
PIE-NET: Parametric Inference of Point Cloud Edges

Xiaogang Wang^{1,2} Yuelang Xu^{2,5} Kai Xu³ Andrea Tagliasacchi⁴
 Bin Zhou¹ Ali Mahdavi-Amiri² Hao Zhang²

¹State Key Laboratory of Virtual Reality Technology and Systems, Beihang University

²Simon Fraser University

³National University of Defense Technology

⁴Google Research

⁵Tsinghua University

Abstract

We introduce an end-to-end learnable technique to robustly identify feature edges in 3D point cloud data. We represent these edges as a collection of parametric curves (i.e., lines, circles, and B-splines). Accordingly, our deep neural network, coined PIE-NET, is trained for *parametric inference of edges*. The network relies on a region proposal architecture, where a first module proposes an over-complete collection of edge and corner points, and a second module ranks each proposal to decide whether it should be considered. We train and evaluate our method on the ABC dataset, the largest publicly available dataset of CAD models, via ablation studies and compare our results to those produced by traditional (non-learning) processing pipelines, as well as a recent deep learning-based edge detector (EC-Net). Our results significantly improve over the state-of-the-art, both quantitatively and qualitatively, and generalize well to novel shape categories.

1 Introduction

Edge estimation is a fundamental problem in image and shape processing. Often regarded as a low-level vision problem, edge detection has been intensely studied and by and large “solved” at the *conceptual* level – there are precise mathematical definitions of what an edge is over an image, or over the surface of a 3D shape. In practice however, even state-of-the-art edge estimators are sensitive to parameter settings and they often underperform near soft edges, noise, and sparse data. This is especially true for acquired point clouds, where these data artifacts are prevalent. We argue this is caused by the fact that edge detection is traditionally achieved by performing decisions based on *manually designed* local surface features; note that this resembles the use of hand-designed descriptors in pre-deep learning computer vision.

In this paper, we advocate for a data-driven approach to feature edge estimation from point clouds – one where priors to make this operation robust are *learned* from training data. More precisely, we develop PIE-NET, a deep neural network that is trained for *Parameter Inference* of feature *Edges* over a 3D point cloud, where the output consists of one or more parametric curves. Our method treats edge inference as a *proposal* and *ranking* problem – a solution that has shown to be extremely effective in computer vision for object detection. More specifically, in a first phase, PIE-NET proposes a large collection of potentially invalid and/or redundant parametric curves, while in a second phase invalid proposals are suppressed, and the final output is generated. The suppression is guided by learnt *confidence* scores estimated by the network, as well as by how well the predicted curves *fit* the data.

Our approach is also motivated by recent success on employing neural networks for other low-level geometry processing tasks such as normal estimation [1], denoising [2], and upsampling [3]. We train PIE-NET on the recently released ABC dataset by [4], which is composed of more than one million

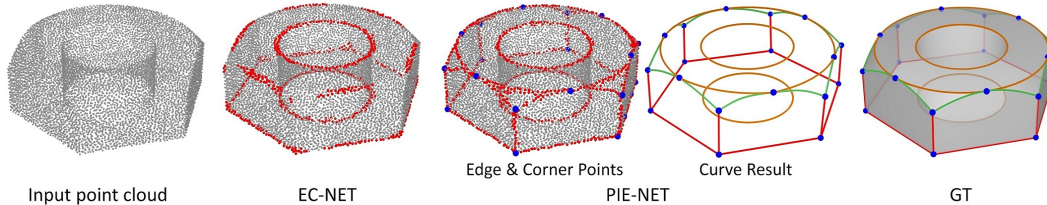


Figure 1: Our deep neural network, PIE-NET, is trained for *parametric inference of edges* from point cloud. It first detects edge and corner points, and then infers a collection of parametric curves representing edge features. In comparison, the only known deep method for this task *only classifies* edge points, and produces results that are inferior to ours both visually and quantitatively.

feature-rich CAD models with parametric edge representations. The combination of large-scale training data and a carefully designed end-to-end learnable pipeline allows PIE-NET to *significantly* outperform traditional (non-learning) edge detection techniques, as well as recent learnable variants from both a quantitative and qualitative standpoint.

2 Related work

Literature on point-based graphics [5] is quite extensive and we refer readers to a recent survey [6]. Since the seminal work of Qi et. al. [7], there has been a proliferation of research on learning deep neural networks for 3D point cloud processing [8, 9, 10, 11, 12, 13, 14, 15, 16, 17]. We now focus on methods that are most closely related: ① primitive inference, ② consolidation, and ③ edge detection.

Parametric primitive inference. Parametric primitive fitting has been a long-standing problem in geometry processing. The detection or fitting of parametric feature curves (such as Bezier curves) in 3D point clouds has been extensively researched, where it is typically formulated in least squares form [18, 19, 20, 21]. Alternatively, one can use random RANSAC-type algorithms [22] to propose the parameters of primitives fitting a given point cloud. Many types of primitive shapes (planes, spheres, quadratic surfaces) have been considered and various applications have been explored – from 3D reconstruction and modeling [23, 24] to robotic grasping [25]. Besides predefined primitives, recent works also studied learning data-driven geometric priors for shape fitting [26, 27]. End-to-end models have been proposed for fitting cuboids [28], super-quadrics [29], and convexes [30, 31]. Li et al. [11] propose a supervised method to detect a variety of primitive patches at various scales. Similarly to our work, their network first predicts per-point properties, and later estimates primitives.

Edge-aware consolidation and reconstruction. Edge feature detection has been applied to enhance 3D reconstruction [32, 33, 34, 35]. Oztireli et al. [32] leverage MLS and robust estimators to automatically identify sharp features and preserve them in the resulting implicit reconstruction. Huang et al. [33] first computes normals reliably away from edges, and then progressively re-samples the point cloud towards edges leading to edge-preserving reconstruction. Edge detection has also been utilized to enhance RGBD reconstruction, where Liu et al. [35] propose to detect edges for the task of wire reconstruction from RGBD sequences. Consolidation has also been adopted in designing deep neural networks. Yu et al. [36] propose PU-Net for point cloud upsampling. It first learns per point features and then expands them with a multi-branch convolution unit. The expanded feature is then split to a multitude of features used to reconstruct a dense point set. EC-Net [10] proposes a deep edge-aware point cloud consolidation framework. The network is trained to regress and recover upsampled 3D points and point-to-edge distances. The reconstruction of surfaces with sharp edges has also recently been tackled by learnable sparse convex decomposition [31], but this method performs a feature-aware reconstruction as a *holistic* task.

Edge feature detection. Edge feature detection from point clouds relies on local geometric properties such as normals [37], curvatures [38], and feature anisotropy [39]. For example, Fleishman et al. [40] employ robust estimators to identify edge features, but shape analysis relies on moving least squares (MLS) to locally model the neighborhood of a point. Daniels et al. [41] extend this method to extract feature *curves* from noisy point clouds based on the reconstructed MLS surface. It is also possible to extract edge features via point cloud segmentation of these local properties [42, 43]. The closest work to ours is EC-Net [10], whose first phase consists of point classification and regression of per-point distances to the edge. Edge points are then detected as the points with a zero point-to-edge

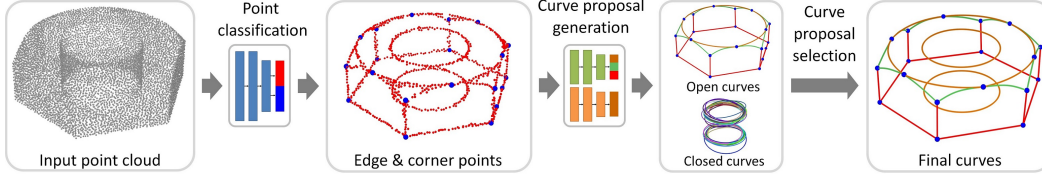


Figure 2: **Pipeline** – Our method treats parametric edge inference as a region proposal task. Given a point cloud, our network first detects edges and corners. Then, for each pair of corners, it performs curve proposal generation and selection to detect feature edges as a set of parametric curves.

distance. To the best of our knowledge, EC-Net is the only prior work on feature edge estimation using deep learning and PIE-NET is the first technique that estimates parametric edge curves with an end-to-end trainable deep network.

