D-FRAME: Direction-Field-Based Wireframe Extraction for Complex CAD Models

Yuan Feng, Honghao Dai, Guangshun Wei, Long Ma, Pengfei Wang, Yuanfeng Zhou, and Ying He, *Member, IEEE*

***Abstract*—Extracting wireframes from CAD models represented by point cloud remains a significant challenge in computer graphics. This difficulty arises from two main factors: first, imperfections in the point cloud data, such as lack of orientation, noise, and sparsity; and second, the inherent complexity of geometric shapes, which often feature a high density of sharp edges in close proximity. In this paper, we propose D-FRAME, a multi-stage wireframe extraction framework that incorporates a novel direction field to improve edge detection quality and connectivity, a refinement strategy to address sparse or noisy edge points, and a final coarse-to-fine connection module to extract a robust wireframe. The direction field not only facilitates connectivity but also enhances the precision of extracted edges by mitigating the impact of misclassified points. By combining the Restricted Voronoi Diagram (RVD) with the extracted wireframes and the original point cloud, our approach also achieves highly faithful reconstruction of CAD model. Experiments conducted on synthetic and real-world scanned CAD datasets demonstrate that D-FRAME effectively manages noise, sparsity, and complex geometries, yielding high-fidelity wireframes.**

***Index Terms*—CAD modeling, direction fields, edge point classification, wireframe extraction.**

I. Introduction

THE extraction of wireframes from CAD-type point clouds is a fundamental and critical challenge in computer graphics and geometric modeling. Accurate wireframes are essential for effectively reconstructing CAD models with sharp features. However, achieving precise wireframe extraction from point clouds is hindered by two primary challenges: (1) defects in point cloud data, such as a lack of orientation, noise, and sparsity, and (2) the complexity of geometric shapes, which often feature a high density of sharp edges in close proximity.

Existing methods for extracting wireframes can be roughly divided into two categories: traditional geometric methods [1], [2], [3] and neural network-based methods [4], [5], [6], [7], [8], [9], [10]. Traditional geometric methods rely on the estimation of curvature or surface normal discontinuities to detect sharp features. Although effective for clean and dense data, these methods struggle with noise, sparsity, and complex geometries typically found in real-world CAD models. Neural network-based methods leverage data-driven approaches to improve adaptability and handle noisy datasets effectively. However, they often produce noisy edge points and incorrect connections due to the lack of explicit geometric constraints. This limitation becomes particularly evident in complex CAD models, especially in scenarios involving closely spaced sharp features. Additionally, existing edge point classification methods [3], [4], [5], [9] often suffer from misclassifications, resulting in inaccurate edge connectivity.

In response to the above challenges, we propose a globally consistent direction field guided approach for wireframe extraction. Specifically, we introduce a direction field as auxiliary information, which enhances edge detection accuracy and guides the connectivity of edge points for more precise wireframe extraction. To mitigate issues such as sparse or noisy local edge points, we develop a refinement strategy to smooth and complete the edge distribution. Finally, a post-processing step leverages the refined edge points and the direction field to generate robust and accurate wireframes.

Our proposed D-FRAME comprises a feature extraction module, followed by two branches for edge point and direction field prediction, a refinement module, and a subsequent wireframe extraction process. Given a CAD-type point cloud, we first employ a feature extraction module to extract latent features, which are then fed into two parallel branches to predict edge points and point-wise direction fields, respectively. Next, we analyze the local neighborhood of each edge point to identify error regions, such as noise or sparsity. The refinement module then generates smooth and dense edge points. Leveraging the direction field, we accurately extract clean wireframes, including various types (e.g., lines, curves, circles). To demonstrate the applicability of the extracted wireframes in downstream tasks, we integrate them into a CAD model reconstruction pipeline using the RVD approach, producing CAD models with faithfully preserved sharp features.

Received 11 April 2025; revised 10 August 2025; accepted 8 September 2025. Date of publication 12 September 2025; date of current version 10 November 2025. This work was supported in part by the National Natural Science Foundation of China under Grant 62172257, Grant 62572284, 62502273, and Grant 62402295, in part by the Natural Science Foundation of Shandong Province (Major Basic Research) under Grant ZR2024ZD12, Grant ZR2025QC1558, and Grant ZR2024QF087, in part by the Shandong Key Laboratory of Digital and Data Convergence Publishing Media, and in part by the Ministry of Education, Singapore, under its Academic Research Fund under Grant RT19/22. Recommended for acceptance by X. Guo. *(Corresponding authors: Guangshun Wei; Yuanfeng Zhou.)*

Yuan Feng, Honghao Dai, Guangshun Wei, Long Ma, Pengfei Wang, and Yuanfeng Zhou are with the School of Software, Shandong University, Jinan 250100, China (e-mail: yfeng@mail.sdu.edu.cn; dhhtang@163.com; guangshunwei@gmail.com; malong@sdu.edu.cn; pfwang@sdu.edu.cn; yfzhou@sdu.edu.cn).

Ying He is with the School of Computer Science and Engineering, Nanyang Technological University, Singapore 639815 (e-mail: yhe@ntu.edu.sg).

Code is available at https://github.com/yuanfeng-01/D-FRAME-test.

Digital Object Identifier 10.1109/TVCG.2025.3609350

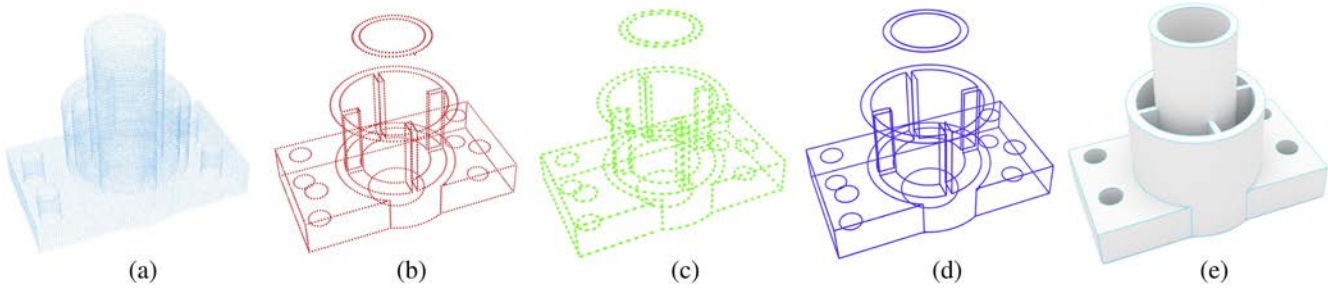


Fig. 1. Starting with an unoriented point cloud (a), D-FRAME involves two parallel branches: one for edge point classification (b) and the other for direction fields prediction (c). These outputs are combined to extract wireframes (d) through refinement and curve extraction. Finally, the Restricted Voronoi Diagram (RVD) and extracted wireframes are used to achieve a high-fidelity reconstruction (e) that preserves sharp features.

The main contributions of this work are summarized as follows:

- We propose a novel wireframe extraction framework for CAD models that accurately detects edge points and their connectivity.
- We introduce a novel direction field that not only improves edge detection accuracy but also provides crucial guidance for linking edge points. This dual functionality ensures robust connectivity in the wireframe, addressing the common issue of misclassified or abnormal edges.
- Based on edge points and direction fields, we propose a coarse-to-fine wireframe extraction algorithm, which is applied to CAD model reconstruction tasks to achieve accurate CAD models with preserved sharp features.

II. Related Works

This paper focuses on edge point classification and wireframe extraction for CAD-type point clouds. This section provides a comprehensive overview of research advancements in the related field, spanning traditional methodologies to modern learning-based approaches. Furthermore, given the prominence of surface reconstruction as a critical downstream task, we also provide a summary of the relevant methods in this research area.

A. Edge Points Classification for 3D Model

Edge points classification plays an essential role in numerous point cloud processing applications, such as point cloud consolidation [4], [11], [12] and shape reconstruction [3], [9]. Traditional methods for detecting edge features in point clouds primarily rely on local geometric characteristics, including eigenvalues [13], normals [14], curvatures [15], and other statistical metrics [16]. A prominent method in this category is the Voronoi Covariance Measure (VCM) [1], which leverages the Voronoi covariance framework and incorporates a Monte Carlo algorithm to compute feature boundaries, allowing efficient and robust extraction of sharp features.

With the rapid advancement of deep learning, particularly its extensive applications in point cloud processing [17], [18], [19], pioneering work EC-Net [4] employs a regression framework and an edge-aware loss to detect edge points. Following this work, recent approaches [6], [20], [21] have further advanced edge extraction by formulating it as a classification problem and leveraging neural networks for learning. Unlike traditional point-based strategies, NerVE [10] introduces a novel neural volumetric edge representation, transforming the point classification problem into an easier voxel classification task, addressing the limitations of previous point-wise methods. While learning-based methods have markedly improved the accuracy of edge point classification, most approaches still rely on traditional backbone networks [18], [19]. As a result, they often misclassify a substantial number of non-edge points as edge points, particularly in the presence of complex geometric details. Furthermore, these methods tend to exhibit considerable performance degradation when applied to noisy or sparse point clouds, thus constraining their effectiveness in real-world applications.

B. Wireframe Extraction

Wireframes play a critical role in CAD modeling, serving as essential tools for designing intricate 3D shapes with sharp and precise geometries. Despite their importance, extracting wireframes from point clouds poses significant challenges due to the diverse types of curves and complex details of CAD models. Traditional approaches such as [22] address this issue by detecting feature points and fitting them with splines to reconstruct wireframes. Subsequent methods have widely adopted this framework. PIE-Net [5] employs a PointNet++ [18]-based backbone to extract point cloud features, enabling the prediction of edge and corner points, followed by curve proposal selection to extract wireframes. Similarly, PC2WF [6] identifies and refines edge points based on extracted features and subsequently constructs wireframes from these points. ComplexGen [23] introduces a sparse CNN encoder coupled with a tri-path transformer decoder to produce geometric primitives and their relationships, achieving wireframe generation through global optimization under structural constraints. Although NerVE [10] adopts voxel-based predictions, it also classifies edge cubes and determines their connection directions to derive piece-wise linear (PWL) curves, which are subsequently assembled into wireframes using a graph search algorithm.

