Neighbor Reweighted Local Centroid for Geometric Feature Identification

Tong Liu, Zhenhua Yang, Shaojun Hu, Zhiyi Zhang, Chunxia Xiao, Xiaohu Guo, Long Yang

Abstract—Identifying geometric features from sampled surfaces is a significant and fundamental task. The existing curvature-based methods that can identify ridge and valley features are generally sensitive to noise. Without requiring high-order differential operators, most statistics-based methods sacrifice certain extents of the feature descriptive powers in exchange for robustness. However, neither of these types of methods can treat the surface boundary features simultaneously. In this paper, we propose a novel neighbor reweighted local centroid (NRLC) computational algorithm to identify geometric features for point cloud models. It constructs a feature descriptor for the considered point via decomposing each of its neighboring vectors into two orthogonal directions. A neighboring vector starts from the considered point and ends with the corresponding neighbor. The decomposed neighboring vectors are then accumulated with different weights to generate the NRLC. With the defined NRLC, we design a probability set for each candidate feature point so that the convex, concave and surface boundary points can be recognized concurrently. In addition, we introduce a pair of feature operators, including assimilation and dissimilation, to further strengthen the identified geometric features. Finally, we test NRLC on a large body of point cloud models derived from different data sources. Several groups of the comparison experiments are conducted, and the results verify the validity and efficiency of our NRLC method.

Index Terms—feature identification, local centroid, convexity and concavity, surface boundary points, assimilation and dissimilation

1 Introduction

The prominent geometric features of a 3D object generally include surface regions of sharp bending or opening boundaries. Although accounting for a very limited proportion of the surface regions [1], [2], these geometric features carry abundant structural information and play an important role in 3D shape analysis. Generally, quality surface reconstruction depends on sophisticated treatment of geometric features [3], [4], [5], [6]. Meanwhile, these features are heavily involved in many semantically related geometry processing applications, ranging from abstract shape representation [7], [8], [9], [10], shape registration [11], [12], segmentation [13], [14] to shape recognition [15], [16] and retrieval [17]. Hence, developing effective detection and recognition methods for geometric features is one of the significant research directions in computer graphics and computer vision communities [18], [19].

With the popularity of 3D scanning technology in the last decade [20], shape analysis and processing of the acquired scene-level models ought to identify not only the surface bending features (ridges and valleys) of individual objects but also the structural features of the considered scenes. Unlike closed objects, a scene often comprises many objects with different shape complexities and quite a number of object parts with noteworthy surface boundaries. Identifying surface boundaries has many practical benefits, including judging the completeness of the considered point clouds, locating and measuring the cracks or deficiencies on the given surfaces, detecting thin sheets with open boundaries and so forth. In this paper, we consider geometric features to be the points that have abrupt shape changes in their local neighborhoods, from which surface boundaries should be considered essential geometric features because they demonstrate abrupt shape occurrence or disappearance in their neighborhoods and convey important perceptual information (see the green points in Fig. 1).

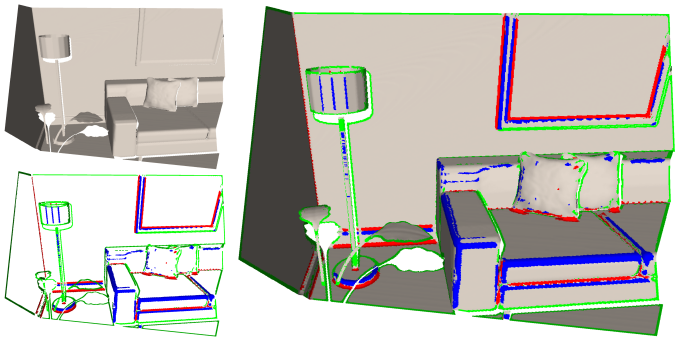


Fig. 1. A single view surface (SVS1) and the identified geometric features. Top-left: the input point cloud. Right: feature-identified results. Bottom-left: feature sets. The convex, concave and surface boundary features are encoded by blue, red and green colors, respectively.

Therefore, geometric feature identification is now expected to recognize ridges and valleys as well as surface boundaries simultaneously. The curvature tensor [21], which can properly present the shape bending extent of the local surface, is extensively used to extract convex and concave features [3], [22]. However, most curvature-based feature identification methods are sensitive to noise. Since

- *Tong Liu, Zhenhua Yang, Shaojun Hu, Zhiyi Zhang and Long Yang are with the College of Information Engineering, Northwest A&F University, Yangling, Shaanxi, China, 712100. Email:{tliu, zhyang1999, hsj, zhangzhiyi, yl}@nwafu.edu.cn. Long Yang is the corresponding author (yl@nwafu.edu.cn).*
- *Chunxia Xiao is with the Computer School, Wuhan University, Hubei, China. Email: cxxiao@whu.edu.cn.*
- *Xiaohu Guo is with the Department of Computer Science, The University of Texas at Dallas, Texas, USA. Email: xguo@utdallas.edu.*

curvature cannot account for the sudden appearance and disappearance of the surface, it lacks the capability to identify the surface boundaries.

