thomas, Charles R Qi, Jean-Emmanuel Deschaud, Beatriz Marcotegui, François Goulette, and Leonidas J Guibas. 2019. Kpconv: Flexible and deformable convolution for point clouds. In *Proceedings of the IEEE/CVF International Conference on Computer Vision*. 6411–6420.

[41] Shubham Tulsiani, Hao Su, Leonidas J. Guibas, Alexei A. Efros, and Jitendra Malik. 2017. Learning Shape Abstractions by Assembling Volumetric Primitives. In *Proceedings of the IEEE Conference on Computer Vision and Pattern Recognition (CVPR)*.

[42] Mikaela Angelina Uy, Yen-Yu Chang, Minhyuk Sung, Purvi Goel, Joseph G Lambourne, Tolga Birdal, and Leonidas J Guibas. 2022. Point2cyl: Reverse engineering 3d objects from point clouds to extrusion cylinders. In *Proceedings of the IEEE/CVF Conference on Computer Vision and Pattern Recognition*. 11850–11860.

[43] Laurens Van der Maaten and Geoffrey Hinton. 2008. Visualizing data using t-SNE. *Journal of machine learning research* 9, 11 (2008).

[44] Weiyue Wang, Ronald Yu, Qiangui Huang, and Ulrich Neumann. 2018. Sgpn: Similarity group proposal network for 3d point cloud instance segmentation. In *Proceedings of the IEEE conference on Computer Vision and Pattern Recognition*. 2569–2578.

[45] Xinlong Wang, Shu Liu, Xiaoyong Shen, Chunhua Shen, and Jiaya Jia. 2019. Associatively segmenting instances and semantics in point clouds. In *Proceedings of the IEEE/CVF Conference on Computer Vision and Pattern Recognition*. 4096–4105.

[46] Xiaogang Wang, Yuelang Xu, Kai Xu, Andrea Tagliasacchi, Bin Zhou, Ali Mahdavi-Amiri, and Hao Zhang. 2020. Pie-net: Parametric inference of point cloud edges. *Advances in Neural Information Processing Systems* 33 (2020), 20167–20178.

[47] Yue Wang, Yongbin Sun, Ziwei Liu, Sanjay E. Sarma, Michael M. Bronstein, and Justin M. Solomon. 2019. Dynamic Graph CNN for Learning on Point Clouds. *ACM Trans. Graph.* 38, 5, Article 146 (2019), 12 pages.

[48] Karl DD Willis, Pradeep Kumar Jayaraman, Hang Chu, Yunsheng Tian, Yifei Li, Daniele Grandi, Aditya Sanghi, Linh Tran, Joseph G Lambourne, Armando Solar-Lezama, et al. 2022. Joinable: Learning bottom-up assembly of parametric cad joints. In *Proceedings of the IEEE/CVF Conference on Computer Vision and Pattern Recognition*. 15849–15860.

[49] Karl DD Willis, Yewen Pu, Jieliang Luo, Hang Chu, Tao Du, Joseph G Lambourne, Armando Solar-Lezama, and Wojciech Matusik. 2021. Fusion 360 gallery: A dataset and environment for programmatic cad construction from human design sequences. *ACM Transactions on Graphics (TOG)* 40, 4 (2021), 1–24.

[50] Rundi Wu, Chang Xiao, and Changxi Zheng. 2021. Deepcad: A deep generative network for computer-aided design models. In *Proceedings of the IEEE/CVF International Conference on Computer Vision*. 6772–6782.

[51] Xianghao Xu, Wenzhe Peng, Chin-Yi Cheng, Karl DD Willis, and Daniel Ritchie. 2021. Inferring cad modeling sequences using zone graphs. In *Proceedings of the IEEE/CVF conference on computer vision and pattern recognition*. 6062–6070.

[52] Dong-Ming Yan, Wenping Wang, Yang Liu, and Zhouwang Yang. 2012. Variational mesh segmentation via quadric surface fitting. *Computer-Aided Design* 44, 11 (2012), 1072–1082.

[53] Siming Yan, Zhenpei Yang, Chongyang Ma, Haibin Huang, Etienne Vouga, and Qixing Huang. 2021. HPNet: Deep Primitive Segmentation Using Hybrid Representations. In *Proceedings of the IEEE/CVF International Conference on Computer Vision*. 2753–2762.

[54] Li Yi, Wang Zhao, He Wang, Minhyuk Sung, and Leonidas J. Guibas. 2019. GSPN: Generative Shape Proposal Network for 3D Instance Segmentation in Point Cloud. In *Proceedings of the IEEE/CVF Conference on Computer Vision and Pattern Recognition (CVPR)*.

