www.cad-journal.net
Computer-Aided Design AND APPLICATIONS™

Surface Reconstruction Based on Geometric Primitive Feature Detection

Bao-Wen Xu[1], Wei-Kang Liu[2], Ji-Xing Li[3] and Li-Yong Shen[4]

[1]AVIC-Digital, Tsinghua University, xubw001@avic.com
[2]University of Chinese Academy of Sciences, liuweikang22@mails.ucas.ac.cn
[3]AVIC-Digital, li_jixing@buaa.edu.cn
[4]University of Chinese Academy of Sciences, lyshen@ucas.ac.cn

Corresponding author: Wei-Kang Liu, liuweikang22@mails.ucas.ac.cn

Abstract. This paper proposes a surface reconstruction scheme based on primitive detection to preserve sharp features. The scheme employs global primitive detection as the fundamental approach for point cloud processing, using primitive segmentation to provide stable normal vector estimation and to extract potentially sharp features at primitive junctions. Specifically, linear sharp feature detection is achieved through normal vector angle analysis and primitive intersection line localization, while point cloud denoising and normal vector updating are accomplished by integrating normal vector information. Finally, surface reconstruction is performed using the Restricted Power Diagram (RPD) method, which incorporates normal vectors and sharp-feature constraints, thereby preserving the sharp features of the mesh model.

Keywords: Feature detection, Surface reconstruction, Geometric primitives, Point cloud denoising.
DOI: https://doi.org/10.14733/cadaps.2026.494-514

1 INTRODUCTION

Three-dimensional (3D) object reconstruction from point clouds is a cornerstone of reverse engineering and a fundamental challenge in computer-aided design (CAD). CAD models are typically composed of smooth surface patches bounded by sharp geometric features — such as edges and corners — that encode critical structural and semantic information. These sharp features serve as essential geometric clues for accurate shape recovery and enable a wide range of downstream tasks, including model editing, segmentation, and parametric reconstruction [18, 36, 49]. However, reconstructing CAD-like surfaces with well-preserved sharp features from raw, noisy point clouds — especially those lacking reliable normal information—remains highly nontrivial.

The core difficulties lie in two interrelated aspects. First, points that precisely lie on sharp features are often extremely sparse or even absent in typical input point clouds due to sampling limitations or sensor noise. This sparsity makes it impossible to directly extract sharp features from the raw data, necessitating strategies to enrich feature regions through point resampling or

inference. Second, even when sharp feature points are identified, faithfully reconstructing a mesh that preserves these features requires careful handling of both geometry and topology during surface generation — a task further complicated by noise and irregular point distribution.

To address these challenges, we propose a primitive detection – based framework for joint sharp feature detection, point cloud denoising, and geometric reconstruction. Our method leverages global geometric priors inherent in CAD models: most surfaces consist of piecewise smooth primitives (e.g., planes, cylinders, spheres), and sharp features naturally arise at their boundaries. By detecting these underlying primitives, we can segment the point cloud into coherent regions, estimate robust normals — even in the absence of initial normal data — and identify candidate sharp feature zones from inter-primitive boundaries. Subsequently, true sharp features are refined using angular analysis of adjacent primitive normals, and linear feature curves are recovered via intersections of fitted primitives. The resulting enhanced point set — augmented with densified edge points and updated normals—is then used in a restricted power diagram – based reconstruction that explicitly respects both normal directions and sharp feature constraints.

In light of these limitations, our work bridges the gap between geometric reasoning and data-driven reconstruction by grounding the entire pipeline in primitive-based analysis—a natural fit for CAD domains. The main contributions of this paper are threefold:

(1) We present a multi-stage algorithm that transforms noisy, unoriented CAD point clouds into clean, feature-preserving triangular meshes.

(2) Sharp feature regions and reliable normals are jointly obtained through primitive detection, enabling simultaneous denoising, feature refinement, and normal correction.

(3) A restricted power diagram reconstruction is employed with adaptive weighting that prioritizes newly added edge points based on normal consistency, ensuring geometric fidelity and topological correctness of sharp features.

By integrating global primitive understanding with local geometric refinement, our method achieves robust, high-quality reconstruction of CAD models directly from imperfect point cloud inputs—without requiring precomputed normals, extensive training data, or manual parameter tuning.

2 RELATED WORK

Our work builds upon and intersects with extensive research in point cloud sharp feature detection and surface reconstruction. As highlighted in the Introduction, the core challenge lies in jointly overcoming data imperfections (noise, sparsity, lack of orientation) and preserving sharp geometric features, a necessity for high-fidelity CAD model reconstruction. Existing approaches often address denoising, feature detection, and reconstruction in sequential or isolated stages, leading to cascading errors or compromised results. Our method, which unifies these steps through primitive-based analysis, contrasts with and complements the following lines of research.

2.1 Point Cloud Sharp Feature Detection

Accurate detection of sharp features, edges, and corners is paramount for reconstructing CAD models with geometric fidelity. These features carry critical structural information but are notoriously difficult to extract from raw, noisy point clouds due to sampling sparsity and sensor noise.

Traditional Methods primarily rely on analyzing local geometric properties. Early approaches detected features by measuring normal variation [32 Resampling (EAR) [20] and FACE [6] were developed to strategically resample and densify points in feature regions. As integrated in libraries like CGAL [12], these techniques are foundational. However, they suffer from significant limitations highlighted in our Introduction: they are highly sensitive to noise, require careful parameter and threshold tuning, and typically need reliable pre-computed normals. Crucially, they treat feature detection as a pre-processing step separate from denoising. and global shape understanding, making

them vulnerable to errors in challenging, unoriented point clouds, or enforcing continuity constraints [16]. Subsequent methods improved robustness by employing Gaussian map clustering of local neighborhoods [43], thresholding the Voronoi Covariance Measure (VCM) [31], or analyzing tensor eigenvalues to assign feature weights [34]. To aid reconstruction, methods like Edge-Aware.

Deep Learning Methods seek to learn feature representations directly from data in an end-to-end manner. Pioneering works like EC-Net [48] and PIE-Net [42] employ specialized architectures for point cloud edge detection, though they can be computationally intensive for complex models. Other networks, such as Self-sample [43] and PCED-Net [17], have been introduced for shape analysis and edge detection. Liu et al. [29] proposed pc2wf, utilizing region proposal and feedforward modules for this task. Similarly, Li et al.' s SEDNet [26] employs a dual-branch architecture to fuse edge and corner features into distinct feature lines. While capable of learning complex patterns, these data-driven approaches face significant hurdles: their performance is contingent on large, high-quality annotated datasets, and they often produce predictions that are sparse and geometrically discontinuous. This sparsity, as noted in the Introduction, renders them suboptimal for driving a high-fidelity, feature-preserving reconstruction pipeline.

Our approach circumvents these issues by using global primitive detection to infer feature regions robustly, simultaneously denoising the data and estimating stable normals, effectively addressing the intertwined challenges of noise and sparsity.

2.2 Point Cloud Triangular Mesh Reconstruction

Surface reconstruction methods can be categorized into explicit and implicit paradigms, both of which struggle to balance smooth surface recovery with sharp feature preservation under noisy conditions.