3 Method

The processing pipeline of **Parametric Inference of Edges Network** is summarized in Figure 2. Our technique treats point cloud curve inference as curve *proposal* process followed by *selection*, a technique inspired by image-based object detection pipelines [44].

Overview. Given a point cloud, a detection module first identifies *edge* and *corner* points (Section 3.1). Corners represent the start/end points of curves, or the locations where two curves touch. Pairs of corners are then given to a curve proposal module (Section 3.2) to identify the corresponding “edge points”, and finally generate a corresponding parametric open curve proposal. For closed curves, as they do not have a well defined start/end, we take inspiration from the Similarity Group Proposal Network [45], and regress the parameters of closed curves by first performing a clustering of the identified edge points, followed by curve fitting. Finally, a simple curve selection scheme merges all the curve proposals to generate the final fitting result.

Training data. We performed a statistical analysis over the ABC dataset [46], a large-scale CAD mechanical part dataset containing ground-truth annotations for edges and corner points. We found that the edges of the mechanical parts largely belong to three types, i.e., lines, circles, and B-spline curves, which account for more than 95% of the edges. We ignored the “ellipse” and “other” types due to their statistical insignificance. Therefore, we only focus on these three curve types in this paper, and filter out other types in the ground truth. For each ABC model, we sample it into a point cloud containing 8,096 points, via uniform point sampling. We then transfer the ground-truth annotations of the CAD models to the point clouds by nearest neighbor assignment.

Parameterization. Our method deals with three types of curves: lines, circles (open arcs plus full closed circles), and B-splines. We now describe the differentiable parameterization of the two latter curve types, where we note that a line segment can be formed by simply connecting two corner points. We parameterize circles as $\beta=(p_1, p_2, p_3)$, where p_1, p_2, p_3 are any three points on the circle that are not collinear. We first transform $\beta=(p_1, p_2, p_3)$ as (n, c, r) , where n is the normal of the circle, and c and r are its center and radius. We then randomly sample points on the circle, and express them as a function of β . To achieve this, we draw $\alpha \in [0, 2\pi]$, and generate random samples $p(\alpha|\beta)=c + r(ucos(\alpha) + vsin(\alpha))$, where $u=p_1 - c$ and $v=u \times n$. We parameterize B-Spline curves with four control points $\beta=\{p_i\}_{i=0}^3$. Given $B_{i,K}(\cdot)$ representing the i -th basis function of a K -th order B-spline, and α sampled uniformly in the $[0, 1]$ range, we draw a uniform random sample as $p(\alpha|\beta)=\sum_i p_i B_{i,K}(\alpha)$. Note that to ensure that $p(0|\beta)=c_1$ and $p(1|\beta)=c_2$, we employ *quasi-uniform* B-splines. We predict residuals as the *displacement* of the two intermediate control points $\{p_i\}_{i=1,2}$.

3.1 Point classification

To classify points in the input point cloud into edge and corners (plus the null class), we use a PointNet++ like architecture [47]. In particular, we devise two separate classification networks outputting edge vs. null and corner vs. null, respectively. The network predicts the probability of a point being on an edge T_e , or a corner T_c , or null otherwise, as well as the 3D *offset vector* D_e that projects the point onto an edge, and the offset D_c that projects the point onto a corner – these predictions are supervised by the ground truth labels \hat{T}_e and \hat{T}_c , as well as the ground truth offsets \hat{D}_e

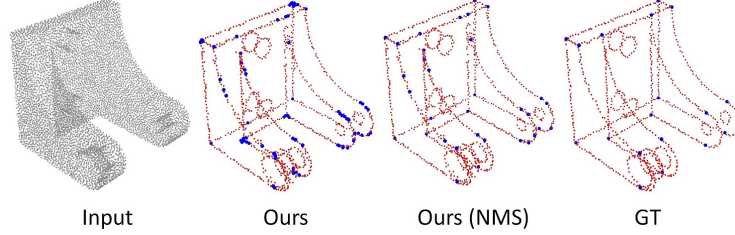


Figure 3: **Point classification** – We determine whether each point belongs to an edge or to a corner. We show the results of our method before and after NMS, as well as the corresponding ground truth.

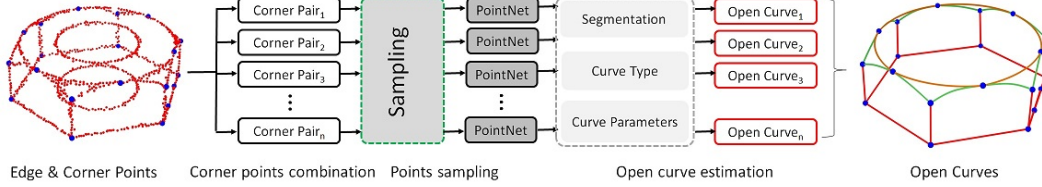


Figure 4: **Curve proposal** – Given the collection of corners, we generate all possible corner pairs and propose an open curve connecting the pair of corners. In doing so, we predict the curve type, the parameters, and the subset of points corresponding to the proposed curve.

and \hat{D}_c . We then threshold the probabilities $T_e > \tau_e$ and $T_c > \tau_c$ to obtain a corresponding set of edge points $E = \{e_i\}_{i=1}^M$ and corner points $C = \{c_i\}_{i=1}^N$. We set the parameters $\tau_e = 0.7$ and $\tau_c = 0.9$ throughout our experiments. We optimize the multi-task loss $\mathcal{L}_{\text{detection}} = \mathcal{L}_{\text{edge}} + \mathcal{L}_{\text{corner}}$:

$$\mathcal{L}_{\text{edge}} = \mathcal{L}_{\text{cls}}(T_e, \hat{T}_e) + \hat{T}_e \cdot \lambda_e \mathcal{L}_{\text{reg}}(D_e, \hat{D}_e), \quad (1)$$

$$\mathcal{L}_{\text{corner}} = \mathcal{L}_{\text{cls}}(T_c, \hat{T}_c) + \hat{T}_c \cdot \lambda_c \mathcal{L}_{\text{reg}}(D_c, \hat{D}_c), \quad (2)$$

where for \mathcal{L}_{cls} , rather than traditional binary cross-entropy, we employ a *focal loss* [48] to deal with the pathological class imbalance in our dataset (the number of edge points is very small relative to the number of non-edge points). For \mathcal{L}_{reg} , we use the *smooth L_1* loss as in [49, 44].

Non-maximal suppression. At test time, we first classify corner points and regress the corresponding offset vectors. However, due to noise, several points may be mislabelled as corners in the proximity of a ground-truth corner, hence necessitating post-processing. To this end, we adopt a point-level Non-Maximal Suppression (NMS). After applying the offset to the detected corner points, we perform agglomerative clustering with a maximum intra-class distance threshold δ ; we use $\delta = 0.05\ell$, with ℓ being the diagonal length of object bounding box. We then perform NMS within each cluster by selecting the corner with the highest classification probability; see Figure 3.

