MSL-Net: Sharp Feature Detection Network for 3D Point Clouds

Xianhe Jiao†, Chenlei Lv†, *Member, IEEE*, Ran Yi, Junli Zhao*, *Member, IEEE*, Zhenkuan Pan, Zhongke Wu, Yong-Jin Liu*, *Senior Member, IEEE*

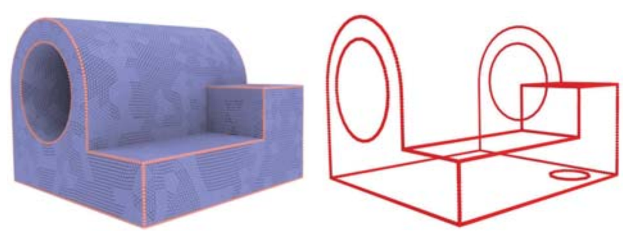
Abstract—As a significant geometric feature of 3D point clouds, sharp features play an important role in shape analysis, 3D reconstruction, registration, localization, etc. Current sharp feature detection methods are still sensitive to the quality of the input point cloud, and the detection performance is affected by random noisy points and non-uniform densities. In this paper, using the prior knowledge of geometric features, we propose a Multi-scale Laplace Network (MSL-Net), a new deep-learning-based method based on an intrinsic neighbor shape descriptor, to detect sharp features from 3D point clouds. Firstly, we establish a discrete intrinsic neighborhood of the point cloud based on the Laplacian graph, which reduces the error of local implicit surface estimation. Then, we design a new intrinsic shape descriptor based on the intrinsic neighborhood, combined with enhanced normal extraction and cosine-based field estimation function. Finally, we present the backbone of MSL-Net based on the intrinsic shape descriptor. Benefiting from the intrinsic neighborhood and shape descriptor, our MSL-Net has simple architecture and is capable of establishing accurate feature prediction that satisfies the manifold distribution while avoiding complex intrinsic metric calculations. Extensive experimental results demonstrate that with the multi-scale structure, MSL-Net has a strong analytical ability for local perturbations of point clouds. Compared with state-of-the-art methods, our MSL-Net is more robust and accurate. The code is publicly available at https://github.com/XianheJiao/Sharp-feature-detection-in-point-cloud.

Index Terms—Sharp feature, 3D point cloud, intrinsic neighbor, multi-scale Laplace network.

1 Introduction

With the development of 3D scanning technology, 3D point clouds are widely collected and gradually becoming one of the most popular data representations in 3D vision tasks. As an important geometric feature in 3D point clouds, sharp features are useful in various applications, including 3D reconstruction, localization, registration, visualization, etc. From the perspective of manifold distribution, sharp features describe the areas in point clouds where the curvature changes abruptly or discontinuously. Such property supports precise semantic feature descriptions for calculations in reconstruction and location. In embedding spaces, compared to other regions, sharp features can represent more prominent geometric details while conforming to human subjective perception in visualization, as shown in Fig. 1. Therefore, sharp feature detection is an important task in point-cloud-based analysis.

To detect sharp features, some geometry-based rules are used to guide the detection framework in traditional solutions [1] [2]. For instance, the edge with sharp features shows the rapid change of normal-vector-based angles in the local region. Once such rules are formulated quantitatively, the detection can be processed by algorithms. Some local shape descriptors, such as normal vectors and curvatures, provide measurement tools. However, the quality of scanned point clouds from real scenes can not support accurate local-region-based analysis and geometric feature extraction. Limited by the performance of scanning devices, randomly noisy points and non-uniform densities in different regions are unavoidable, which have an unpredictable impact on sharp feature detection.



(a) Joint Model (b) Sharp Features

Fig. 1. Instance of shape feature detection for Joint model.

Following the development of deep learning technologies, some researchers propose learning-based frameworks [3] [4] to improve performance. Such frameworks fully utilize the feature learning ability of deep neural networks to extract structured knowledge from training datasets. Then, more complex semantic information establishes efficient and robust sharp feature detection rules. However, the local neighbor detection of these methods does not often follow the manifold constraints, which re-

• *Xianhe Jiao, Junli Zhao and Zhenkuan Pan are with the College of Computer Science and Technology, Qingdao University, Qingdao, China. Chenlei Lv is the College of Computer Science and Software Engineering, Shenzhen University, Shenzhen, China. Ran Yi is department of Computer Science and Engineering, Shanghai Jiao Tong University, Shanghai, China. Zhongke Wu is School of Artificial Intelligence, Beijing Normal University, Beijing, China. Yongjin Liu is BNRist, MOE-Key Laboratory of Pervasive Computing, Department of Computer Science and Technology, Tsinghua University, Beijing, China. Xianhe Jiao and Chenlei Lv contributed equally to this work. Junli Zhao and Yong-Jin Liu are co-coressponding authors. E-mail:(zjl@qdu.edu.cn, liuyongjin@tsinghua.edu.cn).*

Manuscript submitted XXX XX, 2023

duces the detection performance. I.e., these methods focus more on overall edge accuracy but are less sensitive to the local geometric features. Furthermore, the performance of these methods is affected by nonuniform point distribution and noisy points. Therefore, most current deep learning methods cannot achieve stable, accurate, and robust sharp extraction.

To solve the above challenges, we propose a novel point-cloud-based sharp feature detection method, MSL-Net, combining traditional geometric analysis and deep learning. It includes three core parts: discrete intrinsic neighbor (DIN) detection, intrinsic shape operator, and multi-scale Laplace network. The DIN detection aims to search intrinsic neighbors under the geodesic distance metric according to the manifold constraint. It has been shown that the intrinsic neighbors can improve local region representation [5]. The intrinsic neighbors reduce the probability of misidentifying points with small Euclidean distances produced by sharp curvature changes. Based on the intrinsic neighbors, we present an intrinsic shape operator to describe local shape features. This operator makes full use of the geometric homogeneity of the intrinsic neighbors, combines with the cosine field positioning function to provide accurate and robust feature representation based on normal vectors, and improves the sensitivity to sharp features. Finally, we design the backbone of MSL-Net to learn the rules of sharp features with different conditions. With the help of intrinsic neighbors and operators, MSL-Net can be quickly converged. Benefiting from the intrinsic neighborhood and shape descriptor, our proposed MSL-Net has a simple architecture, only requires a few MLP layers while significantly improving the detection accuracy and robustness. The overall pipeline is shown in Fig. 2. The main contributions of the paper are as follows:

- We present a discrete intrinsic neighbor detection for point clouds. It improves the accuracy of neighbor detection without complex geodesic computation and Voronoi-cell-based analysis.
- We propose an intrinsic shape operator to describe the local shape feature. The operator fully considers normal vector distributions based on the intrinsic neighbors. It provides accurate representations for geometric details, making subsequent feature analysis easy.
- We design a multi-scale Laplace network to learn the sharp features from intrinsic shape operators. The network has a multi-channel structure for feature learning in different scales of local regions. It supports accurate sharp feature detection and improves the robustness of noisy point clouds.

The rest of this paper is organized as follows. In Sec. 2, we summarize representative methods for sharp feature detection. In Sec. 3, we describe the design of DIN detection. The intrinsic shape operator and backbone of MSL-Net are introduced in Secs. 4 and 5, respectively. We evaluate the performance of our method and present the comparisons with different measurements in Sec. 6. Finally, we conclude our work in Sec. 7.

2 Related Works

Sharp feature detection methods can be roughly divided into two classes: local-shape-descriptor-based and data-driven-based. For the first class, the main idea is to establish the formulation for the sharp feature based on the related geometric information represented by local shape descriptors. Such descriptors are often constructed with the aid of normal vectors or curvatures, which represent the local shape features. For the second class, the core issue is to train a learning model from the collected data for sharp feature estimation.

Local-shape-descriptor-based methods attempt to formulate sharp features based on geometric information. Pauly *et al.* [6] proposed a PCA-based multi-scale feature extraction to fit sharp lines of point-sampled surface. Xia *et al.* [7] extracted edges by analyzing the ratio between eigenvalues of local point sets. In addition to the above methods, which extracted sharp features from a statistical perspective, more methods extracted features from a geometric perspective. Lin *et al.* [8] established the Line-Segment-Half-Planes (LSHP) structure for point-cloud-based line segments. Hackel *et al.* [9], [10] used graph-based methods for structured edge extraction by extending local sharp feature detectors through global analysis.

Many works focused on normal features to extract sharp features. Mérigot *et al.* [1] utilized normal-vector-based distributions in local Voronoi cells to extract sharp edges from point clouds. Demarsin *et al.* [11] employed a first-order segmentation to extract candidate feature points and reconstructed the sharp lines by a graph. Li *et al.* [12] improved normal estimation for point location in high curvature regions or complex sharp features, and a similar solution was proposed in [13]. Weber *et al.* [14] calculated a Gaussian graph of the samples using normal vectors, which is used to identify sharp features in local regions. Zhang *et al.* [15] proposed a pair consistency voting scheme to estimate normal vectors while preserving sharp features.

In summary, the above methods extract local shape descriptors from point clouds without complex semantic analysis and pre-training. The implementation is relatively concise, and the performance is stable. However, the accuracy of these methods is influenced by the quality of point clouds. Once the non-uniform densities and noisy points have a high proportion in the point cloud, the extracted local shape descriptors may lose the function for sharp feature detection.