Explicit Reconstruction methods construct the mesh directly from the point set. *Direct Connection Methods*: Foundational algorithms like the Ball Pivoting Algorithm [4] and Delaunay Triangulation [37] inspired many subsequent feature-aware variants. Wang et al. [40] used a kernel-based scale estimator for tangent plane estimation and outlier removal. Salman et al. [35] extracted sharp features via Voronoi cell covariance analysis and modified Delaunay refinement to preserve them. Dey et al. [8] combined the Gaussian Weighted Graph Laplacian and Reeb graphs with Weighted Delaunay Triangulation (WDT) to protect feature samples. Digne et al. [9] simplified an initial 3D Delaunay triangulation via optimal transport with a feature-sensitive metric. Xiong et al. [45] proposed a dictionary-learning framework to simultaneously optimize mesh metrics and connectivity for unoriented points. A recent state-of-the-art example is RFEPS [46], which detects features, enhances feature lines, and reconstructs a surface using a restricted power diagram. While demonstrating feature awareness, these methods often inherit the robustness and parameter sensitivity issues of their underlying feature detectors. *Mesh Approximation Methods*: These methods exploit the CAD-domain prior that surfaces are composed of smooth patches. Works like [18, 36, 49] note that reconstruction can proceed by approximating these patches. Groueix et al. [15] proposed AtlasNet, which uses a collection of parametric patches, but struggles with watertight connectivity, leading to cracks and self-intersections. Williams et al. [44] introduced new metrics to improve patch quality but could not fully solve the connectivity learning problem. Chen et al. [7] used BSP-trees to guide convex decomposition, limiting expressivity. Deep Learning-Based Implicit Methods: These use neural networks as continuous shape representations. DeepSDF [33] fits a Signed Distance Function (SDF)[2] but requires ground-truth SDF supervision. Unsupervised methods like IGR [14] use the Eikonal loss but produce overly smooth surfaces. To capture details, SIREN [38] uses periodic activations, and follow-up works employ positional encoding [39] or Spline Positional Encoding (SPE) [41], though these can introduce instability or noise. Methods like Neural Marching Cubes (NMC) [30] and Neural Dual Contouring (NDC) [25] are direct improvements, with NDC excelling at sharp features. Other specialized networks include ComplexGen [28] for B-Rep structures, SECAD-Net [24] for compact CAD models, and DRSF [13], which uses a two-stage training and edge detection to preserve sharp features. As critiqued in the Introduction, these neural

approaches often have a smoothing bias, struggle to generalize without feature supervision, and can be computationally inefficient.

In contrast, our reconstruction stage is explicitly and robustly guided by the output of our primitive-based analysis. By feeding an enriched point set with reliable normals and geometrically consistent edge points into a feature-aware restricted power diagram reconstructor [46], we directly bridge the gap between global shape understanding and local geometric fidelity, ensuring both topological correctness and sharp feature preservation.

Other strategies include using Boundary-Sampled Halfspaces (BSH) [10], robust statistics and nonlinear kernels [1], the Locally Optimal Projection (LOP) operator [27], network deformation [19], leveraging priors [46], and 2D Delaunay triangulation [26]. The central challenge remains accurately identifying and stitching patch boundaries, which are the sharp features.

Implicit Reconstruction methods define the surface as an isosurface of a scalar function. Traditional Implicit Methods: Poisson Surface Reconstruction [21, 22] is a seminal global method that tends to over-smooth sharp features. Its improved version [22] remains a memory-intensive method.

3D point clouds are structurally sparse, making it challenging to directly extract geometric information from the point-based representation. This poses significant difficulties for surface reconstruction from 3D point clouds. CAD models exhibit specific characteristics: they typically consist of large, smooth surfaces joined together by distinct, sharp features at their intersections. Accurately capturing these sharp features is crucial not only for effectively representing the model's geometric information but also for the success of the surface reconstruction process itself. Therefore, this chapter presents a sharp feature detection and surface reconstruction algorithm based on geometric primitives. Its goal is to extract and reconstruct sharp features from 3D point clouds and subsequently reconstruct 3D models that faithfully preserve these sharp features.

2.3 Algorithm Framework

Given a 3D point cloud model, the objectives of this chapter are to inspect feature regions within the model, generate well-defined linear features, and ultimately reconstruct clean CAD models with sharp feature adjustments. The most critical goals are to obtain sharp feature regions and generate these sharp features.

To achieve this, a multi-stage algorithm is proposed as follows:

1. Optimize initial point normals and adjust their orientation. Construct a Restricted Voronoi Diagram (RVD) on the base surface generated by Screened Poisson Reconstruction (SPR).

2. Utilize the RVD as input to the primitive detection module to obtain a segmented representation of the point cloud into primitives.

3. Update point cloud normals based on the primitive segmentation. Identify potential sharp feature regions and extract initial sharp features using the boundaries between primitives.

4. Perform a two-stage optimization process on the point cloud and its normals for denoising and refinement.

5. Reapply SPR to the processed point cloud. Employ a Restricted Power Diagram (RPD) to reconstruct a clean CAD model that faithfully incorporates sharp feature adjustments.

2.4 Normal Vector and Connectivity Generation

Preprocessing the point cloud is necessary before inputting it into the geometric primitive detection module. This involves constructing a point cloud connectivity graph to obtain connectivity properties. Compared to methods using KNN or Ball Pivoting to generate a triangular mesh, constructing the graph by generating a base surface via Poisson reconstruction and inferring point connectivity using an RVD results in fewer holes, fewer edges, and accelerates the point-based primitive detection process.

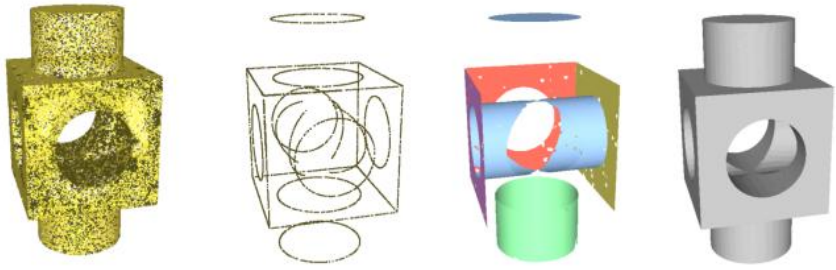

Figure 1: Sharp feature detection and surface reconstruction.

Generating the base surface via Poisson reconstruction requires point normals. These normals are used only for Poisson reconstruction and to assist in determining the correct sign (orientation) for the subsequent true normals. Therefore, Principal Component Analysis (PCA) is employed here: the eigenvector corresponding to the smallest eigenvalue of the covariance matrix constructed from the local neighborhood of each input point is used as its normal vector.

A global consistency-based method propagates and controls normal orientations to ensure consistency. An arbitrary seed point is selected, and its normal orientation is fixed. Breadth-First Search (BFS) traverses the point cloud. For each point, if its normal is opposite to the weighted average normal within its current neighborhood (indicated by a negative dot product), it is flipped. Points already traversed are assigned higher weights.

The Restricted Voronoi Diagram (RVD) [11, 47] is a common mesh generation tool. Given a base surface sufficiently close to the target surface (allowing deviations related to the local feature size), the RVD can generate a high-quality approximating surface. A traditional Voronoi diagram Ω partitions space into regions $\Omega = \{\Omega_i\}_{i=1}^n$, based on a set of points, where each region Ω_i contains points closest to a specific input point x_i.RVD is defined as the intersection of the 3D Voronoi diagram with a surface. Given a smooth 2D manifold surface $S \in R^3$ and a finite set of sample points $X = \{x_i\}_{i=1}^n$ on it, the RVD is the intersection of the 3D Voronoi diagram $\Omega = \{\Omega_i\}_{i=1}^n$ with the surface S :

$$\Omega_{|s} = \Omega \cap S = \{\Omega_i \cap S\}_{i=1}^n,$$

where a Restricted Voronoi Cell (RVC) is defined as:

$$\Omega_{i|S} = \{x \in S \mid d(x,x_i) \leq d(x,x_j), \forall x_j \in X, j \neq i\},$$

And $d(x,y) = \| x - y\|$ denotes the Euclidean distance between two points.

The dual of the RVD is a subcomplex of the 3D Delaunay triangulation, called the Restricted Delaunay Triangulation (RDT). This method is used in this paper to triangulate the input point cloud, which is then input into the next stage, the primitive detection module, as shown in Figure 2. The top-left image represents the input base surface. The top-right image shows the input point cloud (yellow points), with black lines representing the intersection of Voronoi polygons on the surface. Connecting points within one Voronoi polygon (yellow) to points in adjacent Voronoi polygons yields the triangulated surface via Restricted Delaunay Triangulation, shown in the bottom-left and bottom-right images. Using this mesh as input to the primitive detection module significantly enhances detection efficiency.

