NerVE: Neural Volumetric Edges for Parametric Curve Extraction from Point Cloud

Xiangyu Zhu[12*] Dong Du[1*] Weikai Chen[3] Zhiyou Zhao[1]
Yinyu Nie[4] Xiaoguang Han[12†]

[1]SSE, CUHKSZ [2]FNii, CUHKSZ [3]Tencent America [4]Technical University of Munich

Abstract

Extracting parametric edge curves from point clouds is a fundamental problem in 3D vision and geometry processing. Existing approaches mainly rely on keypoint detection, a challenging procedure that tends to generate noisy output, making the subsequent edge extraction error-prone. To address this issue, we propose to directly detect structured edges to circumvent the limitations of the previous point-wise methods. We achieve this goal by presenting NerVE, a novel neural volumetric edge representation that can be easily learned through a volumetric learning framework. NerVE can be seamlessly converted to a versatile piece-wise linear (PWL) curve representation, enabling a unified strategy for learning all types of free-form curves. Furthermore, as NerVE encodes rich structural information, we show that edge extraction based on NerVE can be reduced to a simple graph search problem. After converting NerVE to the PWL representation, parametric curves can be obtained via off-the-shelf spline fitting algorithms. We evaluate our method on the challenging ABC dataset [19]. We show that a simple network based on NerVE can already outperform the previous state-of-the-art methods by a great margin.

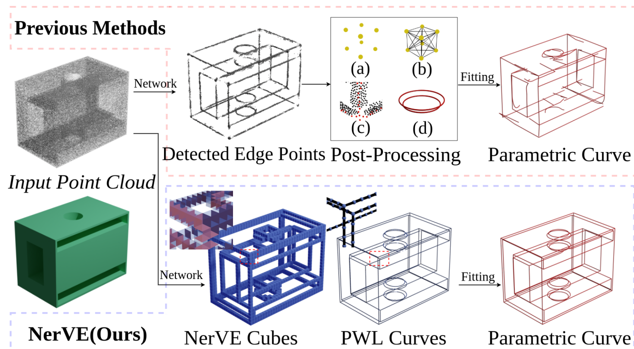

Figure 1. We present NerVE, a neural volumetric edge representation for parametric curve extraction from point clouds. Our method can predict structured NerVE instead of unstructured edge points in previous methods, and directly convert NerVE into piece-wise linear (PWL) curves, reducing the error-prone post-processing of previous methods into simple graph search on PWL curves. Post-processing includes (a) Endpoint detection;(b) Graph structure analysis;(c) Points thinning;(d) Special treatment for closed curves.

1. Introduction

The advances of 3D scanning techniques have enabled us to digitize and reconstruct the physical world, benefiting a wide range of applications including 3D modeling, industrial design, robotic vision, *etc*. However, point clouds, the raw output of a 3D scanner, are typically noisy, unstructured, and can exhibit strong sampling bias. Hence, extracting structured features, such as the feature edges, from an unordered point cloud is a vital geometry processing task. Sharp geometric edges can be used as an abstraction of a complex 3D shape, facilitating downstream tasks including surface reconstruction, normal estimation, and shape classification. Previous state-of-the-art methods mainly resort to a *keypoint fitting* strategy to extract parametric edge curves from a point cloud. Specifically, they first detect a sparse set of keypoints, such as the endpoints or points on sharp edges, and then group these points into individual sets according to predefined topologies. Finally, each point set is converted into a parametric curve using spline fitting.

Recent approaches have strove to improve the accuracy of edge point detection by using hand-crafted features [26] or deep neural networks [2, 25, 36, 41]. Despite the impressive progress that has been made, existing works still have the following limitations. 1) The widely adopted *point-wise* classification approaches tend to generate noisy estimations – the predicted edge points typically contain a spurious set of candidate points (see Fig. 5), which requires further processing for keypoint cleaning and increases the risk of false/missing connections. 2) The grouping procedure highly relies on the accuracy of endpoint detection. However, it remains difficult

*Xiangyu Zhu and Dong Du contribute equally.
†Corresponding author's email: hanxiaoguang@cuhk.edu.cn

to accurately locate endpoints especially when the normal of its surrounding points change smoothly. 3) They require tedious treatments to cope with different curve topologies, including curve type estimation, topology-dependent artifact points removal and curve connection, *etc*.

Our key observation is that the above issues can be resolved if we can *directly predict structured edges* in the form of *piece-wise linear* (PWL) curves from the input point cloud. This bypasses the problematic keypoint detection and avoids the error-prone edge extraction in the curve fitting stage. In addition, PWL curve is a general representation of free-form curves, removing the need of curve topology estimation and the laborious curve fitting and post-processing dependent on curve category. Furthermore, PWL curves can be easily converted to parametric curves using the off-the-shelf solutions. However, unlike its parametric counterpart, PWL curves are notoriously difficult to predict due to its large degrees of freedom.

Towards this end, we propose *NerVE*, a novel neural edge representation in a volumetric fashion. As shown in Fig. 3, NerVE represents 3D structured edges using a regular grid of volumetric cubes – each cube encodes rich structural information including 1) one binary indicator of edge occupancy, 2) edge orientations (if any), and 3) one edge point position (if any). Thereby, NerVE can be readily converted to PWL curves by connecting the edge points enclosed by NerVE cubes according to the encoded point connectivity. The introduction of NerVE brings several advantages. First, the generated NerVE cubes are structured by itself, which greatly simplifies the process of curve extraction. Second, it is fully compatible with the PWL curve representation, and hence, can deal with all types of curves in a unified manner. Third, NerVE cubes can be viewed as a coarse representation of the point cloud. Predicting the occupancy of a volumetric cube is easier and more robust than point-wise classification. Therefore, we are less likely to suffer from the issue of missing curves (see our results in Fig. 6). Lastly, inferring NerVE can be formulated as a voxel-wise classification and regression problem, where the well-developed 3D convolutional networks can be directly employed.

We further propose a volumetric learning framework to predict NerVE from the input point cloud. We first encode the point features into a volumetric feature grid with the same resolution of the output. Then, a multi-head decoder is used to predict the attributes of a NerVE cube from its corresponding feature cell. After converting the NerVE cubes into PWL curves, a specially-tailored post-processing procedure is proposed to correct potential topology errors in the resulting curves. Finally, the parametric curves can be obtained via a straightforward spline fitting algorithm.