3.2 Curve proposal

PIE-NET performs a curve proposal for *open* curves and another for *closed* curves¹. Our *open* curve proposal generation leverages the fact that open curves connect two corner points, and makes the assumption that a corner pair can support at most one curve. Hence, given the set C with N corner points, we start by generating all $O(N^2)$ corner point combinations and create corner pairs $\{\mathcal{P}_i = \{c_{i1}, c_{i2}\} \mid c_{i1}, c_{i2} \in C\}$. Each corner pair will correspond to a curve proposed by the curve proposal generation; see Figure 4. Given a pair \mathcal{P}_i , we need to identify points that should be associated to the corresponding curve. We first localize the search via a heuristic that only considers points in E that lie within a sphere with center $(c_{i1} + c_{i2})/2$, and radius $R = \|c_{i1} - c_{i2}\|/2$. Within this sphere, we then uniformly sample a subset E_i^o that has a cardinality compatible with the input dimension of our multi-headed PointNet networks and feed E_i^o to them; see Figure 4.

Losses. The network heads perform three different tasks, and are trained by three different losses; see **supplementary material** Sections 6 and 8 for network training details. The first network head performs *segmentation*, determining whether a particular point belongs to the candidate curve. The second head performs *classification*, determining whether we need to generate a line, circle, or

¹For technical details on *closed* curve proposal, see Section 6 of the **supplementary material**.

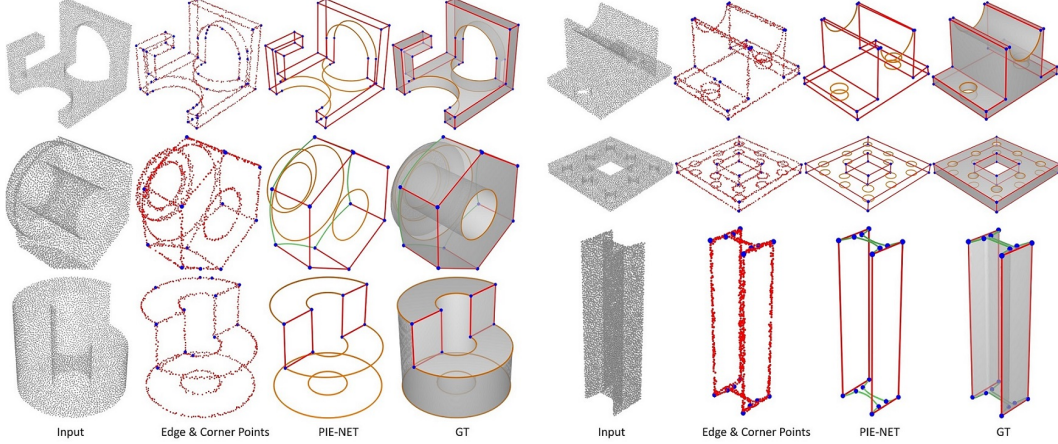


Figure 5: Results on edge and corner detection and parametric curve inference by PIE-NET.

B-spline. The third head performs *regression*, identifying the parameters of the proposed curve. Note that the network outputs the parameters for all curve types and only those corresponding to the type output by the type classifier are regarded as the valid output. More formally:

$$\mathcal{L}_{\text{propos}} = w_m \mathcal{L}_{\text{mask}}(M_p, \hat{M}_p) + w_c \mathcal{L}_{\text{cls}}(T_p, \hat{T}_p) + w_p \mathcal{L}_{\text{para}}(\beta), \quad (3)$$

where M_p and \hat{M}_p are the predicted / ground-truth classifications, T_p and \hat{T}_p are predicted / ground-truth curve types, and β represents the predicted curve parameters. We set $w_m=1$, $w_c=1$, and $w_p=10$ throughout our experiments. Softmax cross-entropy is employed for both $\mathcal{L}_{\text{mask}}(\cdot)$ and $\mathcal{L}_{\text{cls}}(\cdot)$, while for regression:

$$\mathcal{L}_{\text{para}} = \hat{T}_{\text{circle}} \cdot \mathcal{L}_{\text{circle}}(\beta) + \hat{T}_{\text{line}} \cdot \mathcal{L}_{\text{line}}(\beta) + \hat{T}_{\text{spline}} \cdot \mathcal{L}_{\text{spline}}(\beta), \quad (4)$$

where \hat{T}_* encodes the ground truth one-hot labels for the corresponding curve type, while \mathcal{L}_* are Chamfer Distance losses measuring the expectation of Euclidean distance deviation between the curve having parameters β and the ground truth. In order to compute expectations, we need to draw random samples from each curve type which are *parametric* in the degrees of freedom β .

4 Results and evaluation

In Figure 5, we show randomly selected qualitative results produced by PIE-NET, and generalization to object categories that are not part of our training dataset in Figure 8. We evaluate our network via ablation studies (Section 4.1), comparisons to both *traditional* and *learned* pipelines for edge detection (Section 4.2), and stress tests with respect to noise and sampling density (Section 4.3). To evaluate edge classification, we measure precision/recall and the IoU between predictions, while to evaluate the geometric accuracy of the reconstructed edges, we employ the *Edge Chamfer Distance* (ECD) introduced by [31]. Note that, differently from Chen et al. [31], we *do not* need to process the dataset to identify ground-truth edges, as these are provided by the dataset.

4.1 Ablation studies

Sphere radius – R . We validate the spherical sub-sampling heuristic introduced in Section 3.2. Note that we only consider values of R' strictly larger than the default setting, as otherwise we are *guaranteed* to miss edge features. Specifically, we set R' at $\times 1.5$ and $\times 3$ of the default R – in the limit ($R'=\infty$) we would consider the *entire* point cloud for each candidate corner-pair. Our analysis shows that as the sampled point cloud becomes larger and larger, it becomes more and more difficult to identify the right subset of points. This is because our sphere sampling heuristic also provides a *hint* of which curve needs to be sampled – the one whose corners points are touching the sphere.

Classification thresholds – τ_c, τ_e . We also study curve generation performance as we vary how many corner points are accepted. Overall, the performance of the network is stable as we vary τ_c , delivering the expected precision vs. recall trade-off. We find that curve generation quality slightly

Metric	①		②		③		④	PIE-NET
	$R' = 1.5R$	$R' = 3R$	$\tau_c = 0.8$	$\tau_c = 0.95$	$\tau_e = 0.6$	$\tau_e = 0.8$	w/o D_e, D_c	
ECD ↓	0.0326	0.0824	0.0150	0.0144	0.0163	0.0149	0.0186	0.0136
IOU ↑	0.4386	0.2950	0.4875	0.5110	0.4356	0.4964	0.5017	0.5330
Precision ↑	0.5149	0.3244	0.5700	0.6032	0.5180	0.5975	0.5824	0.6219
Recall ↑	0.8570	0.8910	0.8415	0.8230	0.8340	0.8035	0.7849	0.8165

Figure 6: **Ablation studies** – We evaluate the qualitative (top) and quantitative (bottom) performance of our method across a number of metrics as we tweak: ① the radius R controlling the sampling heuristic, ② the corner segmentation threshold τ_c , ③ the edge segmentation threshold τ_e , and ④ the edge and corner offsets D_e and D_c . Recall for PIE-NET: $R'=R$, $\tau_c=.9$, $\tau_e=.7$, and we use D_e, D_c .