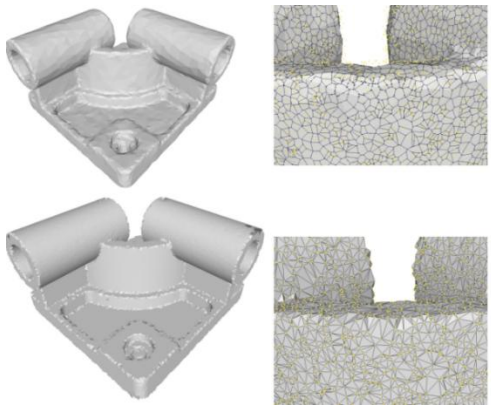

Figure 2: RVD and its dual.

2.5 Primitives Detection and Feature Detection

Primitive detection and fitting essentially involves optimizing segmented results under primitive constraints, then converting the segments into primitive forms. The geometric primitive detection algorithm based on genetic algorithms proceeds as follows:

Input: Point cloud and its connectivity relationships.

Output: Point cloud segmentation and parametric representation of patches.

Step 1. Generate multiple initial segmentation schemes using the point cloud and connectivity relationships to form an initial population.

Step 2. Iteratively perform genetic evolutionary operations on the segments until termination criteria are met:

Step 2.1. Crossover: Exchange segmentation information between individuals.

Step 2.2. Mutation: Modify segmentation information within a single individual.

Step 2.3. Selection: Filter optimal individuals for the next evolutionary iteration.

Step 3. Output the optimal individual as the segmentation scheme, along with each patch's best-fit primitive and parameters.

Convert the 3D point cloud into a graph form to serve as the input for the genetic algorithm. According to the primitive detection requirements, design the objective function as follows:

$$E_{fit} = \omega_1 \times E_{error} + \omega_2 \times (E_{number} + E_{size} + E_{edge}),$$

First, seed points are selected from the point cloud. Then, the point cloud is partitioned using either a greedy primitive-expansion initialization scheme or an average initialization scheme, which serves as the initial population for the genetic algorithm. Subsequently, genetic evolutionary operations are performed iteratively to optimize the segmentation and fitting results. These operations include: 1) **Selection**, employing a family-elitism strategy where each parent and its mutated offspring are treated as a family, and the best individual from each family is selected for the next generation to balance convergence speed and genetic diversity; 2) **Crossover**, which randomly selects two different segmentation solutions, C_0 and C_1, using a linear ranking probability P_i to determine selection weights, and generates a new chromosome by exchanging boundary region information between adjacent patches; and 3) **Mutation**, which locally adjusts a single segmentation solution through five operators: merging two adjacent patches into one, splitting and merging patches with three or fewer points into adjacent ones, expanding a patch by absorbing neighboring points that are closer and within a threshold, contracting a patch by reclassifying distant points as noise and moving them to adjacent patches, and fine-tuning boundaries based on distances from boundary points to neighboring patches. Finally, the segmented point cloud results and the specific parameters

of their corresponding parametric surfaces are output. For normal vector generation, the original point cloud is projected onto the parametric surface of its corresponding primitive, and the surface normal at the projection point is assigned as the point normal. These normals are then corrected for global consistency: if a normal is opposite to the original normal (indicated by a negative dot product), it is flipped.

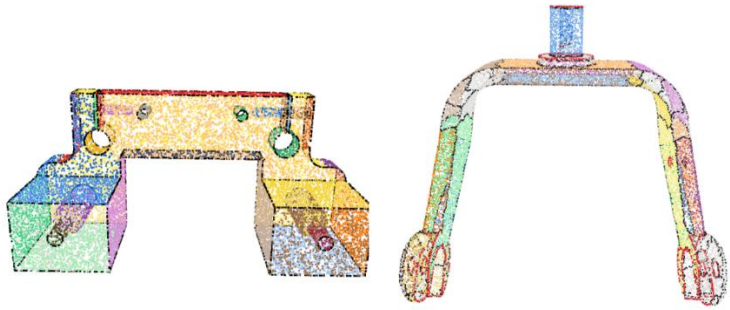

Figure 3: Model of primitive representation.

Once primitive detection is complete for the point cloud, the corresponding primitive parameters are obtained. As shown in Figure 3, each segment represents a primitive, and the boundaries between these primitives become potential locations for sharp features. Relatively accurate, sharp features can be obtained through screening and validation. The most critical parameter during sharp feature detection is the generation of point normals. In this method, the original point cloud is projected onto the parametric surface of its corresponding primitive. The normal vector at the projection point on the primitive’ s surface is assigned as the point cloud normal. These normals are then corrected based on the original normals: if a normal is opposite to the original normal (indicated by a negative dot product), it is flipped to ensure global consistency.

To generate a point cloud that includes sharp features, potential sharp feature points are added to the original point cloud by projecting points from both sides of the primitive intersection line onto the intersection line itself. The primitives used in detection are planes, spheres, cylinders, and cones. First, for all edges that cross different patches, a criterion is applied: if the angle between the normal vectors n_i and n_j of the two endpoints p_i and p_j is angle(n_i,n_j) ≤ π/6$, then the projections of these points onto the intersection line of the parametric surfaces are considered potential sharp feature points. For instance, if p_i lies on a cylinder and p_j on a cone, p_i is moved along the cylinder's axis until it intersects the cone's surface, while p_j is moved along the cone's generatrix until it meets the cylinder's surface, thus obtaining the projection points. As illustrated in Figure 4, this method of generating sharp features from primitive boundaries enables the detection of sharp features in complex structures.

2.6 Point Cloud Denoising and Normal Vector Optimization

The primitive detection module demonstrates robustness against noise in the input point cloud, delivering consistent and unified normal vectors. However, acquiring a clean CAD model with well-defined, sharp features still necessitates a denoising process for the point cloud. For models consisting exclusively of combinations of the four primitive types (planes, spheres, cylinders, and cones) covered in the primitive detection, denoising can be straightforwardly achieved by projecting the point cloud directly onto the parametric surfaces of the corresponding primitives. This direct projection method, however, is clearly inapplicable to surfaces that cannot be represented by these primitives.

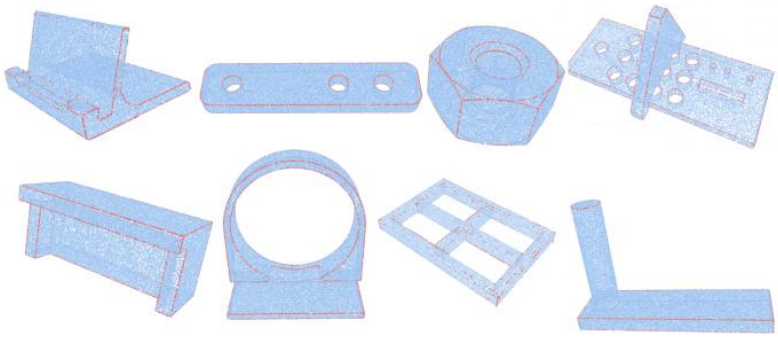

Figure 4: Point clouds and sharp features.

For point clouds lying on smooth surfaces, a covariance matrix M_i is constructed within a local neighborhood:

$$M_3^i \triangleq \sum_{p_j \in Neigh(p_i)} (p_i - p_j)(p_i - p_j)^T,$$

where $Neigh(p_i)$ denotes a spherical neighborhood with radius r. In the default setting, $r = 2\delta$ (twice the average distance between points). The eigenvectors of the covariance matrix encode the local surface orientation. Point positions are updated by moving each point along its normal direction: $p_{i'} = p_i + \epsilon_i \mathbf{n_i}$.