We evaluate our method on the ABC dataset [19], a large-scale collection of computer-aided design (CAD) models with challenging topology variations. In particular, we compare with the state-of-the-art approaches on two different tasks: edge estimation and parametric curve extraction. Experimental results show that by leveraging the proposed NerVE representation, our method can faithfully extract complete and accurate edges and parametric curves from intricate CAD models, outperforming the other methods both qualitatively and quantitatively.

We summarize our contributions as follows:

- We propose *NerVE*, a learnable neural volumetric edge representation that supports direct estimation of structured 3D edges, seamless conversion with general PWL curves, and compatibility with latest volumetric learning framework.
- A pipeline for parametric curve extraction from point cloud that consists of a learning-based framework for faithful NerVE cubes estimation and a post-processing module for curve topology correction.
- We set a new state-of-the-art on the ABC dataset in the task of parametric curve extraction from point cloud.

2. Related Works

Edge Feature Detection. Traditional edge feature detection from a point cloud is based mainly on local geometric features, such as the eigenstructure of the covariance matrix [1, 13, 31, 40], normals [8, 9, 11, 37], curvatures [15, 21], or other statistical metrics [29, 38]. To improve the robustness of edge detection for noisy point clouds, Daniels et al. [7] use a multistep refinement method with robust moving least squares to fit the surface to potential features. Ni et al. [28] combine RANSAC and the angular map metric to detect edges. VCM [26] is presented by measuring Voronoi covariance and applying the Monte-Carlo algorithm to compute feature boundaries, which is widely used in geometry processing. With the bloom of deep learning, PIE-Net [36], EDC-Net [2], and PCEDNet [16] are proposed to formulate the edge detection as a classification task and utilize neural networks to learn it. On the other hand, EC-Net [41] reformulates it as a regression problem, then learns residual point coordinates and point-to-edge distances to identify edge points. All of these learning-based methods significantly improve the accuracy of edge detection. In this paper, we also utilize advanced neural networks but avoid explicit edge detection. We propose a novel NerVE representation to directly learn the positions and topology of edge points.

Parametric Curve Extraction. Parametric curves have been widely used in CAD modeling to design complex 3D shapes with sharp geometries. However, extracting parametric curves from a point cloud is challenging due to the various types and complex connections of curves. Pioneering work [13] attempts to detect feature points and fits them with

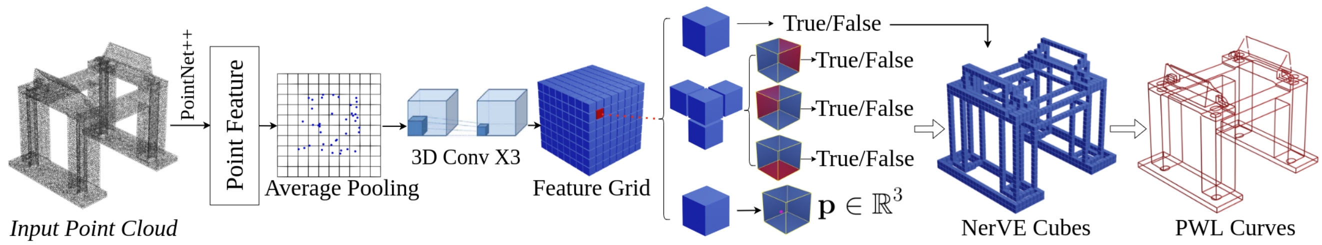

Figure 2. Overview of our proposed network for learning NerVE. Given a point cloud, we first utilize a simplified PointNet++ [32] module and a 3D CNN module to obtain the feature grid (has the same resolution of our NerVE output). Three individual decoders are applied to process cube features in the grid to predict the corresponding three attributes of NerVE, i.e., edge occupancy, edge orientation, and edge point position. By converting the NerVE cubes into PWL curves, we can obtain the parametric curves of the shape with a post-processing.

splines to recover feature lines. Recent deep learning methods [14, 22, 24, 25, 34, 36] leverage a keypoint fitting strategy that first detects a sparse set of keypoints (e.g., corner points and edge points), then groups and connects them, followed by a determination of the target parametric curve type and performs curve fitting. Specifically, PIE-Net [36] utilizes a PointNet++ [32]-like network to extract point features for the classification of edge points, corners, and others, then generates curve proposals for parametric curve extraction. ComplexGen [14] formulates the prediction of validness and primitive types as classification tasks, and recovers corners, curves, and patches simultaneously along with their mutual topology constraints. In contrast, DEF [25] regresses a continuous distance field to represent the distance from the input points to the closest feature lines, and then extracts parametric feature curves from the inferred field. Other works [22, 24, 34] simplify the curves into lines only, and focus on 3D wireframe reconstruction. However, complicated corner detection and connection estimation make these keypoint fitting methods prone to produce artifacts (e.g., missing curves). To this end, we propose NerVE, a novel neural-based edge representation that supports the prediction of structured edges in the form of PWL curves, making the following parametric curve extraction easy and efficient.

Neural Representation Learning. Many well-known 3D representations, such as voxels [6], point clouds [10], meshes [12, 35], occupancy fields [27], and signed distance functions [4, 30], have been introduced into deep learning to resolve the problem of 3D reconstruction and achieve impressive results. Liao et al. [20] propose a differentiable learning architecture to represent the classical Marching Cubes [23] algorithm for shape reconstruction. Chen et al. [3, 5] further extend the algorithms of Marching Cubes [23] and Dual Contouring [18] with data-driven approaches. Specifically, they implicitly represent triangle meshes in compact per-cube parameterizations that are compatible with neural learning. With the learned implicit field, a high-quality triangle mesh with sharp features can be directly extracted. Inspired by the tendency, we introduce the traditional PWL curve representation into deep learning for parametric curve estimation. To make it easy to learn PWL curves, we propose NerVE to parameterize PWL curves in a uniform volumetric field and apply advanced neural networks for learning.

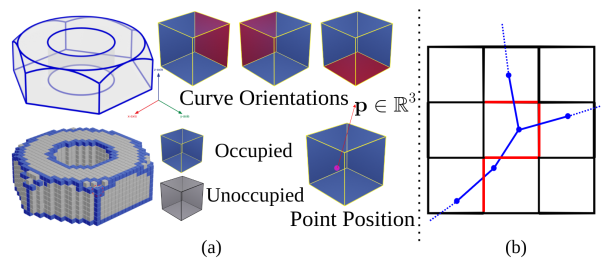

Figure 3. (a) Definition of the three attributes in NerVE. (b) Illustration of PWL curve extraction from NerVE. A red edge is corresponding to the red face in (a), showing there exists at least one curve passing through the face, i.e., the edge orientation is true.