	VCM			EAR			EC-Net	PIE-NET
	$\tau=0.12$	$\tau=0.17$	$\tau=0.22$	$\tau=0.03$	$\tau=0.035$	$\tau=0.04$		
ECD ↓	0.0321	0.0430	0.0569	0.0679	0.0696	0.0864	0.0360	0.0088
IOU ↑	0.2841	0.2854	0.2855	0.3404	0.3250	0.2844	0.3561	0.6223
Precision ↑	0.3063	0.3244	0.3456	0.5560	0.4149	0.6523	0.4872	0.6918
Recall ↑	0.8385	0.7644	0.6937	0.4820	0.5910	0.3578	0.5736	0.8584

Figure 7: **Comparisons to state-of-the-art methods** – Qualitative (top) and quantitative (bottom) comparisons against edge detection techniques – VCM [50], EAR [51], and EC-Net [10].

improves when we increase τ_c , but as the threshold gets too large, the quality begins to decline as several corner points are filtered out, resulting in the absence of some feature curves. Analogous trade-offs are observed as we adjust the values of edge classification threshold τ_e .

4.2 Comparisons to the state-of-the-art

Two of the most classical, pre-deep learning, methods for point cloud edge detection are: ① Merigot et al. [50], where edges are detected by thresholding the Voronoi Covariance Measure (VCM), and ② Huang et al. [51], where edges are identified as part of an Edge-Aware Resampling (EAR) routine. We consider these two methods as representative of the state of the art, as both have been adopted in the point-set processing routines of the well known CGAL library [52]. As reported in Figure 7, our end-to-end method completely outperforms these classical baselines.

Voronoi Covariance Measure (VCM) [50]. We compute VCM for each point $VCM(p_i)$, where we set *offset radius* to 0.5 and *convolution radius* to 0.25. These parameters were found by a parameter sweep evaluated over the test set of ABC [4]. We then consider a point to belong to an edge if $VCM(p_i) > \tau$, and select three pareto-optimal thresholds $\tau = \{0.12, 0.17, 0.22\}$ for evaluation.

Edge Aware Resampling (EAR) [51]. This method first re-samples away from edges via anisotropic locally optimal projection (LOP), an operation that leaves *gaps* near sharp edges. We classify points as edges by detecting whether they fall within these gaps. This is achieved by computing the average distance $D_{10}(p_i)$ to the ten nearest neighbors – we consider a point to belong to an edge if $D_{10}(p_i) > \tau$.

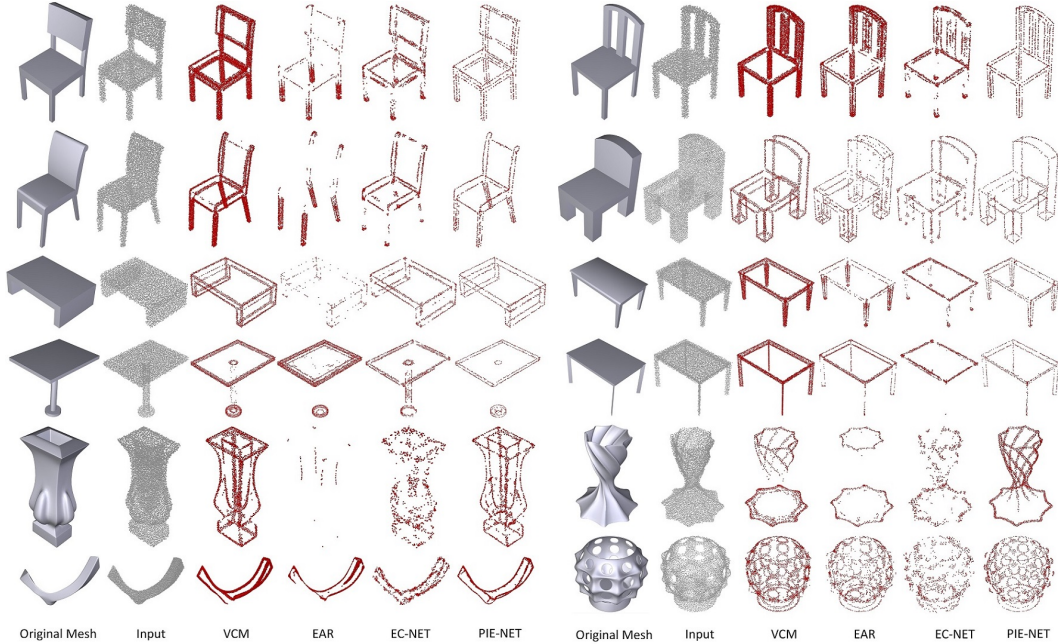


Figure 8: **Generalization to novel object categories** – While PIE-NET is trained on CAD models of mechanical assemblies from the ABC dataset [46], our edge detector is immediately applicable to 3D point clouds of general 3D objects and consistently outperforms VCM, EAR, and EC-Net.

We report the results for $\tau = \{0.03, 0.035, 0.04\}$ – the best performing thresholds as selected by a dense sweep in the $[0.01 \dots 0.05]$ range as suggested in the paper. As illustrated in Figure 7, our end-to-end method performs significantly better, as immediately quantified by the fact that ECD is an *order of magnitude* larger than the one reported for PIE-NET.

Edge-Aware Consolidation Network (EC-Net) [10]. We train EC-Net on our dataset, not the author’s dataset [10] as it only contained 24 CAD models with manually annotated edges, while ABC contains more than a million models. The comparison results demonstrate how PIE-NET achieves significantly better performance; see Figure 7. In particular, our qualitative analysis revealed how EC-Net struggles in capturing areas with *weak-curvature*, as well as *short edges*.

In Figure 8, we show additional qualitative comparison results on *novel* object categories such as tables, chairs, and vases, etc., which are *not* part of the ABC training set. These results were obtained using the same methods and trained networks as those that produced Figure 7. The point cloud inputs were obtained by uniform sampling on the original mesh shapes, which are shown in Figure 8 to reveal the edge features; there were no GT edges associated with these shapes.

4.3 Stress tests

Point cloud noise. We stress test PIE-NET by increasing the level of noise. Specifically, we randomly apply different perturbations to the point samples along the surface normal direction with a scale factor in the $[-1.0 - X, 1.0 + X]$ range, where we tested four values of $X = \{0, 0.01, 0.02, 0.05\}$. In each case, the network was trained with the noise-added data. Figure 9 shows some visual results and quantitative measures. As we can observe, our network, even when trained with noisy data, can still out-perform VCM, EAR, and EC-Net when they are tested on or trained on clean data.

Point density. We also train PIE-NET on point clouds at a reduced density. Specifically, for each CAD shape, we sampled a different number (P) of points to verify whether our network could handle the sparser point clouds, where $P = \{8,096, 4,096, 2,048, 1,024\}$. Results in Figure 9 reveal a similar trend as from the previous stress test. Namely, our network, when trained on sparser point clouds, can still outperform VCM, EAR, and EC-Net when they are tested on or trained on data at full resolution (8,096 points).