To avoid the computation of curvature and more complicated high-order differential operators, detecting geometric features proceeds along the statistics-based direction as the number of point cloud models increases [23], [24], [25], [26]. This type of method classifies each point in terms of its local statistical quantities without any requirements of cleaning or preprocessing for input models. The existing statistics-based methods show high robustness but cannot classify convex and concave features. Although they have been used in point clouds of large scenes and received growing popularity in the 3D vision field [26], statistics-based approaches are still incompetent to identify surface boundaries. To the best of our knowledge, identifying convex, concave and surface boundary features simultaneously from a point cloud, especially a large scene, is still an unsolved problem.

In this paper, we propose a novel method of geometric feature identification for point clouds accompanied by normal orientations. It exploits a neighbor reweighted local centroid (hereafter, we call it **NRLC**) computation to detect geometric features and concurrently perform feature classification. Overall, the contributions of our work include:

- Defining a robust local centroid for each point considering both the distances and shape variations from its neighbors.
- Designing a unified feature detection and classification algorithm to effectively identify geometric features and assign them to be convex, concave or surface boundary points.
- Introducing assimilation and dissimilation operators that can readily refine geometric features for postprocessing.

Before elaborating our method, we will analyze the related geometric feature identification methods concisely.

2 Related work

There are many approaches related to geometric feature identification. The existing techniques can be divided into four main categories: curvature-related, statistics-based, data-driven and surface boundary-oriented methods.

As an essential surface property, curvature has been defined in various forms on mesh surfaces and has been used to identify geometric features. Taubin [21] defines the directional curvature for a pair of vertices by measuring the offset of the neighbor along the normal direction of the central vertex. Then, these directional curvatures within the 1-ring neighborhood of the central vertex are used to estimate its curvature tensor. Lengagne et al. [3] use least-squares to fit a quadric for each vertex of the mesh within its local neighborhood and compute the principal curvatures for each vertex. Ohtake et al. [22] implement an MPU (multi-level partition of unity implicits) fitting [27] for the given mesh at multiple scales and compute the curvature and the derivative of curvature to extract the ridge and valley lines.

Rusinkiewicz [28] first calculates the curvature tensor for each facet of the given mesh in terms of the directional derivatives of the surface normal and then constructs the curvature tensor at each vertex by accumulating the curvature tensor of the related facets. Kalogerakis [29] formulates the computation of the curvature tensor for each vertex as a maximum likelihood estimation problem. This method exploits an iterative reweighted least-square to fit the directional curvatures formed by the variations of all normal pairs in a local neighborhood. Since it does not require any topological connections between vertexes, this method is also used on point clouds for the first time to estimate the curvature tensor for each point.

Pauly et al. [30] adopt the eigenanalysis of the covariance matrix within each local neighborhood to estimate surface properties for point clouds. The proportion of the minimum eigenvalue over the total variation is taken as the shape variation along the surface normal. They use surface variation to build multiscale feature descriptors and to further extract feature lines [31]. Most PCA (principal component analysis) -related methods [32], [33], [34] are based on this eigenanalysis of the covariance matrix of the local neighbors. Monga et al. [35] estimate the principal curvatures and directions for medical voxel images by formulizing its first- and second-order partial derivatives. Furthermore, the third-order differential property, derivative of curvature, is also designed for feature detection.

Since numerical integration can effectively resist noise interference, to improve robustness, integral invariants (IIs) [36] over specific domains are used to estimate the local curvatures. Yang et al. [37] compute the IIs to estimate surface curvatures, which are then used to extract different scales of geometric features. IIs-based methods generally require a multistep procedure: fit the local points (with a paraboloid in [37]), compute the IIs on a closed domain, then utilize the IIs to estimate surface curvatures and finally extract the convex and concave features. Feature extraction based on IIs from large-scale neighborhoods cannot identify the meso- and microscale geometric features [37]. Lai et al. [38] also define IIs on meshes to detect ridge, valley and prong features. Guennebaud et al. employ the algebraic sphere to fit the point set surfaces (APSS) [39]. The radius of the fitted sphere can be used to compute the mean curvature of the surface naturally.

Generally, curvature extremes along the principal curvature direction correspond to ridge or valley regions. Zero values of the curvature derivative are required to identify those feature points. However, surfaces with constant high curvature (e.g., sphere, cylinder, see examples in Fig. 5) hold zero values for the first-order derivative of curvature everywhere. Consequently, the fourth-order differential, i.e., the second-order derivative of curvature, should be introduced to exclude the trivial zero value of the curvature derivative. As a substitute, the zero-crossing of the curvature derivative along the principal curvature directions [3], [22] is employed to distinguish the local curvature extremes from the degenerated constant curvatures so that the feature lines can be extracted. For simplicity, empirical thresholds are adopted to screen the high curvatures [35] and surface variations [31] for extracting geometric features.