3. Method

Overview. Different from traditional keypoint fitting workflows [2, 25, 36, 41], we present a new paradigm to generate accurate parametric curves using a neural volumetric edge representation, named *NerVE*. First, we introduce the definition of NerVE and then propose a dedicated network to learn NerVE that supports direct estimation of structured 3D edges. These edges compose PWL curves that can not only provide precise edge positions but also contain valuable topology information, which eases the extraction of parametric curves. For the inferred PWL curves, we further utilize a simple post-processing procedure to correct their topology errors. Finally, parametric curves can be obtained using a straightforward graph search and spline fitting algorithm.

3.1. NerVE Representation

Learning 3D parametric curves from a point cloud is a challenging task due to the complexity of both curve types and connections. Existing works mainly resort to a keypoint fitting strategy and tend to generate noisy or error-prone results. In this work, we propose a unified edge representation, NerVE, that is compatible with different types of curves and is also feasible to be inferred with the latest volumetric learning techniques.

3.1.1 Volumetric Edge Definition

We follow the voxel representation of a 3D shape [6] to discretize continuous 3D curves into a volumetric grid. As shown in Fig. 3, each cube in NerVE contains three attributes: 1) edge occupancy o. $o \in \{0, 1\}$, which is a binary value to define whether the cube contains edges; 2) edge orientations e. $e_i \in \{0, 1\}, i = 1, 2, 3$, which are 3 binary values that represent whether the cube should connect with its adjacent cubes to construct edge pieces. Note that a cube has 6 faces and shares with its neighbors, thus we define only three statuses on the left, bottom, and back faces in a cube; 3) edge point position p, which defines an edge point in the cube. With the three defined attributes, we can discretize continuous curves in a unified and regular volumetric representation, which can be learned directly via neural networks.

For NerVE, the higher resolution could keep more curve information but also increase the calculation consumption for network training and curve extraction. In our method, we use the resolution of 32^3, which we found it works well in our experiments. We provide the ablation of different NerVE resolutions in Sec. 4.4. Regarding the determination of ground-truth p, we generally choose the midpoint of the truncated curve inside a cube, which is natural and works well. For a special case where multiple curves appears in a cube, we calculate the average midpoint position or pick the endpoint (if any) as p. We also provide an ablation study about the selection of p in Sec. 4.4.

3.1.2 PWL Curve Extraction

Given our volumetric edge representation (i.e., NerVE), the underlying curves can be easily extracted in the form of PWL curves. Specifically, we pick the cube points $\{p\}$ when their occupancy statuses are true, and connect them with their neighbor points if the edge orientation statuses are true. Hence, a PWL curve is obtained, as illustrated in Fig. 2.

3.2. NerVE Learning

Given a point cloud $\mathcal{X}$ in the 3D space, we adopt a dense and simplified PointNet++ [32] module and a simplified 3D CNN module [6] to extract efficient local features for the unorganized point cloud, then use an MLP-based module to generate our NerVE, denoted as $\{y_{(i,j,k)}\}$, where $y = (o, e, p)$ is the volumetric cube attributes at the position of (i, j, k). We can formulate it as learning a function $\mathcal{F}_\theta$,

$$\mathcal{F}_\theta : \mathcal{X} \longrightarrow \{y_{(i,j,k)}\}, \forall (i, j, k), \tag{1}$$

where θ denotes the parameters in a neural network.

3.2.1 Network Architecture

Our network adopts an encoder-decoder paradigm to individually learn each attribute (o, e, p) in NerVE cubes (see Fig. 2). Note that the encoder is comprised of point convolutions (1-dimensional convolution) and volume convolutions (3D CNN). We use the average pooling to connect these two kinds of convolutions, providing a feature grid for the following learning. The decoder is based on MLP layers.

Encoder. Our encoder is a dense and simplified PointNet++ [32] combined with a 3D CNN module. Specifically, for each input point $x \in \mathcal{X}$, we first calculate its k-nearest neighbors ($k = 8$ in our experiments, following [3]), and then apply 4 MLP layers, each layer followed by a leaky ReLU activation, to extract 128-dimension point features. Then we utilize average pooling to fuse the features of points if they fall into the same cubes. Hence, we encode the point features into cubes. For these cubes without points inside, we initialize their features with zeros for the following 3D CNN learning. The 3D CNN has 3 3D-convolution layers, each having a kernel size of 3, stride length of 1, and padding size of 1. A leaky ReLU activation is appended after each convolution layer.

Decoder. We apply three decoders to learn the three cube attributes, respectively. Each decoder is comprised of 5 MLP layers with a leaky ReLU activation except the final one. Using the 128-dimension volumetric features from our encoder as input, we decode each cube feature to predict an occupancy o and a point p. For the learning of edge orientation, we concatenate a cube feature with its three adjacent ones before decoding, as shown in Fig. 2. Furthermore, we use a sigmoid function for the outputs of o and e.

3.2.2 Training

Compared to the 3D shape volume, the edge volume representation is much more sparse. To address the data imbalance problem in training and reduce the calculation consumption, we learn each cube attribute with specific masks, denoted $\mathcal{M}_o$, $\mathcal{M}_e$, $\mathcal{M}_p$. Specifically, given a point cloud, we pick all the volumetric surface cubes ($\mathcal{M}_o$) to train their edge occupancy $\{o\}$. The edge orientations $\{e\}$ and edge points $\{p\}$ are learned from those occupied edge cubes. Thus, we use the ground-truth edge cubes as masks $\mathcal{M}_e$, $\mathcal{M}_p$ to constrain that the edge orientations and edge point positions are learned in edge cubes during training. During inference, we use the inferred edge occupancies $\{o\}$ as the mask to predict their edge orientations $\{e\}$ and edge points $\{p\}$.