With the bloom of neural implicit representations, another category of wireframe extraction methods has emerged. These methods bypass explicit edge point classification by utilizing implicit representation to extract wireframes. DEF [7] is a representative work in this domain, which introduces the regression of

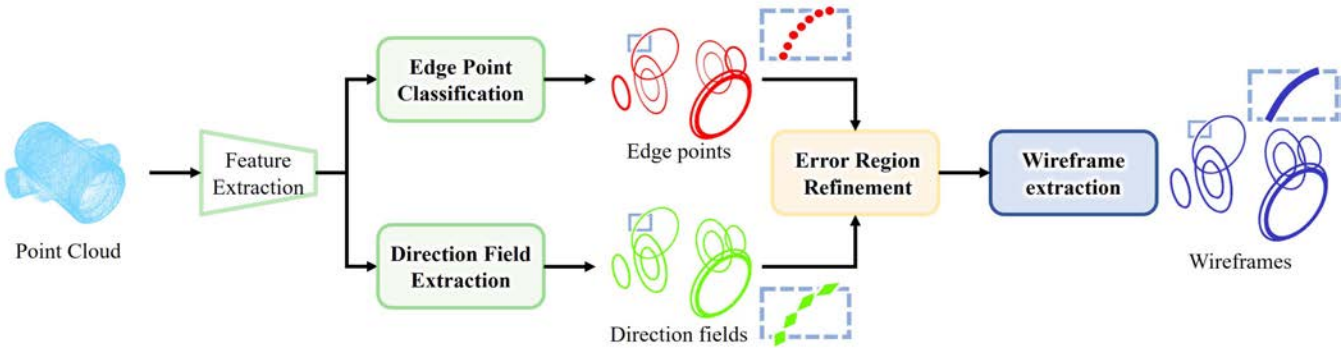


Fig. 2. **Overview of D-FRAME.** Given an input CAD-type point cloud, D-FRAME incorporates three core components for wireframe extraction. First, sharp features are effectively captured using a feature extraction module, followed by two branches for predicting edge points and direction fields (Section III-B1 and Section III-B2). To address noisy or sparse regions, an optional lightweight refinement module is introduced (Section III-C). Finally, based on the refined edge points and direction fields, we propose a novel algorithm for extracting piecewise-linear (PWL) curves and perform curve fitting to generate wireframes with accurate connections.

a scalar field within local patches representing the distance from an input point to the closest feature lines, enabling the extraction of wireframes from the continuous field. Similarly, NEF [8] proposes a Neural Edge Field to model the density distribution of 3D edges, eliminating the need for 3D ground truth edge supervision and relying solely on rendered 2D boundaries for optimization. Compared to edge point classification-based approaches, these methods effectively mitigate the influence of point cloud noise. However, they may exhibit inferior performance when extracting wireframes with intricate details. Our method seamlessly integrates the strengths of the two aforementioned categories of approaches, the superior capacity of our network and the introduced direction field enable us to achieve remarkable performance in edge point classification and wireframe extraction.

C. Surface Reconstruction of CAD Models

Surface reconstruction for CAD models is typically based on point cloud reconstruction techniques, a longstanding and extensively studied problem. Traditional implicit reconstruction methods [24], [25], [26] commonly employ Poisson Surface Reconstruction [27] (PSR) as a foundational approach. In contrast, learning-based methods have widely adopted unsupervised (fitting-based) approaches [28], [29], [30], which directly fit implicit surfaces from raw point clouds without relying on ground truth shape supervision. However, it is important to highlight that these methods are not specifically designed to address the unique challenges of surface reconstruction in CAD models.

To tackle the challenges, one category of methods [9], [31], [32], [33] identifies and segments multiple primitive types (e.g., planes, cones, cylinders, spheres) for individual fitting. Unfortunately, these methods often fail when faced with substantial sparse regions, particularly in areas involving multiple neighboring primitives. In contrast, RFEPS [3] introduces a multi-stage algorithm aimed at reconstructing noisy CAD-type point clouds, effectively addressing the inability of traditional point cloud reconstruction methods to preserve sharp features. This is achieved by enhancing edge regions and employing surface reconstruction based on Restricted Power Diagram [34] (RPD). NeurCADRecon [35] advances the field by incorporating a developability term into its loss function, encouraging Gaussian curvature toward zero while maintaining fidelity to the input point cloud. Additionally, it employs a dynamic sampling strategy to manage sparse or incomplete point clouds. Despite utilizing a double-trough curve to handle tip points and achieving overall fidelity in CAD shape reconstruction, the reconstructed sharp features by NeurCADRecon still show gaps from the ground truth. In contrast, our approach requires only an unoriented point cloud while achieving superior performance in reconstructing CAD models with accurately aligned sharp features. Moreover, D-FRAME demonstrates robust reconstruction performance even for scanned point clouds.

III. Method

A. Overview

As illustrated in Fig. 2, we propose D-FRAME, a multi-stage framework for robust wireframe extraction. This section details its three core components: edge point and direction field prediction, error region refinement, and wireframe extraction. First, we employ a backbone based on the Point Transformer V3 (PTv3) [36] to effectively extract latent features from the input point cloud. Two parallel branches are subsequently utilized to predict edge points and direction fields. To enhance the quality of the extracted edge points, we introduce an error region refinement module to filter out noisy and sparse points. Leveraging the direction field, we propose a novel piecewise-linear (PWL) curve extraction and fitting-based post-process to ensure accurate and efficient edge point connectivity. Finally, we incorporate the extracted wireframes into a CAD model reconstruction task using the Restricted Voronoi Diagram (RVD), achieving high quality as well as sharp feature preservation.

B. Edge Point and Direction Field Prediction

1) Edge Point Classification: Given an unoriented point cloud denoted as $P = \{\mathbf{p}_i \in \mathbb{R}^3 \mid i = 1, 2, \ldots, N\}$, where N represents the total number of points, our objective is to predict $\mathcal{Y}_p = \{y_i \in [0, 1] \mid i = 1, 2, \ldots, N\}$ for each point p_i, indicating its probability of belonging to an edge point, using an edge point classification network. This process identifies the subset

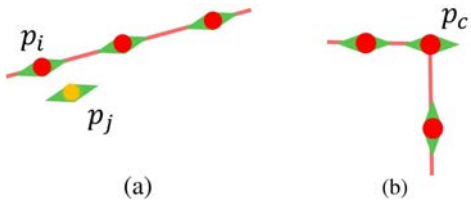


Fig. 3. Ground truth wireframes are shown as light red lines, with the direction field represented by green arrows. (a) At an edge point p_i, the direction field points to its nearest neighbor, while at a non-edge point p_j, it aligns with the nearest wireframe. (b) At a corner point p_c, the direction field also points to its nearest neighbor for simplicity.

of edge points $P_e \in \mathbb{R}^{N_e \times 3}$, where N_e denotes the number of edge points in the point cloud. In D-FRAME, we use the PTv3 as the backbone for accurate and efficient edge point prediction. It integrates both local and global information for sharp feature detection. After feature extraction, a simple multi-layer perceptron (MLP) is used as the decoder, and the final per-point class probabilities are predicted through a softmax function. The overall process is shown in (1).

$$\begin{aligned} \mathcal{F}_p &= PT\left(P\right), \\ \bar{\mathcal{Y}}_p &= Softmax\left(MLP_c\left(\mathcal{F}_p\right)\right), \end{aligned} \tag{1}$$

where PT denotes the PTv3-based backbone, and MLP_c is the classification head. The output $\bar{\mathcal{Y}}_p \in \mathbb{R}^{N \times 2}$ denotes per-point class probabilities, from which $\mathcal{Y}_p$ is derived naturally (i.e., the second column of $\bar{\mathcal{Y}}_p$).

2) Direction Field Prediction: The cross field and its variants are widely used in geometry processing, and their capability to preserve sharp features has been demonstrated in previous studies [37], [38], [39]. Inspired by them, we define a direction field which can be easily learned and integrate it with edge point classification to achieve more precise connections. As illustrated in Fig. 3, the direction field, denoted as $\mathcal{D}_p = \{d_p^{(i)} \in \mathbb{R}^3 \mid i = 1, 2, \ldots, N\}$ at edge points is defined as the vector pointing toward its nearest neighbor. For non-edge points, the direction field aligns with the direction of the nearest wireframe. It is important to note that the sign of the directional vectors is not considered, meaning that $d_p^{(i)}$ and $-d_p^{(i)}$ are regarded as equivalent. Additionally, the direction field at corner points is defined in the same manner as at regular points. Further discussion of the direction field is left in Section VI. To predict the direction field, we also use MLP as an independent decoder to acquire the direction field $\mathcal{D}_p$. The entire process is defined as follows:

$$\mathcal{D}_p = MLP_d\left(\mathcal{F}_p\right), \tag{2}$$

where MLP_d is the MLP head for predicting the direction field.

3) Loss Function: Since the number of edge points in the input point cloud is significantly less than that of non-edge points, addressing the class imbalance during network training is crucial. To tackle this issue, we introduce Focal Loss [40] as the loss function for training the edge point classification module, defined as:

$$\mathcal{L}_c = -\alpha(1-p_t)^{\gamma}\log(p_t), \tag{3}$$

$$p_t = \begin{cases} \mathcal{Y}_p & \mathcal{Y}_{gt} = 1, \\ 1 - \mathcal{Y}_p & else, \end{cases}$$

where $\mathcal{Y}_p$ represents the predicted probability, $\mathcal{Y}_{gt}$ is the ground truth class label, α and γ serve as moderators to control the balance between positive and negative samples, as well as the difficulty of samples.

For direction field prediction, since the directional vectors are required only at the identified edge points for wireframe extraction, we assign higher weights to these points based on the probabilities generated by the classification network. Similarly, the direction field is supervised using the corresponding ground truth, with a cosine similarity-based loss function to avoid the adverse effects of direction flipping. The loss function can be formulated as follows:

$$\mathcal{L}_d = \frac{1}{N}\sum_{i=1}^{N} abs\left(1 - \frac{< d_p^{(i)}, d_{gt}^{(i)} >}{||d_p^{(i)}|| \cdot ||d_{gt}^{(i)}||}\right) \otimes \mathcal{Y}_p(i), \tag{4}$$

where $\mathcal{Y}_p(i)$ serves as the weight mentioned above. $d_p^{(i)}$ and $d_{gt}^{(i)}$ denote the predicted and ground truth direction field at point p_i respectively; $< \cdot, \cdot >$ represents the dot product of two vectors; $||\cdot||$ denotes the vector magnitude and $\otimes$ represents element-wise product operation. Furthermore, to account for the local continuity of the direction field, a smoothness loss is introduced as a regularization term, promoting smoother predictions of the direction field by the network, defined as:

$$\mathcal{L}_s = \frac{1}{N}\sum_{i=1}^{N}\sum_{j=1}^{K} abs\left(1 - \frac{< d_p^{(i)}, d_p^{(j)} >}{||d_p^{(i)}|| \cdot ||d_p^{(j)}||}\right), \tag{5}$$

where K is the number of nearest neighbors to be considered. Finally, our overall loss function can be summarized as follows:

$$\mathcal{L} = w_1\mathcal{L}_c + w_2\mathcal{L}_d + w_3\mathcal{L}_s, \tag{6}$$

where w_1, w_2 and w_3 are weights used to balance the contributions of the respective loss terms.

C. Error Region Refinement

Due to the inevitable presence of misclassified or missed key points in the edge point classification network, particularly when handling noisy or sparse point clouds, directly using these results for wireframe extraction easily leads to incorrect connections. To address these error regions, we propose an optional lightweight point refinement module that leverages the direction field to refine detected edge points, yielding more regularized and accurate results.