[55] Fenggen Yu, Zhiqin Chen, Manyi Li, Aditya Sanghi, Hooman Shayani, Ali Mahdavi-Amiri, and Hao Zhang. 2022. CAPRI-Net: learning compact CAD shapes with adaptive primitive assembly. In *Proceedings of the IEEE/CVF Conference on Computer Vision and Pattern Recognition*. 11768–11778.

[56] Lequan Yu, Xianzhi Li, Chi-Wing Fu, Daniel Cohen-Or, and Pheng-Ann Heng. 2018. EC-Net: an Edge-aware Point set Consolidation Network. In *European Conference on Computer Vision*.

[57] Mulin Yu and Florent Lafarge. 2022. Finding Good Configurations of Planar Primitives in Unorganized Point Clouds. In *IEEE Conference on Computer Vision and Pattern Recognition*.

[58] Long Zhang, Jianwei Guo, Jun Xiao, Xiaopeng Zhang, and Dong-Ming Yan. 2022. Blending Surface Segmentation and Editing for 3D Models. *IEEE TVCG* 28, 8 (2022), 2879–2894.

[59] Hengshuang Zhao, Li Jiang, Jiaya Jia, Philip HS Torr, and Vladlen Koltun. 2021. Point transformer. In *IEEE International Conference on Computer Vision (ICCV)*. 16259–16268.

[60] Chuhang Zou, Ersin Yumer, Jimei Yang, Duygu Ceylan, and Derek Hoiem. 2017. 3D-PRNN: Generating Shape Primitives With Recurrent Neural Networks. In *Proceedings of the IEEE International Conference on Computer Vision (ICCV)*.

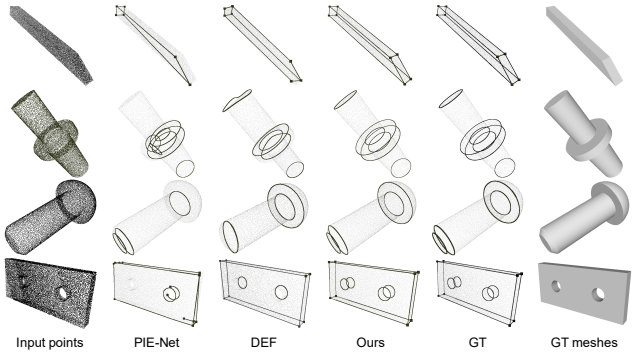

Figure 9: We compare our method with PIE-Net [Wang et al. 2020] and DEF [Matveev et al. 2022] which extract parametric edges with corners from point clouds.

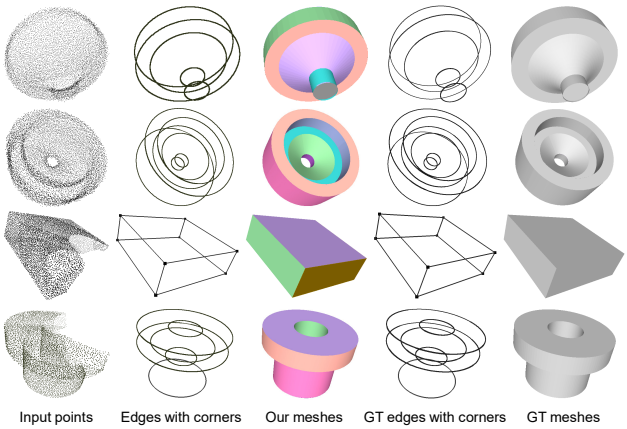

Figure 10: Our primitive detection and reconstruction results on noisy (top two rows) and partial (bottom two rows) data.

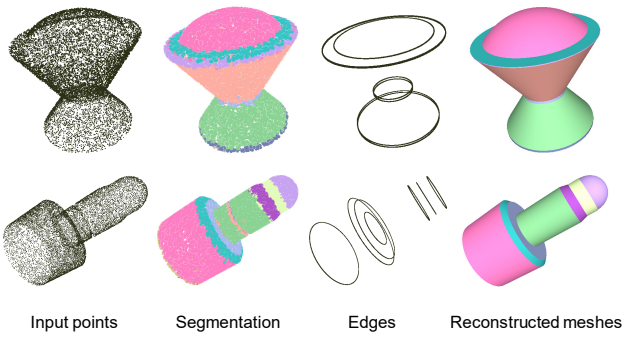

Figure 11: Testing results on two real-scanned point clouds validate the generalization of our approach.

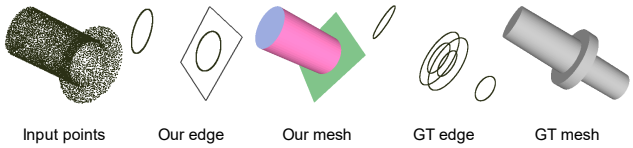

Figure 12: A failure example of SED-Net when handling a large missing region.