To train our network, we choose the binary cross entropy (BCE) loss for the learning of edge occupancy o and edge orientation e, and adopt the L_1 loss for the regression of edge point p, i.e.,

$$L_o = 1/|\mathcal{M}_o| \sum_{o \in \mathcal{M}_o} \mathrm{BCE}(o, o_{gt}), \tag{2}$$

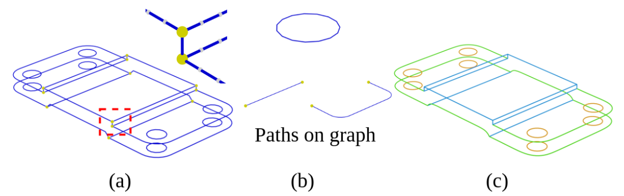

Figure 4. Parametric curve extraction via graph search. (a) The points with degree$>$ 2 are regarded as endpoints. (b) Finding a path between two endpoints or a closed path by searching on the PWL curve graph. (c) Final parametric curves after fitting the path.

$$L_e = 1/|\mathcal{M}_e| \sum_{e \in \mathcal{M}_e} \text{BCE}(e, e_{gt}), \tag{3}$$

$$L_p = 1/|\mathcal{M}_p| \sum_{p \in \mathcal{M}_p} \|p - p_{gt}\|_1 , \tag{4}$$

where o_{gt}, e_{gt}, p_{gt} are the ground-truth cube attributes in the corresponding masked cubes.

3.3. Parametric Curve Extraction

Most existing methods for parametric curve estimation struggle with the endpoint estimation & connection, and the curve type recognition (e.g., line, circle, B-spline) from a point cloud in the learning stage. In contrast, our representation of NerVE is unified for learning various types of curves. NerVE can directly predict structured edges in the form of PWL curves, establishing a topology graph of edge points, as shown in Fig. 4. The parametric curve extraction is then reduced to a simple graph-search problem. In this paper, we focus on curve extraction of CAD models.

For CAD models which are known as boundary representation (B-Rep), there are two constraints for parametric curves: 1) one open curve has only two endpoints, and two open curves can only be intersected at one endpoint; 2) the closed curves have zero endpoints [17, 33]. Given the PWL curves estimated from the inferred NerVE, we first choose the edge points with degree>2 [33, 39] as endpoints. Here, the point degree is the number of edges that connect with the point in the PWL curves. Then we start from an endpoint and search along the PWL curve to obtain another endpoint. We thus fit with these two endpoints and all edge points on the path using an off-the-shelf spline fitting library. For the closed curve extraction, we just pick an edge point with a degree of 2 and search along the PLW curve graph until it ends up, and then fit with all points on the path. Our PWL curve graph greatly simplifies parametric curve extraction compared to previous methods.

However, it is inevitable to produce artifacts for a learning-based method, reducing the accuracy of curve extraction. To this end, we propose a straightforward post-processing method to refine the topology of PWL curves extracted from NerVE. Specifically, we connect two points with degree 1 if the distance between these two points is smaller than a given threshold and their tangent vectors are consistent. We also remove superfluous curve segments if their lengths are very short. More details are provided in the supplemental.

4. Experiment

In this section, we evaluate the effectiveness of our method with learning-based methods [25, 36, 41] and a traditional method [26]. Then we test the robustness by adding noise or changing the sampling density of a point cloud, and provide an ablation study to explore the effects of certain choices in our method.

4.1. Experiment Setup

Dataset. We train and test our model on the ABC dataset [19], which has one million CAD models in total created by human users. We use the first chunk for our experiments, which is already enough for use. Following [3, 25, 36], we consider *sharp* curves in CAD models. However, some models need to be filtered due to data missing, shape repetition, or lack of sharp edges. The final filtered dataset contains 2364 models and is randomly split into a train set (80%) and a test set (20%). We provide additional details on dataset cleaning and data preparation in the supplemental.

Implementation. In our experiments, we set the resolution of NerVE grid at 32^3. The effects of using higher resolutions are discussed in our ablation study (see Sec. 4.4). All input point clouds are normalized into $[-1, 1]^3$. We train our networks on a NVIDIA RTX 3090 for 60 epochs. The Adam optimizer is used with an initial learning rate of 5e-4. We follow NDC [3] and set the batch size as 1. The input point number varies for different shapes. On average, it is around 22,000. For more details like layer specifications and time cost, we provide them in the supplemental.

Metrics. We quantitatively evaluate the prediction of our networks using average recall R_o and precision P_o for edge occupancy, average correct rate C_e for edge orientations and average L_2 distance D_p for edge point positions. To define C_e, in the evaluation of edge orientation, a cube is regarded correct if all of its predictions of three faces are identical to GT, otherwise the cube is wrong. C_e is defined as the ratio of correct cubes in all considered cubes in an input. Note that the metrics are calculated under the cube masks defined in Sec. 3.2.2.

To compare the quality of both PWL curves and parametric curves with other methods, we adopt the typical Chamfer distance (CD) and average Hausdorff distance (HD), which assess the similarity between two point sets. Assume X, Y are two finite point sets, CD and HD can be calculated as:

$$\text{CD} = \frac{1}{|X|} \sum_{p \in X} \min_{q \in Y} \|p - q\|_2^2 + \frac{1}{|Y|} \sum_{q \in Y} \min_{p \in X} \|q - p\|_2^2,$$

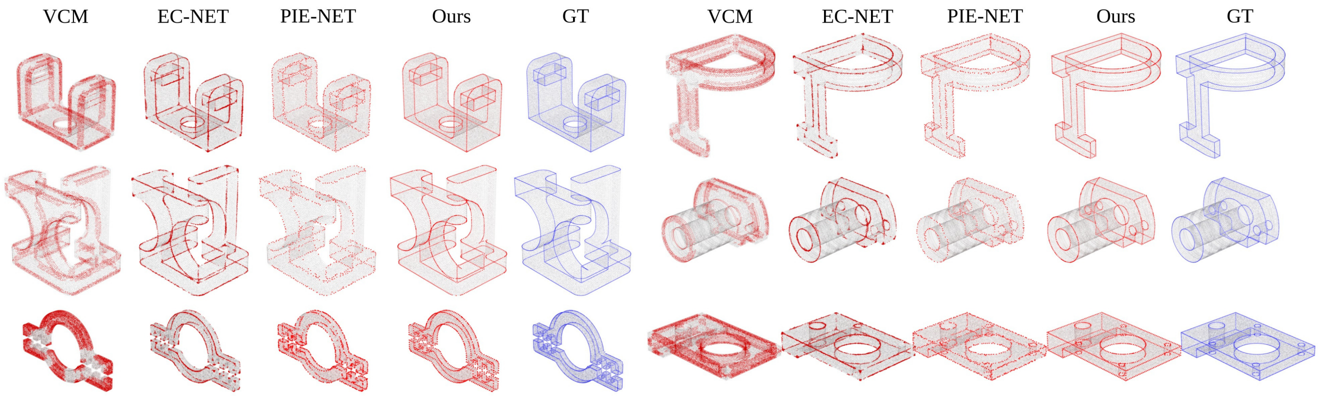

Figure 5. Qualitative comparisons on the edge point estimation with VCM [26], EC-NET [41] and PIE-NET [36].