Data-driven-based methods try to learn the latent features from training data for sharp feature detection. With the development of deep learning, such methods achieved increasing attention. Based on classic CNN networks, several methods have been proposed. ”Feng *et al.* [16] utilized the U-Net architecture in combination with attention mechanism for edge point classification. Subsequently, they employed bilateral high-pass filtering to filter the edge points, which can effectively represent the overall characteristics of the model. Raina *et al.* [17] proposed a CNN-based structure to predict the sharpness field (ShF) for edge point classification. In the same way, Himeur *et al.* [18] trained a CNN to learn the description of edges and use it to efficiently detect edges in 3D point cloud. Matveev *et al.*

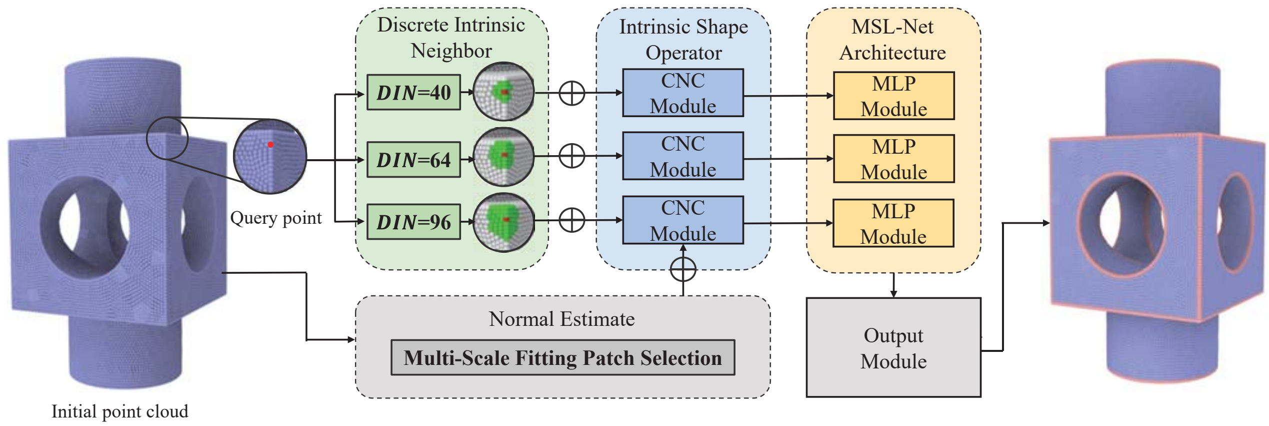


Fig. 2. The pipeline of our method. Three core components: discrete intrinsic neighbor detection, intrinsic shape operator, and MSL-Net Architecture. The discrete intrinsic neighbor detection is used to extract intrinsic neighbors for each point. The intrinsic shape operator describes the local shape feature based on the intrinsic neighbors. The MSL-Net Architecture learns the sharp feature detection model based on the intrinsic shape operators.

[3] also proposed a CNN-based network, DEF-Net, to detect and enhance edge points. Loizou $et\ al.$ [19] constructed a graph convolutional network architecture for parts' boundary detection from point clouds.

A few methods constructed neural networks using point clouds as input for extracting sharp features. Yu $et\ al.$ [4] designed an edge-aware network (EC-Net) based on the upsampling framework [20], which sampled the point cloud and regressed the distance from each point to the edge curve. Wang $et\ al.$ [21] proposed an end-to-end learnable network, PIE-Net, for parametric inference of edges. The network is trained based on PointNet++ [22] to implement edge and corner point classification. Zhang $et\ al.$ [23] proposed a denoising framework with an encoder and a decoder structure for sharp feature preserving. Edirimuni $et\ al.$ [24] designed a deep-learning-based method to filter point clouds while keeping sharp features. Zhao $et\ al.$ [25] enhanced the noisy robustness by estimating displacement vectors according to the training dataset. Zhu $et\ al.$ [26] employed the backbone of PointNet++ to encode point features for sharp feature detection. Himeur $et\ al.$ [18] proposed to formulate edge detection as a classification task and utilize neural networks to learn it. Cherenkova $et\ al.$ [27] achieved edge point detection and line fitting through a network to effectively realize clear and continuous edge classification.

The above mentioned methods fully utilized the advantages of deep learning to learn sharp features from point clouds. However, most of them implement feature concentration based on the Euclidean space but not the manifold space. Moreover, the k-nearest neighbor (KNN) detection extracts unstable neighborhood relations between points in local regions, which reduces the accuracy for sharp feature detection. In addition, edge extraction requires both local accuracy and global consistency. It is difficult to achieve this goal with a single-scale network architecture.

In this paper, we propose a new solution that combines the advantages of the two classes. It extracts the intrinsic neighbors to fit the manifold surface and defines the intrinsic local shape descriptors. Using a multi-scale Laplace network, the descriptors are further trained to formulate a judgment for sharp features.

3 Method

3.1 Overview

We propose a new discrete intrinsic neighbor (DIN) detection for point clouds (Section 3.2). Based on the intrinsic neighbors, we present an intrinsic shape operator to describe the local shape feature (Section 3.3). Finally, we design the backbone of MSL-Net to learn the rules of sharp features with different conditions (Section 3.4). The pipeline of our method is shown in Fig. 2.

3.2 Discrete Intrinsic Neighbor Detection

As aforementioned, the local neighbor detection on point clouds should fit the manifold constraint. If there is an implicit surface corresponding to a point cloud as a continuous form, the manifold constraint means that the point-based distance should be defined "on" the surface, which is consistent with the first fundamental form. It can be represented as

$$d(p_i, p_j) = \int_0^1 \sqrt{E(\frac{du}{dt})^2 + 2F\frac{du}{dt}\frac{dv}{dt} + G(\frac{dv}{dt})^2}\,dt, \quad (1)$$

where E, F, and G are second partial derivatives of a curve parametric function with respect to the u and v that are directions in the parameter domain, $p_i = (u(0), v(0)), p_j = (u(1), v(1))$. Once the curve parametric function is provided based on the point cloud, the distance can be computed that satisfies the manifold constraint. In general, the curve parametric function is defined by the geodesic path, and the intrinsic neighbor is detected based on the geodesic distance. It is shown in [5] that the performance of intrinsic neighbor detection is good in point cloud-based applications, including simplification, resampling, and reconstruction. However, the implementation of intrinsic neighbor detection is complicated in practice. Due to the significant computational cost and sensitivity to the quality of the

This article has been accepted for publication in IEEE Transactions on Visualization and Computer Graphics. This is the author's version which has not been fully edited and content may change prior to final publication. Citation information: DOI 10.1109/TVCG.2023.3346907

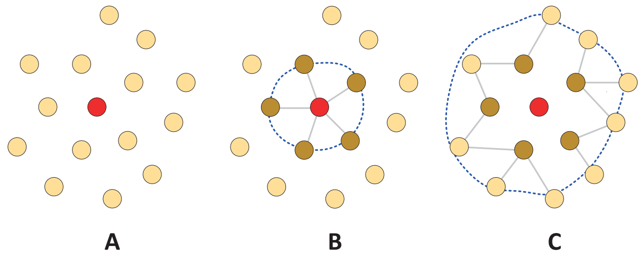


Fig. 3. Instance of iterative neighbor searching strategy. A: initial state; B: 1-ring neighborhood L_1 of source point (red); C: iterative neighbor searching for L_2. Blue dashed circles represent the wavefront for different values of σ, which is the scale of the neighborhood.

target point cloud, calculating the geodesic path is time-consuming and difficult. Although some methods have simplified the computation of geodesic, the time cost is still relatively high. Inspired by the Laplace graph theory [28] and the fast marching algorithm [29], we propose a new discrete intrinsic neighbor (DIN) detection for point clouds. This detection can be regarded as an iterative neighbor searching strategy based on the Laplace graph. The neighbor searching process adopts the core idea of fast marching [29] to keep an interface of the wavefront. The advantage is that it does not require computing geodesic distance or Voronoi cells to implement intrinsic control. The implementation of the search process is simple and efficient while satisfying the manifold constraint.

Let P represent the input point cloud, p_i is a point of P, $L(p_i)_\sigma$ is the intrinsic neighborhood of p_i, σ is the scale of the neighborhood. Once the σ is provided, the neighbor detection is implemented iteratively as

$$L(p_i)_\sigma = L(p_i)_{\sigma-1} + \{p_i\}_\sigma, \text{ and} \tag{2}$$

$$\{p_i\}_\sigma = \{p_j | p_j \in \sum_{p_l \in \{p_i\}_{\sigma-1}} L(p_l)_1, p_j \notin L(p_i)_{\sigma-1}\}, \tag{3}$$

where $\{p_i\}_\sigma$ is the discrete form for interface of wavefront, which is collected from the adjacent neighborhood L_1 (defined by Voronoi cell in tangent space, it can be approximated as a k-neighbor region, $k = 6$) of the previous wavefront $\{p_i\}_{\sigma-1}$. In this way, the detection is an iterative search based on the previous neighborhood, which roughly simulates the propagation process of the wave equation. The initial neighbors $L(p_i)_1$ or wavefront $\{p_i\}_1$ can be directly extracted from adjacency based on the Laplace graph. An example is shown in Fig. 3.

It is worth noting that the quality of $L(p_i)$ is affected by the point cloud density. Non-uniform densities take anisotropic neighbors that reduce the performance of our detection. To address this challenge, we implement the isotropic simplification [30] before the neighbor detection. Such pre-processing has two advantages. First, the number of $L(p_i)_1$ is clear (i.e., $|L(p_i)_1| \approx 6$) in an isotropic point cloud, which is guaranteed by the equilateral relationships between neighbors. Second, the isotropic property makes the iterative searching close to geodesic computation while avoiding complex weight calculations [31]. The reason is that the weights between the point and its neighbors in an isotropic point cloud can be represented by the distance directly. The update of σ explicitly reflects the distance change. Based on the above algorithmic considerations, we present the implementation details for DIN detection in Algorithm 1.

Algorithm 1: DIN Detection

Input : Raw Point cloud P with σ
Output: $L(p)_\sigma$ for each point

```
Pre-processing P with n points by [30]
Adjacent neighbor detection for each point by KNN, k = 6
Initial Laplace adjacency matrix Lg(P) for P, the scale of the matrix is n × n. All values of Lg(P) are assigned zero.
if points p_i and p_j are adjacent points according to the adjacent neighbor detection result, set Lg(P)_ij = 1 and Lg(P)_ji = 1,
for p_i ∈ P do
    for h ∈ σ do
        Search {p_i}_h by Eq.3 with Lg(P)
        Add {p_i}_h into L(p_i)_σ
Achieve L(p)_σ for each point
```

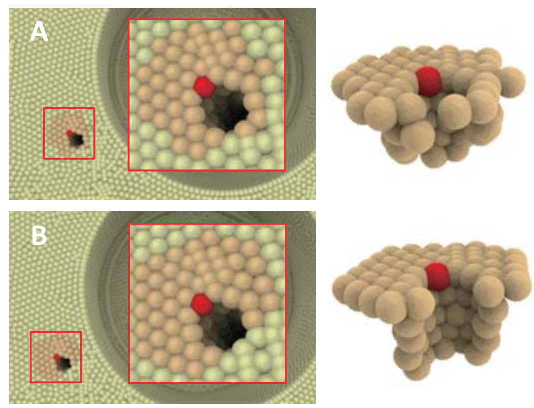


Fig. 4. Comparisons between KNN-based neighbors (A) and DIN-based neighbors (B). KNN-based neighbors cover some unrelated regions that reduces the accuracy for local region representation. The DIN-based neighbors represent more adjacent geometric details according to the manifold constraint.