If the normal vector $\mathbf{n_i}$ at point p_i accurately reflects the true surface orientation, then $M_i \mathbf{n_i}$ should approximate a null vector. Leveraging the primitive fitting results from the previous stage: when a patch exhibits high fitting quality, both the normals and positions of points on that patch are more accurate. Therefore, moving these points and their normals should incur a higher penalty. Combining these two principles, the following objective function is designed for point cloud denoising:

$$\min_{\{\epsilon_i\}} \{\sum_{i=1}^{n} || M_3^i \mathbf{n_i} ||^2 + \xi_i \sum_{i=1}^{n} \epsilon^2\}, \tag{3.1}$$

Here, ξ_i is a parameter balancing denoising and fidelity. It is computed as the ratio of the global mean squared fitting error to the mean squared fitting error of the patch containing p_i, multiplied by a constant $\xi = 0.01$.

$$\xi_i = \frac{E_{error} \times N_p}{E_p \times N(v)} \xi,$$

When normal information is highly accurate, optimizing this function achieves surface smoothing for denoising. To differentiate between sharp regions and points on different surfaces, points in $Neigh(p_i)$ are filtered: $Neigh(p_i)' \subseteq Neigh(p_i)$, requiring $\langle \mathbf{n_i}, \mathbf{n_j} \rangle \leq \pi / 6$. The covariance matrix is then reconstructed using this filtered neighborhood.

$$M_3^i \triangleq \sum_{p_j \in Neigh(p_i)'} (p_i - p_j)(p_i - p_j)^T,$$

Figure 5: Point cloud denoising results.

For models represented by primitives, this scheme yields results similar to direct projection onto primitive surfaces. For complex surfaces, it produces smoother results while effectively preserving features. Additionally, outlier points are detected and removed based on the average distance from each point p_i to its k-nearest neighbors p_j.

The denoising results can further optimize the normals. A similar optimization function can be designed:

$$\min_{\{\mathbf{n_i}\}} \sum_{i=1}^{n} || M_3^j \mathbf{n_i} ||^2,$$

By changing the optimized variable to the normals $\mathbf{n_i}$, this function smooths the normal field, addressing normal generation for non-primitive surfaces.

The principal advantage of this integrated denoising and normal optimization scheme is its ability to preserve sharp features. By initializing the process with the accurate normals from primitive fitting (or a robust estimation for non-primitive regions) and employing a filtered neighborhood that respects normal discontinuity, the method avoids blurring across edges and corners.

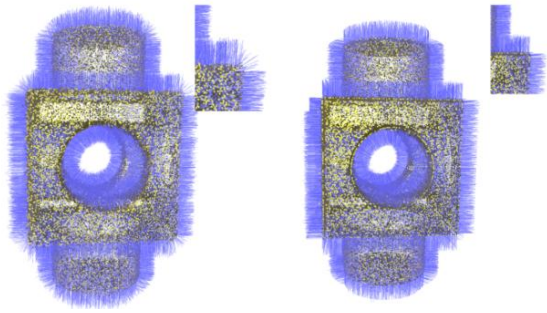

Figure 6: Optimization results of normal vectors.

2.7 Surface Reconstruction

After denoising the original point cloud, surface reconstruction incorporating sharp feature points is required to generate models that preserve sharp features. Since implicit methods struggle to handle

linear sharp features, this section utilizes and optimizes the Restricted Power Diagram (RPD)[3] for high-quality model generation.

By replacing the Euclidean distance with the power distance

$$d_P(x,w_x;y,w_y) = (x-y)^2 - w_x - w_y,$$

(where w_x, w_y are the weights assigned to points x and y), the RVD generalizes to the RPD.

While Voronoi diagrams assign equal importance to all points —potentially failing to explicitly align triangulations with potential feature lines if edge point density is insufficient —Power Diagrams, as extensions of Voronoi diagrams, provide a superior solution. By assigning larger weights to edge points, they prioritize connections along sharp features, thereby generating models that preserve these features.

In our implementation, A base surface is generated by running the Screened Poisson Reconstruction (SPR) solver on the denoised point cloud augmented with sharp feature points. Each point is projected onto this reconstructed surface. Point connectivity is inferred using the RPD. Compared to the RFEPS algorithm [46], which assigns different fixed weights to feature points (E) and non-feature points, our approach dynamically assigns larger weights to regions of higher surface distortion. This yields superior surface reconstruction. The weight w_i for each point p_i is determined by the maximum normal angle within its k-nearest neighbors:

$$w_i = \begin{cases} \max_{j\in N(i)} \left(\frac{\langle \mathbf{n_i}, \mathbf{n_j} \rangle}{\pi} \right) \delta & , p_i \notin E \\ \delta & , p_i \in E, \end{cases} \tag{3.2}$$

Let δ be the average gap between points. Finally, the connectivity information is copied back to the original point set to complete the reconstruction task. As shown in Figure 7: The left image represents the input base surface. In the top image, yellow points denote the input point cloud, while black lines represent the Restricted Power Diagram (RPD). Note that the generated polygons are non-uniform. Connecting points (yellow) within one polygon to points in adjacent polygons yields the triangulated meshed surface shown in the bottom and right images. As demonstrated in Figure 8, the improved RPD method effectively reconstructs sharp feature-preserving models from point clouds augmented with sharp features.

Figure 7: RPD and its dual.

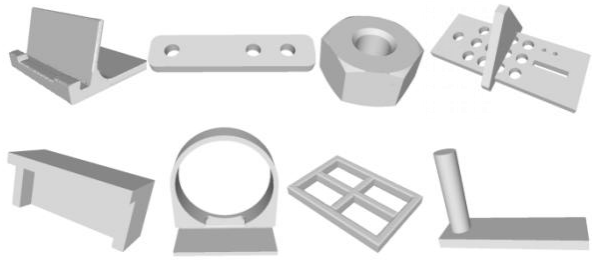

Figure 8: Surface reconstruction results.

3 EXPERIMENTAL RESULTS

3.1 Experimental Setting

The proposed algorithm was implemented in C++, utilizing the Eigen library for solving linear systems. Experiments were conducted on a workstation equipped with an Intel Xeon(R) W-2245

processor (3.90GHz, 16 threads) and 64.0GiB RAM. This section evaluates the algorithm's performance across multiple test cases. For comparisons with learning-based methods, these approaches were executed on an NVIDIA RTX 4000 GPU. All point cloud models were normalized to the range $[-0.5,0.5]^3$. The constrained optimization problem defined in Section 3.4 was solved using the LBFGS function from the ALGLIB solver [5], with constraints configured via the `minbleicsetlc` function.

Datasets: Experiments utilized synthetic data: Synthetic point clouds were sampled from the ABC dataset [23], selecting 100 defect-free models (e.g., no self-intersections). 50,000 surface samples were uniformly extracted per model. Noisy variants were generated by adding Gaussian noise at three intensity levels: $0.1\%,0.5\%,0.2\%$ of the bounding box diagonal length.

Evaluation Metrics. Addressing the dual focus of point cloud optimization/feature detection and surface reconstruction, this chapter employs the following metrics: Point cloud adherence to the true surface is evaluated using One-way Chamfer Distance (OCD), Hausdorff Distance (HD), and denoising precision; sharp feature construction accuracy is assessed via One-way Chamfer Distance (OCD), Chamfer Distance (CD), and F-score (F1); the fidelity of reconstructed meshes is quantified using Chamfer Distance (CD), F-score (F1), and Normal Consistency (NC).

3.2 Point Cloud Denoise Quality

In this section, our method is benchmarked against state-of-the-art point cloud optimization approaches, including EC-Net [48], Dis-PU [25], and RFEPS [46]. Denoising capabilities were evaluated by introducing Gaussian noise at three intensity levels: $0.1\%,0.5\%,0.2\%$ of the bounding box diagonal length.

Method	OCD $(\times 10^4)\downarrow$			HD $(\times 10^2)\downarrow$			Precision $(\tau = 0.005)\uparrow$		
	0.2%	0.5%	1.0%	0.2%	0.5%	1.0%	0.2%	0.5%	1.0%
noise cloud	0.18	0.29	0.70	1.044 8	2.040 6	3.544 8	0.923 3	0.835 6	0.526 7
EC-Net	0.17	0.19	0.24	1.441 5	**1.563 2**	2.450 5	0.993 3	0.978 8	0.908 6
RFEPS	0.21	0.19	0.23	**0.830 9**	1.999 1	2.703 1	0.996 8	**0.994 9**	0.964 6
Ours	**0.16**	**0.17**	**0.22**	0.911 1	1.659 0	**2.375 9**	**0.997 2**	0.982 9	**0.923 2**

Note: **Red and bolded** indicate optimal, blue indicates secondary. ↓: smaller values better; ↑: larger values better.

Table 1: Comparison of point cloud denoising performance.