Another direction of geometric feature identification is the statistics-based technique. The spin image representation [23] constructs 2D histograms of points falling into a cylindrical volume via a plane that spins around the

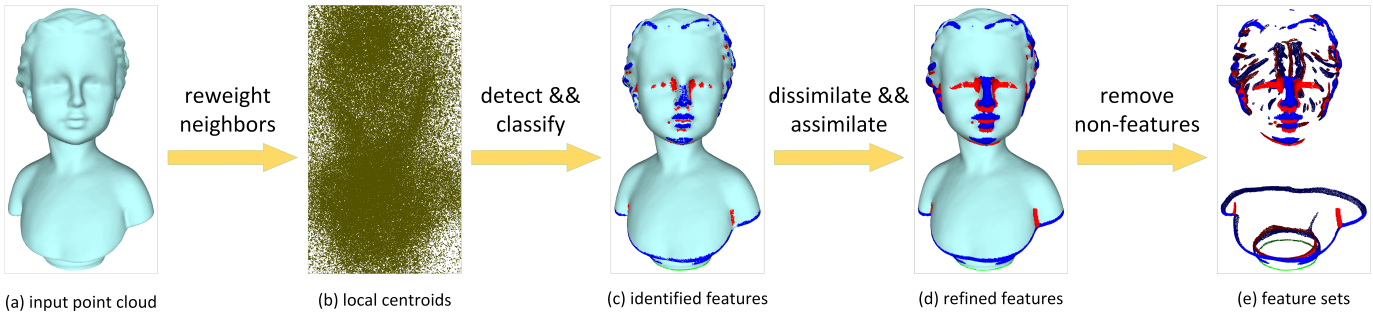


Fig. 2. Overview of the proposed NRLC method. (a) The input point cloud model. (b) Neighbor reweighted local centroid for each point. (c) Identified geometric features. (d) Refined feature points via dissimilating and assimilating operations. (e) Abstract surface represented by feature point sets.

normal of the specific point. SHOT [25] counts the normal histograms within a set of sphere bins to describe the local shape. PFH [24] combines four basic feature descriptors to build a 16D histogram vector as a point's signature. It considers every pair of points within the neighborhood of a given point for several scales. A small portion of points with persistent signatures at different scales will be extracted as geometric features. Although FPFH [40] speeds up the PFH, it is still time-consuming, especially for large scenes. Most of these methods [23], [24], [25] are designed for 3D vision tasks.

The projected images from point clouds are also used to detect geometric features for outdoor buildings [2], [41], [42]. This kind of method generates linear or curved segments and still cannot differentiate convexity and concavity. The sectional 2D profile curves are exploited to match the targeting object and its pose for robot bin picking tasks [43].

There are a few data-driven methods [44], [45], [46], [47] related to explicit geometric features. Kalogerakis [45] designed a feature training method to extract the feature lines for dynamic objects. Guerrero et al. [46] utilize the local patches to train a deep neural network for estimating normals and curvatures on object-level point clouds. EC-Net [47] focuses on enhancing the sharp features. It exploits the annotation of sharp edges on virtual scanned models that are derived from mesh data to produce the training set. EC-Net cannot respond to surface boundaries and does not distinguish convexity and concavity.

In addition to the convex and concave features, finding the surface boundary for point clouds is not a trivial task. The half-disc criterion [48], [49], [50] is designed to detect the hole boundary for point clouds. The angle criterion (AC) [48] checks the maximum gap between two neighbors and the central point. The approach [51] projects the neighbors of a central point onto its local tangent plane and then uses a disc template with six 60° slices to choose the boundary features. A semi-automatic operator is taken in [52] to determine the hole boundary for point clouds with sharp features.

Overall, discrete curvature operators are often defined on manifold meshes with high-quality vertex distributions. They are sensitive to noise and rarely defined on point clouds. Simplified surface variation (PCA-like) methods improve the robustness of feature detection while losing the distinguishing capability for convexity and concavity. Statistics-based methods care much about robustness and do not require high-order differential operators at the cost of sacrificing the description capability of convexity and concavity. The specific methods for surface boundaries cannot find the surface bending features. Data-driven methods require sufficient types of labeled data sets and incur a high time cost to train the network. Hence, in this paper, we intend to identify the convex, concave and surface boundary features in a unified algorithm framework for point cloud models.