Using different values of σ, we can extract multi-scale intrinsic neighborhoods useful for subsequent feature learning. For deep network training, the number of neighbors should be controlled by a uniform value. To implement the accurate control, we search the neighbors ($L(p_i)_\sigma$) according to the σ. Once the point number of $L(p_i)_\sigma$ is larger than the specified value, the searching process stops. We randomly delete some points from the latest wavefront $\{p_i\}_\sigma$ to achieve neighbors with the uniform value. As shown in Fig.4, the DIN-based neighbors are better than KNN-based neighbors for local implicit surface modeling. In practice, we specify the multi-scale number $\sigma = \{3, 4, 5\}$ to define the neighborhoods. According to the different values of σ, we estimate the related neighbor point numbers $\{40, 64, 96\}$ as the uniform values for related $L(p_i)_\sigma$.

This article has been accepted for publication in IEEE Transactions on Visualization and Computer Graphics. This is the author's version which has not been fully edited and content may change prior to final publication. Citation information: DOI 10.1109/TVCG.2023.3346907

3.3 Intrinsic Shape Descriptor

To describe the sharp feature, a proper local shape operator is useful. It can be regarded as a kind of formulation for geometric information of the local region, as presented in Eq. (6). In the previous section, we show that the KNN-based neighbors cannot provide accurate local implicit surface modeling (ref. Fig. 4), which degrades the performance of related local shape operators. We propose the following intrinsic shape descriptor as a new local shape operator. Benefited from the intrinsic neighbors, the intrinsic shape descriptor is consistent with the manifold constraint, which provides a more accurate formulation for the local geometric structure. It inherits the advantages of intrinsic neighborhoods and achieves a good trade-off between manifold consistency and computational efficiency.

Basically, the intrinsic shape descriptor is a normal-vector-based shape operator consisting of a set of cosine values based on normal intersection angles. The cosine value can be regarded as a simple discrete curvature related to the mean curvature. The computation of its value is formulated as

$$CNC_{p_i} = cos(N_{p_i}, N_{p_j}), \tag{4}$$

where CNC_{p_i} represents the cosine normal curvature of point p_i, p_j is one of the intrinsic neighbor points of p_i, N_{p_i} is the normal vector of p_i, and N_{p_j} is the normal vector of p_j. We compute the absolute cosine value of the intersection angle between the normal vectors, which can be regarded as a concise encoding for the local implicit surface. Since normal estimation has a crucial impact on the calculation of the descriptor, we employ the advanced normal calculation method Multi-Scale Fitting Patch Selection (MFPS) [32] to estimate normal vectors.

Even if we obtain accurate normal vectors in local regions, they are not globally oriented, which reduces the performance of the related shape descriptor. Without global orientation, the normal vectors may generate conflicting shape descriptors. As shown in Fig.5, the normal vector of points B and C will point towards different directions. This will result in a sudden change in the $\cos\beta$, causing the classifier to falsely detect a sharp change in the normal direction, which contradicts the fact that the curve is smooth and continuous at that point. For the curve (B, C), an angle γ between the negative normal vector of B and the normal vector of C should reflect the change in the normal direction of this curve correctly. Inspired by Sharpness fields [17], we use the absolute value of cosine to represent the normal vector-based curvature. Fortunately, the absolute values of cosine from these two formulations are the same, shown as

$$|\cos\beta| = |\cos\gamma|, \tag{5}$$

where β, and γ represent intersections in the above two formulations, and they have the same absolute value of cosine that can be used to represent normal-vector-based curvature. Based on the above reason, we define the new representation of *CNC*, formulated as

$$CNC_{p_i} = |cos(N_{p_i}, N_{p_j})|. \tag{6}$$

Once the computation of *CNC* is provided, the intrinsic shape descriptor can be established. The descriptor is a set of *CNC*s extracted from the intrinsic neighborhood $L(p_i)_\sigma$.

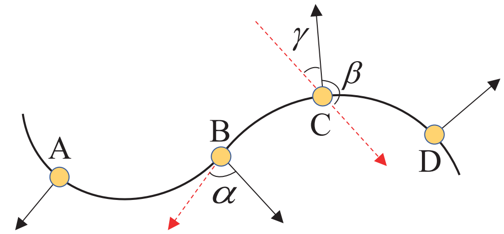


Fig. 5. Instance of normal vectors. α is the angle between the normal vectors of A and B; β is the angle between the normal vectors of B and C; γ is the angle between the negative normal vector of B and the normal vector of C.

It can be regarded as a concise representation of the local region, similar to fast point feature histograms (FPFH) [33]. The advantages of the descriptor include convenient serialization, orientation independence, and facilitating the established quantitative analysis. Even if the descriptors indicate rough geometric features, their advantages are still important for subsequent training tasks. With the intrinsic shape descriptor, we can use a simple network structure as presented in Section 3.4 to learn an accurate and robust sharp feature model.

3.4 Multi-Scale Laplace Network

Due to the inherent diversity of point clouds, different types of sharp features have significantly different characteristics. To formulate the sharp features, suitable neighborhoods need to be detected to achieve local shape analysis. In some traditional solutions, the neighborhood scale is fixed, which may reduce the flexibility for sharp feature learning. Considering the successful application of shifted window techniques in the field of image processing [34], we propose the multi-scale Laplace network (MSL-Net) as the learning model. It is a multi-channel deep network constructed by several MLP layers and a weighted average pooling module for final probability estimation. Each channel corresponds to a specified scale that simulates the shifted window across the image. Collecting all features from different channels, the MSL-Net can estimate the probability of sharp feature judgment. Benefited from the multi-scale analysis, the estimation is more accurate than single-scale networks.

Multi-Scale Input. As aforementioned, we can use *CNC* to describe the local shape feature. For MSL-Net training, the multi-scale input is based on the collected *CNC* with different scales of intrinsic neighborhoods. It is represented as

$$MSLN_{input} = \{MSLN_{\sigma_1}, ..., MSLN_{\sigma_f}\}, \sigma_1 < ... < \sigma_f, \tag{7}$$

$$MSLN_\sigma = \{CNC_{p_j} | p_j \in L(p_i)_\sigma\}, \tag{8}$$

where the $MSLN_\sigma$ is a set of *CNC* extracted from the intrinsic neighborhood $L(p_i)_\sigma$. For each point in the cloud, we establish the $MSLN_{input}$ with different values of σ. In practice, we specify the channel number $f = 3$. As mentioned in Sec. 3, we can specify an accurate number of neighbor points by deleting some points to control the number in the range of $(|L(p_i)|, |L(p_i)_{\sigma-1}|)$. Then, we can specify $MSLN_\sigma$ with a

This article has been accepted for publication in IEEE Transactions on Visualization and Computer Graphics. This is the author's version which has not been fully edited and content may change prior to final publication. Citation information: DOI 10.1109/TVCG.2023.3346907

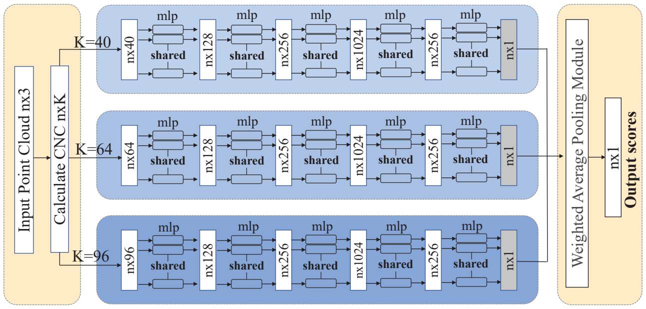


Fig. 6. **MSL-Net Architecture.** It can be regarded as a multi-channel classification network for sharp feature detection. As the input feature, the multi-scale input includes three *CNC*-based vectors that have been serialized. The three vectors are respectively inputted into three channels and transformed by a series of MLP layers. Then, the MSL-Net obtains three feature vectors of reduced dimensions. Finally, the classification is implemented based on the vectors by average pooing.

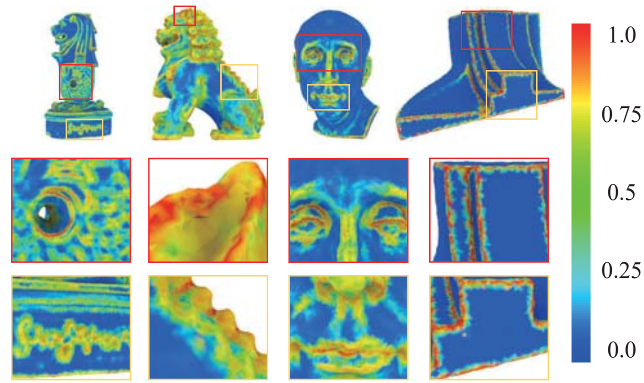


Fig. 7. Visualizations of sharp feature probability estimation by color maps.

certain scale convenient for subsequent training. We set the multi-scales as 40, 64 and 96. The quantitative analysis for the scale selection is provided in Section 4.

Serialization. The disorder problem should be solved for point cloud-based deep learning. The classical solution is to implement max pooling [35] for the feature vector. However, such implementation neglects more geometric details. Therefore, we implement the serialization to solve the problem. We consider two kinds of serialization, which include *CNC*-based and Euclidean distance-based serializations. For the first one, the input *CNC*-based feature vector is sorted based on their values. It can be regarded as a statistical analysis according to the curvatures in the related DIN region. Such serialization lost the local spatial correspondence to achieve better robustness for non-uniform densities. The Euclidean distance-based serialization keeps the local spatial correspondence to a certain degree. However, it is sensitive to the density. Compared to the max pooling, the Euclidean-distances-based serialization preserves more geometric information. Although the serialization disrupts the accurate point-based correspondence of *CNC* values according to the σ, it still has significant statistical significance and keeps rough correspondence. Some details are discussed in Section 4.4.

Network Architecture. The proposed architecture of MSL-Net is shown in Fig. 6. The multi-scale input vectors extracted from intrinsic neighborhoods with three scales are inserted into their corresponding channels. Inspired by the PointNet [35], MLP layers are used to perform dimension expansion and reduction on the *CNC*-based feature vectors. Each MLP layer can be regarded as a cross-analysis of the *CNC*-based neighborhood for a point, which represents the statistical shape feature. Since the initial vector has already been serialized, there is no need to use a max pooling layer when reducing the dimension of features, which maximally preserves the geometric features in the training data. Finally, we obtain three vectors whose dimensions have been reduced.