Figure 9 provides visual comparisons of optimization results, while Table 1 quantitatively compares the denoised point clouds using three metrics: One-way Chamfer Distance (OCD) between noise-free and denoised point clouds, Hausdorff Distance (HD) between denoised point clouds and the original surface, and denoising precision. Qualitative assessments were conducted based on experimental testing. To ensure fair comparison, EC-Net utilized its publicly available pre-trained model, and RFEPS retained its original parameters due to identical input dimensions.

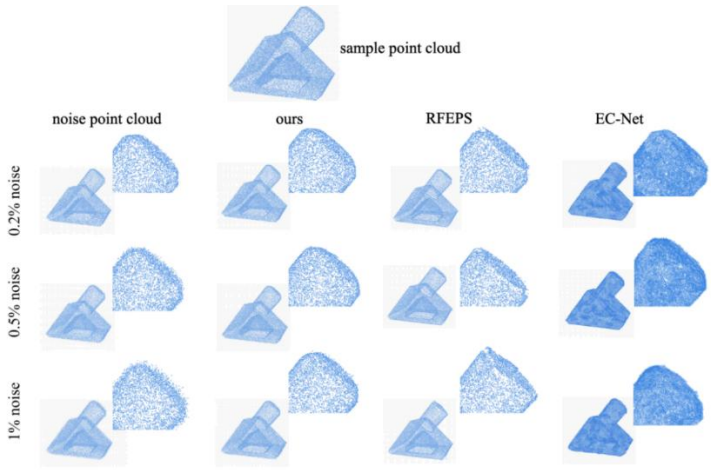

Figure 9: Surface reconstruction results.

The figure demonstrates that the proposed method achieves superior denoising for CAD models, particularly in processing point clouds within sharp feature regions. The core challenge in CAD model point cloud optimization lies in accurately inferring underlying linear features, yet most models struggle to reproduce geometric sharpness, resulting in significant losses at sharp features. Although EC-Net and RFEPS methods also aim to preserve sharp features, they still inadequately handle complex sharp geometric edges. While EC-Net focuses on upsampling tasks where increasing point cloud density can significantly improve the Chamfer Distance (CD) metric, visual comparisons reveal that the newly added points fail to align accurately with the underlying surfaces, especially under noise interference.

In summary, leveraging the structural characteristics of CAD point clouds —where holistic geometric analysis maximizes effective information extraction —significantly mitigates noise impact on reconstructed models. Our algorithm's core innovation lies in obtaining regionally consistent normals via primitive detection. The two-stage denoising and normal optimization process delivers enhanced point cloud restoration quality.

3.3 Feature Detection Quality

This section benchmarks our method against EC-Net [48], PIE-Net [25], RFEPS [46], and DeepSF [13] in detecting and reconstructing sharp features. Ground-truth sharp feature point clouds were generated from triangular meshes: edges where adjacent face normals exceeded $\pi/6$ were classified as sharp, with 10K points uniformly sampled along these edges.

Model	Methods	Ours	RFEPS	DeepSF	PIE-Net	EC-Net
model65	OCD $(\times 10^4)\downarrow$	0.03	13.79	0.89	1.69	0.08
	CD $(\times 10^4)\downarrow$	0.51	8.33	0.52	1.34	8.33
	F1 $\uparrow$	0.9764	0.5517	0.5341	0.4347	0.8541
model80	OCD $(\times 10^4)\downarrow$	0.08	0.37	0.75	1.27	0.45
	CD $(\times 10^4)\downarrow$	0.20	17.95	1.18	0.99	4.35

Model	Methods	Ours	RFEPS	DeepSF	PIE-Net	EC-Net
	F1 ↑	**0.8865**	**0.3966**	**0.5645**	**0.4841**	**0.4067**
model32	**OCD** $(\times 10^4)\downarrow$	**0.81**	**0.25**	**0.75**	**1.27**	**0.45**
	CD $(\times 10^4)\downarrow$	**0.49**	**10.98**	**0.16**	**2.10**	**0.50**
	F1 ↑	**0.9697**	**0.2606**	**0.8615**	**0.5334**	**0.7847**

Note: **bolded** indicates optimal; ↓: smaller values better; ↑: larger values better.

Table 2: Comparison of feature detection values.

As shown in Table 2, our method demonstrates superior representation of sharp features in most models numerically. For model 32 (Fig. 11), the restoration of trapezoidal thread edges into sharp edges results in slightly lower performance on the Chamfer Distance (CD) and Oriented Chamfer Distance (OCD) metrics. Figure 10 illustrates that the enhanced points generated by our method more accurately reflect geometric edges.

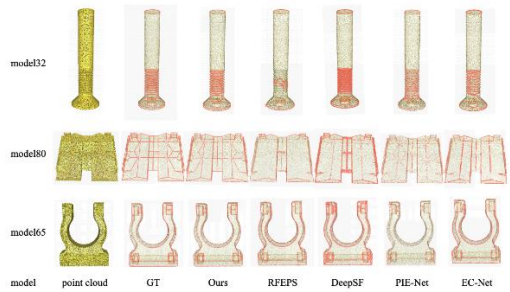

Figure 10: Surface reconstruction results.

Detecting sharp features in a model requires both precise identification of sharp regions and avoidance of misclassifying rounded transition areas as sharp features. The figures demonstrate that our method achieves more accurate detection of sharp feature regions on complex models, producing reinforced linear sharpness. Although EC-Net and DeepSF detect sharp feature regions, their newly added points fail to align precisely with the underlying feature lines. While PIE-Net and RFEPS achieve linear sharpness, they still exhibit errors in detecting sharp features within complex topological regions.

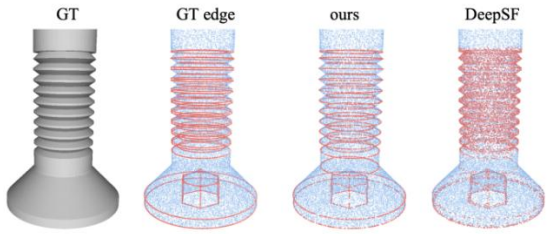

Figure 11: Surface reconstruction results.

In summary, the extensive smooth patches characteristic of CAD point clouds enable effective representation through geometric primitives. Leveraging primitive boundaries as model sharp features, coupled with the reinforcement of linear feature structures, thereby establishes a robust foundation for reconstructing models that faithfully preserve sharp geometric characteristics.

3.4 Surface Reconstruction Quality

This study aims to recover geometric structures from CAD point clouds and generate corresponding meshes, encompassing two critical stages: point cloud optimization and surface reconstruction. Consequently, identifying the optimal "point cloud optimization + surface reconstruction" scheme is essential. Evaluation metrics for different schemes are presented in Table 3.

Method	OCD $(\times 10^3) \downarrow$		F1 ↑		NC ↑	
	model65	model78	model65	model78	model65	model78
noise cloud	2.016	1.672	0.9922	0.9946	0.9749	0.9787
EC-Net	2.966	3.421	0.9653	0.8505	**0.9862**	0.9695
RFEPS	1.942	1.396	0.9875	0.9942	0.9748	0.9847
Ours	**1.908**	**1.293**	**0.9996**	**0.9998**	0.9853	**0.9896**

Note: **Red and bolded** indicate optimal, blue indicates secondary. ↓: smaller values better; ↑: larger values better.

Table 3: Comparison of surface reconstruction quality.

The evaluated surface reconstruction approaches include Screened Poisson Reconstruction (SPR) [22], DeepSF [13], and RFEPS [46], where DeepSF and SPR represent implicit methods while RFEPS employs direct connectivity. The enhanced Restricted Power Diagram (RPD) reconstruction requires base surface support, for which SPR output serves as the experimental base surface (replaceable by alternatives). Crucially, our method assigns special labels to edge points generated during point cloud optimization, enabling higher weight assignment during RPD computation. Without such edge labels, reconstruction defaults to an equal-weight Restricted Voronoi Diagram (RVD).