Noise: Given the edge points $P_e \in \mathbb{R}^{N_e \times 3}$ and the corresponding direction field $\mathcal{D}_e \in \mathbb{R}^{N_e \times 3}$, we first detect the presence of noisy points. Specifically, for each edge point $p_i \in P_e$, we compute the value e_{ik} defined in (7),

$$e_{ik} = abs\left(\frac{< d_e^{(i)}, \overrightarrow{p_i p_{ik}} >}{||d_e^{(i)}|| \cdot ||\overrightarrow{p_i p_{ik}}||}\right), \tag{7}$$

where $p_{ik}(k \in \{0, 1\})$ is one of the two nearest neighbors of p_i, $d_e^{(i)}$ denote the direction field at p_i. If all of the cosine values $e_{ik}(k \in \{0, 1\})$ are below a given threshold e, we classify p_i as a

Fig. 4. Illustration of error regions detection. The red points represent regular edge points, with their direction fields indicated by green arrows. (a) The orange point is identified as a noisy point. (b) The point p_i is considered to be near a sparse region.

noisy point, as shown in Fig. 4(a). Note that for corner points, it is almost unlikely that both e_{i0} and e_{i1} fall below the threshold e, which helps prevent misclassification of corners as noise while enabling efficient noise detection.

Sparsity: To detect sparse regions of the edges, we construct a cone (as shown in Fig. 4(b)) with a radius r and a height h along the direction $\pm d_e^{(i)}$ for each edge point $p_i \in P_e$. If no neighboring points are found within any of the cones, the corresponding point is classified as belonging to a sparse region. Since sparsity mainly occurs in the middle of wireframes, and occasional sparse corner points rarely affect connectivity, we also construct a cone at corner points. For detected points, we insert additional edge points $P_{ins} \in \mathbb{R}^{N_{ins}\times 3}$ along the direction of the direction field within a specified range to enhance the completeness of the edge representation in sparse regions.

Refinement: To address detected erroneous regions in the edge points, we design a refinement module R to filter out these issues. Specifically, after feature extraction, we employ a Node Shuffle operation [41] to generate offsets for each point, resulting in refined edge points $P_r \in \mathbb{R}^{N_r\times 3}$, where $N_r = N_e + N_{ins}$. To ensure that points in non-noisy or non-sparse regions remain unaffected, as even small offsets could distort clean edges, the refinement process is applied only to noisy points and inserted points. This is guided by a mask M obtained through noisy and sparse points detection. The whole process is defined in (8):

$$P_r = concat\left(P_e, P_{ins}\right) + R\left(cocat(P_e, P_{ins}), \mathcal{D}_r\right) \otimes M, \tag{8}$$

where $\mathcal{D}_r$ represents the direction fields of all N_r points, with the direction fields of the inserted points set to those of the detected sparse points.

D. Wireframe Extraction

1) PWL Curve Extraction: Given the predicted (or refined) edge points P_r and their corresponding direction fields D_r, we integrate the predicted direction fields with traditional graph-based methods to generate the connection set $\mathcal{E}$ that forms the PWL curves.

Step 1. Coarse Connection: Given the edge points and direction fields, we can directly obtain the coarse connection through simple operations, as described in Algorithm 1. Specifically, we calculate the distances between points, and any point with a distance less than $dist_1$ is considered part of the current point's neighborhood $N(P_i)$. Then we calculate the cosine similarity θ_{ik} for each neighboring point $p_k \in N(p_i)$, and the two closest points that satisfy $\theta_{ik} > \theta_1$ are selected. If the edge $(p_i, p_k) \notin \mathcal{E}$, it will be added to the edge set $\mathcal{E}$ as a new connection.

Algorithm 1: PWL Curve Connection Algorithm.

Input:
$P_r \in \mathbb{R}^{N_r\times 3}$ and $\mathcal{D}_r \in \mathbb{R}^{N_r\times 3}$.
θ_1, θ_2: Cosine similarity thresholds.
$dist_1$, $dist_2$: Distance thresholds.
Output:
Connections $\mathcal{E} = \{(p_i, p_j) \mid p_i, p_j \in P_r\}$
▷ Step 1: Coarse Connection
for *each point* $p_i \in P_r$ **do**
  Compute $dist(p_i, p_j)$ between p_i and $p_j \in P_r$.
  Select valid neighbors:
  $N(p_i) = \{p_j \mid p_j \in P_r \text{ and } dist(p_i, p_j) \le dist_1\}$.
  for *each point* $p_k \in N(p_i)$ **do**
    Compute $\theta_{ik} = abs(\frac{\langle d_r^{(i)}, \overrightarrow{p_i p_k}\rangle}{||d_r^{(i)}||\cdot||\overrightarrow{p_i p_k}||})$.
  end
  Select top 2 nearest points in $N(p_i)$ with $\theta_{ik} \ge \theta_1$.
  Add (p_i, p_k) to $\mathcal{E}$ if it does not already exist.
end
▷ Step 2: Refinement of Connections
for *each point* $p_i \in P_r$ **do**
  Compute the adjacency matrix A based on $\mathcal{E}$.
  if $Degree(p_i) > 2$ **then**
    for *each pair of neighboring edges* (p_j, p_k) **do**
      Compute $\theta_{ijk} = abs(\frac{\langle \overrightarrow{p_i p_j}, \overrightarrow{p_i p_k}\rangle}{||\overrightarrow{p_i p_j}||\cdot||\overrightarrow{p_i p_k}||})$,
      $\theta_j = abs(\frac{\langle \overrightarrow{p_j p_i}, \overrightarrow{p_j p_k}\rangle}{||\overrightarrow{p_j p_i}||\cdot||\overrightarrow{p_j p_k}||})$ and
      $\theta_k = abs(\frac{\langle \overrightarrow{p_k p_i}, \overrightarrow{p_k p_j}\rangle}{||\overrightarrow{p_k p_i}||\cdot||\overrightarrow{p_k p_j}||})$
      Remove connection with $max(\theta_j, \theta_k)$ if $\theta_{ijk} > \theta_2$.
    end
  end
  else if $Degree(p_i) = 1$ **then**
    Connect to nearest point p_n with
    $dist(p_i, p_n) < dist_2$.
    Add (p_i, p_n) to $\mathcal{E}$ if it does not already exist.
  end
end
Resolve multi-path connections. ▷ Step 3
Remove isolated points. ▷ Step 4

Step 2. Refinement of Connections: Based on P_r and $\mathcal{E}$, we construct an adjacency matrix A. Through observation, we identify several types of abnormal edges or points, including the following situations: (*Case 1*) invalid connections caused by noisy points, (*Case 2*) broken connections at corner points, (*Case 3*) multi-path connections, and (*Case 4*) isolated points. The specific handling procedures are as follows:

- *Case 1:* As illustrated in Fig. 5(a), noise near sharp features often leads to incorrect edge connections. To mitigate this issue, our algorithm first identifies all points with $degree > 2$. For each candidate pair (p_j, p_k) connected to p_i, we compute the absolute cosine similarity between $\overrightarrow{p_i p_j}$ and $\overrightarrow{p_i p_k}$. If it is larger than θ_2, we then calculate the value θ_j and θ_k, removing either: (1) the point with $degree = 1$, or

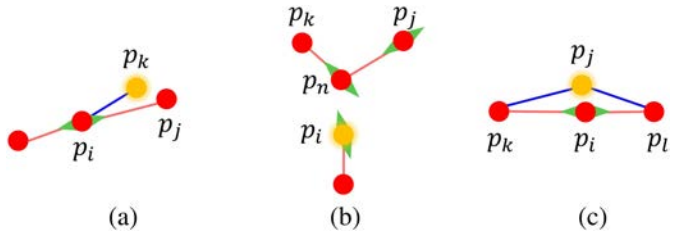


Fig. 5. Visualization of *Case 1* to *Case 3*. The red and orange points represent regular edge points and abnormal edge points, respectively. The light red lines and blue lines denote standard connections and abnormal connections, respectively. Green arrows indicate the direction field at key points.

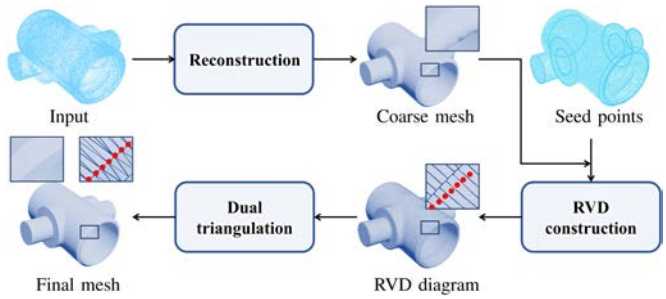


Fig. 6. CAD model reconstruction based on wireframes and RVD. The red points represent edge points. By combining the reconstructed mesh with seed points obtained through adaptive edge point densification, D-FRAME generates high-quality CAD meshes that accurately preserve sharp features.

(2) the point forming the larger angle $max(\theta_j, \theta_k)$, along with all its associated edges.

- *Case 2:* As shown in Fig. 5(b), at corner points, variations in the direction field may cause neighboring points to have a degree of 1. We then select the nearest point p_n in the direction of p_i's direction field, ensuring that the distance is less than $dist_2$, and establish a direct connection.
- *Case 3:* For multi-path connection, as shown in Fig. 5(c), we compare the angle ($\overrightarrow{p_kp_i}$, $\overrightarrow{p_ip_l}$) and angle ($\overrightarrow{p_kp_j}$, $\overrightarrow{p_jp_l}$), and then remove the point with the smaller angle and its corresponding connection to keep the smoothness of the curves.
- *Case 4:* For isolated points (i.e., points with a degree of 0) and edges shorter than a predefined length (e.g., 5), we simply remove them.

2) Wireframe Fitting: Based on the extracted PWL curves, our goal is to obtain the final wireframes, including specific curve types (e.g., lines, circles, or splines). Following NerVE [10], edge points with a degree larger than 2 are selected as endpoints. For open curves, we start from one endpoint and search along the PWL curve until reaching another endpoint. Subsequently, we fit the two endpoints along with all edge points on the path. For the untraversed points, we can also easily detect and fit closed circles.

E. CAD Model Reconstruction

Our proposed D-FRAME not only efficiently extracts wireframes from CAD-type point clouds but also effectively tackles the challenging task of preserving sharp features during CAD model reconstruction. Specifically, given an unoriented input point cloud P, we first apply a point cloud reconstruction algorithm (e.g., Poisson reconstruction) to generate a coarse mesh. Inspired by previous studies [42], [43], [44], we enhance edge features using the RVD. However, sparse edge points can lead to the generation of triangles that cross sharp features, as noted in RFEPS [3]. To mitigate this issue, we adopt an interpolation-based strategy typically used in geometric modeling methods, such as [45], to generate dense edge points by sampling along the wireframes. These points are integrated with the input point cloud to serve as seed points for subsequent processing. As demonstrated in the Delaunay mesh study by [46], sufficiently dense sampling of edge points results in adjacent Voronoi cell formations among these points, thereby preventing the dual triangulation from traversing sharp geometric features. While RPD used in RFEPS allows edge points to exert greater influence via weighting, we achieve a similar effect in RVD by increasing the density of edge points among the seeds. Besides, RVD eliminates the need to manage weights, resulting in lower computational cost and improved stability compared to RPD. By combining the coarse mesh from the reconstruction process with the generated seed points, we construct the RVD and perform dual triangulation, ultimately producing a high-quality CAD model mesh that faithfully preserves sharp features, as shown in Fig. 6.