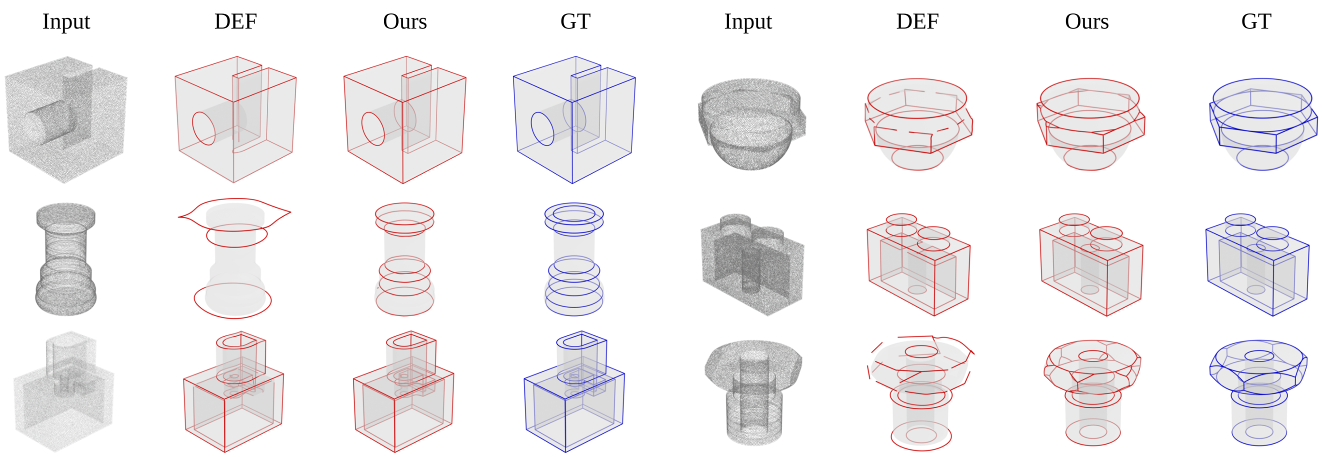

Figure 6. Qualitative comparisons with DEF on parametric curve extraction.

$$\text{HD} = \frac{1}{2}(\max_{p \in X} \min_{q \in Y} \|p - q\|_2 + \max_{q \in Y} \min_{p \in X} \|p - q\|_2).$$

4.2. Comparisons

Edge Estimation. We evaluate our predicted PWL curves on our test set by comparing them with the baseline methods: VCM [26], EC-NET [41] and PIE-NET [36]. To measure the error between predicted edge points and the ground-truth, we convert PWL curves into point sets by sampling the midpoints of all edges. Numerical results are reported in Table 1 and visual comparisons are shown in Fig. 5. We provide the settings of baseline methods in the supplemental.

	VCM [26]	EC-NET [41]	PIE-NET [36]	Ours
CD↓	0.0226	0.0037	0.0074	**0.0012**
HD↓	0.1941	0.1284	0.1318	**0.0714**

Table 1. Quantitative comparisons on edge points estimation.

The quantitative and qualitative results show that our method significantly outperforms the baselines, where we produce accurate and uniformly distributed points visually. VCM and EC-NET are prone to generate redundant points around the ground-truth edges, thus producing higher CD and HD values. PIE-NET has a thinner band of outputs, but it tends to miss some edge points.

Parametric Curves. With the predicted PWL curves as input, our method fit them to generate parametric curves. We compare our parametric curves with the state-of-the-art method DEF [25] both quantitatively and qualitatively on the DEF-Sim [25] dataset, which has 68 carefully selected shapes from ABC dataset. Since the official code of DEF is not available, we use the input data in DEF-Sim and their results of parametric curves provided by DEF's authors. We directly use the reported results of PIE-Net and DEF from DEF's paper [25] for quantitative comparisons (shown in Table 2). Since PIE-NET does not provide the official implementation for its parametric curve inference, we only compare with DEF qualitatively in Fig. 6.

	PIE-NET [36]	DEF [25]	Ours(32^3)	Ours(64^3)
CD↓	0.97	0.04	0.008	**0.005**
HD↓	2.19	0.55	0.224	**0.184**

Table 2. Quantitative comparisons on parametric curve extraction.