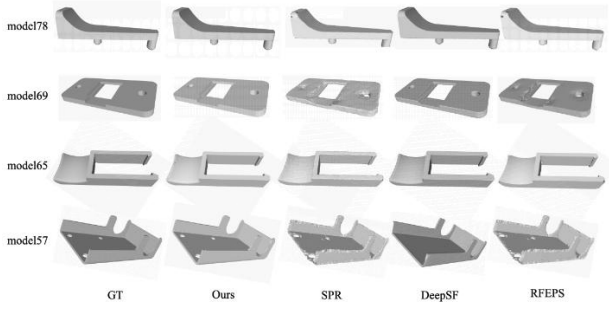

Figure 12: Comparison of surface reconstruction results.

Figure 12 demonstrates our method's superior capability in preserving sharp features during reconstruction, particularly excelling in complex thin-walled structures by prioritizing sharp boundary formation while maintaining smooth planar regions. Quantitative results in Table 3 confirm our surface reconstruction algorithm outperforms alternatives on Model 65 and Model 78. Critical case studies reveal: DeepSF exhibits topological errors in its implicit surface for Model 57, whereas our algorithm preserves significantly sharper features in Model 69.

3.5 Runtime Performance

The algorithm comprises five sequential stages: T1 (point cloud connectivity generation), T2 (geometric primitive detection), T3 (sharp feature generation), T4 (point cloud denoising & normal optimization), and T5 (RPD mesh generation). All stages except T2 exhibit linear time complexity relative to point cloud size, requiring approximately 20 seconds for standard 50,000-point clouds. The T2 geometric primitive detection stage demonstrates exponential time complexity due to its search mechanism, with convergence time positively correlating with model complexity and noise intensity (e.g., ~60 minutes for 50,000-point clouds with 0.2% Gaussian noise).

3.6 Influence of Parameters

Figure 13: Parameter influence.

Figure 13 illustrates the impact of parameters ξ and ω through three configurations: $(\omega = 0.5, \xi = 0.01)$, $(\omega = 1, \xi = 0.01)$, and $(\omega = 1, \xi = 0.1)$. The ξ parameter in Equation (3.1) governs denoising intensity: larger values suppress point displacement while smaller values promote position and normal smoothing. Our experiments adopt $\xi = 0.01$ as default, noting limited sensitivity to ξ variations due to constrained normal optimization. The ω parameter in Equation (3.2) controls weights in the Restricted Power Diagram (RPD), exhibiting minimal impact on smooth regions and quantitative reconstruction metrics, yet contributing to sharp edge preservation.

Parameters	OCD $(\times 10^3) \downarrow$	F1 $\uparrow$	NC $\uparrow$
$\xi = 1.0$	2.455	0.936 4	0.924 8
$\xi = 0.1$	2.359	0.955 2	0.929 2
$\xi = 0.01$	**1.919**	**0.9990**	**0.969 4**
$\xi = 0.0001$	2.617	0.991 0	0.967 0
$\omega = 2.0$	2.156	0.979 1	0.0951 1
$\omega = 1.0$	**1.919**	**0.9990**	**0.969 4**
$\omega = 0.5$	2.526	0.963 3	0.949 7

Table 4: Impact of parameters on reconstruction quality.

3.7 Robustness to Point Density

Most existing reconstruction methods exhibit sensitivity to point density. To validate our method's robustness against density variations, we constructed: Synthetic point clouds with non-uniform density distributions; Sampled versions of identical models at different densities. Experimental results demonstrate that our reconstruction pipeline effectively handles non-uniform sampling while maintaining performance under reduced point densities.

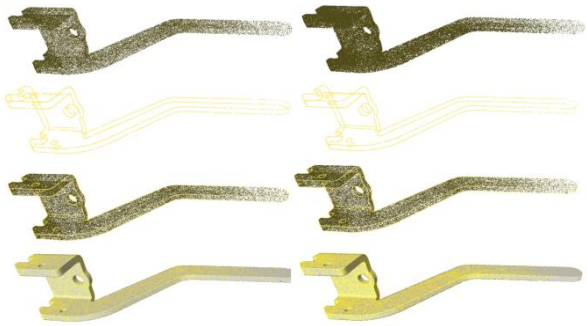

Figure 14: Non-uniform point cloud.

Figure 14 demonstrates algorithm performance on non-uniformly sampled point clouds, confirming robust sharp feature detection and surface reconstruction regardless of sampling uniformity. Similarly, Figure 15 evaluates decreasing point densities (from left to right: 10,000, 5,000, and 2,500 points), showcasing maintained reconstruction accuracy and sharp feature recovery even under sparse sampling conditions.

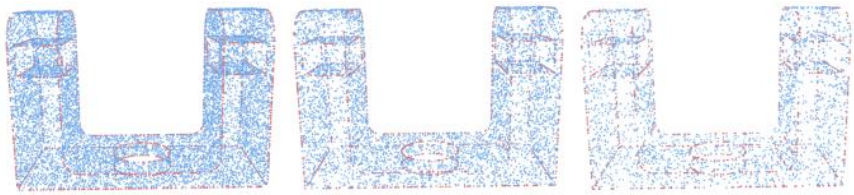

Figure 15: Different density point clouds.

Figure 15 evaluates decreasing sampling densities (left to right: 10,000, 5,000, and 2,500 points). The reconstruction consistently preserves geometric fidelity, accurately recovering sharp features even under sparse sampling conditions.

3.8 Limitation

Dihedral Angle Constraints and Algorithmic Limitations. The current algorithm detects and generates sharp features within dihedral angles of $[\pi/6, 5\pi/6]$. This constraint addresses two critical challenges:

Near-flat features ($\approx \pi$) risk confusion with false primitive intersection lines generated during freeform surface representation. Acute angles ($\approx 0^{\circ}$) and ultra-thin structures may induce cross-surface connection errors in RPD.

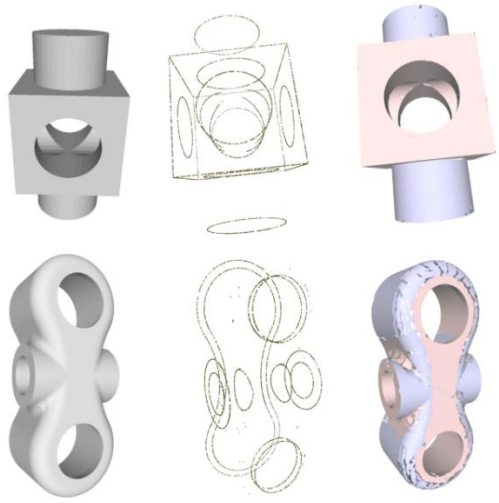

Figure 16: Geometric primitives represent free-form surfaces.

As demonstrated in Figure 16, the exclusion of certain cylindrical intersection ellipses from sharp feature preservation (upper image) prevents reconstruction artifacts shown in the lower figure. Here, primitive-based representation of freeform surfaces produces fragmented intersection lines that shouldn't be classified as sharp features.

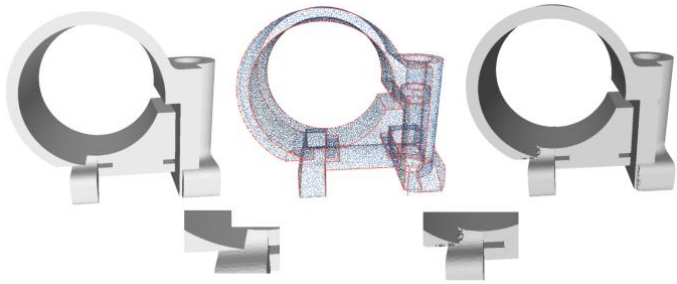

Figure 17: Error representation of extremely thin structure.

RPD Limitations: Figure 17 reveals inherent constraints: while correct surface boundaries are generated at features, acute angles trigger Voronoi polyhedron misalignment during base surface polygon generation, causing erroneous cross-surface connections.

Feature Point Density Constraint: Our feature generation method projects neighborhood points onto feature lines followed by optimization. This approach inherently limits sharp feature densification, preventing high-resolution feature representation. Figure 18 illustrates this bottleneck: insufficient edge points from a 1,000-point input cloud result in compromised boundary reconstruction.