IV. Experiments

In this section, we first introduce the datasets used in our experiments and provide a detailed description of the evaluation metrics. We then conduct a comprehensive evaluation of our model on three core tasks: edge point classification, wireframe extraction, and CAD model reconstruction. Furthermore, we demonstrate the robustness of D-FRAME in processing noisy and sparse point clouds, as well as its effectiveness in real-world scanned data applications.

A. Datasets

The ABC dataset [47], a widely used CAD dataset containing more than a million synthetic models, has served as the foundation for numerous previous studies. To ensure fair and consistent comparisons, we adopt the dataset created by NerVE [10], which includes 2,364 models filtered from ABC. Leveraging the ground truth edges provided by this dataset, we generate edge point labels and compute the ground truth direction field. For dataset partitioning, we adhere strictly to NerVE's predefined configuration, employing an 80%–20% split for training and testing, respectively. For wireframe extraction, we utilize the datasets provided by DEF [7] and NEF [8], which are also derived from ABC. Additionally, to evaluate CAD model reconstruction performance, we perform experiments not only on these synthetic datasets but also on real-world scanned datasets.

B. Implementation Details

Network training: We train our networks on 4 NVIDIA RTX 4090 GPUs with a total batch size of 8 for 800 epochs. The

AdamW optimizer with a weight decay of 0.05 is used for optimization. An one-cycle learning rate scheduler is employed with an initial learning rate of 6e-4, a maximum learning rate of 6e-3, and an increasing percentage of 0.05. The hyperparameters α and γ in the focal loss are set to 0.8 and 2, respectively. The number of nearest neighbors K in smooth loss is set to 5 for computational efficiency, while the loss weights w_1, w_2 and w_3 are set to 10, 1 and 0.5 respectively. For noisy points detection, the cosine similarity threshold e is set to 0.8. In sparse region identification, the radius r and height h of cones are set to 0.01 and 0.05, respectively, while the distance constraint for point insertion is set to 0.05. For the backbone architecture, we follow the default configuration of PTv3, setting the output feature channels to 64. Additionally, we adjust the grid size to 0.002, as our point cloud contains a larger number of points (typically 10K$\sim$ 30K) compared to those used for classification or segmentation. As for data augmentation, we apply slight random jittering to the input points to mitigate overfitting to the synthetic data.

Wireframe extraction: The cosine similarity thresholds θ_1 and θ_2 in Algorithm 1 are set to 0.9 and 0.98 respectively. The distance thresholds $dist_1$ and $dist_2$ are both set to a large value 0.05, as they are both used to filter out neighbors with excessively large distances.

CAD model reconstruction: For dense edge generation in Section III-E, up to five intermediate points are adaptively inserted between endpoints based on edge length. Additionally, to prevent the generated triangular mesh from including connections between non-edge points that cross sharp features, we remove non-edge points located within a distance of $dist_3$ from any edge point, where $dist_3$ is defined as the maximum gap between adjacent edge points.

C. Metrics

Following previous works [7], [8], [10], we quantitatively evaluate the accuracy of edge points and the wireframes using a set of metrics, including Chamfer Distance (CD) [48], Hausdorff Distance (HD) [49] (only for edge points), Intersection over Union (IoU), Precision, Recall, and F-score (only for wireframes). Consistent with the evaluation protocol of NEF [8], we consider a point to be matched if at least one ground truth point has an L2 distance smaller than 0.02 from it when evaluating IoU, Precision, Recall, and F-score.

To assess the overall reconstruction quality, we compute the ℓ^1 version of the CD between the vertices of the reconstructed meshes and the ground truth meshes. Additionally, to evaluate the preservation of sharp features, we calculate the dihedral angle difference (DAD) between the reconstructed and ground truth meshes in sharp regions, focusing on vertices with a degree of 2 for simplicity.

D. Comparisons

1) Edge Point Prediction: We evaluate our predicted edge points against several state-of-the-art methods, including VCM [1], EC-NET [4], PIE-NET [5], RFEPS [3], SED-NET [9], and NerVE [10]. Quantitative results are reported in Table I and visual comparisons are shown in Fig. 7. Both the numerical and qualitative results demonstrate that our method significantly outperforms the baseline approaches, producing visually accurate edge points. EC-NET and SED-NET tend to generate redundant points around the ground-truth edges, resulting in higher CD and HD values. PIE-NET and RFEPS produces a thinner output region but often result in sparse edge point distributions.

TABLE I
QUANTITATIVE RESULTS ON EDGE POINTS CLASSIFICATION. OURS$^-$ REPRESENTS THE RESULT WITHOUT THE ERROR REGION REFINEMENT MODULE, WHICH IS APPLIED AFTER THE EDGE POINT PREDICTION.

Methods	CD↓	HD↓	Precision↑	Recall↑	IoU↑
VCM	0.0226	0.1941	-	-	-
EC-NET	0.0037	0.1284	0.769	0.751	0.455
PIE-NET	0.0074	0.1318	0.687	0.864	0.606
RFEPS	0.0028	0.0659	0.921	0.916	0.869
SED-NET	0.0037	0.1063	0.648	0.907	0.611
NerVE	0.0022	0.0714	0.965	0.930	0.871
Ours$^-$	**0.0010**	**0.0370**	**0.989**	**0.985**	**0.968**

TABLE II
QUANTITATIVE RESULTS ON WIREFRAME EXTRACTION. OURS$^-$ REPRESENTS THE RESULT WITHOUT THE ERROR REGION REFINEMENT MODULE.

Methods	CD↓	Precision↑	Recall↑	F-score↑	IoU↑
PIE-NET	0.0708	0.9072	0.7204	0.7846	0.6709
PC2WF	0.1382	0.9043	0.5525	0.6348	0.5074
RFEPS	0.0165	0.9031	0.8544	0.9272	0.9173
DEF	0.0402	0.8343	0.7802	0.8099	0.7368
NEF	0.0353	0.9387	0.8838	0.9044	0.8283
NerVE	0.0084	0.9837	0.9941	0.9833	0.9764
Ours$^-$	0.0021	0.9903	0.9960	0.9896	0.9825
Ours	**0.0017**	**0.9937**	**0.9962**	**0.9943**	**0.9897**

Although NerVE outperforms earlier methods, it often produces outliers or sparse regions when handling complex CAD geometries due to the absence of global geometric constraints in sharp feature extraction. Furthermore, it may fail to detect edge points in cases where two curves are in close proximity. In contrast, D-FRAME demonstrates superior performance in edge point prediction, producing accurate and uniformly distributed edge points even in challenging scenarios such as closely spaced circles, narrow slits, or intricate curves.

2) Wireframe Extraction: We compare the wireframes extracted by our method with those produced by state-of-the-art approaches [3], [5], [6], [7], [8], [10] on the ABC-NEF dataset introduced by NEF [8]. The quantitative results in Table II demonstrate that our method consistently outperforms all competing methods across all evaluation metrics. PIE-Net [5] often struggles to accurately detect edge points and frequently misses critical edge points. PC2WF [6], while effective for linear structure reconstruction, is fundamentally limited to straight-line detection and demonstrates poor performance on other types of curve. Thus, we only include PIE-NET and PC2WF in the quantitative comparison.

As shown in Fig. 8, while RFEPS, NEF, and NerVE perform well on simple CAD models, they struggle with complex geometries, often resulting in gaps in intricate features and erroneous

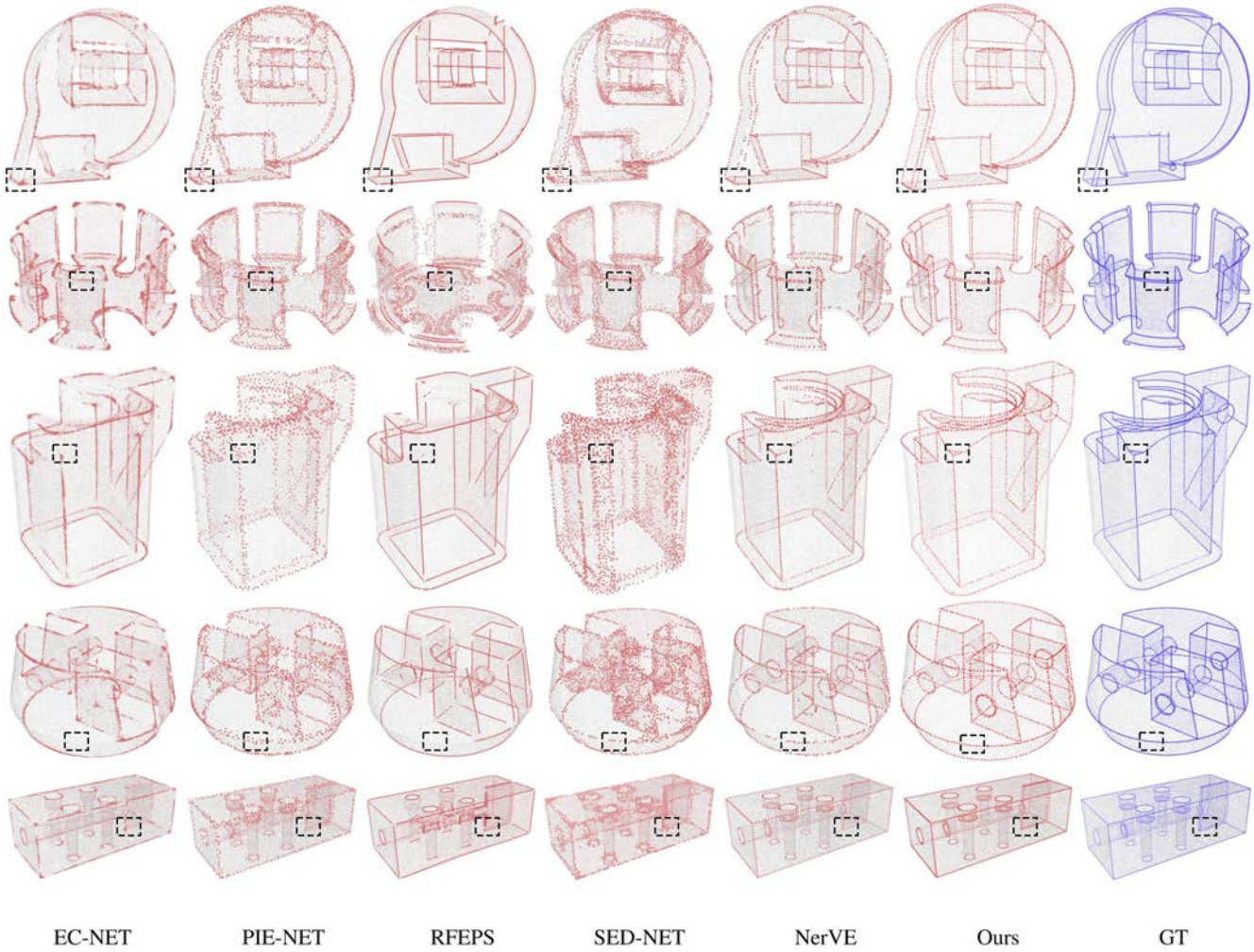


Fig. 7. Visual comparisons on edge point prediction.