Probability Estimation. Based on the output feature vectors from the three channels, we estimate the probability of sharp feature judgment. A weighted average pooling module is used to combine the three vectors into a single one. The probability estimation is implemented based on the combined feature vector. In this paper, the MSL-Net classification threshold is set to 0.6. The judgment threshold of probability is trained to fit the training data, which can also be regarded as the output score for sharp feature estimation. Once the threshold is set, the construction of MSL-Net is completed. To facilitate a clear illustration of the relationship between probability estimation and sharp features, we visualize the probabilities by color maps as shown in Fig.7. It is clearly observed that the probability estimation successfully quantifies sharp features.

Loss function. The loss function uses the cross-entropy function to implement binary classification of the network, which is represented as

$$Loss = -\sum_{i=1}^{q} y_i \log(\overset{\wedge}{y_i}), \tag{9}$$

where y_i represents the conditional probability of any input being a sharp feature point, $\overset{\wedge}{y_i}$ represents one-hot label vector, and q represents the number of classification categories. In Section 4, we evaluate the performance of MSL-Net and the functions of its modules.

4 Experiments

We evaluate the performance of MSL-Net in this section. All experiments are run on the computer equipped with AMD Ryzen 7 5800H, 16GB RAM, RTX3060, and with windows 11 as its running system and pycharm as the development platform. The learning rate of the network is set to 10^{-4}, and the optimizer is selected for Adam for model optimization iteration. The experiments include the following parts: 1) we explain the basic introduction to the ABC dataset; 2) we introduce the selected quantitative analysis tools for quality measurement of sharp feature detection; 3) we discuss the ablation study for the MSL-Net to demonstrate the effects of different modules and parameters; 4) we provide a comprehensive analysis with existing limitations.

4.1 ABC Dataset

We conducted experiments on ABC Dataset [36] that is widely used in sharp feature estimation. The ABC

This article has been accepted for publication in IEEE Transactions on Visualization and Computer Graphics. This is the author's version which has not been fully edited and content may change prior to final publication. Citation information: DOI 10.1109/TVCG.2023.3346907

TABLE 1
Seven datasets were used to compare the performance of different sharp feature detection methods.

$\mathbf{A_s}$	selected subset of ABC with more accurate sharp features with 1000 point clouds
$\mathbf{A}_{all}$	original ABC with 1500 sets of data
$\mathbf{A_g}$	A_s add Gaussian noise with 0.12% noisy intensity
$\mathbf{A_{g2}}$	A_s add Gaussian noise with 0.2% noisy intensity
$\mathbf{A_{g3}}$	A_s add Gaussian noise with 0.3% noisy intensity
Sparse	1000 models from the ABC [36] dataset and implement down-sampling [37]
Real	125 real scanning point clouds provided by reference [3]

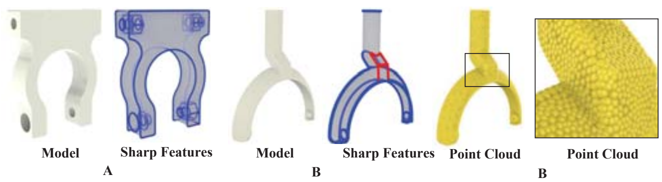


Fig. 8. Visualizations of sharp feature labels of point clouds from the ABC dataset. The blue labels represent the ground truth of sharp features provided by the ABC dataset. The red labels represent the sharp features that are not labeled as the ground truth. It means that the ground truth labels of the ABC dataset have flaws.

dataset [36] is collected with one million computer-aided design (CAD) models for research of geometric deep learning and related applications. Each model is established by parametrized curves and surfaces that provide ground truth for geometric feature detection. The surfaces mainly include planes, cylinders, cones, spheres, torus, surface of revolution or extrusion, and NURBS patches [38]. The parametrized curves mainly include lines, circles, ellipses, parabolas, hyperbolas, or NURBS curves [38]. In this paper, we consider the parametrized curves of the CAD model as its sharp features.

The formats of the dataset include various types, including step, parasolid, yml, stl, obj, etc. Using files with original parameter representations (step, parasolid) makes it difficult to construct large training datasets because boundary representation (B-Rep) requires the use of off-the-shelf geometric kernels (Open Cascade [39]), which is not designed for batch processing. To avoid these issues, we use yml files for sharp feature extraction. The yml files include the type of parametrized curve that is a kind of sharp feature representation and the index of each curve corresponding to its points. With the curve-based sharp feature labels, we can classify the point cloud into two types: sharp points and normal points.

Due to the fact that the CAD models in the ABC dataset are automatically generated, the sharp feature ground truth provided by the ABC dataset is not comprehensive, as shown in Fig. 8. Some sharp points are not labeled, which leads to defects in the ground truth calibrated based on the parametrized curves. In order to establish more convincing experimental tests, we selectively extract a subset from the ABC dataset with accurate labels. The test results are reported based on the subset and the original one at the same time.

On the ABC dataset, all point cloud models are noise-free with relatively uniform distributions. The training dataset is collected by 100 models with clear edges of different types (straight lines, spline curves, circles, etc.) from the ABC dataset. We also add Gaussian noise with an intensity of 0.12% times the diagonal length to extend the diversity. During the training process, it is necessary to consider the ratio of positive and negative samples. The reason is that the proportion of positive edge points is lower in general. Without the sample balance, the training process will be biased towards more negative points or normal samples. We select all edge points as positive samples and randomly choose normal points as negative ones from models.

We have conducted performance tests on seven datasets, which include A_s(selected subset of ABC with more accurate sharp features with 1000 point clouds), A_{all}(original ABC with 1500 sets of data), we adopt a Gaussian distribution with a mean of 0.12%, 0.2% and 0.3% of the diagonal length of the bounding box and a standard deviation of one to generate N (number of points) random numbers for the X, Y, and Z axes, respectively. Then, we add the random numbers to the point cloud coordinates to generate a point cloud with Gaussian noise. A_g, A_{g2}, A_{g3}(A_s add Gaussian noise with 0.12%, 0.2%, 0.3% noisy intensity respectively), sparse point clouds(we extract 1000 models from the ABC [36] dataset and implement down-sampling [37]), and real point clouds(we collect 125 real scanning point clouds provided by reference [3]. Such point clouds are achieved from 3D models of ABC data with 3D printer and scanner), the test dataset as shown in Table 1. In the following parts, we report the performance of sharp feature detection based on the test sets.

4.2 Metrics

Geometric Consistency. To evaluate the performance of sharp feature detection, we need a set of quantitative analysis tools to provide metrics. Here we use D_P, D_G and D_{mean} to represent the geometric consistency between the detected results and the ground truth.

$$D_P = \frac{1}{|P'|} \sum_{p_i \in P'} d_{min}(p_i^{'}, G), \tag{10}$$

$$D_G = \frac{1}{|G|} \sum_{g_i \in G} d_{min}(g_i, P^{'}), \tag{11}$$

where D_P represents error of minimum distance from detected sharp point $p_i^{'}$ to the ground truth G, $|P^{'}|$ is the number of sharp points. Correspondingly, D_G is the error from ground truth point g_i to the $P^{'}$. Both evaluation indicators can reflect the effectiveness of sharp feature detection to a certain extent. However, there are some limitations: if there are fewer detected edge points, D_P is lower unreasonably. For instance, if the model only detects one single correct edge point, the value of D_P is zero, $\sum_{p_i \in P'} d_{min}(p_i^{'}, G) = 0$. Similarly, if there are redundant detected edge points, the value of D_G is unreliably higher. To mitigate the impact of the two extreme cases, we calculated the mean of D_P and D_G,

$$D_{mean} = \frac{D_P + D_G}{2}, \tag{12}$$

It is used to avoid bias in classification and provide a balanced measurement for the geometric consistency of sharp features. The details are discussed in the next subsection.

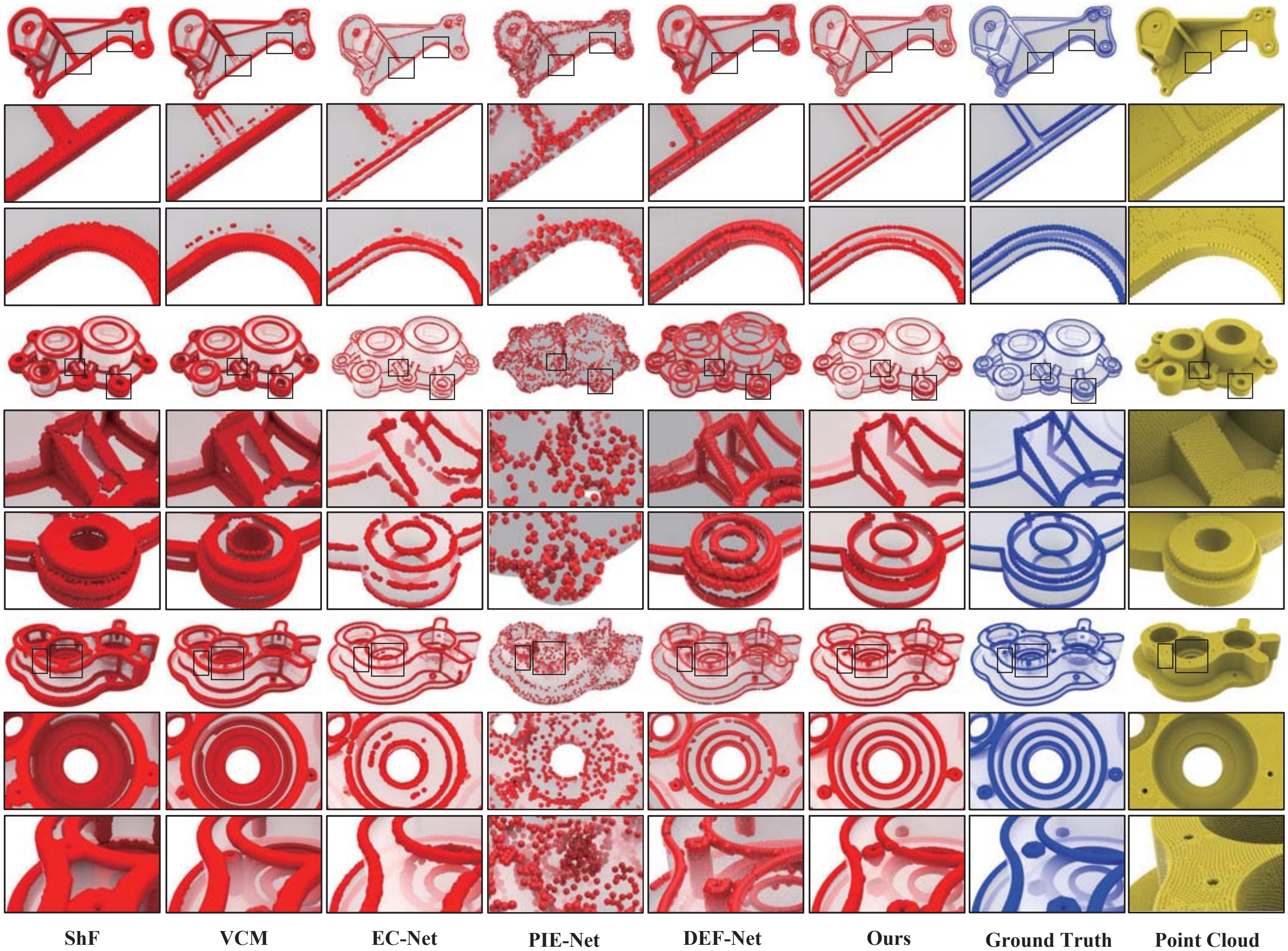


Fig. 9. Comparisons of different sharp feature detection methods. ShF [17],VCM [1], EC-Net [4],PIE-Net [21],DEF-Net [3].