3 Overview of Our Method

Our method utilizes the neighbor reweighted local centroid to detect and classify geometric features. It takes as input the initial point cloud $P = \{p_1, p_2, \cdots, p_m\}$($m$ is the number of points) equipped with normal set $N = \{\overrightarrow{n_1}, \overrightarrow{n_2}, \cdots, \overrightarrow{n_m}\}$, as shown in Fig. 2(a). For those raw point clouds, we could exploit PCA combined with orientation propagation and viewpoint assistance to produce reliable normals. We first compute the new local centroid c_i for each point p_i by weighting its neighbors $p_j \in \Omega(p_i)(j = 1, 2, \cdots, k, \Omega(p_i)$ as the k nearest neighbor set of point $p_i)$ considering both the distance and the normal variation factors and accumulating these reweighted neighbors together, as depicted in Fig. 2(b). Then, the feature vector $\overrightarrow{v_i}(i = 1, \ldots, m)$ from the current point p_i to its corresponding centroid c_i is used to detect and classify the potential geometric features, as shown in Fig. 2(c). To denoise and enhance the detected features, our method specifically introduces a pair of feature operators, namely, assimilation and dissimilation, so that the geometric features can be further refined (Fig. 2(d)). Finally, we obtain the sparse feature sets to represent the input model abstractly (Fig. 2(e)). An overview of our NRLC method is illustrated in Fig. 2.

4 Geometric Feature Identification

The centroid of a local surface is highly related to its neighboring distribution and is much easier to calculate than curvature and other point signatures. Relying on this observation, we design and realize the NRLC method.

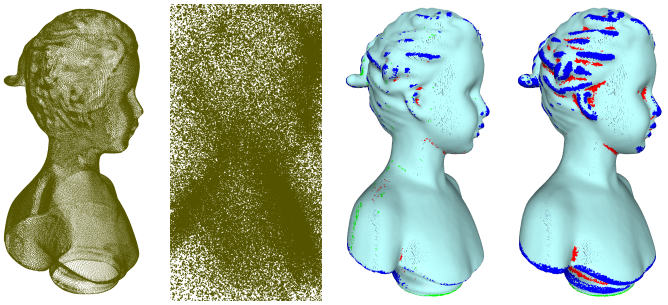
4.1 Neighbor reweighted local centroid

4.1.1 *Definition of the local centroid*

To identify different kinds of geometric features, we modify the general centroid definition and make it applicable to our

problem setting. The most significant changes of the concept are the surface scope covered by a centroid and the computational approach of the centroid. Accordingly, we define a new centroid for each point based on its local neighbors, rather than just one centroid for an entire model. Moreover, we redesign the local centroid computational algorithm depending on the distances and the shape variations between the currently considered point and its neighbors rather than simply averaging its local neighbors.

Specifically, for a given point $p_i(i = 1, 2, \cdots, m)$, we compute the new local centroid c_i from its neighbor set $\Omega(p_i)$. Each neighbor $p_j \in \Omega(p_i)(j = 1, 2, \cdots, k)$ contributes nonequivalently to the final centroid according to both the distance weight $\omega_d(i, j)$ and the shape variation weight $\omega_s(i, j)$ (defined in Section 4.1.2). Implementing the neighbor reweighted local centroid computation for all points, we will obtain the centroid set $C = \{c_1, c_2, \cdots, c_m\}$ (see Fig. 2(b) and Fig. 3(b)) that is used to identify geometric features subsequently (subsections 4.2 and 4.3).



(a) GCs (b) NRLCs (c) fea. (GC) (d) fea. (NRLC)

Fig. 3. Centroids and feature-identified results for a sculpture model. (a) GCs adhering to the target. (b) NRLCs. (c) Features identified by GCs. (d) Features created by NRLCs. Both results of (c) and (d) are generated without using dissimilation and assimilation.

4.1.2 Local centroid computation

The general centroids (GCs) computed by averaging the local neighbors adhere to the target surface, as shown in Fig. 3(a). Since we attempt to identify geometric features by analyzing the location of the centroid for each point, the limited small offset from a point to its general local centroid makes our centroid-based feature detection insensitive to practical shape changes. The clustered feature vectors aggravate the difficulty of geometric feature identification (see an example in Fig. 3(c)). Differing from just averaging the k nearest neighbors, we would like to push the centroids far away from the local surfaces for potential feature points. These operations contribute to the manifesting of different geometric features. In particular, for a given point $p_i(i = 1, 2, \cdots, m)$, its centroid can be expressed as Eq. (1).

$$c_i = p_i + \overrightarrow{v_i}. \tag{1}$$

Vector decomposition. The feature vector $\overrightarrow{v_i}$ defined based on the general centroid contains both surface bending and surface boundary information. However, it is insufficient to distinguish the convex, concave and surface boundary features clearly and lacks robustness. To improve the expression and detection capabilities of geometric features and simplify the feature classification process, we need to separate the surface bending and boundary measurements from the feature vector and treat them specifically.