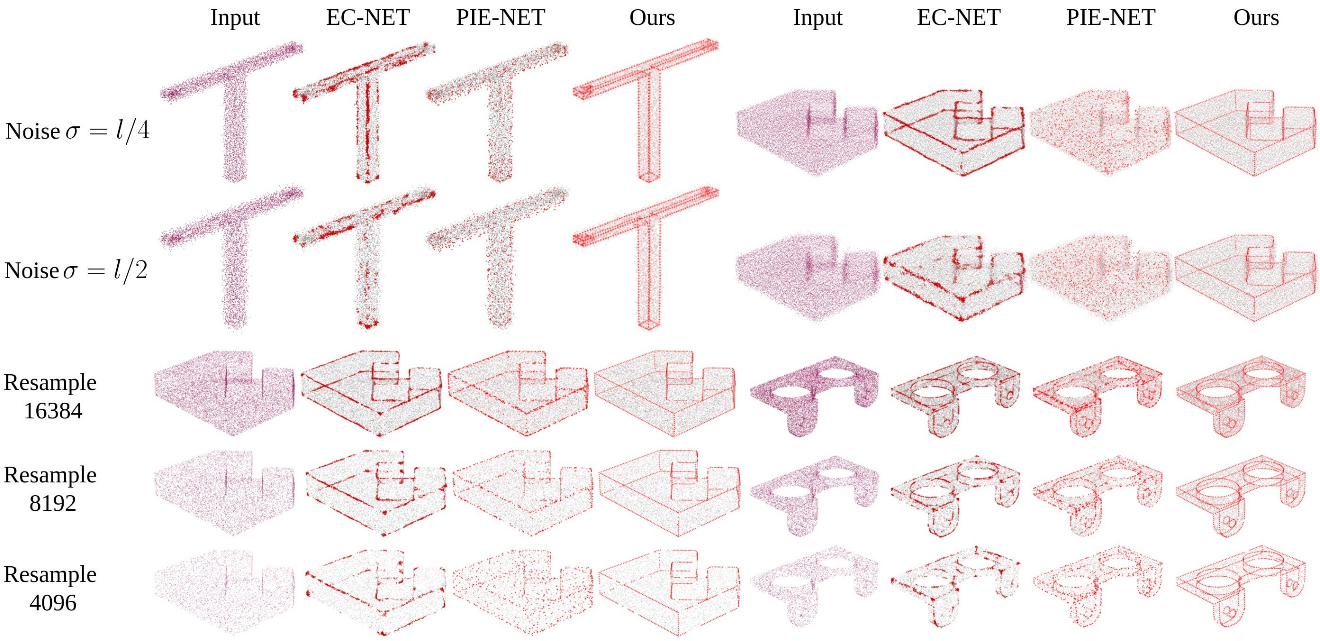

Figure 7. Qualitative comparisons with EC-NET [41] and PIE-NET [36] on noisy input points or input points with varying sampling density. $l = 2/32$ is the edge length of a cube in the grid.

The results in Table 2 show that our method presents significantly better performance than DEF and PIE-NET, where the CD and HD errors of PIE-NET primarily arise from missing curve instances, which usually happen when keypoints (e.g., corner points or edge points) are wrongly predicted or simply missed. As shown in Fig. 6, DEF performs well around sharp corners but struggles in processing structures with circle curves. In contrast, our method can handle closed curves and sharp corners consistently and produce convincing parametric curves in visual.

4.3. Stress Tests

We investigate the robustness of our method using point clouds with varying noise or sampled point numbers. We train and test networks on our dataset with augmentation by adding Gaussian noise and random resampling.

NerVE Prediction. Table 3 shows the effects of noise intensity and sampled point number on the prediction of our NerVE representation. We observe that even with a large noise intensity ($\sigma = l/2$, where l is the edge length of a cube in the grid) or much fewer sampled points (4,096), our method still can produce reasonable results of PWL curves with low-level CD and HD errors.

Edge Estimation. We also compare with VCM [26], EC-NET [41] and PIE-NET [36] on noisy or resampled inputs. Table 4 shows that our method achieves the best numerical performance. Fig. 7 further demonstrates that our method outperforms baseline methods and presents robustness against noisy or resampled inputs.

		$R_o\uparrow$	$P_o\uparrow$	$C_e\uparrow$	$D_p\downarrow$	CD$\downarrow$	HD$\downarrow$
Clean		**0.965**	**0.965**	**0.940**	**0.0030**	**0.0012**	**0.071**
Noise	$\sigma = l/4$	0.919	0.937	0.887	0.0053	0.0019	0.097
	$\sigma = l/2$	0.892	0.923	0.854	0.0079	0.0027	0.110
#Sample Points	16384	0.952	0.955	0.919	0.0044	0.0019	0.095
	8192	0.938	0.951	0.902	0.0048	0.0028	0.112
	4096	0.923	0.941	0.876	0.0053	0.0057	0.146

Table 3. Influence of input points with varying noise or sampled point numbers. $l = 2/32$ is the edge length of a cube in the grid.

		VCM [26]	EC-NET [41]	PIE-NET [36]	Ours
Clean		0.0226	0.0037	0.0074	**0.0012**
Noise	$\sigma = l/4$	0.0253	0.0055	0.0323	**0.0019**
	$\sigma = l/2$	0.0250	0.0132	0.0420	**0.0027**
#Sample Points	16384	0.0211	0.0039	0.0060	**0.0019**
	8192	0.0214	0.0060	0.0145	**0.0028**
	4096	0.0223	0.0130	0.0216	**0.0057**

Table 4. Edge estimation errors (CD, the smaller the better) of four methods on noisy or resampled inputs. $l = 2/32$ is the edge length of a cube in grid. Results of HD are provided in the supplemental.

4.4. Ablation Study

Resolution. Table 5 shows the performance of our method under different NerVE grid resolution, where we observe that using resolution 32^3 achieves slightly better performance on occupancy prediction of edge cubes. As a binary classification problem, data imbalance of edge occupancy is aggravated as resolution increases since the number of non-edge points grows faster than edge points number. Therefore, using resolution 64^3 meets a more challenging classification problem, producing slightly worse R_o and P_o than using resolution 32^3. Nevertheless, we notice that using resolution

	R_o↑	P_o↑	C_e↑	D_p↓	CD↓	HD↓
Reso 32^3	**0.965**	**0.965**	**0.940**	0.003	0.0012	0.0714
Reso 64^3	0.958	0.960	**0.940**	**0.001**	**0.0009**	**0.0523**

Table 5. Influence of grid resolution on NerVE prediction and PWL curves generation.

64^3 can achieve better CD and HD. Meanwhile, the parametric curves under resolution 64^3 present better performance as shown in Table 2. Fig. 8 shows the visualizations and provides some insights to explain the phenomenon.

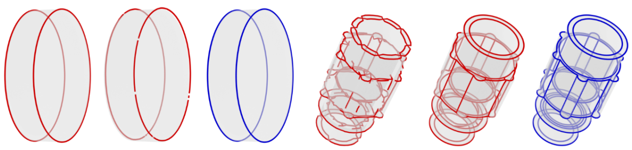

Figure 8. Qualitative comparisons on the resolution of 32^3 and 64^3 for NerVE. The resolution 64^3 can represent complicated curves but also have imperfections in simple instances.

For simple shapes, e.g., the left side of Fig. 8, using a higher resolution (e.g., 64^3) could disconnect at several positions on curves, which will degrade the edge occupancy accuracy of its PWL curves, but these artifacts (e.g., disconnection) can be well addressed by a simple post-processing in the following parametric curve extraction. On the contrary, using resolution 32^3 suffers from representing complicated shapes (right side in Fig. 8) while using resolution 64^3 performs better and brings more geometric details. More experiments and analysis are provided in the supplemental.

Cube Point Choice. As stated in Sec. 3, edge point position in a cube is defined as the midpoint of the truncated curve. Another option is to use a point which minimizes a quadratic error function (QEF), similar to the point definition in Dual Contouring [18]. We provide the details in our supplemental. We validate our choice by restoring curves from ground-truth NerVE with two different definitions of point position. As shown in Table 6, current definition of the point position clearly has better performance on curve restoration.