Metric	$X = 0.05$	$X = 0.02$	$X = 0.01$	$X = 0.00$	$P=1,024$	$P=2,048$	$P=4,096$	$P=8,096$
ECD ↓	0.0924	0.0261	0.0159	0.0088	0.0473	0.0251	0.0185	0.0159
IOU ↑	0.5037	0.5905	0.6049	0.6223	0.4187	0.4972	0.5843	0.6049
Precision ↑	0.5973	0.6364	0.6672	0.6918	0.4985	0.5673	0.6549	0.6672
Recall ↑	0.7781	0.8385	0.8691	0.8584	0.7936	0.8445	0.8344	0.8691

Figure 9: **Network behavior with respect to noise and sampling density** – We show the qualitative (top) and quantitative (bottom) performance of PIE-NET on input point clouds degraded by noise of different magnitudes: $X=\{0.05, 0.02, 0.01, 0\}$ or sampled at a reduced density: with P going from 8,096 down to 1,024. All the numbers in **boldface** outperform VCM, EAR, and EC-Net without added noise or reduced point density; see the table in Figure 7 for reference.

5 Conclusion, limitation, and future work

The detection of edge features in images has been shown to play a fundamental role in low-level computer vision — for example, the first layers in deep CNNs have been shown to be nothing but edge detectors. Notwithstanding, limited work to detect features of visual importance exists in 3D computer vision. With this objective, we present PIE-NET, a deep neural network which is trained to extract parametric curves from a point cloud that compactly describe its edge features. We demonstrate that a *region proposal* network architecture can already significantly outperform the state-of-the-art, which includes both traditional (non-learning) methods, and a recently developed deep model [10], the only other learning-based edge-point classifier, to the best of our knowledge.

Our contribution does not lie in merely surpassing the state-of-the-art from traditional graphics and geometry processing techniques. More importantly, we are introducing a *new learning problem* to the machine learning community, as well as defining the corresponding challenge for the recently published ABC dataset [4]. We also show that, yet again, networks can perform excellently even without resorting to excessive amounts of domain-specific (i.e., differential geometry) knowledge, as the only domain knowledge we assume is a simple curve parameterization.

Future works (learning). From an architectural perspective, it would be natural to consider self-attention mechanisms to remove the need for ad-hoc solutions (E_i^o in Section 3.2), as well as to employ deep representations of curves rather than explicit ones [53, 54]. Integrating our edge detector with current deep models for 3D reconstruction [53, 55], scan/shape completion [56, 57], and novel shape synthesis [58, 59] may be the key missing ingredient which would significantly improve their results in terms of high-frequency geometric details.

Future works (geometry). Our current method still has several technical limitations. The first is related to the current method’s inability to distinguish between close-by detected features. For example, as we generate open curve proposals based on corner pairs, for very short edges, the two corner points can be close to each other, and the network might assume that there is no connectivity between the two corner pairs. Also, our method cannot deal with scenarios where *multiple* curves connect two corners, or when the curve connecting the corners is (partially) *outside* the sphere associated with corners. We have not explored the full potential of a learning-base edge detector in handling *significant noise* and *missing data*. Another avenue for future investigation would be to transfer or extend our learning framework to handle large-scale 3D scenes.

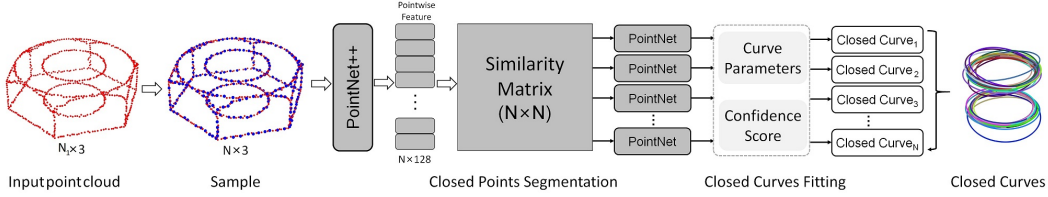


Figure 10: **Closed curve proposal** – We first identify the subset of points belonging to closed curves via feature-based clustering and then fit a closed curve to each cluster. The network outputs both curve parameters and confidence scores.

6 Closed curve proposal generation

Our closed curve proposal module is inspired by [45]. In particular, we first identify the subset of points belonging to closed curves via feature *clustering*, and then *fit* a closed curve to each proposed cluster, while simultaneously estimating the *confidence* of the fit; see Figure 10. Here we use the edge/corner classification described in Section 3.1 of the main paper. Note our method currently only handles curves with a *circular* profile, but it can be easily extended to other types of closed curves.

Clustering. We train an equivariant PointNet++ network to produce a point-wise feature $F(\cdot)$ for each of the M input edge points. Based on such features, we then create a similarity matrix $S \in \mathbb{R}^{M \times M}$, where $S_{ij} = \|F(p_i) - F(p_j)\|_2$. We can then interpret each of the M rows of S as a *proposal*, and consider the set $C_m = \{j \text{ s.t. } S_{m,j} < \bar{S}\}$ as the edge points of the m -th proposal. \bar{S} is a threshold to filter out the points attaining very different feature scores (i.e., they probably do not belong to the same curve). Potentially redundant proposals are dealt with in the *selection* phase of our pipeline, as described in Section 7.

Fitting. We take each proposal C_m , and regress the parameters β of the corresponding curve, as well as its confidence γ . We parameterize each circle proposal via three points $\beta = \{p_a + \Delta_a, p_b + \Delta_b, p_c + \Delta_c\}$. We obtain $\{p_a, p_b, p_c\}$ by furthest point sampling in C_m initialized with $p_a = p_m$. We then train a PointNet architecture with two fully-connected heads. The first head regresses the offsets $\{\Delta_a, \Delta_b, \Delta_c\}$, while the second head predicts γ .

Losses. We train our network to predict similarity matrices S given ground truth \hat{S} , confidences $\Gamma = \{\gamma_m\}$ given ground truth $\hat{\Gamma}$, and a collection of points sampling the ground truth curve:

$$\mathcal{L}_{\text{closed}} = \mathcal{L}_{\text{sim}}(S, \hat{S}) + \mathcal{L}_{\text{score}}(\Gamma, \hat{\Gamma}) + \mathcal{L}_{\text{para}}(\beta). \quad (5)$$

Given that $\hat{S}_{ij} = 0$ if points p_i and p_j belong to the same ground truth curve and $\hat{S}_{ij} = 1$ otherwise, we supervise for similarity via:

$$\mathcal{L}_{\text{sim}} = \sum_{ij} S_{ij}, \quad (6)$$

where

$$S_{ij} = \begin{cases} \|F(p_i) - F(p_j)\|_2, & \hat{S}_{ij} = 1 \\ \max\{0, K - \|F(p_i) - F(p_j)\|_2\}, & \hat{S}_{ij} = 0 \end{cases}$$

where F is point-wise feature computed with PointNet++. K controls the dissimilarity between elements in different parts, which is set to $K=100$ in our experiments. For $\mathcal{L}_{\text{score}}(\cdot)$ we employ L_2 loss, where $\hat{\Gamma}$ is the segmentation confidence. Positive training examples come from seed points belonging to ground truth closed curves, and their IoU with ground truth segmentations is larger than 0.5. Negative training examples are those with IoU smaller than 0.5. For parameter regression, we minimize

$$\mathcal{L}_{\text{para}} = \hat{T}_{\text{circle}} \cdot \mathcal{L}_{\text{circle}}(\beta), \quad (7)$$

where \hat{T}_{circle} is the ground truth one-hot labels, and $\mathcal{L}_{\text{circle}}(\cdot)$ is the Chamfer distance between the curve represented by β and the ground truth. In particular, we first compute the circle according to the estimated β , and then sample it by points. We then compute the Chamfer distance between the estimated circle and its corresponding point-sampled ground-truth.