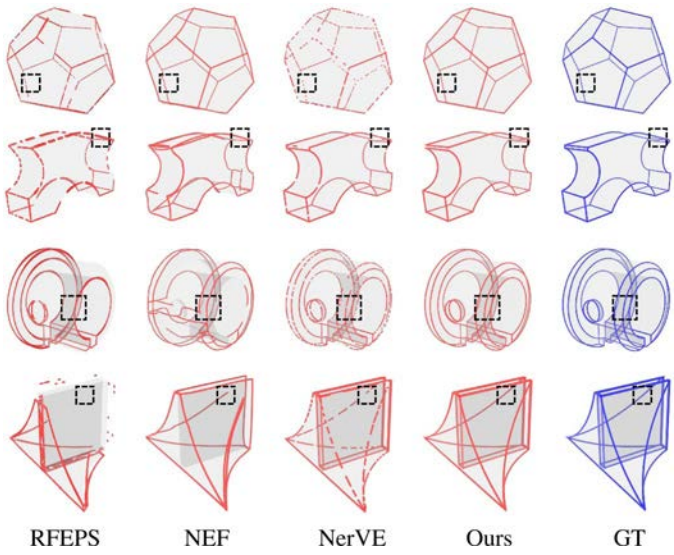


Fig. 8. Visual comparisons on wireframe extraction on the ABC-NEF dataset.

connections in closely spaced circles. We further perform additional qualitative comparisons with DEF [7] on the DEF-Sim dataset, which consists of 68 carefully selected shapes from the ABC dataset. As shown in Fig. 9, DEF tends to approximate curves as straight lines particularly when its endpoint detection produces sparse results.

Different from previous approaches, D-FRAME employs a coarse-to-fine connection and leverages direction fields to ensure accurate connections. This enables D-FRAME to handle complex curves and sharp corners robustly, producing visually convincing and geometrically accurate wireframes, even for challenging models from the datasets provided by NerVE, as illustrated in Fig. 10.

3) CAD Model Reconstruction: As a critical downstream task, we compare the reconstruction performance of D-FRAME with state-of-the-art CAD model reconstruction methods [3], [35]. Specifically, RFEPS employs an explicit reconstruction approach, while NeurCADRecon utilizes a self-supervised implicit reconstruction strategy. For a fair comparison, we use the raw input data provided by NerVE. Fig. 11 presents reconstruction results from D-FRAME and other methods across various

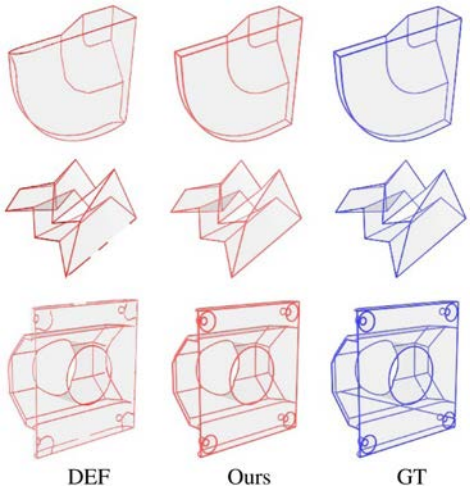


Fig. 9. Visual comparisons on wireframe extraction on the DEF-Sim dataset.

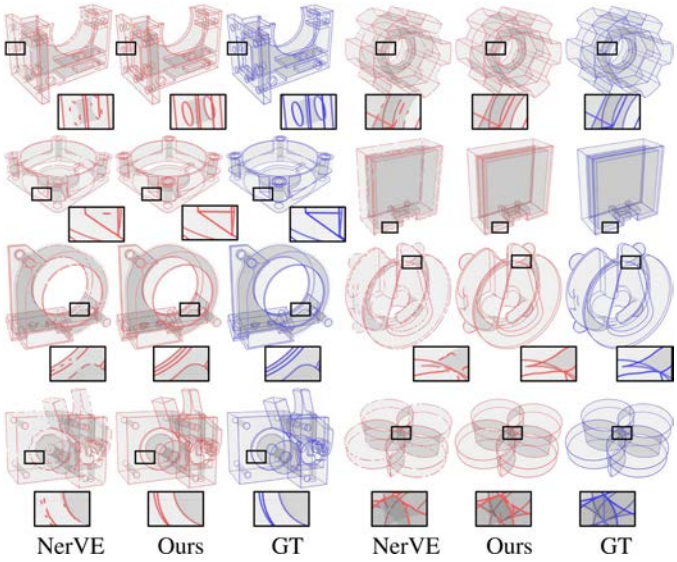


Fig. 10. Visual comparisons with NerVE on wireframe extraction for complex models. The presented examples include various types of wireframes, regions with high density of sharp edges, and highly intricate wireframe intersections.

models, along with the CD and DAD metrics. Lower CD values indicate higher reconstruction quality, while lower DAD values reflect better sharp feature preservation in CAD models.

As shown in Fig. 11, RFEPS struggles with the reconstruction of sparse point clouds and tends to produce artifacts in CAD model regions where the dihedral angle is either less than $\pi/3$ or too close to π (see the first row). NeurCADRecon prioritizes smoothness by minimizing the overall absolute Gaussian curvature, but its reconstructed sharp features are less pronounced compared to RFEPS. The qualitative and quantitative results demonstrate that our approach addresses these challenges and achieves high-quality, smooth surfaces comparable to NeurCADRecon, while also preserving sharp edge features as RFEPS. Additionally, Fig. 12 illustrates the predicted direction

TABLE III
QUANTITATIVE RESULTS ON EDGE POINTS CLASSIFICATION BASED ON VARIOUS NOISE OR SAMPLED POINT NUMBERS. OURS$^-$ REPRESENTS THE RESULT WITHOUT THE ERROR REGION REFINEMENT MODULE.

Type	Level	Methods	CD↓	HD↓	Precision↑	Recall↑	IoU↑
Non-uniform	90%	RFEPS	0.0026	0.0667	0.918	0.907	0.864
		NerVE	0.0024	0.0807	0.962	0.907	0.827
		Ours$^-$	**0.0011**	0.0528	**0.987**	0.948	0.908
		Ours	**0.0011**	**0.0476**	0.986	**0.953**	**0.938**
	80%	RFEPS	0.0031	0.0675	0.911	0.889	0.856
		NerVE	0.0024	0.0955	0.961	0.875	0.771
		Ours$^-$	**0.0012**	0.0639	0.981	0.903	0.833
		Ours	**0.0012**	**0.0533**	**0.983**	**0.933**	**0.879**
Noise	0.4%	RFEPS	0.0033	0.0781	0.903	0.889	0.846
		NerVE	0.0080	0.2390	0.869	0.846	0.666
		Ours$^-$	0.0021	0.0851	0.968	0.951	0.906
		Ours	**0.0015**	**0.0747**	**0.973**	**0.961**	**0.922**
	0.8%	RFEPS	0.0047	0.142	0.879	0.867	0.831
		NerVE	0.0119	0.2439	0.747	0.757	0.507
		Ours$^-$	0.0034	0.1200	0.940	0.911	0.845
		Ours	**0.0026**	**0.0933**	**0.957**	**0.926**	**0.851**

TABLE IV
QUANTITATIVE RESULTS ON WIREFRAME EXTRACTION BASED ON VARIOUS NOISE OR SAMPLED POINT NUMBERS. OURS$^-$ REPRESENTS THE RESULT WITHOUT THE ERROR REGION REFINEMENT MODULE.

Type	Level	Methods	CD↓	Precision↑	Recall↑	F-score↑	IoU↑
Non-uniform	90%	RFEPS	0.0081	0.9418	0.9297	0.9564	0.9585
		NerVE	0.0117	0.9771	0.9563	0.9743	0.9374
		Ours$^-$	0.0032	0.9914	0.9941	0.9936	0.9876
		Ours	**0.0024**	**0.9924**	**0.9949**	**0.9938**	**0.9880**
	80%	RFEPS	0.0096	0.9366	0.9213	0.9547	0.9519
		NerVE	0.0125	0.9763	0.9477	0.9691	0.9248
		Ours$^-$	0.0048	0.9922	0.9865	0.9902	0.9784
		Ours	**0.0034**	**0.9916**	**0.9873**	**0.9927**	**0.9806**
Noise	0.4%	RFEPS	0.0076	0.9416	0.9288	0.9613	0.9565
		NerVE	0.0319	0.8669	0.9257	0.8912	0.7935
		Ours$^-$	0.0037	0.9863	0.9755	0.9842	0.9709
		Ours	**0.0031**	**0.9874**	**0.9819**	**0.9871**	**0.9803**
	0.8%	RFEPS	0.0214	0.9305	0.9291	0.9527	0.9214
		NerVE	0.0445	0.7457	0.8993	0.8098	0.6758
		Ours$^-$	0.0106	0.9714	0.9426	0.9634	0.9344
		Ours	**0.0089**	**0.9755**	**0.9507**	**0.9718**	**0.9413**

fields and wireframes from D-FRAME, further highlighting its capability in accurately capturing sharp features.

4) Robustness Tests: To evaluate the robustness of D-FRAME, we compare its performance against RFEPS, which is known for strong robustness to noise, and NerVE, a state-of-the-art method for wireframe extraction. The comparison is conducted under varying noise levels and point cloud densities. It is important to note that no additional retraining or fine-tuning is performed, other than the data augmentation applied during training.

Noise: To assess the robustness of our method against noise, we test it on point clouds with two levels of Gaussian noise, with intensities of $0.4\%l$ and $0.8\%l$ (where l is the maximum length of the bounding box). As presented in Table III and Table IV, our method significantly outperforms RFEPS and NerVE in edge classification and wireframe extraction across both levels of noise. Additionally, we conducted an ablation study on the refinement module and compared the performance with or without the module across all metrics. The results clearly demonstrate that the proposed refinement module significantly enhances the robustness of our method to noise.