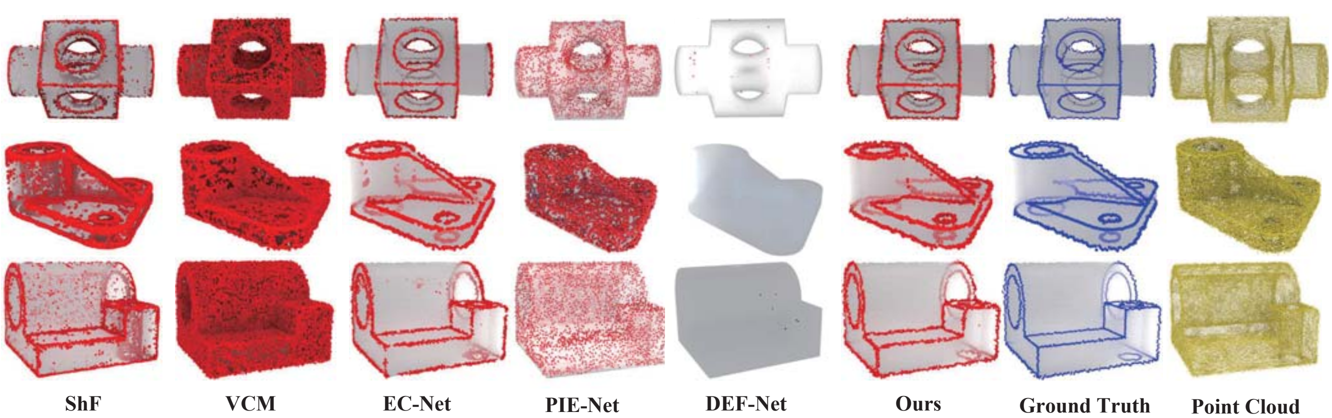


Fig. 10. Comparisons of different sharp feature detection methods (ShF [17],VCM [1], EC-Net [4],PIE-Net [21],DEF-Net [3]) for noisy point clouds.

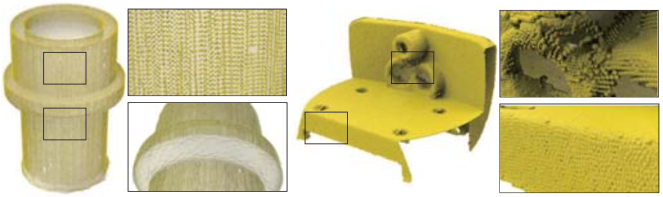


Fig. 11. Real-World scanning point cloud.

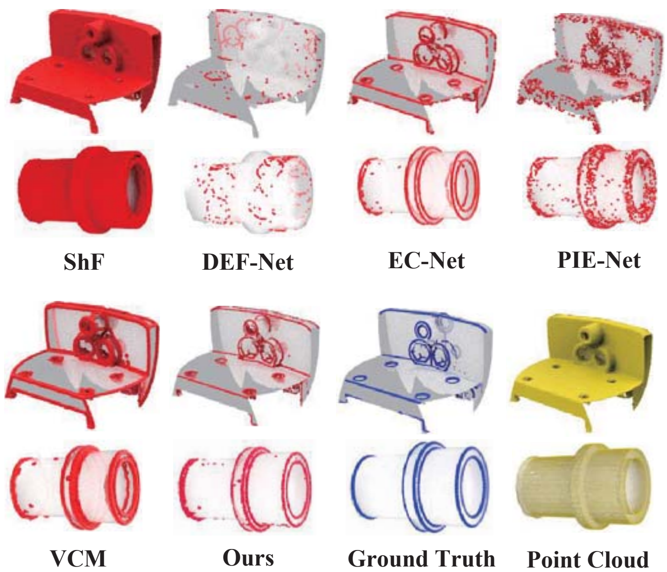


Fig. 12. Comparisons of different sharp feature detection methods for real-world point clouds. ShF [17],VCM [1], EC-Net [4],PIE-Net [21],DEF-Net [3].

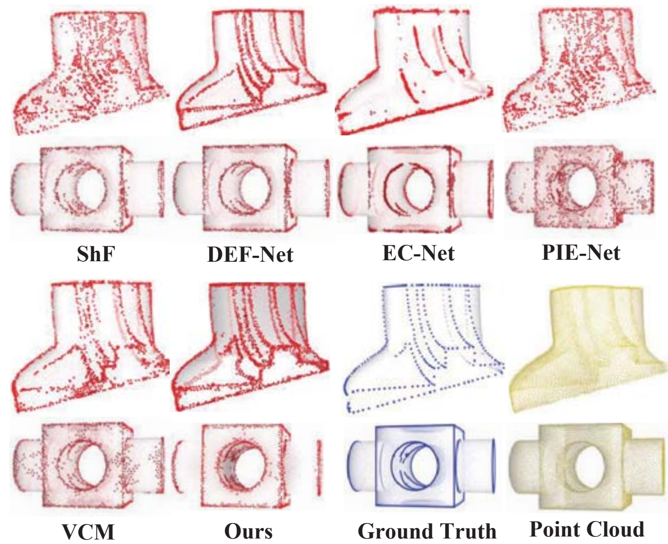


Fig. 13. Comparisons of different sharp feature detection methods (ShF [17],VCM [1], EC-Net [4],PIE-Net [21],DEF-Net [3]) for sparse point clouds.

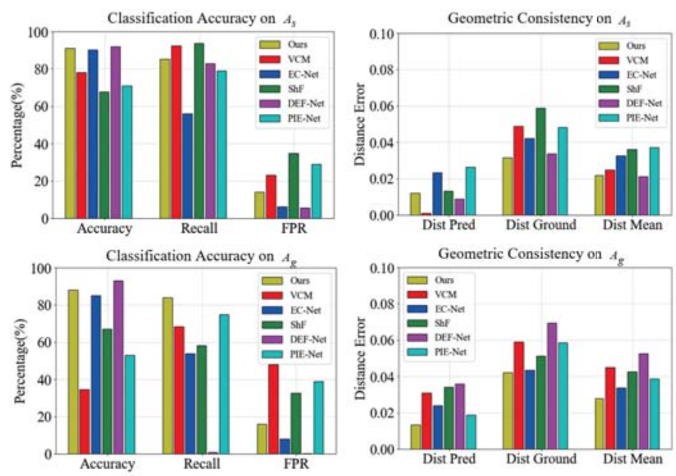


Fig. 14. Visualizations of quantitative metrics for different methods. First column: classification accuracy results of different methods; second column: geometric consistency metrics of different methods.

Classification Accuracy.To reveal a more intuitive performance for sharp feature detection, we also employ some traditional metrics for classification measurement, including the percentage of classification accuracy, recall, and FPR (False Positive Rate). Such metrics are used to estimate the sharp point judgment directly.

The sharp feature detection is a binary classification task (classify sharp points and normal points). Therefore, some classical classification indexes can be employed to estimate the accuracy, which includes accuracy, recall, FPR, and ROC (Receiver Operating Characteristic) curves. The classification accuracy can be directly counted by the percentage of correct classification of sharp points and normal points. To evaluate the overall performance of network classification, we use accuracy as a quality assessment metric, as shown in Eq.13.

$$Accuracy = \frac{TP + TN}{N}, \tag{13}$$

where TP (True Positives) is the number of cases where the actual value is positive and the predicted value is also positive, TN (True Negatives) is the number of cases where the actual value is negative, and the predicted value is also negative, and N is the total number of samples.

In order to measure whether the sharp points are correctly predicted, we use the recall rate as an indicator. The recall rate with a higher value means that the method is sensitive to the sharp feature. The calculation of the recall rate is shown in Eq.14

$$Recall = \frac{TP}{(TP + FP)}, \tag{14}$$

where FP (False Positives) is the number of cases where the actual value is negative but the predicted value is positive. To measure the insensitivity of the method for normal points, we use FPR as another indicator. The lower the FPR, the fewer normal points are misjudged as sharp points. The calculation of FPR is shown as

$$FPR = \frac{FP}{(TN + FP)}. \tag{15}$$

4.3 Comparisons

Experimental Comparisons with Different Methods. We compare the sharp feature detection performance of different methods based on the seven test sets. The mentioned

This article has been accepted for publication in IEEE Transactions on Visualization and Computer Graphics. This is the author's version which has not been fully edited and content may change prior to final publication. Citation information: DOI 10.1109/TVCG.2023.3346907

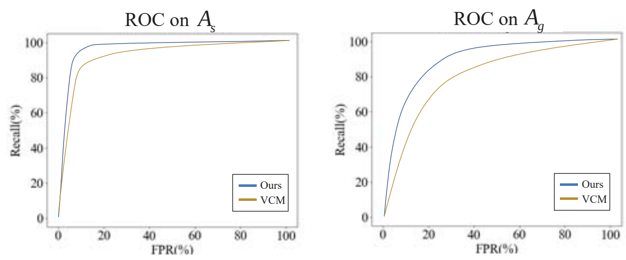


Fig. 15. Visualizations of ROC for Ours and VCM methods.

TABLE 2
Quantitative metrics of different sharp feature detection methods based on the test set A_s.

Methods	D_P	D_G	D_{mean}	Accuracy	Recall	FPR
VCM [1]	**0.0010**	0.0490	0.0250	78.1	92.4	23.1
ShF [17]	0.0133	0.0589	0.0361	67.7	**93.7**	34.8
EC-Net [4]	0.0233	0.0422	0.0327	90.3	56.0	**6.10**
PIE-Net [21]	0.0263	0.0483	0.0373	71.9	79.1	29.3
DEF-Net [3]	0.0089	0.0338	**0.0213**	**92.8**	83.2	5.52
MSL-Net	0.0122	**0.0315**	0.0218	91.2	85.2	14.1

TABLE 3
Quantitative metrics of different sharp feature detection methods based on the test set A_g.