	DC QEF	Our Choice
CD ↓	0.002799	**0.000136**
HD ↓	0.052306	**0.024263**

Table 6. Ablation study of edge point estimation on ground-truth NerVE. We measure the errors of generated PWL curves using two types of point positions in cubes, i.e, DC QEF [18] and ours.

5. Conclusions

In this paper, we propose NerVE, a neural volumetric edge representation, for the extraction of parametric curves from a point cloud. This edge structure representation is fully compatible with the volumetric learning framework and can be easily converted to explicit PWL curves, which greatly reduces the complexity of following parametric curve fitting. The quantitative and qualitative results in the experiments evidently show the superiority of our method.

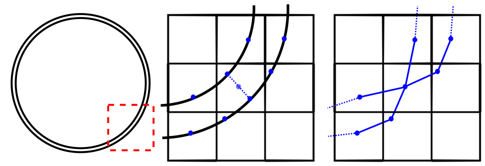

Figure 9. Limitations of our method. When two curves are close to each other (a), there is a possibility that two edge points respectively from them fall into the same cube, which would be processed into one edge point (c) with a low-resolution NerVE.

Limitations and Future Works. One limitation is that our method may produce unexpected junction points. As illustrated in Figure 9, when two curves are so close and pass through the same cube, it will produce a junction point since we only predict one point in one cube by definition. This issue is essentially the cost of discretization, but it is insignificant in a statistical sense. In fact, there are only 1.62% cubes with described junction points in all edge-occupied cubes in our dataset at resolution 32^3, and the number decreases to 0.71% as the resolution increases from 32^3 to 64^3. Another limitation is the vertex in one cube can only have at most 6 connections, due to the definition of NerVE and the structure of cube grids. In the future, we would like to devise a better representation that can completely solve these problems.

Designing a differentiable architecture for the extraction of PWL curves or parametric curves is also an interesting future work. Furthermore, notice NerVE can essentially represent general edges, we plan to test its ability boundary. For example, we wish to apply it not only to CAD models but also to some other shapes that have thin structures, such as the neural nerves of humans.

Acknowledgment. This work was partially supported by NSFC-62172348, the Basic Research Project No. HZQB-KCZYZ-2021067 of Hetao Shenzhen-HK S&T Cooperation Zone, the National Key R&D Program of China with grant No. 2018YFB1800800, by Shenzhen Outstanding Talents Training Fund 202002, by Guangdong Research Projects No. 2017ZT07X152 and No. 2019CX01X104, by the Guangdong Provincial Key Laboratory of Future Networks of Intelligence (Grant No. 2022B1212010001), and by Shenzhen Key Laboratory of Big Data and Artificial Intelligence (Grant No. ZDSYS201707251409055). It was also supported in part by Outstanding Yound Fund of Guangdong Province with No. 2023B1515020055 and Shenzhen General Project with No. JCYJ20220530143604010. It was also sponsored by CCF-Tencent Open Research Fund.

References

[1] Dena Bazazian, Josep R Casas, and Javier Ruiz-Hidalgo. Fast and robust edge extraction in unorganized point clouds. In *2015 international conference on digital image computing: techniques and applications (DICTA)*, pages 1–8. IEEE, 2015. 2

[2] Dena Bazazian and M Eulàlia Parés. Edc-net: Edge detection capsule network for 3d point clouds. *Applied Sciences*, 11(4):1833, 2021. 1, 2, 3

[3] Zhiqin Chen, Andrea Tagliasacchi, Thomas Funkhouser, and Hao Zhang. Neural dual contouring. *arXiv preprint arXiv:2202.01999*, 2022. 3, 4, 5

[4] Zhiqin Chen and Hao Zhang. Learning implicit fields for generative shape modeling. In *Proceedings of the IEEE Conference on Computer Vision and Pattern Recognition*, pages 5939–5948, 2019. 3

[5] Zhiqin Chen and Hao Zhang. Neural marching cubes. *ACM Transactions on Graphics (TOG)*, 40(6):1–15, 2021. 3

[6] Christopher B Choy, Danfei Xu, JunYoung Gwak, Kevin Chen, and Silvio Savarese. 3d-r2n2: A unified approach for single and multi-view 3d object reconstruction. In *European conference on computer vision*, pages 628–644. Springer, 2016. 3, 4

[7] Joel II Daniels, Linh K Ha, Tilo Ochotta, and Claudio T Silva. Robust smooth feature extraction from point clouds. In *IEEE International Conference on Shape Modeling and Applications 2007 (SMI'07)*, pages 123–136. IEEE, 2007. 2

[8] Kris Demarsin, Denis Vanderstraeten, Tim Volodine, and Dirk Roose. Detection of closed sharp feature lines in point clouds for reverse engineering applications. In *International Conference on Geometric Modeling and Processing*, pages 571–577. Springer, 2006. 2

[9] Kris Demarsin, Denis Vanderstraeten, Tim Volodine, and Dirk Roose. Detection of closed sharp edges in point clouds using normal estimation and graph theory. *Computer-Aided Design*, 39(4):276–283, 2007. 2

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

[11] Shachar Fleishman, Daniel Cohen-Or, and Cláudio T Silva. Robust moving least-squares fitting with sharp features. *ACM transactions on graphics (TOG)*, 24(3):544–552, 2005. 2

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

[13] Stefan Gumhold, Xinlong Wang, Rob S MacLeod, et al. Feature extraction from point clouds. In *IMR*, pages 293–305, 2001. 2

[14] Haoxiang Guo, Shilin Liu, Hao Pan, Yang Liu, Xin Tong, and Baining Guo. Complexgen: Cad reconstruction by b-rep chain complex generation. *ACM Transactions on Graphics (TOG)*, 41(4):1–18, 2022. 3

[15] Timo Hackel, Jan D Wegner, and Konrad Schindler. Contour detection in unstructured 3d point clouds. In *Proceedings of the IEEE conference on computer vision and pattern recognition*, pages 1610–1618, 2016. 2

[16] Chems-Eddine Himeur, Thibault Lejemble, Thomas Pellegrini, Mathias Paulin, Loic Barthe, and Nicolas Mellado. Pcednet: A lightweight neural network for fast and interactive edge detection in 3d point clouds. *ACM Transactions on Graphics (TOG)*, 41(1):1–21, 2021. 2