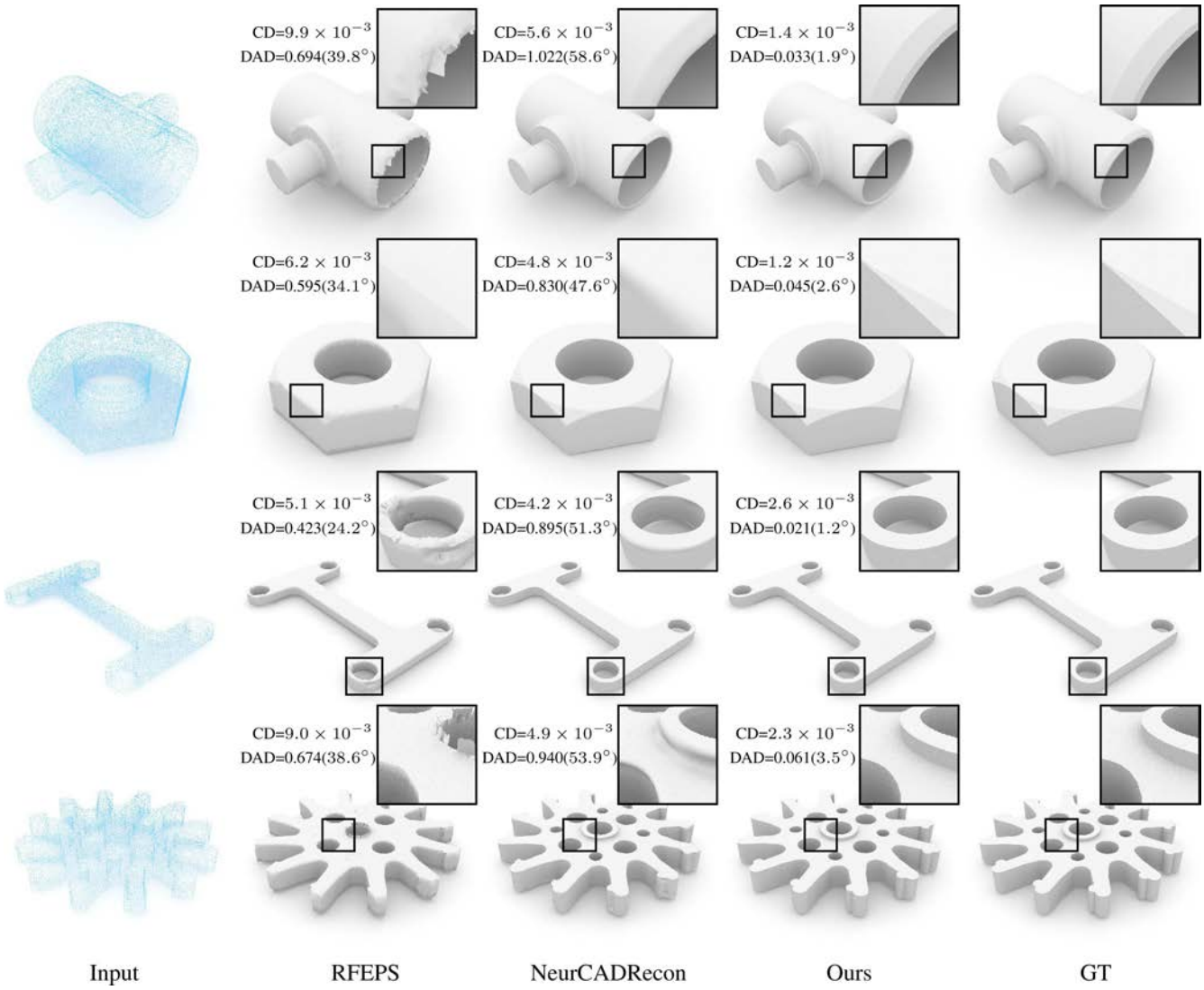


Fig. 11. Comparisons with RFEPS and NeurCADRecon on CAD models. For each reconstruction result, we report the chamfer distance (CD) and the dihedral angle difference (DAD) relative to ground-truth meshes.

As shown in Fig. 13, RFEPS exhibits strong noise resistance; however, as the noise level increases, it occasionally fails to extract edges in some local regions. On the other hand, NerVE shows sensitivity to noise, leading to substantial errors in edge point prediction. In contrast, our method outperforms others even at high noise level (e.g., $0.8\%l$), demonstrating the enhanced robustness of D-FRAME against noise. This noise immunity can be attributed to the refinement step and the direction field proposed by our approach, which effectively mitigate the impact of noisy points, resulting in more accurate edge point detection and wireframe extraction.

Non-uniform: We also evaluate the performance of RFEPS, NerVE, and D-FRAME on point clouds with varying levels of sparsity. Different from previous approaches, we assign higher sampling probabilities to points with larger z-values, creating more challenging non-uniform point clouds compared to random sampling. In our experiments, two downsampling levels of 90% and 80% are employed.

As presented in the Table III and Table IV, our method outperforms others in both edge classification and wireframe extraction tasks. Especially with the refinement module, our method can accurately extract wireframes even in the presence of sparse point clouds. In contrast, RFEPS and NerVE face challenges in such scenarios, as the sparse input often results in significant edge point omissions.

Qualitative comparisons in Fig. 13 show that D-FRAME generates more continuous and uniformly distributed edge points than RFEPS and NerVE under sparse input conditions. The non-uniformity of point clouds significantly impacts RFEPS and NerVE, particularly in regions with complex geometric features.

Robustness of direction field prediction: In D-FRAME, the direction field plays a critical role in guiding wireframe extraction, so we also conduct robustness test on direction field prediction. As shown in Table V, sparsity has limited impact on direction field prediction, since downsampling does not significantly change the overall geometry. In contrast, noise can

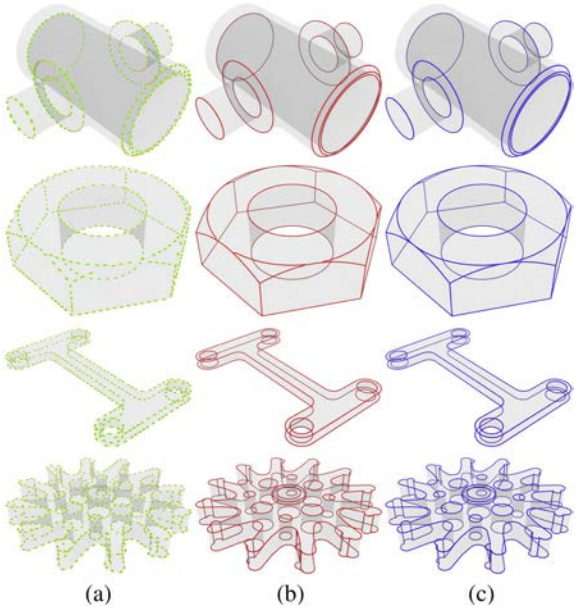


Fig. 12. Additional visual results of the four models presented in Fig. 11. (a) Visualization of direction field predicted by D-FRAME. We moderately downsample the predicted edge points for visualization purpose. The green arrows represent the directional vectors at each downsampled point (highlighted in orange). (b) Predicted wireframes. (c) GT wireframes.

TABLE V
QUANTITATIVE RESULTS ON DIRECTION FIELD PREDICTION UNDER VARIOUS NOISE LEVELS OR SAMPLED POINT NUMBERS. WE REPORT THE LOSS $\mathcal{L}_d$ INTRODUCED IN (4).

Type	Normal	Non-uniform		Noise	
	-	90%	80%	0.4%	0.8%
$\mathcal{L}_d \downarrow$	**0.0093**	0.0107	0.0125	0.0158	0.0204

affect prediction accuracy, as direction field estimation depends on local geometric features that are relatively sensitive to perturbations. As illustrated in Fig. 16, D-FRAME exhibits errors in direction field prediction when processing severely noisy point clouds.

E. Results on Real Scans

Reconstructing real CAD scans is a critical task in fields such as reverse engineering. To evaluate the performance of our method, we tested D-FRAME on real scanned CAD models, with the scan data provided by [3]. Note that we apply only moderate downsampling and normalize the point clouds into $[-1, 1]^3$ to improve computational efficiency and to align with the grid size used in PTv3. No additional preprocessing steps such as denoising are applied. The processed point clouds contain 20 K$\sim$ 30K points, which present challenges such as noise and non-uniform density, posing difficulties for accurate reconstruction. We performed a qualitative comparison with state-of-the-art CAD reconstruction methods, RFEPS [3]. As illustrated in the close-up views of Fig. 14, while RFEPS achieves satisfactory overall shape reconstruction, it often produces unnatural sharp edges especially in regions adjacent to holes. In contrast, our method, with the assistance of the refinement module, preserves the original sharp features more accurately.

TABLE VI
A COMPARATIVE ANALYSIS OF THE TIME COSTS PER MODEL FOR EDGE POINT CLASSIFICATION, WIREFRAME EXTRACTION, AND CAD MODEL RECONSTRUCTION

Methods	Edge points classification	Wireframes extraction	CAD model reconstruction
VCM	2.06s	-	-
EC-NET	0.84s	-	-
PIE-NET	0.52s	3.01s	-
NerVE	**0.15s**	**0.02s**	-
RFEPS	$\sim$3s	$\sim$4s	$\sim$9s
NeurCADRecon	-	-	$\sim$20min
Ours	0.19s	**0.02s**	$\sim$**3.4s**

V. RUNTIME PERFORMANCE

In Table VI, we compare VCM [1], EC-NET [4], PIE-NET [5], RFEPS [3], NerVE [10] and our method in terms of runtime performance for different tasks, including edge point classification, wireframe extraction, and CAD model reconstruction. Under the same hardware conditions, D-FRAME and NerVE exhibit similar runtimes for edge point classification and wireframe extraction tasks, yet outperform other methods in runtime. For CAD model reconstruction, NeurCADRecon requires an average of 20 minutes due to its iterative strategy for obtaining neural SDF. RFEPS involves several optimization steps, resulting in a slightly longer runtime. In contrast, the reconstruction process of D-FRAME is highly efficient, with only the Poisson reconstruction step for generating the base mesh being time-consuming, while all other steps operate at the millisecond level.

VI. DISCUSSION

In the following, we focus on discussing the issues within the direction field definition introduced in Section III-B2.

A. Direction Field Vs. Cross Field

We choose to regress a plain vector field rather than a more complex 4-symmetric cross field as described in [37]. This decision is driven by the nature of wireframe extraction, which only requires the direction that align with sharp edges. A deterministic vector field simplifies PWL curve connections and enhances robustness in complex geometries with dense edges. While cross fields effectively represent symmetric tangent directions, they introduce ambiguity in direction prediction, potentially hindering model convergence. In contrast, a vector field offers a clear prediction target, improving stability during training. We believe this choice strikes a balance between theoretical considerations, simplicity, and practical performance.