Methods	D_P	D_G	D_{mean}	Accuracy	Recall	FPR
VCM [1]	0.0310	0.0590	0.0450	34.6	68.3	58.1
ShF [17]	0.0341	0.0514	0.0427	67.1	58.2	32.6
EC-Net [4]	0.0241	0.0434	0.0337	85.2	54.2	**8.90**
PIE-Net [21]	0.0189	0.0587	0.0388	53.9	75.7	39.8
DEF-Net [3]	0.0359	0.0694	0.0526	93.8	0.96	0.17
MSL-Net	**0.0135**	**0.0423**	**0.0279**	**88.5**	**84.2**	16.8

TABLE 4
Quantitative metrics of different sharp feature detection methods based on the original ABC dataset A_{all}.

Methods	D_P	D_G	D_{mean}	Accuracy	Recall	FPR
VCM [1]	**0.0103**	0.0591	0.0347	73.1	87.3	30.7
ShF [17]	0.0172	0.0624	0.0398	66.7	**90.1**	38.9
EC-Net [4]	0.0264	**0.0502**	0.0383	82.1	53.9	**10.2**
PIE-Net [21]	0.0279	0.0512	0.0395	68.6	77.1	35.7
DEF-Net [3]	0.0127	0.0397	**0.0262**	**90.7**	78.3	6.78
MSL-Net	0.0136	0.0529	0.0332	83.4	80.9	16.3

TABLE 5
Quantitative metrics of different sharp feature detection methods based on real scanned point clouds.

Methods	D_P	D_G	D_{mean}	Accuracy	Recall	FPR
VCM [1]	0.0098	0.0196	0.0147	82.6	**86.9**	22.3
ShF [17]	0.1348	0.0007	0.0677	15.7	94.6	89.7
EC-Net [4]	**0.0083**	**0.0154**	**0.0118**	**86.9**	78.1	13.8
PIE-Net [21]	0.0257	0.0438	0.0347	72.9	56.7	21.8
DEF-Net [3]	0.0473	0.0531	0.0502	58.3	21.7	14.4
MSL-Net	0.0111	0.0166	0.0138	84.8	40.4	**6.79**

metrics are used to measure the related indexes of the detection methods, which contain geometric consistency, classification accuracy, and noisy robustness. The comparison methods include sharpness fields-based detection (ShF) [17],Voronoi-based feature estimation (VCM) [1], DEF-Net [3], PIE-Net [21] and EC-Net [4] that cover the mainstream technology solutions. In Tables 2∼4, the estimated results based on the mentioned indexes are reported. It is clear that our method achieves more accurate classification results. In Fig. 14, more intuitive histograms are visualized to represent the advantage of our method in classification. As MSL-Net and VCM provide threshold parameters, we can extract different numbers of sharp feature points by setting different thresholds. However, EC-Net and ShF cannot manually set threshold parameters, so we only compared the ROC curves of MSL-Net and VCM in Fig. 15. Some instances are visualized in Figs. 9 and 10. Even in the presence of noise interference, our method is still able to better detect sharp points.

Experiments on Sparse Point Clouds. In order to generate sparse point clouds, we extract 1000 models from the ABC [36] dataset and implement down-sampling [37]. The point number is controlled to 10,000. The sharp feature detection results are shown in Fig.13 with quantitative analysis in Table 6. Results demonstrate that both DEF-Net and our method yield favorable results on sparse point clouds. Due to the detection of neighborhoods in Euclidean space, ShF produces more errors on sparse point clouds. It is important that our method implements feature detection on simplified models directly. The DEF-Net performs slightly better performance than our method, which benefits from the input point cloud with clean and uniform distributed points. As shown in Fig.12 and Fig.10, we prove that the DEF-Net notably under-performs our method on non-uniform and noisy models. Despite employing up-sampling during edge detection, EC-Net still overlooks a considerable number of edge points. Similarly, the PIE-Net and VCM methods exhibit inferior performance in edge point detection compared to our method.

Experiments on Different Levels of Noisy Point Clouds. We have conducted comparative experiments with different levels of noise (0.12%, 0.2% and 0.3%), which are reported in Tables 7 and 8. Among the evaluation indicators, Recall represents the accuracy of positive edge point detection in total proportion. On the contrary, FPR refers to the false positive proportion in the detection. The two metrics should be combined to evaluate the performance of sharp feature detection. For VCM, ShF, and PIE-Net, the related values of FPR are higher, which means these methods tend to extend the edge points in regions with sharp features. For DEF-Net, its recall is lower than others, which means it considers more points to be normal points and neglects real edge points. The EC-Net achieves a relatively balanced result. However, its recall value is significantly lower than our method. The reason is that our method adopts a more aggressive recognition strategy. It helps to identify more edge points in the sharp feature region. Overall, our method

TABLE 6
Quantitative metrics of different sharp feature detection methods based on sparse point clouds

Methods	D_P	D_G	D_{mean}	Accuracy	Recall	FPR
VCM [1]	0.0243	0.0376	0.0309	78.1	79.6	21.7
ShF [17]	0.0257	0.0435	0.0346	73.8	80.3	25.3
EC-Net [4]	**0.0127**	0.0478	0.0302	79.6	81.7	14.9
PIE-Net [21]	0.0278	0.0369	0.0323	70.2	77.3	31.5
DEF-Net [3]	0.0129	**0.0365**	**0.0247**	**85.1**	88.7	**13.7**
MSL-Net	0.0134	0.0389	0.0261	82.9	**90.8**	23.5

TABLE 7
Recall of different method based on the different levels of noisy model.

Levels	0.12%	0.2%	0.3%
VCM [1]	68.3	71.5	72.1
ShF [17]	58.2	67.1	81.3
EC-Net [4]	54.2	53.9	53.0
PIE-Net [21]	75.7	74.1	73.4
DEF-Net [3]	0.96	0.78	0.77
MSL-Net	84.2	84.8	82.1

achieves better sharp feature results with more stable performance.

Experiments on Real Scanning Point Clouds. We collect 125 real scanning point clouds provided by reference [3]. These point clouds are achieved from 3D models of ABC data with a 3D printer and scanner. In Fig.11, our method predicts edge points that are closest to the ground truth. Due to the sparsity and non-uniformity of the scanned point clouds, as shown in Fig.12, calculating the neighborhood in Euclidean space for ShF produces more errors. In comparison, our method uses intrinsic neighborhood detection, which effectively improves the accuracy of detection. The real scanning point clouds contain more random noisy points. The performance of DEF-Net is reduced for noisy point clouds, which are proved in Table 3. Benefiting from up-sampling, EC-Net can achieve better results. However, it is sensitive to the point distribution that has been proved in Table 6. Our method achieves a balanced scheme with a more stable performance.

TABLE 8
FPR of different method based on the different intensity of noisy model.

Levels	0.12%	0.2%	0.3%
VCM [1]	58.1	62.1	63.4
ShF [17]	32.6	38.1	42.6
EC-Net [4]	8.90	9.34	9.67
PIE-Net [21]	39.8	41.7	42.3
DEF-Net [3]	0.17	0.14	0.13
MSL-Net	16.8	18.7	22.6

TABLE 9
Quantitative metrics of MSL-Net with different normal calculate methods on A_s.

Methods	D_P	D_G	D_{mean}	Accuracy	Recall	FPR
PCA [40]	0.0131	0.0462	0.0296	86.1	82.0	19.4
Deepfit [41]	0.0289	0.0512	0.0400	82.4	79.5	23.1
NH-Net [32]	0.0163	0.0405	0.0284	78.4	80.1	29.1
MPFS [32]	**0.0122**	**0.0315**	**0.0218**	**91.2**	**85.2**	**14.1**

4.4 Ablation study

In this part, we evaluate the function of each module in MSL-Net to verify the rationality of its structure. The modules include normal estimation, neighbor detection, structure of multi-scale, and serialization processing.

Normal Estimation. To achieve the intrinsic shape descriptor, normal estimation is necessary for MSL-Net training. We compare the performance of different normal estimation methods, including PCA-based estimation [40], MFPS [32], DeepFit [41], and NH-Net [32]. The quantitative analysis results are shown in Table 9. It can be seen that MFPS helps to detect more accurate sharp points while ensuring the stability for judgment of normal points as much as possible. In comparison, other methods are more sensitive to non-uniform densities. Therefore, the normal estimation of MSL-Net is implemented by MFPS in practice.

Neighbor Detection. As an important module of MLS-Net, the DIN detection extracts intrinsic neighbors to represent local shape features. It is completely different from the traditional KNN detection that has been shown in Fig. 4. The improvement of the DIN detection should be evaluated. We compare the performance of MLS-Net with KNN detection. In Table 10, the geometric consistency and classification accuracy results are shown. It is clear that DIN detection significantly improves the performance of sharp feature detection.

Multi-Scale Structure. Different scales of DIN also take influence for MSL-Net. To reveal the relationship between the scale and the sharp feature detection, we select different k values (8, 40, 64, 96) in DIN detection to compare the performance. The results are also reported in Table 10. It reveals that the smaller scale of DIN tends to reduce the sharp points. On the contrary, the larger scale predicts more sharp points around the ground truth. The two conditions explain that the multi-scale DIN is to avoid the two extreme situations and balance the accuracy and robustness. In Fig. 17, we visualize the sharp feature detection results by MSL-Net with different scales of DIN. Compared to the single-scale DIN, the multi-scale DIN achieves more accurate and robust sharp features.

The Multi-Scale Grouping (MSG) module used in PointNet++ and the multi-scale network used in MSL-Net share similar ideas for local geometric feature learning. The difference is that the MSG module shares the learning parameters in the network with one branch and MSL-Net trains the independent parameters in related channels with corresponding scales. Obviously, the multi-channels improve the accuracy and robustness for detection, which have been proven in ablation, as shown in Fig.17 and Table 10. MSG is a module for calculating multi-scale neighborhoods in

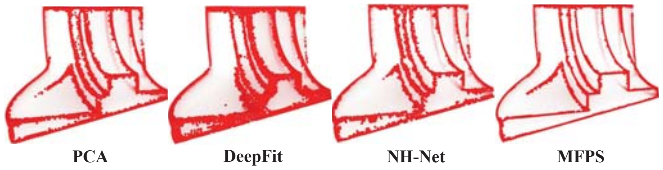


Fig. 16. Comparison of different normal estimation methods.