[17] Christoph M Hofmann. *Geometric and solid modeling: an introduction*. Morgan Kaufmann, 1989. 5

[18] Tao Ju, Frank Losasso, Scott Schaefer, and Joe Warren. Dual contouring of hermite data. In *Proceedings of the 29th annual conference on Computer graphics and interactive techniques*, pages 339–346, 2002. 3, 8

[19] Sebastian Koch, Albert Matveev, Zhongshi Jiang, Francis Williams, Alexey Artemov, Evgeny Burnaev, Marc Alexa, Denis Zorin, and Daniele Panozzo. Abc: A big cad model dataset for geometric deep learning. In *Proceedings of the IEEE/CVF Conference on Computer Vision and Pattern Recognition*, pages 9601–9611, 2019. 1, 2, 5

[20] Yiyi Liao, Simon Donne, and Andreas Geiger. Deep marching cubes: Learning explicit surface representations. In *Proceedings of the IEEE Conference on Computer Vision and Pattern Recognition*, pages 2916–2925, 2018. 3

[21] Yangbin Lin, Cheng Wang, Jun Cheng, Bili Chen, Fukai Jia, Zhonggui Chen, and Jonathan Li. Line segment extraction for large scale unorganized point clouds. *ISPRS Journal of Photogrammetry and Remote Sensing*, 102:172–183, 2015. 2

[22] Yujia Liu, Stefano D'Aronco, Konrad Schindler, and Jan D Wegner. Pc2wf: 3d wireframe reconstruction from raw point clouds. In *International Conference on Learning Representations (ICLR)*, 2021. 3

[23] William E Lorensen and Harvey E Cline. Marching cubes: A high resolution 3d surface construction algorithm. *ACM siggraph computer graphics*, 21(4):163–169, 1987. 3

[24] Albert Matveev, Alexey Artemov, Denis Zorin, and Evgeny Burnaev. 3d parametric wireframe extraction based on distance fields. In *2021 4th International Conference on Artificial Intelligence and Pattern Recognition*, pages 316–322, 2021. 3

[25] Albert Matveev, Ruslan Rakhimov, Alexey Artemov, Gleb Bobrovskikh, Vage Egiazarian, Emil Bogomolov, Daniele Panozzo, Denis Zorin, and Evgeny Burnaev. Def: Deep estimation of sharp geometric features in 3d shapes. *ACM Transactions on Graphics (TOG)*, 41(4):1–22, 2022. 1, 3, 5, 6

[26] Quentin Mérigot, Maks Ovsjanikov, and Leonidas J Guibas. Voronoi-based curvature and feature estimation from point clouds. *IEEE Transactions on Visualization and Computer Graphics*, 17(6):743–756, 2010. 1, 2, 5, 6, 7

[27] Lars Mescheder, Michael Oechsle, Michael Niemeyer, Sebastian Nowozin, and Andreas Geiger. Occupancy networks: Learning 3d reconstruction in function space. In *Proceedings of the IEEE Conference on Computer Vision and Pattern Recognition*, pages 4460–4470, 2019. 3

[28] Huan Ni, Xiangguo Lin, Xiaogang Ning, and Jixian Zhang. Edge detection and feature line tracing in 3d-point clouds by analyzing geometric properties of neighborhoods. *Remote Sensing*, 8(9):710, 2016. 2

[29] Jianhui Nie. Extracting feature lines from point clouds based on smooth shrink and iterative thinning. *Graphical Models*, 84:38–49, 2016. 2

[30] Jeong Joon Park, Peter Florence, Julian Straub, Richard Newcombe, and Steven Lovegrove. 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. 3

[31] Mark Pauly, Richard Keiser, and Markus Gross. Multi-scale feature extraction on point-sampled surfaces. In *Computer graphics forum*, volume 22, pages 281–289. Wiley Online Library, 2003. 2

[32] Charles Ruizhongtai Qi, Li Yi, Hao Su, and Leonidas J Guibas. Pointnet++: Deep hierarchical feature learning on point sets in a metric space. *Advances in neural information processing systems*, 30, 2017. 3, 4

[33] Ian Stroud. *Boundary representation modelling techniques*. Springer Science & Business Media, 2006. 5

[34] Xuefeng Tan, Dejun Zhang, Long Tian, Yiqi Wu, and Yilin Chen. Coarse-to-fine pipeline for 3d wireframe reconstruction from point cloud. *Computers & Graphics*, 106:288–298, 2022. 3

[35] Nanyang Wang, Yinda Zhang, Zhuwen Li, Yanwei Fu, Wei Liu, and Yu-Gang Jiang. Pixel2mesh: Generating 3d mesh models from single rgb images. In *Proceedings of the European Conference on Computer Vision (ECCV)*, pages 52–67, 2018. 3

[36] Xiaogang Wang, Yuelang Xu, Kai Xu, Andrea Tagliasacchi, Bin Zhou, Ali Mahdavi-Amiri, and Hao Zhang. Pie-net: Parametric inference of point cloud edges. *Advances in neural information processing systems*, 33:20167–20178, 2020. 1, 2, 3, 5, 6, 7

[37] Christopher Weber, Stefanie Hahmann, and Hans Hagen. Sharp feature detection in point clouds. In *2010 Shape Modeling International Conference*, pages 175–186. IEEE, 2010. 2

[38] Christopher Weber, Stefanie Hahmann, and Hans Hagen. Methods for feature detection in point clouds. In *Visualization of Large and Unstructured Data Sets-Applications in Geospatial Planning, Modeling and Engineering (IRTG 1131 Workshop)*. Schloss Dagstuhl-Leibniz-Zentrum fuer Informatik, 2011. 2

[39] Kevin J Weiler. *Topological structures for geometric modeling (Boundary representation, manifold, radial edge structure)*. Rensselaer Polytechnic Institute, 1986. 5

[40] Shaobo Xia and Ruisheng Wang. A fast edge extraction method for mobile lidar point clouds. *IEEE Geoscience and Remote Sensing Letters*, 14(8):1288–1292, 2017. 2

[41] Lequan Yu, Xianzhi Li, Chi-Wing Fu, Daniel Cohen-Or, and Pheng-Ann Heng. Ec-net: an edge-aware point set consolidation network. In *Proceedings of the European conference on computer vision (ECCV)*, pages 386–402, 2018. 1, 2, 3, 5, 6, 7