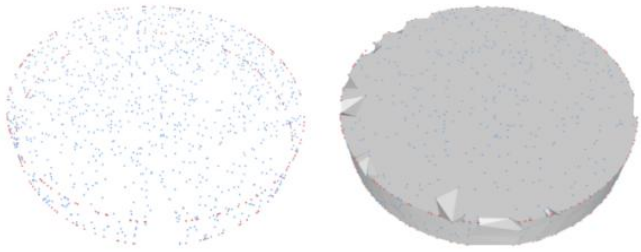

Figure 18: Sharp feature density limitation.

4 CONCLUSIONS

3D reconstruction technology serves as the critical bridge converting unstructured point clouds into functional CAD models. Since CAD model functionality is intrinsically linked to geometric structure, reconstruction inaccuracies may compromise component functionality. As essential geometric information in reconstruction, sharp features remain challenging for existing methods to accurately capture and preserve. To address this fundamental limitation, this paper proposes a primitive detection-based sharp feature detection and surface reconstruction method for high-quality CAD model reconstruction from point clouds.

Our methodology sequentially: (1) generates potential sharp feature regions from primitive boundaries, (2) processes point clouds through two-stage denoising and normal optimization, and (3) optimizes RPD methodology to achieve feature-preserving surface reconstruction.

Algorithmic Innovations: The core contributions comprise three breakthroughs: (1) Sharp feature identification through geometric primitive segmentation, leveraging primitive boundaries to detect features while generating model features via point movement along parametric surfaces; (2) Development of a two-stage denoising and normal optimization algorithm incorporating parametric surface normals and primitive fitting quality assessment, with integrated outlier removal; (3) During surface reconstruction: implementation of RPD to interpolate augmented point sets while prioritizing topological connections between edge points, coupled with optimized RPD weighting for enhanced reconstruction quality.

Our algorithm effectively exploits CAD-specific structural characteristics, demonstrating superior performance in both feature detection and sharp feature-preserving geometric reconstruction compared to state-of-the-art methods.

Bao-Wen Xu, https://orcid.org/0009-0002-2717-3377
Wei-Kang Liu, https://orcid.org/0009-0001-4123-0481
Ji-Xing Li, https://orcid.org/0000-0001-7136-2234
Li-Yong Shen, https://orcid.org/0000-0001-5769-4814

REFERENCES

[1] Alexa, M.; Behr, J.; Cohen-Or, D.; Fleishman, S.; Levin, D.; Silva, C.: Point set surfaces. In Proceedings Visualization, 2001. VIS '01., 21–29, 537, 2001. http://doi.org/10.1109/VISUAL.2001.964489

[2] Atzmon, M.; Lipman, Y.: Sal: Sign agnostic learning of shapes from raw data. In Proceedings of the IEEE/CVF Conference on Computer Vision and Pattern Recognition (CVPR), 2565–2574, 2020.

[3] Basselin, J.; Alonso, L.; Ray, N.; Sokolov, D.; Lefebvre, S.; Lévy, B.: Restricted power diagrams on the gpu. Computer Graphics Forum, 40(2), 1–12, 2021. http://doi.org/https://doi.org/10.1111/cgf.142610.

[4] Bernardini, F.; Mittleman, J.; Rushmeier, H.; Silva, C.; Taubin, G.: The ball-pivoting algorithm for surface reconstruction. IEEE Transactions on Visualization and Computer Graphics, 5(4), 349–359, 1999. http://doi.org/10.1109/2945.817351.

[5] Bochkanov, S.: Alglib, 2023. http://www.alglib.net/. Cross-platform numerical analysis library.

[6] Cai, S.; Ye, Y.; Cao, J.; Chen, Z.: Face: Feature-preserving cad model surface reconstruction. Graphical Models, 136, 101230, 2024. ISSN 1524-0703. http://doi.org/https://doi.org/10.1016/j.gmod.2024.101230.

[7] Chen, Z.; Tagliasacchi, A.; Zhang, H.: Bsp-net: Generating compact meshes via binary space partitioning. In Proceedings of the IEEE/CVF Conference on Computer Vision and Pattern Recognition (CVPR), 45– 54, 2020.

[8] Dey, T.K.; Ge, X.; Que, Q.; Safa, I.; Wang, L.; Wang, Y.: Feature-preserving reconstruction of singular surfaces. Computer Graphics Forum, 31(5), 1787–1796, 2012. http://doi.org/https://doi.org/10.1111/j.1467-8659.2012.03183.x.

[9] Digne, J.; Cohen-Steiner, D.; Alliez, P.; De Goes, F.; Desbrun, M.: Feature-preserving surface reconstruction and simplification from defect-laden point sets. Journal of mathematical imaging and vision, 48, 369–382, 2014.

[10] Du, X.; Zhou, Q.; Carr, N.; Ju, T.: Boundary-sampled halfspaces: a new representation for constructive solid modeling. ACM Trans. Graph., 40(4), 2021. ISSN 0730-0301. http://doi.org/10.1145/ 3450626.3459870.

[11] Edelsbrunner, H.; Shah, N.R.: Triangulating topological spaces. In Proceedings of the Tenth Annual Symposium on Computational Geometry, SCG '94, 285–292. Association for Computing Machinery, New York, NY, USA, 1994. ISBN 0897916484. http://doi.org/10.1145/177424.178010.

[12] Fabri, A.; Pion, S.: Cgal: the computational geometry algorithms library, 2009. http://doi.org/10.1145/1653771.1653865.

[13] Feng, Y.F.; Shen, L.Y.; Yuan, C.M.; Li, X.: Deep shape representation with sharp feature preservation. Computer-Aided Design, 157, 103468, 2023. ISSN 0010-4485. http://doi.org/https://doi.org/10.1016/j.cad.2022.103468.

[14] Gropp, A.; Yariv, L.; Haim, N.; Atzmon, M.; Lipman, Y.: Implicit geometric regularization for learning shapes, 2020. https://arxiv.org/abs/2002.10099.

[15] Groueix, T.; Fisher, M.; Kim, V.G.; Russell, B.C.; Aubry, M.: A papier-mâché approach to learning 3d surface generation. In Proceedings of the IEEE Conference on Computer Vision and Pattern Recognition (CVPR), 216–224, 2018.

[16] Guy, G.; Medioni, G.: Inference of surfaces, 3d curves, and junctions from sparse, noisy, 3d data. IEEE Transactions on Pattern Analysis and Machine Intelligence, 19(11), 1265–1277, 1997. http://doi.org/10.1109/34.632985.

[17] Himeur, C.E.; Lejemble, T.; Pellegrini, T.; Paulin, M.; Barthe, L.; Mellado, N.: Pcednet: A lightweight neural network for fast and interactive edge detection in 3d point clouds. ACM Trans. Graph., 41(1), 2021. ISSN 0730-0301. http://doi.org/10.1145/3481804.

[18] Huang, H.; Ascher, U.: Surface mesh smoothing, regularization, and feature detection. SIAM Journal on Scientific Computing, 31(1), 74–93, 2008. http://doi.org/10.1137/060676684.

[19] Huang, H.; Li, D.; Zhang, H.; Ascher, U.; Cohen-Or, D.: Consolidation of unorganized point clouds for surface reconstruction. ACM Trans. Graph., 28(5), 1–7, 2009. ISSN 0730-0301. http://doi.org/10. 1145/1618452.1618522.

[20] Huang, H.; Wu, S.; Gong, M.; Cohen-Or, D.; Ascher, U.; Zhang, H.R.: Edge-aware point set resampling. ACM Trans. Graph., 32(1), 2013. ISSN 0730-0301. http://doi.org/10.1145/2421636. 2421645.

[21] Kazhdan, M.; Bolitho, M.; Hoppe, H.: Poisson surface reconstruction. In Proceedings of the fourth Eurographics symposium on Geometry processing, vol. 7, 2006.

[22] Kazhdan, M.; Hoppe, H.: Screened poisson surface reconstruction. ACM Trans. Graph., 32(3), 2013. ISSN 0730-0301. http://doi.org/10.1145/2487228.2487237.

[23] Koch, S.; Matveev, A.; Jiang, Z.; Williams, F.; Artemov, A.; Burnaev, E.; Alexa, M.; Zorin, D.; Panozzo, D.: Abc: A big cad model dataset for geometric deep learning. In The IEEE Conference on Computer Vision and Pattern Recognition (CVPR), 9601–9611, 2019.