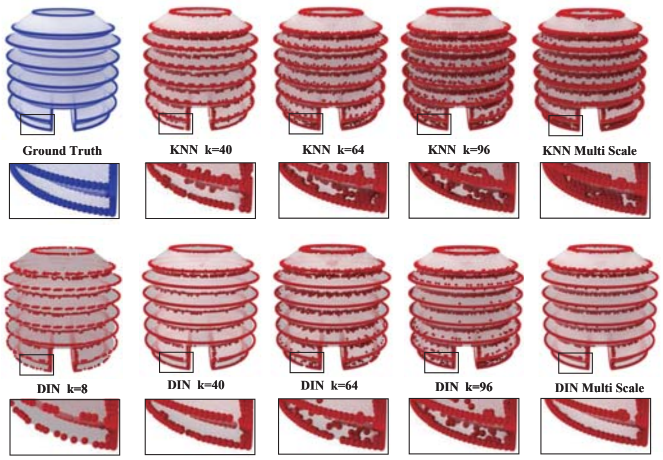


Fig. 17. Visualizations of sharp features detected by MSL-Net with different kinds of neighbors and related scales.

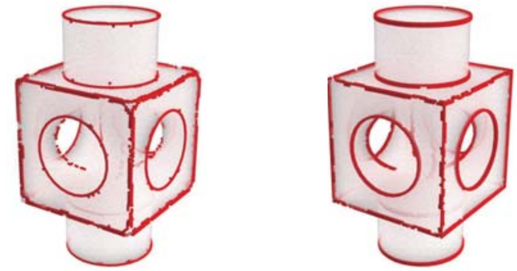


Fig. 18. Visualizations of different serialization methods, Left: *CNC*-based serialization; right: Euclidean distance-based serialization

TABLE 10
Quantitative metrics of MSL-Net with different neighbor regions on A_s. **KNN**: k-nearest neighbors based on Euclidean distance; **DIN**: discrete intrinsic neighbor.

Neighbor	D_P	D_G	D_{mean}	Accuracy	Recall	FPR
KNN k=40	0.0193	0.0315	0.0254	89.3	69.9	7.8
KNN k=64	0.0132	0.0535	0.0333	82.3	82.7	17.7
KNN k=96	0.0113	0.0576	0.0344	81.5	84.6	26.1
DIN k=8	0.0217	**0.0281**	0.0249	87.1	79.5	**6.1**
DIN k=40	0.0145	0.0311	0.0228	89.9	85.1	14.0
DIN k=64	0.0132	0.0335	0.0233	89.4	85.4	14.6
DIN k=96	0.0124	0.0381	0.0252	86.1	**86.0**	18.4
KNN Multi-Scale	**0.0110**	0.0540	0.0325	82.6	82.8	17.0
DIN Multi-Scale	0.0122	0.0315	**0.0218**	**91.2**	85.2	14.1

Euclidean space, which is equivalent to the KNN Multi-Scale module in the paper. From Table 10, it can be observed that the use of DIN significantly outperforms MSG in most indicators.

Serialization. The disorder problem should be solved for point cloud-based deep learning. The classical solution is to implement max pooling [35] for the feature vector. However, such implementation neglects more geometric details. Therefore, we implement the serialization to solve the problem. Two kinds of serialization can be used, which include *CNC*-based and Euclidean distance-based serializations. For the first one, the input *CNC*-based feature vector is sorted based on their values. It can be regarded as a statistical analysis according to the curvatures in the related DIN region. Such serialization lost the local spatial correspondence to achieve better robustness for non-uniform densities. The Euclidean distance-based serialization keeps the local spatial correspondence to a certain degree. However, it is sensitive to the density. To show the performance of the two serializations, we report the quantitative analysis results in Table 11. In Fig. 18, two instances are used to visualize the difference between the two serializations. The results demonstrate that local spatial correspondence is more important for sharp feature learning.

4.5 Comprehensive Analysis

It has been proved that our method achieves more accurate and robust results for sharp feature detection tasks. Especially for noisy point clouds, the multi-scale structure of MSL-Net with DIN can achieve more effective feature estimation that fully considers the local surface property and statistical shape feature in larger regions. It has been shown that a multi-scale structure is better than a single-scale structure.

From the comparison results with the SOTA methods in Sec 4.3, it can be seen that the ShF [17] predicts more edge points around the regions with sharp features, which lacks accurate analysis in the local region. For noisy point clouds, it produces more independent edge points that violate the continuity of sharp features. For the same reason, the performance of VCM [1] is significantly reduced for noisy points. Some negative results are shown in Fig. 10. The continuity of edge points is not well by EC-Net [4] and PIE-Net [21], which is affected by the noisy points. As the SOTA method, DEF-Net [3] can achieve better performance on point clouds with uniform distribution. However, it is also sensitive to noisy points and affected by non-uniform densities, which are proved in Fig.10 and Table 3. Overall, our method achieves a better balance between robustness and accuracy.

Even though our method employs multi-scale analysis and DIN detection, it still has some advantages in compu-

TABLE 11
Quantitative metrics of MSL-Net with different serializations on A_s. **ED**: Euclidean distance-based serialization; **CNC**: *CNC*-based serialization.

	D_P	D_G	D_{mean}	Accuracy	Recall	FPR
CNC	0.0139	0.0437	0.0288	87.2	81.5	15.2
ED	**0.0122**	**0.0315**	**0.0218**	**91.2**	**85.2**	**14.1**

TABLE 12
Time reports of different methods for point clouds with different point numbers.

Points	VCM [1]	DEF-Net [3]	ShF [17]	EC-Net [4]	MSL-Net
10,000	2.5s	4.1s	3.0s	5.1s	**1.5s**
25,000	7.2s	10.3s	7.1s	13.5s	**3.1s**
50,000	14.5s	20.7s	16.8s	27.4s	**6.5s**
100,000	30.1s	40.2s	31.9s	55.0s	**13.2s**

tational efficiency. The DIN detection effectively simplifies the calculation for intrinsic neighbors. It avoids complex computations like geodesic searching and Voronoi diagram construction. We report the time cost of different sharp feature detection methods in Table 12. For point clouds with different point numbers, MSL-Net achieves faster computation speed. The reason is that the structure of MSL-Net is simpler, which facilitates feature learning. It doesn't require complex geometric feature-based pre-modulation and analysis.

Limitations. Although the MSL-Net achieves significant improvements in sharp feature detection, some limitations still exist, including normal vector dependency, ambiguity in sharp point detection on thin surface boundaries, and sensitivity to non-uniform densities. In Table 9, different normal vector estimation methods produce different performances for sharp feature detection. It means that our method is sensitive to normal vector estimation. Thin surfaces increase the difficulty of searching the intrinsic neighbors. Continuous normal transformation pattern in the area of the thin surface is difficult to model and recognize. For the same reason, the non-uniform density has some potential effects on feature learning. In addition, it reduces the accuracy of Euclidean distance-based serialization for *CNC*. Our method uses isotropic simplification to reduce the influence as much as possible. However, the influence can not be eliminated completely.

5 Conclusion

We propose an accurate and robust sharp feature detection method MSL-Net. It extracts DIN from point clouds to improve the accuracy of local region representation. Based on the DIN, we design the intrinsic shape operator that describes the local shape feature while keeping the manifold distribution. With a multi-scale structure, the MSL-Net learns sharp features from input intrinsic shape operators extracted from different scales of DIN. The scheme achieves a balance between local surface property and statistical shape features. Experiments show that the DIN and multi-scale structure achieve significant improvement for point clouds even when there are random noisy points. In future work, we will employ a new deep learning structure to handle normal vector dependence problems and improve the serialization.

Acknowledgments

The authors gratefully appreciated the anonymous reviewers for all of their helpful comments. This work was supported by the National Natural Science Foundation of China under Grant Nos.62172247, 61702293, Natural Science Foundation of Shandong Province (No.ZR2019LZH002).

References

[1] Q. Mérigot, M. Ovsjanikov, and L. J. Guibas, "Voronoi-based curvature and feature estimation from point clouds," *IEEE Transactions on Visualization and Computer Graphics*, vol. 17, no. 6, pp. 743–756, 2010.

[2] H. Huang, S. Wu, M. Gong, D. Cohen-Or, U. Ascher, and H. Zhang, "Edge-aware point set resampling," *ACM transactions on graphics (TOG)*, vol. 32, no. 1, pp. 1–12, 2013.

[3] A. Matveev, R. Rakhimov, A. Artemov, G. Bobrovskikh, V. Egiazarian, E. Bogomolov, D. Panozzo, D. Zorin, and E. Burnaev, "Def: Deep estimation of sharp geometric features in 3d shapes," *ACM Transactions on Graphics (TOG)*, vol. 41, no. 4, pp. 1–22, 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 *Proceedings of the European conference on computer vision (ECCV)*, 2018, pp. 386–402.

[5] C. Lv, W. Lin, and B. Zhao, "Intrinsic and isotropic resampling for 3d point clouds," *IEEE Transactions on Pattern Analysis and Machine Intelligence*, vol. 45, no. 3, pp. 3274–3291, 2022.

[6] M. Pauly, R. Keiser, and M. Gross, "Multi-scale feature extraction on point-sampled surfaces," in *Computer graphics forum*, vol. 22, no. 3. Wiley Online Library, 2003, pp. 281–289.

[7] S. Xia and R. Wang, "A fast edge extraction method for mobile lidar point clouds," *IEEE Geoscience and Remote Sensing Letters*, vol. 14, no. 8, pp. 1288–1292, 2017.

[8] Y. Lin, C. Wang, J. Cheng, B. Chen, F. Jia, Z. Chen, and J. Li, "Line segment extraction for large scale unorganized point clouds," *ISPRS Journal of Photogrammetry and Remote Sensing*, vol. 102, pp. 172–183, 2015.

[9] T. Hackel, J. D. Wegner, and K. Schindler, "Contour detection in unstructured 3d point clouds," in *Proceedings of the IEEE conference on computer vision and pattern recognition*, 2016, pp. 1610–1618.

[10] ——, "Joint classification and contour extraction of large 3d point clouds," *ISPRS Journal of Photogrammetry and Remote Sensing*, vol. 130, pp. 231–245, 2017.

[11] K. Demarsin, D. Vanderstraeten, T. Volodine, and D. Roose, "Detection of closed sharp edges in point clouds using normal estimation and graph theory," *Computer-Aided Design*, vol. 39, no. 4, pp. 276–283, 2007.