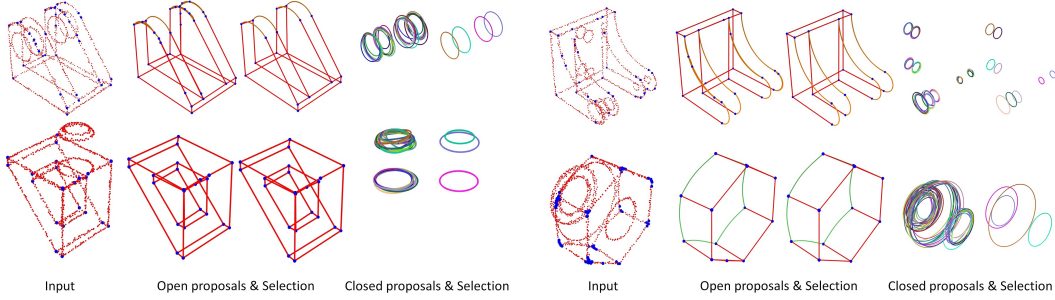


Figure 11: **Proposal selection** – The curves generated by open/closed proposals, and the ones that were accepted by our selection process.

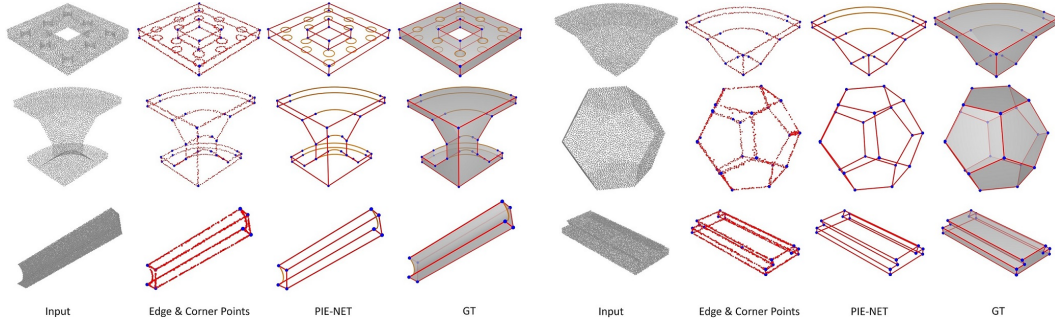


Figure 12: Results on edge and corner detection and parametric curve inference by PIE-NET.

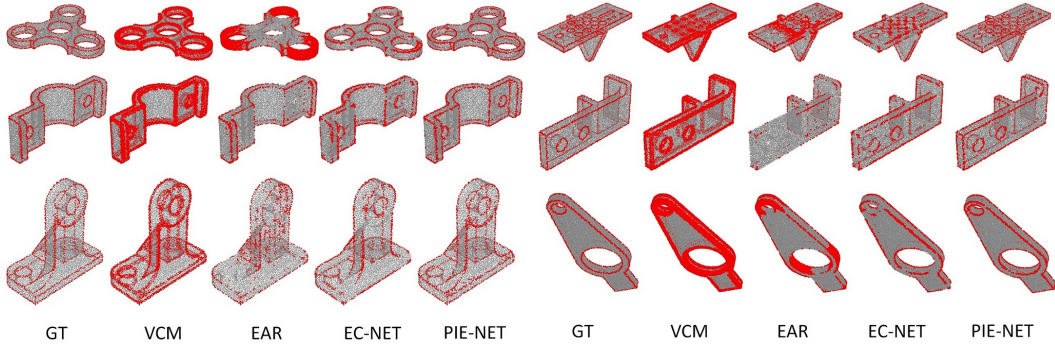


Figure 13: **Comparisons to state-of-the-art methods** – Qualitative comparisons against edge detection techniques – VCM [50], EAR [51], and EC-Net [10].

7 Curve proposal selection

Similar to proposal-based object detection for images [44], the final stage of our algorithm is a non-differentiable process for redundant/invalid proposal filtering; see Figure 11. We adopt slightly different solutions for *open* and *closed* curves.

Open curve selection. Given the segmentations, i.e., a set of points associated with a curve (see Figure 5 in the main paper) corresponding with two proposals, we first measure overlap via $O(A, B) = \max\{I(A, B)/A, I(A, B)/B\}$, where $I(A, B)$ is the cardinality of the intersection between the sets. We then merge the two candidates, if $O(A, B) > \tau_o$ and retain the curve with larger cardinality, where we use $\tau_o=0.8$ as determined by hyper-parameter tuning.

Closed curve selection. The similarity matrix produces a closed-curve proposal for *each* of its N rows. Even after discarding proposals with confidence score $\gamma_n < \tau_\gamma$, many are *non-closed* curves, or represent the *same* closed curve; see Figure 11. We perform agglomerative clustering for proposals when $\text{IoU}(A, B) > \tau_{\text{IoU}}$, and retain the proposal in the cluster with the highest confidence. We use

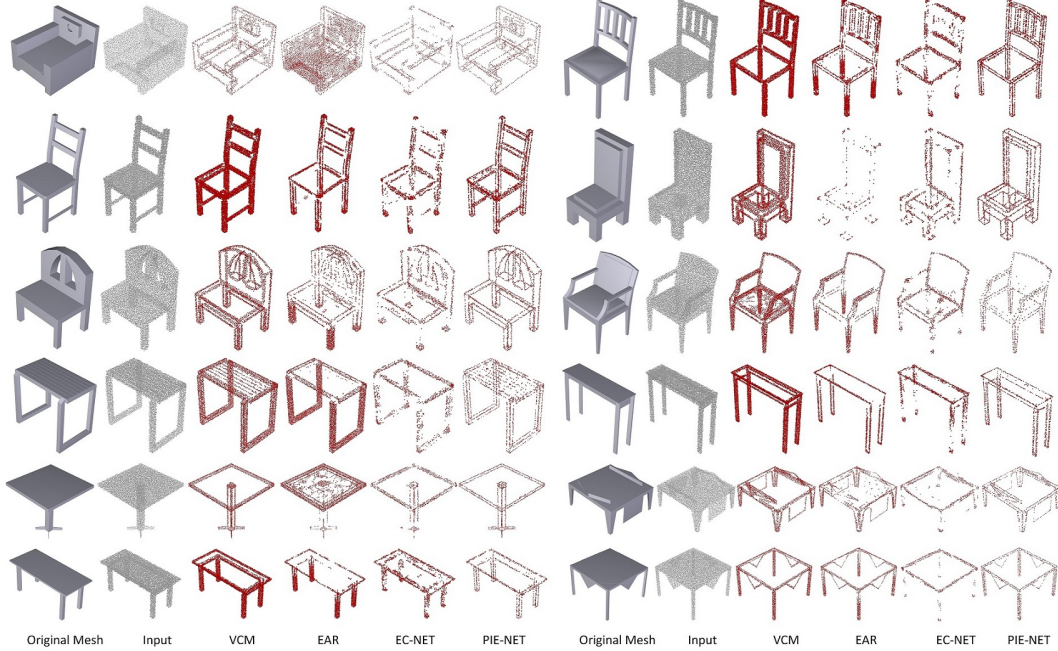


Figure 14: **Generalization to novel object categories** – While PIE-NET is trained on CAD models of mechanical assemblies from the ABC dataset [46], our edge detector is immediately applicable to 3D point clouds of general 3D objects and consistently outperforms VCM, EAR, and EC-Net.