To this end, we decompose the feature vector $\overrightarrow{v_i}$ into two components along the parallel direction $\overrightarrow{v_{i\parallel}}$ of the normal $\overrightarrow{n_i}$ and the perpendicular direction $\overrightarrow{v_{i\perp}}$ in its tangent plane; see an illustration in Fig. 4(a). Therefore, these two components can be presented as Eq. (2).

$$\begin{cases} \overrightarrow{v_{i\parallel}} = < \overrightarrow{v_i}, \overrightarrow{n_i} > \cdot \overrightarrow{n_i}, \\ \overrightarrow{v_{i\perp}} = \overrightarrow{v_i} - \overrightarrow{v_{i\parallel}}. \end{cases} \tag{2}$$

Here, $< \cdot, \cdot >$ indicates the dot-product of two vectors.

Similarly, for a local neighbor $p_j \in \Omega(p_i)(j = 1, 2, \cdots, k)$, we also decompose the neighboring vector $\overrightarrow{v_{ij}}$ along the parallel direction $\overrightarrow{v_{ij\parallel}}$ of normal $\overrightarrow{n_i}$ and the corresponding perpendicular direction $\overrightarrow{v_{ij\perp}}$, shown in Fig. 4(b). Thereafter, we can accumulate the decomposed neighboring vectors $\overrightarrow{v_{ij\parallel}}$ and $\overrightarrow{v_{ij\perp}}$ for each neighbor p_j. Then, the two components of feature vector $\overrightarrow{v_i}$ are calculated with Eq. (3).

$$\begin{cases} \overrightarrow{v_{i\parallel}} = \sum_{j=1}^{k} \overrightarrow{v_{ij\parallel}}, \\ \overrightarrow{v_{i\perp}} = \sum_{j=1}^{k} \overrightarrow{v_{ij\perp}}. \end{cases} \tag{3}$$

In Eq. (3), if we accumulate each neighbor with an equal weight $1/k$ and treat all points in the same way, we would produce the GCs for a given point cloud (Fig. 3(a)).

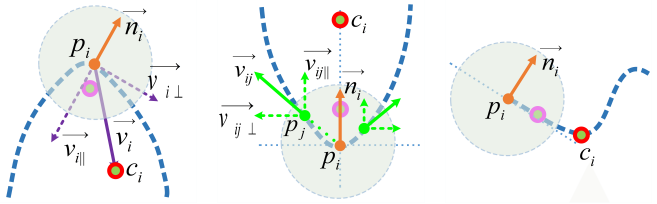
Distance weight. The larger the lengths of feature vectors vary, the easier our centroid-based feature identification will be. To exaggerate the contrast of feature vectors, we increase the lengths of feature vectors for those points located at the ridge, valley and boundary regions while holding the lengths of feature vectors for nonfeature points.

Therefore, we intentionally construct a distance weight $\omega_d(i, j)$ for each neighbor $p_j \in \Omega(p_i)$ of the current point p_i. Since those distant (rather than close) neighbors often make the feature pattern around a considered point more distinct, our distance weight $\omega_d(i, j)$ will assign more weights to those neighbors far away from the current point p_i and fewer weights to the closer neighbors. Specifically, $\omega_d(i, j)$ is defined as Eq. (4).

$$\omega_d(i, j) = \frac{1}{1 + e^{-\|\overrightarrow{v_{ij}}\|+3}} + 1. \tag{4}$$

Note that the carefully designed distance weight $\omega_d(i, j)$ varies in the interval $(1.0, 2.0)$, and the empirically added 3 in the exponential part makes $\omega_d(i, j)$ approximately start from 1.0. The distance weight is imposed on both components of Eq. (3) for each neighbor point. For feature points, their neighbors located at a far distance will draw the centroids deviating from the local surfaces. Those centroids of nonfeature points will still adhere to the local surfaces. Hence, those distant neighbors of a potential feature point make its feature pattern clearer. The red circles in Fig. 4 show that the centroids of points in different shape regions are dominantly determined by those distant neighbors. In Fig. 2(b) and Fig. 3(b), we can see that the NRLCs of feature points are pushed far away from the local surfaces.

Shape variation weight. The points at the convex or concave regions intuitively have long feature vectors. How-



(a) convex point (b) concave point (c) boundary point

Fig. 4. A 2D illustration of the local centroids c_i of a point p_i and its computation. (a), (b) and (c) show three different cases of a point located at the convex, concave and boundary regions, respectively. The red circles are the NRLCs, and the purples are the GCs.