B. Definition of the Direction Field At Corner Points

In Section III-B2, we define the direction field at corner points as the vector pointing toward the nearest neighbor, consistent

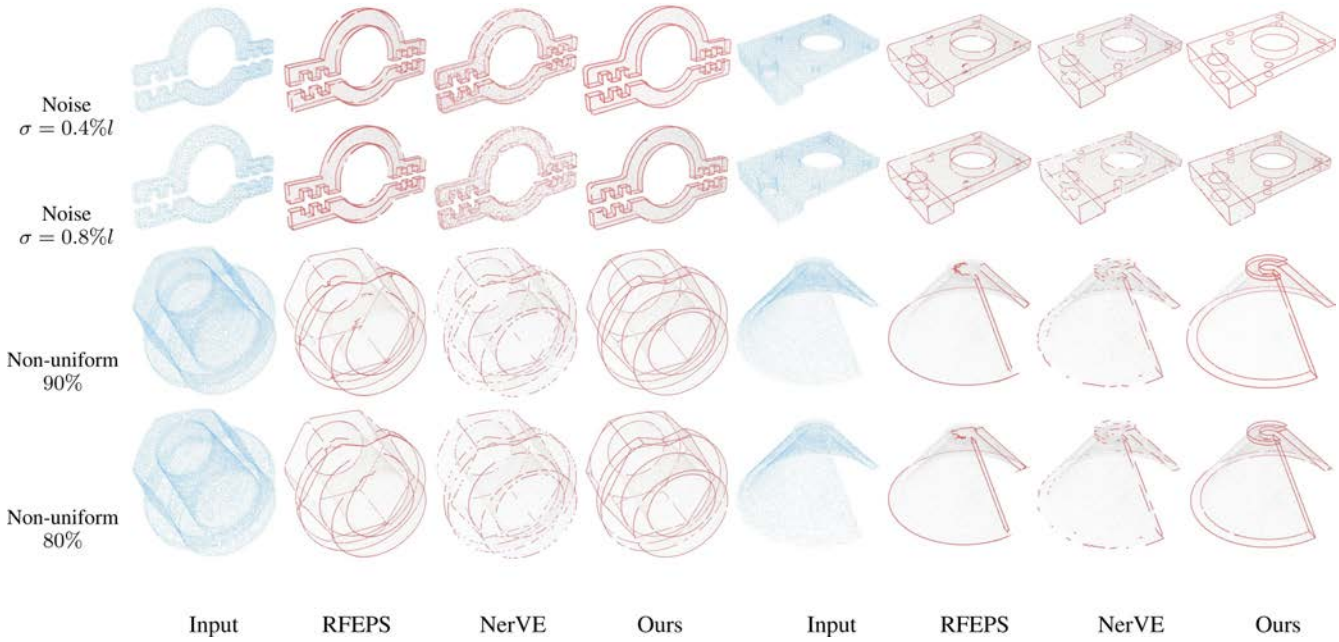


Fig. 13. Qualitative comparison of wireframe extraction results produced by D-FRAME, RFEPS, and NerVE on inputs with varying noise levels and non-uniform sampling densities.

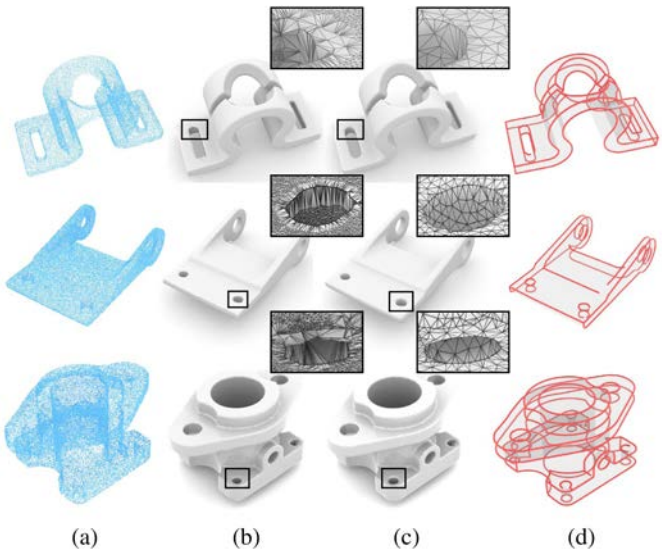


Fig. 14. (a) Input point cloud. (b) and (c) Reconstruction results produced by RFEPS and D-FRAME, respectively. (d) Wireframes extracted by D-FRAME. The close-up views demonstrate that our method achieves superior reconstruction quality and improved sharp feature preservation compared to RFEPS.

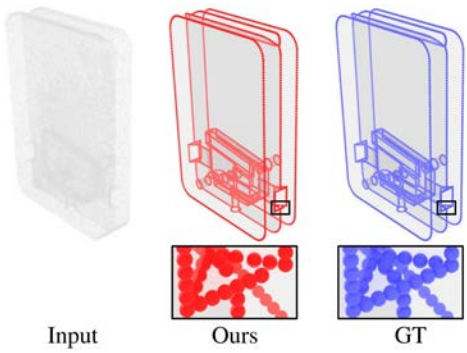


Fig. 15. Detailed visualization of the first limitation. Some edge points near regions with highly complex intersections tend to be misclassified by D-FRAME.

with the definition at regular edge points. While this choice may seem to introduce ambiguity in the connection process, our algorithm is designed to effectively handle such cases through the process described in *Step 1* and *Case 2* in *Step 2*.

Specifically, as shown in Fig. 5, the direction field at point p_n points toward the nearest point p_k, instead of the points p_i or p_j. For point p_j, the direction field remains reliable, allowing a connection to p_n via *Step 1*. For point p_i, although its direction vector may be less accurate due to geometric complexity near the corner, our method resolve this case by connecting it to the nearest neighbor if the distance is within the threshold $dist_2$. Therefore, this unified definition does not affect the stability of the wireframe connection in complex regions and helps avoid redundant handling of special cases.

C. Limitation

Highly complex intersections: In practice, a small number of misclassified edge points can be effectively handled through the combination of error region refinement and the PWL curve connection strategy. However, for highly complex intersecting regions, such as the area highlighted in the bottom-right corner of Fig. 10 and depicted in Fig. 15, a large number of misclassified edge points tend to occur, which poses a significant challenge for wireframe connectivity. Although our method achieves better results compared to state-of-the-art approaches in these cases, a gap with respect to the ground truth remains.

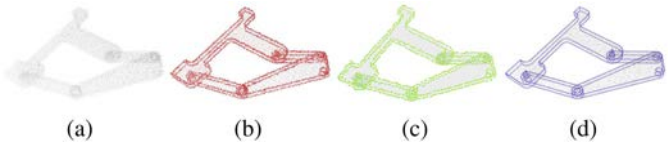


Fig. 16. D-FRAME encounters challenges when processing point clouds with high noise levels, particularly in regions with intricate geometric details. (a) Input point cloud. (b) and (c) Predicted edge points and direction field. (d) GT edge points.

Large Noise: The accuracy of wireframe extraction in this work relies on the precision of edge point classification and direction field extraction. However, for point clouds with high levels of noise, our method faces challenges in accurately predicting the direction field. This limitation arises because direction field estimation depends on local geometric features, which can be significantly disrupted by excessive noise. As illustrated in Fig. 16, D-FRAME demonstrates notable errors in edge point classification and direction field prediction when processing heavily noisy point clouds ($2\%l$).

VII. Conclusion

In this paper, we introduce D-FRAME, a novel approach for wireframe extraction from CAD-type point clouds. Our method integrates edge point detection with implicit direction fields, combined with a refinement module and a piecewise linear (PWL) extraction algorithm, to produce high-quality and robust wireframes. Equipped with point cloud reconstruction techniques and the Restricted Voronoi Diagram (RVD), D-FRAME achieves high-quality CAD model reconstruction with faithful preservation of sharp features.

References

[1] Q. Mérigot, M. Ovsjanikov, and L. J. Guibas, "Voronoi-based curvature and feature estimation from point clouds," *IEEE Trans. Vis. Comput. Graph.*, vol. 17, no. 6, pp. 743–756, Jun. 2011.

[2] H. Chen et al., "Multiscale feature line extraction from raw point clouds based on local surface variation and anisotropic contraction," *IEEE Trans. Automat. Sci. Eng.*, vol. 19, no. 2, pp. 1003–1016, Apr. 2022.

[3] R. Xu et al., "RFEPS: Reconstructing feature-line equipped polygonal surface," *ACM Trans. Graph.*, vol. 41, no. 6, pp. 1–15, 2022.

[4] L. Yu, X. Li, C.-W. Fu, D. Cohen-Or, and P.-A. Heng, "EC-Net: An edge-aware point set consolidation network," in *Proc. Eur. Conf. Comput. Vis.*, 2018, pp. 386–402.

[5] X. Wang et al., "PIE-NET: Parametric inference of point cloud edges," in *Proc. Adv. Neural Inf. Process. Syst.*, 2020, vol. 33, pp. 20167–20178.

[6] Y. Liu, S. D'Aronco, K. Schindler, and J. D. Wegner, "PC2WF: 3D wireframe reconstruction from raw point clouds," in *Proc. Int. Conf. Learn. Representations*, 2021. [Online]. Available: https://openreview.net/forum?id=8X2eaSZxTP

[7] A. Matveev et al., "DEF: Deep estimation of sharp geometric features in 3D shapes," *ACM Trans. Graph.*, vol. 41, no. 4, 2022, Art. no. 108.

[8] Y. Ye, R. Yi, Z. Gao, C. Zhu, Z. Cai, and K. Xu, "NEF: Neural edge fields for 3D parametric curve reconstruction from multi-view images," in *Proc. IEEE/CVF Conf. Comput. Vis. Pattern Recognit.*, 2023, pp. 8486–8495.

[9] Y. Li, S. Liu, X. Yang, J. Guo, J. Guo, and Y. Guo, "Surface and edge detection for primitive fitting of point clouds," in *Proc. ACM SIGGRAPH 2023 Conf. Proc.*, 2023, pp. 1–10.

[10] X. Zhu, D. Du, W. Chen, Z. Zhao, Y. Nie, and X. Han, "NerVE: Neural volumetric edges for parametric curve extraction from point cloud," in *Proc. IEEE/CVF Conf. Comput. Vis. Pattern Recognit.*, 2023, pp. 13601–13610.

[11] G. Metzer, R. Hanocka, R. Giryes, and D. Cohen-Or, "Self-sampling for neural point cloud consolidation," *ACM Trans. Graph.*, vol. 40, no. 5, pp. 1–14, 2021.

[12] T. Zhao, M. Yu, P. Alliez, and F. Lafarge, "Sharp feature consolidation from raw 3D point clouds via displacement learning," *Comput. Aided Geometric Des.*, vol. 103, 2023, Art. no. 102204.

[13] D. Bazazian, J. R. Casas, and J. Ruiz-Hidalgo, "Fast and robust edge extraction in unorganized point clouds," in *Proc. Int. Conf. Digit. Image Comput., Techn. Appl.*, 2015, pp. 1–8.

[14] S. Fleishman, D. Cohen-Or, and C. T. Silva, "Robust moving least-squares fitting with sharp features," *ACM Trans. Graph.*, vol. 24, no. 3, pp. 544–552, 2005.

[15] T. Hackel, J. D. Wegner, and K. Schindler, "Contour detection in unstructured 3D point clouds," in *Proc. IEEE Conf. Comput. Vis. Pattern Recognit.*, 2016, pp. 1610–1618.