[24] Li, P.; Guo, J.; Zhang, X.; Yan, D.M.: Secad-net: Self-supervised cad reconstruction by learning sketchextrude operations. In Proceedings of the IEEE/CVF Conference on Computer Vision and Pattern Recognition (CVPR), 16816–16826, 2023.

[25] Li, R.; Li, X.; Heng, P.A.; Fu, C.W.: Point cloud upsampling via disentangled refinement. In Proceedings of the IEEE/CVF Conference on Computer Vision and Pattern Recognition (CVPR), 344–353, 2021.

[26] Li, Y.; Liu, S.; Yang, X.; Guo, J.; Guo, J.; Guo, Y.: Surface and edge detection for primitive fitting of point clouds. In ACM SIGGRAPH 2023 Conference Proceedings, SIGGRAPH '23. Association for Computing Machinery, New York, NY, USA, 2023. ISBN 9798400701597. http://doi.org/10.1145/3588432.3591522.

[27] Liao, B.; Xiao, C.; Jin, L.; Fu, H.: Efficient feature-preserving local projection operator for geometry reconstruction. Computer-Aided Design, 45(5), 861–874, 2013. ISSN 0010-4485. http://doi.org/ https://doi.org/10.1016/j.cad.2013.02.003.

[28] Lipman, Y.; Cohen-Or, D.; Levin, D.; Tal-Ezer, H.: Parameterization-free projection for geometry reconstruction. ACM Trans. Graph., 26(3), 22–es, 2007. ISSN 0730-0301. http://doi.org/10.1145/1276377.1276405.

[29] Liu, Y.; D'Aronco, S.; Schindler, K.; Wegner, J.D.: Pc2wf: 3d wireframe reconstruction from raw point clouds, 2021. https://arxiv.org/abs/2103.02766.

[30] Lu, X.; Schaefer, S.; Luo, J.; Ma, L.; He, Y.: Low rank matrix approximation for 3d geometry filtering. IEEE Transactions on Visualization and Computer Graphics, 28(4), 1835–1847, 2022. http://doi.org/10.1109/TVCG.2020.3026785.

[31] Mérigot, Q.; Ovsjanikov, M.; Guibas, L.J.: Voronoi-based curvature and feature estimation from point clouds. IEEE Transactions on Visualization and Computer Graphics, 17(6), 743–756, 2011. http://doi.org/10.1109/TVCG.2010.261.

[32] Öztireli, A.C.; Guennebaud, G.; Gross, M.: Feature preserving point set surfaces based on non-linear kernel regression. Computer Graphics Forum, 28(2), 493–501, 2009. http://doi.org/https://doi.org/10.1111/j.1467-8659.2009.01388.x.

[33] Park, J.J.; Florence, P.; Straub, J.; Newcombe, R.; Lovegrove, S.: Deepsdf: Learning continuous signed distance functions for shape representation. In Proceedings of the IEEE/CVF Conference on Computer Vision and Pattern Recognition (CVPR), 165–174, 2019.

[34] Park, M.K.; Lee, S.J.; Lee, K.H.: Multi-scale tensor voting for feature extraction from unstructured point clouds. Graphical Models, 74(4), 197–208, 2012. ISSN 1524-0703. http://doi.org/https: //doi.org/10.1016/j.gmod.2012.04.008. GMP2012.

[35] Salman, N.; Yvinec, M.; Merigot, Q.: Feature preserving mesh generation from 3d point clouds. Computer Graphics Forum, 29(5), 1623–1632, 2010. http://doi.org/https://doi.org/10.1111/j.1467-8659.2010.01771.x.

[36] Shen, Y.; Fu, H.; Du, Z.; Chen, X.; Burnaev, E.; Zorin, D.; Zhou, K.; Zheng, Y.: Gcn-denoiser: Mesh denoising with graph convolutional networks. ACM Trans. Graph., 41(1), 2022. ISSN 0730-0301. http://doi.org/10.1145/3480168.

[37] Shewchuk, J.R.: Triangle: Engineering a 2d quality mesh generator and delaunay triangulator. In M.C. Lin; D. Manocha, eds., Applied Computational Geometry Towards Geometric Engineering, 203–222. Springer Berlin Heidelberg, Berlin, Heidelberg, 1996. ISBN 978-3-540-70680-9.

[38] Sitzmann, V.; Martel, J.; Bergman, A.; Lindell, D.; Wetzstein, G.: Implicit neural representations with periodic activation functions. In H. Larochelle; M. Ranzato; R. Hadsell; M. Balcan; H. Lin, eds., Advances in Neural Information Processing Systems, vol. 33, 165–1747462–7473. Curran Associates, Inc., 2020.

[39] Tancik, M.; Srinivasan, P.; Mildenhall, B.; Fridovich-Keil, S.; Raghavan, N.; Singhal, U.; Ramamoorthi, R.; Barron, J.; Ng, R.: Fourier features let networks learn high frequency functions in low dimensional domains. In H. Larochelle; M. Ranzato; R. Hadsell; M. Balcan; H. Lin, eds., Advances in Neural Information Processing Systems, 33, 7537–7547. Curran Associates, Inc., 2020.

[40] Wang, J.; Yu, Z.; Zhu, W.; Cao, J.: Feature-preserving surface reconstruction from unoriented, noisy point data. Computer Graphics Forum, 32(1), 164–176, 2013. http://doi.org/https://doi.org/ 10.1111/cgf.12006.

[41] Wang, P.S.; Liu, Y.; Yang, Y.Q.; Tong, X.: Spline positional encoding for learning 3d implicit signed distance fields, 2021. https://arxiv.org/abs/2106.01553.

[42] Wang, X.; Xu, Y.; Xu, K.; Tagliasacchi, A.; Zhou, B.; Mahdavi-Amiri, A.; Zhang, H.: Pie-net: Parametric inference of point cloud edges. In H. Larochelle; M. Ranzato; R. Hadsell; M. Balcan; H. Lin, eds., Advances in Neural Information Processing Systems, vol. 33, 20167–20178. Curran Associates, Inc., 2020.

[43] Weber, C.; Hahmann, S.; Hagen, H.: Sharp feature detection in point clouds. In 2010 Shape Modeling International Conference, 175–186, 2010. http://doi.org/10.1109/SMI.2010.32.

[44] Williams, F.; Schneider, T.; Silva, C.; Zorin, D.; Bruna, J.; Panozzo, D.: Deep geometric prior for surface reconstruction. In Proceedings of the IEEE/CVF Conference on Computer Vision and Pattern Recognition (CVPR), 10130–10139, 2019.

[45] Xiong, S.; Zhang, J.; Zheng, J.; Cai, J.; Liu, L.: Robust surface reconstruction via dictionary learning. ACM Trans. Graph., 33(6), 2014. ISSN 0730-0301. http://doi.org/10.1145/2661229.2661263.

[46] Xu, R.; Wang, Z.; Dou, Z.; Zong, C.; Xin, S.; Jiang, M.; Ju, T.; Tu, C.: Rfeps: Reconstructing feature-line equipped polygonal surface. ACM Trans. Graph., 41(6), 2022. ISSN 0730-0301. http://doi.org/10.1145/3550454.3555443.

[47] Yan, D.M.; Bao, G.; Zhang, X.; Wonka, P.: Low-resolution remeshing using the localized restricted Voronoi diagram. IEEE Transactions on Visualization and Computer Graphics, 20(10), 1418–1427, 2014. http://doi.org/10.1109/TVCG.2014.2330574.

[48] Yu, L.; Li, X.; Fu, C.W.; Cohen-Or, D.; Heng, P.A.: Ec-net: an edge-aware point set consolidation network. In Proceedings of the European Conference on Computer Vision (ECCV), 2018.

[49] Zhang, W.; Deng, B.; Zhang, J.; Bouaziz, S.; Liu, L.: Guided mesh normal filtering. Computer Graphics Forum, 34(7), 23–34, 2015. http://doi.org/https://doi.org/10.1111/cgf.12742.