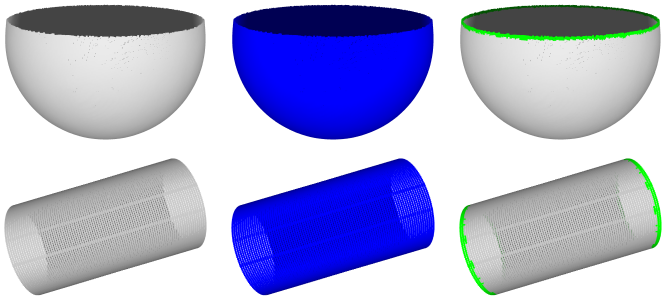


Fig. 5. Surfaces with identical high curvatures. Top-left: a semisphere; Bottom-left: a cylinder. Our feature-identified results without and with the shape variation weight are shown in the middle and right columns respectively.

ever, not all of these points should be viewed as geometric features. The degenerated case is the surface bending uniformly. Spherical (or cylindrical) surfaces are those cases with high curvatures but zero curvature derivatives (see examples in the first column of Fig. 5). Although all points of these shapes are located at the local curved surfaces, they do not have shape bending variations, so we should not consider them convex or concave features.

The identical high curvature makes the zero curvature derivative insufficient for feature detection. Lengagne et al. [3] and Ohtake et al. [22] used the zero-crossing of the curvature derivatives to identify features. It avoids the requirement of the fourth-order differential. To exclude the high curvature nonfeature points, the offset of the feature vector along the parallel direction of the considered normal should be compressed, but the offset along the perpendicular direction should be preserved. Vector decomposition presents an opportunity to treat these two components individually.

Herein, we introduce a shape variation weight $\omega_s(i,j)$ for the parallel component $\overrightarrow{v_{ij\|}}$ of each neighboring vector $\overrightarrow{v_{ij}}$. It is defined as a switch weight considering the local shape variation between the current point p_i and the neighbor p_j, namely,

$$\omega_s(i,j) = \begin{cases} 0, & \triangle\sigma(i,j) < \epsilon, \\ \alpha, & \triangle\sigma(i,j) \geq \epsilon, \end{cases} \tag{5}$$

where α is an adjustable magnification factor, $\triangle\sigma(i,j)$ represents an approximate estimation of the local shape variation for p_i and p_j, while ϵ is a specified small threshold indicating to what extent the two local surfaces should be deemed the same shape.

For a currently considered point pair p_i and p_j, its shape variation $\triangle\sigma(i,j)$ is approximately measured by the difference of two local normal variances $\sigma_n(i)$ and $\sigma_n(j)$, namely, $\Delta\sigma(i,j) = |\sigma_n(i) - \sigma_n(j)|$. Accordingly, the normal variance $\sigma_n(i)$ for point p_i can be formulated as Eq. (6).

$$\sigma_n(i) = \frac{\sum_{q\in\Omega(p_i)} \|\overrightarrow{n_q} - \overrightarrow{\mu_{n_i}}\|^2}{k}. \tag{6}$$

Here, $\overrightarrow{\mu_{n_i}}$ is the averaging normal of the neighboring points located at the local surface around p_i.

With the shape variation weight defined in Eq. (5), we can impose $\omega_s(i,j)$ on the parallel component $\overrightarrow{v_{ij\|}}$ for each neighboring vector $\overrightarrow{v_{ij}}$. Thus, the two components of feature vector $\overrightarrow{v_i}$ are finally accumulated through Eq. (7).

$$\begin{cases} \overrightarrow{v_{i\|}} = \sum_{j=1}^{k} \omega_d(i,j)\omega_s(i,j)\overrightarrow{v_{ij\|}}, \\ \overrightarrow{v_{i\perp}} = \sum_{j=1}^{k} \omega_d(i,j)\overrightarrow{v_{ij\perp}}. \end{cases} \tag{7}$$

Thus far, we have provided the necessary weights for the two components $\overrightarrow{v_{i\|}}$ and $\overrightarrow{v_{i\perp}}$. The feature vector $\overrightarrow{v_i}$ can be composed by $\overrightarrow{v_i} = \overrightarrow{v_{i\|}} + \overrightarrow{v_{i\perp}}$. Hence, we can use Eq. (1) to compute the NRLC c_i for current point p_i. Relying on the NRLCs, we implement our feature identification (presented in Section 4.2) on both the semisphere and cylinder models. The comparison results of feature identification without and with the shape variation weight are shown in the second and third columns of Fig. 5, respectively.

4.2 Feature identification

4.2.1 *Feature detection*

The local centroid intrinsically relates to the shape information of the local surface. NRLC substantially strengthens its feature descriptive capability. Intuitively, the length of a feature vector encodes the bending or bounding extent of the local surface, as shown in Fig. 4. Consequently, our method employs the length of the feature vector to find the potential geometric feature points.

For each point $p_i(i = 1, 2, \cdots, m)$ on a point cloud P, we compute the NRLC c_i and obtain its corresponding feature vector $\overrightarrow{v_i}$. Then, incrementally sort the centroids c_i depending on the length $d_i = \|\overrightarrow{v_i}\|$ of feature vector $\overrightarrow{v_i}$. Since the geometric features commonly show a very limited proportion, we take the $80th$ percentile of the length distribution of the feature vectors $Q_\lambda(d_i)(\lambda = 0.8)$ as the threshold ξ to judge each point. If the length d_i is greater than ξ, we regard the point p_i as a candidate feature point and forward it to the second stage of feature identification.