[16] Y. Zhang, G. Geng, X. Wei, S. Zhang, and S. Li, "A statistical approach for extraction of feature lines from point clouds," *Comput. Graph.*, vol. 56, pp. 31–45, 2016.

[17] C. R. Qi, H. Su, K. Mo, and L. J. Guibas, "PointNet: Deep learning on point sets for 3D classification and segmentation," in *Proc. IEEE Conf. Comput. Vis. Pattern Recognit.*, 2017, pp. 652–660.

[18] C. R. Qi, L. Yi, H. Su, and L. J. Guibas, "PointNet++: Deep hierarchical feature learning on point sets in a metric space," in *Proc. Adv. Neural Inf. Process. Syst.*, 2017, vol. 30, pp. 5099–5108.

[19] Y. Wang, Y. Sun, Z. Liu, S. E. Sarma, M. M. Bronstein, and J. M. Solomon, "Dynamic graph CNN for learning on point clouds," *ACM Trans. Graph.*, vol. 38, no. 5, pp. 1–12, 2019.

[20] C.-E. Himeur, T. Lejemble, T. Pellegrini, M. Paulin, L. Barthe, and N. Mellado, "PCEDNet: A lightweight neural network for fast and interactive edge detection in 3D point clouds," *ACM Trans. Graph.*, vol. 41, no. 1, pp. 1–21, 2021.

[21] X. Jiao et al., "MSL-Net: Sharp feature detection network for 3D point clouds," *IEEE Trans. Vis. Comput. Graph.*, vol. 30, no. 9, pp. 6433–6446, Sep. 2024.

[22] S. Gumhold et al., "Feature extraction from point clouds," in *Proc. 10th Int. Meshing Roundtable*, 2001, pp. 293–305.

[23] H. Guo, S. Liu, H. Pan, Y. Liu, X. Tong, and B. Guo, "Complexgen: Cad reconstruction by b-rep chain complex generation," *ACM Trans. Graph.*, vol. 41, no. 4, pp. 1–18, 2022.

[24] M. Kazhdan and H. Hoppe, "Screened poisson surface reconstruction," *ACM Trans. Graph.*, vol. 32, no. 3, pp. 1–13, 2013.

[25] F. Hou, C. Wang, W. Wang, H. Qin, C. Qian, and Y. He, "Iterative poisson surface reconstruction (IPSR) for unoriented points," Association for Computing Machinery, New York, NY, USA, Jul. 2022, vol. 41, no. 4, Article ID 128, doi: 10.1145/3528223.3530096.

[26] S. Lin, D. Xiao, Z. Shi, and B. Wang, "Surface reconstruction from point clouds without normals by parametrizing the gauss formula," *ACM Trans. Graph.*, vol. 42, no. 2, pp. 1–19, 2022.

[27] M. Kazhdan, M. Bolitho, and H. Hoppe, "Poisson surface reconstruction," in *Proc. 4th Eurographics Symp. Geometry Process.*, 2006, vol. 7, no. 4, pp. 61–70.

[28] V. Sitzmann, J. Martel, A. Bergman, D. Lindell, and G. Wetzstein, "Implicit neural representations with periodic activation functions," in *Proc. Adv. Neural Inf. Process. Syst.*, 2020, vol. 33, pp. 7462–7473.

[29] Y. Ben-Shabat, C. H. Koneputugodage, and S. Gould, "DiGS: Divergence guided shape implicit neural representation for unoriented point clouds," in *Proc. IEEE/CVF Conf. Comput. Vis. Pattern Recognit.*, 2022, pp. 19323–19332.

[30] Z. Wang et al., "Neural-singular-hessian: Implicit neural representation of unoriented point clouds by enforcing singular hessian," *ACM Trans. Graph.*, vol. 42, no. 6, pp. 1–14, 2023.

[31] G. Sharma, D. Liu, S. Maji, E. Kalogerakis, S. Chaudhuri, and R. Měch, "PARSENET: A parametric surface fitting network for 3D point clouds," in *Proc. 16th Eur. Conf. Comput. Vis.*, 2020, pp. 261–276.

[32] M. A. Uy et al., "Point2Cyl: Reverse engineering 3D objects from point clouds to extrusion cylinders," in *Proc. IEEE/CVF Conf. Comput. Vis. Pattern Recognit.*, 2022, pp. 11850–11860.

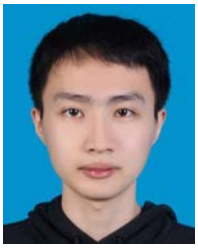
[33] F. Yu et al., "CAPRI-Net: Learning compact cad shapes with adaptive primitive assembly," in *Proc. IEEE/CVF Conf. Comput. Vis. Pattern Recognit.*, 2022, pp. 11768–11778.

[34] J. Basselin, L. Alonso, N. Ray, D. Sokolov, S. Lefebvre, and B. Lévy, "Restricted power diagrams on the GPU," *Comput. Graph. Forum*, vol. 40, no. 2, pp. 1–12, 2021.

[35] Q. Dong et al., "NeurCADRecon: Neural representation for reconstructing cad surfaces by enforcing zero Gaussian curvature," Association for Computing Machinery, New York, NY, USA, Jul. 2024, vol. 43, no. 4, Article ID 51, doi: 10.1145/3658171.

[36] X. Wu et al., "Point transformer V3: Simpler faster stronger," in *Proc. IEEE/CVF Conf. Comput. Vis. Pattern Recognit.*, 2024, pp. 4840–4851.

[37] G. Wei, H. Pan, S. Zhuang, Y. Zhou, and C. Li, "iPUNet: Iterative cross field guided point cloud upsampling," *IEEE Trans. Vis. Comput. Graph.*, vol. 30, no. 9, pp. 6089–6103, Sep. 2024.
[38] L. Qi et al., "RFF-meshing: A parallel and anisotropic quad-dominant mesh generation framework based on Riemann frame field," *Eng. Comput.*, vol. 41, no. 4, pp. 2267–2289, Aug. 2025. doi: 10.1007/s00366-024-02096-7
[39] W. Zheng, H. Wu, G. Xu, R. Ling, and R. Gu, "Feature-preserving quadrilateral mesh boolean operation with cross-field guided layout blending," *Comput. Aided Geometric Des.*, vol. 111, 2024, Art. no. 102324.
[40] T.-Y. Lin, P. Goyal, R. Girshick, K. He, and P. Dollár, "Focal loss for dense object detection," in *Proc. IEEE Conf. Comput. Vis. Pattern Recognit.*, 2017, pp. 2980–2988.
[41] G. Qian, A. Abualshour, G. Li, A. Thabet, and B. Ghanem, "PU-GCN: Point cloud upsampling using graph convolutional networks," in *Proc. IEEE/CVF Conf. Comput. Vis. Pattern Recognit.*, 2021, pp. 11683–11692.
[42] D.-M. Yan, B. Lévy, Y. Liu, F. Sun, and W. Wang, "Isotropic remeshing with fast and exact computation of restricted Voronoi diagram," *Comput. Graph. Forum*, vol. 28, no. 5, Wiley Online Library, pp. 1445–1454, 2009.
[43] D.-M. Yan, G. Bao, X. Zhang, and P. Wonka, "Low-resolution remeshing using the localized restricted voronoi diagram," *IEEE Trans. Vis. Comput. Graph.*, vol. 20, no. 10, pp. 1418–1427, Oct. 2014.
[44] M. Khoury and J. R. Shewchuk, "Fixed points of the restricted delaunay triangulation operator," in *Proc. 32nd Int. Symp. Comput. Geometry*, 2016, pp. 47:1–47:15.
[45] J. Xie et al., "Interpolatory Catmull-Clark volumetric subdivision over unstructured hexahedral meshes for modeling and simulation applications," *Comput. Aided Geometric Des.*, vol. 80, 2020, Art. no. 101867.
[46] Y.-J. Liu, C.-X. Xu, D. Fan, and Y. He, "Efficient construction and simplification of Delaunay meshes," *ACM Trans. Graph.*, vol. 34, no. 6, pp. 1–13, 2015.
[47] S. Koch et al., "ABC: A big CAD model dataset for geometric deep learning," in *Proc. IEEE/CVF Conf. Comput. Vis. Pattern Recognit.*, 2019, pp. 9601–9611.
[48] H. Fan, H. Su, and L. Guibas, "A point set generation network for 3D object reconstruction from a single image," in *Proc. IEEE Conf. Comput. Vis. Pattern Recognit.*, 2017, pp. 2463–2471.
[49] M. Berger, J. A. Levine, L. G. Nonato, G. Taubin, and C. T. Silva, *ACM Trans. Graph.*, vol. 32, no. 2, pp. 1–17, Apr. 2013.



Yuan Feng received the BS degree from the School of Computer Science and Technology, Shandong University of Finance and Economics, in 2023. He is currently working toward the master's degree with the School of Software, Shandong University, Jinan, China. His research interests include point cloud processing and geometric analysis.

Honghao Dai received the B.S. and M.S. degrees from the School of Computer Science and Technology, Shandong University of Finance and Economics, in 2021 and 2024, respectively. He is currently working toward a Ph.D. with the School of Software, Shandong University. His research interests include machine learning, medical image processing, and computer vision

Guangshun Wei received the PhD degree from the School of Software, Shandong University, Jinan, China, in 2022. From 2023 to 2024, he was a postdoctoral fellow with the Department of Computer Science, The University of Hong Kong. He is currently an associate research fellow with the School of Software, Shandong University. His research interests include machine learning, image processing, and geometric analysis.

Long Ma received the PhD degree from the School of Computer Science and Technology, Shandong University in 2017. He is currently an associate researcher with the School of Software, Shandong University. His research interests include computer graphics, geometry modeling, and mechanical simulation.

Pengfei Wang received the BS and PhD degrees from the School of Computer Science and Technology, Shandong University, in 2013 and 2022, respectively. He was a postdoctoral with the Department of Computer Science, The University of Hong Kong, Hong Kong. He is currently an associate researcher with the School of Software, Shandong University. His research interests include computer graphics, geometric computing, and 3D reconstruction.

Yuanfeng Zhou received the master's and PhD degrees from the School of Computer Science and Technology, Shandong University, Jinan, China, in 2005 and 2009, respectively. From 2009 to 2011, he was a postdoctoral with Graphics Group, Department of Computer Science, The University of Hong Kong, Hong Kong. He is currently a professor with the School of Software, Shandong University, where he is also the leader of IGIP Laboratory. His research interests include geometric modeling, information visualization, and image processing.

Ying He (Member, IEEE) is currently an associate professor with the School of Computer Engineering, Nanyang Technological University, Singapore. His research focuses on geometric computing and analysis. He actively participates in the technical program committees of major conferences in geometric modeling and is/was on the editorial boards of *IEEE Transactions on Visualization and Computer Graphics*, *Computer Graphics Forum*, and *Computational Visual Media*. He was also a general/program co-chair for the Shape Modeling International conference in 2022, Symposium on Solid and Physical Modeling in 2022 and 2023, Geometric Modeling and Processing conference in 2014 and 2021, and Conference on Computational Visual Media in 2020.