$\tau_\gamma=0.6$ and $\tau_{\text{ou}}=0.6$ for all our experiments. Finally, for each closed segment, we select the best matching closed curve. Specifically, we use the Chamfer Distance to measure the matching score.

8 Backbone architectures

8.1 Classification

We use PointNet++ [47] as the feature backbone network. Following the same notations in PointNet++, $\text{SA}(K, r, n, [l_1, l_2, \dots, l_d])$ is a set abstract layer with K local regions of balls with radius r . n points are sampled for each local region. The SA layer uses d 1×1 conv layers with output channels l_1, l_2, \dots, l_d respectively. $\text{FP}(l_1, l_2, \dots, l_d)$ is a feature propagation (FP) layer with d 1×1 conv layers, whose output channels are l_1, l_2, \dots, l_d . In our task, we use four set abstract (SA) layers and four feature propagation (FP) layers. The parameters of set abstract (SA) and feature propagation (FP) layers are as follows: $\text{SA}(4096, 0.05, 32, [32, 32, 64])$, $\text{SA}(2048, 0.1, 32, [64, 64, 128])$, $\text{SA}(1024, 0.2, 32, [18, 128, 256])$, $\text{SA}(512, 0.4, 32, [256, 256, 512])$, $\text{FP}(256, 256)$, $\text{FP}(256, 256)$, $\text{FP}(256, 128)$, $\text{FP}(128, 128, 128)$. Following PointNet++, we also added four separate fully-connected layers in order to predict the following tasks: edge point classification \hat{T}_e , point-to-curve distance vector regression \hat{D}_e , corner point classification \hat{T}_c and corner point residual vector regression \hat{D}_c .

8.2 Open curve proposal

For each corner pair, we use a PointNet++ as the backbone with two multi-layer perceptrons (MLP) layers. $\text{MLP}([l_1, \dots, l_n])$ indicates several MLPs with output channels l_1, \dots, l_n . The parameters of these two MLPs are set to $[128, 256, 512]$, $[256, 256]$. Following PointNet, we added three separate fully-connected layers in order to predict the following tasks: curve segmentation, curve type classification, and curve parameters estimation.

8.3 Clustering

We use a PointNet++ as the clustering backbone. In our task, we use four set abstract (SA) layers and four feature propagation (FP) layers. The parameters of set abstract (SA) and feature propagation

(FP) layers are as follows: SA(512, 0.1, 32, [32,32,64]), SA(256, 0.2, 32, [64,64,128]), SA(128, 0.4, 32, [18,128,256]), SA(64, 0.8, 32, [256,256,512]), FP(256,256), FP(256,256), FP(256,128), FP(128,128,128). Following PointNet, we added one MLP layer in order to predict similarity matrix.

8.4 Closed curve fitting

In this task, we use a PointNet as the backbone with two MLP layers. The parameters of the two MLPs are set to [128,256, 512], [256, 256]. Following PointNet, we added two separate fully-connected layers to predict the curve parameters and the confidence scores.

Broader impact

We introduce a methodology that can benefit a range of current and new applications, from 3D data acquisition to object recognition. On the broader societal level, this work remains largely academic in nature, and does not pose foreseeable risks regarding defense, security, and other sensitive fields.

References

- [1] Guerrero, P., Y. Kleiman, M. Ovsjanikov, et al. PCPNET: Learning local shape properties from raw point clouds. In *Computer Graphics Forum (Proc. of Eurographics)*, vol. 37, pages 75–85. 2018.
- [2] Wang, P.-S., Y. Liu, X. Tong. Mesh denoising via cascaded normal regression. *ACM Trans. on Graphics*, 35(6):232:1–232:12, 2016.
- [3] Yu, L., X. Li, C.-W. Fu, et al. PU-Net: Point cloud upsampling network. In *Proc. IEEE Conf. on Computer Vision & Pattern Recognition*. 2018.
- [4] Koch, S., A. Matveev, Z. Jiang, et al. Abc: A big cad model dataset for geometric deep learning. In *Proc. IEEE Conf. on Computer Vision & Pattern Recognition*. 2019.
- [5] Gross, M., H. Pfister. *Point-Based Graphics*. Morgan Kaufmann, 2007.
- [6] Berger, M., A. Tagliasacchi, L. Seversky, et al. A survey of surface reconstruction from point clouds. *Computer Graphics Forum*, 36:301–329, 2017.
- [7] Qi, C. R., H. Su, K. Mo, et al. 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.
- [8] Guo, Y., H. Wang, Q. Hu, et al. Deep learning for 3d point clouds: A survey. *arXiv preprint arXiv:1912.12033*, 2019.
- [9] Duan, C., S. Chen, J. Kovacevic. 3d point cloud denoising via deep neural network based local surface estimation. In *ICASSP 2019-2019 IEEE International Conference on Acoustics, Speech and Signal Processing (ICASSP)*, pages 8553–8557. IEEE, 2019.
- [10] Yu, L., X. Li, C.-W. Fu, et al. EC-Net: an edge-aware point set consolidation network. In *Proc. Euro. Conf. on Computer Vision*, pages 398–414. 2018.
- [11] Li, L., M. Sung, A. Dubrovina, et al. Supervised fitting of geometric primitives to 3d point clouds. In *Proceedings of the IEEE Conference on Computer Vision and Pattern Recognition*, pages 2652–2660. 2019.
- [12] Su, H., V. Jampani, D. Sun, et al. Splatnet: Sparse lattice networks for point cloud processing. In *Proceedings of the IEEE Conference on Computer Vision and Pattern Recognition*, pages 2530–2539. 2018.
- [13] Graham, B., M. Engelcke, L. van der Maaten. 3d semantic segmentation with submanifold sparse convolutional networks. In *Proceedings of the IEEE Conference on Computer Vision and Pattern Recognition*, pages 9224–9232. 2018.
- [14] Yu, F., K. Liu, Y. Zhang, et al. Partnet: A recursive part decomposition network for fine-grained and hierarchical shape segmentation. In *Proceedings of the IEEE Conference on Computer Vision and Pattern Recognition*. 2019.
- [15] Yi, L., H. Huang, D. Liu, et al. Deep part induction from articulated object pairs. *ACM Transactions on Graphics (TOG)*, 37(6):209, 2019.
- [16] Wang, X., B. Zhou, Y. Shi, et al. Shape2motion: Joint analysis of motion parts and attributes from 3d shapes. In *Proceedings of the IEEE Conference on Computer Vision and Pattern Recognition*, pages 8876–8884. 2019.