4.2.2 *Feature classification*

Our method utilizes the feature vector $\overrightarrow{v_i}$ from the current point p_i to its centroid c_i to classify the candidate feature point p_i. As shown in Fig. 4, the plus and minus signs as well as the zero value of the dot-product $< \overrightarrow{v_i}, \overrightarrow{n_i} >$ reasonably distinguish three types of essential shape features. Due to the shape variations with small scale and

the magnification of the feature vectors, it is insufficient to identify features just relying on this dot-product. If we consider the centroid position c_i as a random variable for a specified point p_i, c_i may distribute randomly on a unit sphere centered at p_i and expanded by $\overrightarrow{v_i}$ (a 2D case is depicted in Fig. 6). To effectively distinguish those notable feature candidates and discard the shallow feature noise, we design an NRLC-based classifier for geometric feature identification.

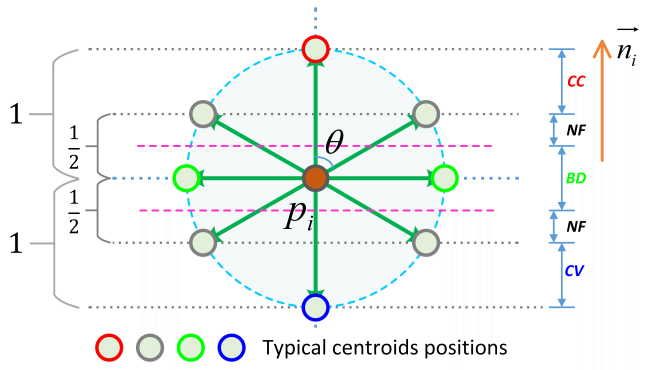


Fig. 6. A 2D illustration of the feature classification. A candidate feature point will be viewed as a convex, concave, boundary or nonfeature point if its centroid is located at one of the CV, CC, BD or NF regions accordingly.

We construct four indicating quantities to divide the potential distributions of the centroid c_i for a candidate feature p_i. If the angle expanded from the normal $\overrightarrow{n_i}$ to the feature vector $\overrightarrow{v_i}$ is denoted by θ, as shown in Fig. 6, we first define the concavity-related quantity CC as Eq. (8).

$$CC = |1 - cos\theta|. \tag{8}$$

CC describes the closeness of the centroid c_i to the most definite concave feature centroid (the red circle in Fig. 6) along the normal direction $\overrightarrow{n_i}$ of point p_i. Similarly, the convexity-related quantity can be defined as Eq. (9).

$$CV = |1 + cos\theta|. \tag{9}$$

CV means the distance from c_i to the most definite convex feature centroid (the blue circle in Fig. 6) along the direction $\overrightarrow{n_i}$. The quantity of the boundary is defined as Eq. (10).

$$BD = |0 - cos\theta| = |cos\theta|. \tag{10}$$

BD denotes the proximity extent from c_i to the most definite boundary feature centroid (the green circles in Fig. 6) along the direction $\overrightarrow{n_i}$. Finally, we define the last quantity, which indicates that a candidate would be a nonfeature point with a high probability, as Eq. (11).

$$NF = \min_{>0}\{\frac{1}{A} - cos\theta, \frac{1}{A} + cos\theta, \frac{1}{A}\}. \tag{11}$$

Here, A is the separating factor controlling the regions of CC, CV, BD and NF in Fig. 6. Those points whose centroids fall into the NF regions will be considered noisy feature points. Noisy feature points possess exaggerated feature vectors but shallow surface changes will be excluded outside the feature sets. Without a specific setting, we assign A as a constant 2 in our experiments.

We take the sum of these four quantities as $S = CC + CV + BD + NF$ and define a group of judging probabilities as Eq. (12).

$$\begin{cases} \mathscr{P}_{CC} = 1 - CC/S, \\ \mathscr{P}_{CV} = 1 - CV/S, \\ \mathscr{P}_{BD} = 1 - BD/S, \\ \mathscr{P}_{NF} = 1 - NF/S. \end{cases} \tag{12}$$

Thus, the category probabilities that the candidate feature p_i belongs to will form a set $\mathscr{P}_f(p_i) = \{\mathscr{P}_{CC}, \mathscr{P}_{CV}, \mathscr{P}_{BD}, \mathscr{P}_{NF}\}$. Let f be the mark of the feature category and take the value from the set $\{CC, CV, BD, NF\}$; then, we take the type of maximum probability as the label for candidate p_i.

$$label(p_i) = \arg\max_f \mathscr{P}_f(p_i). \tag{13}$$

After undergoing the above feature classification, except for a few proportions of noisy feature points, all the remaining candidate feature points will be identified as one of three basic types. Finally, we will produce three feature sets $P_F(F \in \{CC, CV, BD\})$ corresponding to the concave, convex and surface boundary points, respectively.