[12] B. Li, R. Schnabel, R. Klein, Z. Cheng, G. Dang, and S. Jin, "Robust normal estimation for point clouds with sharp features," *Computers & Graphics*, vol. 34, no. 2, pp. 94–106, 2010.

[13] Y. Wang, H.-Y. Feng, F.-É. Delorme, and S. Engin, "An adaptive normal estimation method for scanned point clouds with sharp features," *Computer-Aided Design*, vol. 45, no. 11, pp. 1333–1348, 2013.

[14] C. Weber, S. Hahmann, and H. Hagen, "Sharp feature detection in point clouds," in *2010 Shape Modeling International Conference*. IEEE, 2010, pp. 175–186.

[15] J. Zhang, J. Cao, X. Liu, H. Chen, B. Li, and L. Liu, "Multi-normal estimation via pair consistency voting," *IEEE transactions on visualization and computer graphics*, vol. 25, no. 4, pp. 1693–1706, 2018.

[16] Y.-F. Feng, L.-Y. Shen, C.-M. Yuan, and X. Li, "Deep shape representation with sharp feature preservation," *Computer-Aided Design*, p. 103468, 2023.

[17] P. Raina, S. Mudur, and T. Popa, "Sharpness fields in point clouds using deep learning," *Computers & Graphics*, vol. 78, pp. 37–53, 2019.

[18] 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 Transactions on Graphics (TOG)*, vol. 41, no. 1, pp. 1–21, 2021.

[19] M. Loizou, M. Averkiou, and E. Kalogerakis, "Learning part boundaries from 3d point clouds," in *Computer Graphics Forum*, vol. 39, no. 5. Wiley Online Library, 2020, pp. 183–195.

[20] L. Yu, X. Li, C.-W. Fu, D. Cohen-Or, and P.-A. Heng, "Pu-net: Point cloud upsampling network," in *Proceedings of the IEEE conference on computer vision and pattern recognition*, 2018, pp. 2790–2799.

[21] X. Wang, Y. Xu, K. Xu, A. Tagliasacchi, B. Zhou, A. Mahdavi-Amiri, and H. Zhang, "Pie-net: Parametric inference of point cloud edges," *Advances in neural information processing systems*, vol. 33, pp. 20 167–20 178, 2020.

This article has been accepted for publication in IEEE Transactions on Visualization and Computer Graphics. This is the author's version which has not been fully edited and content may change prior to final publication. Citation information: DOI 10.1109/TVCG.2023.3346907

[22] C. R. Qi, L. Yi, H. Su, and L. J. Guibas, "Pointnet++: Deep hierarchical feature learning on point sets in a metric space," *Advances in neural information processing systems*, vol. 30, 2017.

[23] D. Zhang, X. Lu, H. Qin, and Y. He, "Pointfilter: Point cloud filtering via encoder-decoder modeling," *IEEE Transactions on Visualization and Computer Graphics*, vol. 27, no. 3, pp. 2015–2027, 2020.

[24] D. de Silva Edirimuni, X. Lu, G. Li, and A. Robles-Kelly, "Contrastive learning for joint normal estimation and point cloud filtering," *IEEE Transactions on Visualization and Computer Graphics*, 2023.

[25] T. Zhao, M. Yu, P. Alliez, and F. Lafarge, "Sharp feature consolidation from raw 3d point clouds via displacement learning," *Computer Aided Geometric Design*, vol. 103, p. 102204, 2023.

[26] 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 *Proceedings of the IEEE/CVF Conference on Computer Vision and Pattern Recognition*, 2023, pp. 13 601–13 610.

[27] K. Cherenkova, E. Dupont, A. Kacem, I. Arzhannikov, G. Gusev, and D. Aouada, "Sepicnet: Sharp edges recovery by parametric inference of curves in 3d shapes," in *Proceedings of the IEEE/CVF Conference on Computer Vision and Pattern Recognition*, 2023, pp. 2726–2734.

[28] J. Zeng, G. Cheung, M. Ng, J. Pang, and C. Yang, "3d point cloud denoising using graph laplacian regularization of a low dimensional manifold model," *IEEE Transactions on Image Processing*, vol. 29, pp. 3474–3489, 2019.

[29] C. Moenning and N. A. Dodgson, "Fast marching farthest point sampling for implicit surfaces and point clouds," *Computer Laboratory Technical Report*, vol. 565, pp. 1–12, 2003.

[30] C. Lv, W. Lin, and B. Zhao, "Approximate intrinsic voxel structure for point cloud simplification," *IEEE Transactions on Image Processing*, vol. 30, pp. 7241–7255, 2021.

[31] C. Lv, W. Lin, and J. Zheng, "Adaptively isotropic remeshing based on curvature smoothed field," *IEEE Transactions on Visualization and Computer Graphics*, 2022.

[32] H. Zhou, H. Chen, Y. Feng, Q. Wang, J. Qin, H. Xie, F. L. Wang, M. Wei, and J. Wang, "Geometry and learning co-supported normal estimation for unstructured point cloud," in *Proceedings of the ieee/cvf conference on computer vision and pattern recognition*, 2020, pp. 13 238–13 247.

[33] R. B. Rusu, N. Blodow, and M. Beetz, "Fast point feature histograms for 3d registration," in *2009 IEEE international conference on robotics and automation*. IEEE, 2009, pp. 3212–3217.

[34] Z. Liu, Y. Lin, Y. Cao, H. Hu, Y. Wei, Z. Zhang, S. Lin, and B. Guo, "Swin transformer: Hierarchical vision transformer using shifted windows," in *Proceedings of the IEEE/CVF international conference on computer vision*, 2021, pp. 10 012–10 022.

[35] C. R. Qi, H. Su, K. Mo, and L. J. Guibas, "Pointnet: Deep learning on point sets for 3d classification and segmentation," in *Proceedings of the IEEE conference on computer vision and pattern recognition*, 2017, pp. 652–660.

[36] S. Koch, A. Matveev, Z. Jiang, F. Williams, A. Artemov, E. Burnaev, M. Alexa, D. Zorin, and D. Panozzo, "Abc: A big cad model dataset for geometric deep learning," in *Proceedings of the IEEE/CVF conference on computer vision and pattern recognition*, 2019, pp. 9601–9611.

[37] D. Dunbar and G. Humphreys, "A spatial data structure for fast poisson-disk sample generation," *ACM Transactions on Graphics (TOG)*, vol. 25, no. 3, pp. 503–508, 2006.

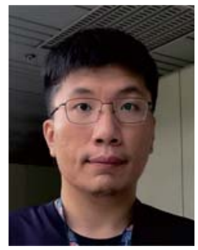
[38] G. Farin, *Curves and surfaces for CAGD: a practical guide*. Morgan Kaufmann, 2002.

[39] O. Cascade, "Open cascade technology, www.opencascade.com," 2011.

[40] H. Hoppe, T. Derose, T. Duchamp, J. Mcdonald, and W. Stuetzle, "Surface reconstruction from unorganized points," *ACM SIGGRAPH Computer Graphics*, vol. 26, no. 2, pp. 71–78, 1992.

[41] Y. Ben-Shabat and S. Gould, "Deepfit: 3d surface fitting via neural network weighted least squares," in *Computer Vision–ECCV 2020: 16th European Conference, Glasgow, UK, August 23–28, 2020, Proceedings, Part I 16*. Springer, 2020, pp. 20–34.

Xianhe Jiao is a postgraduate. His supervisor is Junli Zhao. His current research projects include Craniofacial Landmarks Localization, sharp feature detection in point cloud, point cloud simplification, and he has published an paper at International Conference on Multimedia and Expo. He received his BBM degree from Qingdao University in 2021. His research interests is computer graphics.



Chenlei Lv is currently an assistant professor with the College of Computer Science and Software Engineering, Shenzhen University. He received PhD degree in College of information science and technology, Beijing Normal University (BNU). After that, he started the research work as a Research Fellow in School of Computer Science and Engineering, Nanyang Technological University (NTU). His research interests include Computer Vision, 3D Biometrics, Computer Graphics, Discrete Differential Geometry and Conformal Geometric. He has published several papers in IEEE TPAMI, TIP, TVCG, TMM, ACM TOMM, PR, etc. The personal page of the link is aliexken.github.io.

Ran Yi is an assistant professor with the Department of Computer Science and Engineering, Shanghai Jiao Tong University. She received the BEng degree and the PhD degree from Tsinghua University, China, in 2016 and 2021. Her research interests include computer vision, computer graphics and computational geometry.

Junli Zhao is a professor and doctoral supervisor in College of Computer Science and Technology, Qingdao University, Qingdao. She received her Ph.D. degree in Computer applied technology major in 2015 from Beijing Normal University, Beijing. She was a visiting scholar of State University of New York At Stony Brook. She is currently engaged in craniofacial informatics, computer graphics, computer vision, and virtual reality research. She has presided over and participated in more than ten projects, such as National Natural Science Foundation of China, Key Research and Development Projects of Shandong Province, Natural Science Foundation of Shandong Province and etc., published more than fifty papers in conference (such as ICCV) and journals on related topics.

Zhenkuan Pan is a professor in College of Computer Science and Technology, Qingdao University, Qingdao. He received his Ph.D. degree in mechanics major from Shanghai Jiao Tong University, Shanghai, in 1992. He was a visiting scholar of University of California, Los Angeles. His main research interests focus on computer vision, image processing, multi-body system dynamics and control.

Zhongke Wu is Full Professor in School of Artificial Intelligence, Beijing Normal University (BNU), China from 2006. Prior to joining in BNU, he worked in Nanyang Technological University (NTU), Singapore, Institut National de Recherche en Informatique et en Automatique (INRIA) in France, Institute of High Performance Computing (IHPC) and Institute of Software, Chinese Academy of Sciences in China from 1995 to 2006. Prof.Wu's current research interests include computer graphics, computer animation, virtual reality, geometric modeling, shape analysis and medical imaging. He has published several papers in Computer Aided Geometric Design, Virtual Reality, Pattern Recognition, Computers & Graphics, MICCAI, etc. He achieved the second prize of National Science Progress Award in 2009, the second prize of Science and Technology Progress Award of China Computer Federation in 2013, second prize of Science and Technology Progress Award of Ministry of Education in 2006 and 2015.

Yong-Jin Liu is a tenured full professor with the Department of Computer Science and Technology, Tsinghua University, China. He received the BEng degree from Tianjin University, China, in 1998, and the PhD degree from the Hong Kong University of Science and Technology, Hong Kong, China, in 2004. His research interests include computer vision, computer graphics and computer-aided design. For more information, visit http://cg.cs.tsinghua.edu.cn /people/ Yongjin/Yongjin.htm.