- [17] Chen, D. Z., A. X. Chang, M. Nießner. Scanrefer: 3d object localization in rgb-d scans using natural language. *arXiv preprint arXiv:1912.08830*, 2019.
- [18] Wang, W., H. Pottmann, Y. Liu. Fitting b-spline curves to point clouds by curvature-based squared distance minimization. *ACM Transactions on Graphics (ToG)*, 25(2):214–238, 2006.
- [19] Park, H., J.-H. Lee. B-spline curve fitting based on adaptive curve refinement using dominant points. *Computer-Aided Design*, 39(6):439–451, 2007.
- [20] Flöry, S., M. Hofer. Constrained curve fitting on manifolds. *Computer-Aided Design*, 40(1):25–34, 2008.
- [21] Zheng, W., P. Bo, Y. Liu, et al. Fast b-spline curve fitting by l-bfgs. *Computer Aided Geometric Design*, 29(7):448–462, 2012.
- [22] Schnabel, R., R. Wahl, R. Klein. Efficient ransac for point-cloud shape detection. *Computer Graphics Forum*, 26(2):214–226, 2007.
- [23] Sanchez, V., A. Zakhor. Planar 3d modeling of building interiors from point cloud data. In *2012 19th IEEE International Conference on Image Processing*, pages 1777–1780. IEEE, 2012.
- [24] Wang, J., K. Xu. Shape detection from raw lidar data with subspace modeling. *IEEE transactions on visualization and computer graphics*, 23(9):2137–2150, 2016.
- [25] Gallardo, L. F., V. Kyrki. Detection of parametrized 3-d primitives from stereo for robotic grasping. In *2011 15th International Conference on Advanced Robotics (ICAR)*, pages 55–60. IEEE, 2011.
- [26] Remil, O., Q. Xie, X. Xie, et al. Surface reconstruction with data-driven exemplar priors. *Computer-Aided Design*, 88:31–41, 2017.
- [27] Wang, R., L. Liu, Z. Yang, et al. Construction of manifolds via compatible sparse representations. *ACM Transactions on Graphics (TOG)*, 35(2):14, 2016.
- [28] Tulsiani, S., H. Su, L. J. Guibas, et al. Learning shape abstractions by assembling volumetric primitives. In *Proceedings of the IEEE Conference on Computer Vision and Pattern Recognition*, pages 2635–2643. 2017.
- [29] Paschalidou, D., A. O. Ulusoy, A. Geiger. Superquadrics revisited: Learning 3d shape parsing beyond cuboids. In *Proceedings of the IEEE Conference on Computer Vision and Pattern Recognition*, pages 10344–10353. 2019.
- [30] Deng, B., K. Genova, S. Yazdani, et al. Cvxnets: Learnable convex decomposition. *arXiv preprint*, 2019.
- [31] Chen, Z., A. Tagliasacchi, H. Zhang. Bsp-net: Generating compact meshes via binary space partitioning. *arXiv preprint*, 2019.
- [32] Öztireli, A. C., G. Guennebaud, M. Gross. Feature preserving point set surfaces based on non-linear kernel regression. In *Computer Graphics Forum*, vol. 28, pages 493–501. Wiley Online Library, 2009.
- [33] Huang, H., S. Wu, M. Gong, et al. Edge-aware point set resampling. *ACM transactions on graphics (TOG)*, 32(1):9, 2013.
- [34] Zhou, Q.-Y., V. Koltun. Depth camera tracking with contour cues. In *Proceedings of the IEEE Conference on Computer Vision and Pattern Recognition*, pages 632–638. 2015.
- [35] Liu, L., N. Chen, D. Ceylan, et al. Curvefusion: reconstructing thin structures from rgbd sequences. In *SIGGRAPH Asia 2018 Technical Papers*, page 218. ACM, 2018.
- [36] Yu, L., X. Li, C.-W. Fu, et al. Pu-net: Point cloud upsampling network. In *Proceedings of the IEEE Conference on Computer Vision and Pattern Recognition*, pages 2790–2799. 2018.

- [37] Weber, C., S. Hahmann, H. Hagen. Sharp feature detection in point clouds. In *2010 Shape Modeling International Conference*, pages 175–186. IEEE, 2010.
- [38] Hackel, T., J. D. Wegner, K. Schindler. Contour detection in unstructured 3d point clouds. In *Proceedings of the IEEE Conference on Computer Vision and Pattern Recognition*, pages 1610–1618. 2016.
- [39] Gumhold, S., X. Wang, R. S. MacLeod. Feature extraction from point clouds. In *IMR*. Citeseer, 2001.
- [40] Fleishman, S., D. Cohen-Or, C. T. Silva. Robust moving least-squares fitting with sharp features. In *ACM transactions on graphics (TOG)*, vol. 24, pages 544–552. ACM, 2005.
- [41] Daniels Ii, J., T. Ochotta, L. K. Ha, et al. Spline-based feature curves from point-sampled geometry. *The Visual Computer*, 24(6):449–462, 2008.
- [42] Demarsin, K., D. Vanderstraeten, T. Volodine, et al. Detection of closed sharp edges in point clouds using normal estimation and graph theory. *Computer-Aided Design*, 39(4):276–283, 2007.
- [43] Feng, C., Y. Taguchi, V. R. Kamat. Fast plane extraction in organized point clouds using agglomerative hierarchical clustering. In *2014 IEEE International Conference on Robotics and Automation (ICRA)*, pages 6218–6225. IEEE, 2014.
- [44] Ren, S., K. He, R. Girshick, et al. Faster r-cnn: Towards real-time object detection with region proposal networks. In *Advances in neural information processing systems*, pages 91–99. 2015.
- [45] Wang, W., R. Yu, Q. Huang, et al. Sgpn: Similarity group proposal network for 3d point cloud instance segmentation. In *Proceedings of the IEEE Conference on Computer Vision and Pattern Recognition*. 2018.
- [46] Koch, S., A. Matveev, Z. Jiang, et al. Abc: A big cad model dataset for geometric deep learning. In *Proceedings of the IEEE Conference on Computer Vision and Pattern Recognition*, pages 9601–9611. 2019.
- [47] Qi, C. R., L. Yi, H. Su, et al. Pointnet++: Deep hierarchical feature learning on point sets in a metric space. In *Advances in neural information processing systems*, pages 5099–5108. 2017.
- [48] Lin, T.-Y., P. Goyal, R. Girshick, et al. Focal loss for dense object detection. In *Proceedings of the IEEE international conference on computer vision*, pages 2980–2988. 2017.
- [49] Girshick, R. Fast r-cnn. In *Proceedings of the IEEE international conference on computer vision*, pages 1440–1448. 2015.
- [50] Merigot, Q., M. Ovsjanikov, , et al. Voronoi-based curvature and feature estimation from point clouds. *IEEE Trans. Visualization & Computer Graphics*, 17(6):743–756, 2011.
- [51] Huang, H., S. Wu, M. Gong, et al. Edge-aware point set resampling. *ACM Trans. on Graphics*, 32:9:1–9:12, 2013.
- [52] Alliez, P., S. Giraudot, C. Jamin, et al. Point set processing. In *CGAL User and Reference Manual*. CGAL Editorial Board, 5.0 edn., 2019.
- [53] Groueix, T., M. Fisher, V. Kim, et al. Atlasnet: A papier-mâché approach to learning 3d surface generation. In *CVPR*. 2018.
- [54] Gadelha, M., R. Wang, S. Maji. Deep manifold prior. *arXiv preprint*, 2020.
- [55] Richter, S. R., S. Roth. Matryoshka networks: Predicting 3d geometry via nested shape layers. In *Proc. IEEE Conf. on Computer Vision & Pattern Recognition*. 2018.
- [56] Park, J. J., P. Florence, J. Straub, et al. Deepsdf: Learning continuous signed distance functions for shape representation. In *Proceedings of the IEEE Conference on Computer Vision and Pattern Recognition*, pages 165–174. 2019.

- [57] Dai, A., A. X. Chang, M. Savva, et al. Scannet: Richly-annotated 3d reconstructions of indoor scenes. *CoRR*, abs/1702.04405, 2017.
- [58] Wu, J., C. Zhang, T. Xue, et al. Learning a probabilistic latent space of object shapes via 3d generative-adversarial modeling. In *Advances in Neural Information Processing Systems (NIPS)*, pages 82–90. 2016.
- [59] Chen, Z., H. Zhang. Learning implicit fields for generative shape modeling. In *IEEE Computer Vision and Pattern Recognition (CVPR)*. 2019.