4.3 Feature refinement operators

To enhance the identified geometric features, we introduce a pair of refinement operators including assimilation and dissimilation. It resembles morphological dilation and erosion operations. The differences exist in two aspects. First, the refinement does not actually add (or remove) any 3D points but just relabels the undetected (or detected) feature points. Second, we place a restriction on the assimilation operator so that only the nearest nonfeature neighbors with shape variation will be relabeled as the feature points.

4.3.1 *Feature assimilation*

Intuitively, feature assimilation, which labels the undetected feature points, will expand the contour of the specified type of features. It usually strengthens the geometric features and makes the feature structure of the target more remarkable.

Given a specified feature set $P_F(F \in \{CV, CC, BD\})$, for each feature point $p \in P_F$, we search a neighbor $q_t \in \Omega(p) - \Omega(p) \cap P_F$ that satisfies Eq. (14),

$$q_t = \arg\min_q\{\|p - q\|\}, \tag{14}$$

where $q \in \Omega(p) - \Omega(p) \cap P_F$ is the nonfeature neighbors of feature point p. If q_t exists, we further check an additional condition shown in Eq. (15).

$$\begin{cases} \sigma_n(q_t) > \varphi, F = CV/CC, \\ d_{q_t} > \xi, F = BD. \end{cases} \tag{15}$$

Here, φ and ξ are the thresholds of normal variance and feature vector length, respectively. If the point q_t satisfies Eq. (15), which means it is located at a feature region, our method will assimilate it to be the feature point as the type of point p.

4.3.2 *Feature dissimilation*

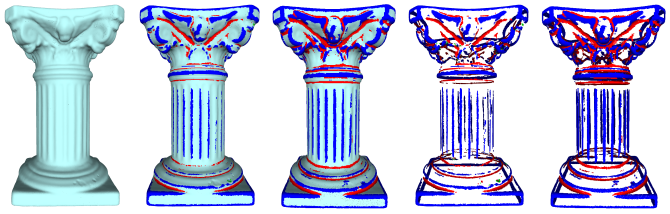
In contrast to assimilation, dissimilation will turn the peripheral labels of the feature region back to be nonfeatures.

Once we implement a dissimilating operation, generally, the contour of a specified feature type will be contracted.

The dissimilating operation is realized by enforcing the assimilating operation for nonfeature points. Specifically, we first construct a nonfeature point set P_{NF} by selecting the nonfeature neighbors for each feature point in P_F. Then, for each point $p' \in P_{NF}$, we find its nearest feature neighbor $q'_t \in P_F$ as Eq. (16),

$$q'_t = \arg\min_{q'} \{\|p' - q'\|\}, \tag{16}$$

where $q' \in \Omega(p') \bigcap P_F$ is the feature neighbor of the nonfeature point p'. q'_t is the closest feature neighbor of p' and will be dissimilated as a nonfeature point. Feature dissimilation can be used to reduce the detected noisy features.



(a) (b) (c) (d) (e)

Fig. 7. The combination result of feature dissimilation and assimilation. (a) The input column model. (b) The initial NRLC feature-identified result. (c) The result of feature dissimilation and assimilation. (d) Feature sets of subfigure (b). (e) Feature sets of subfigure (c).

In one pass of assimilation or dissimilation processing, we allow the assimilating (/dissimilating) target $q_t(/q'_t)$ to be repeatedly labeled by different feature (/nonfeature) neighbors. The repeatable labeling makes our assimilation (/dissimilation) operator expand (/contract) just one step outward (/inward) of the peripheral feature points per pass. Fig. 7 shows an example of the feature refinement for the identified feature sets.

5 Experiment Results

We conduct multiple groups of experiments to test the NRLC method. In our experiments, the identified convex, concave and boundary features are colored blue, red and green, respectively. All experiments are implemented on a computer with an Intel i7-9700K 3.60 GHz CPU with 16GB RAM.

5.1 Implementation and parameters

NRLC can be implemented following the pipeline of Fig. 2. It is also compared with the curvature method APSS [39], the statistical FPFH [40], variational PCA [32], feature linear-fitted F3D [42] and boundary-oriented technique angle criterion (AC) [48]. APSS is run on the Meshlab platform, FPFH and AC with Point Cloud Library (PCL), PCA and F3D with the codes released by their authors. Similar to other existing methods, NRLC also involves a set of parameters. Unless stated otherwise, these parameter settings simply abide by the following principles.

The first parameter is the k nearest neighbors. We set a variable nearest neighbor number k, which depends on the normal variance for different points. More specifically, each point will obtain a basic neighbor number k_0 and an additional variant neighbor number k_v multiplied by a decreasing weight ω, namely, $k(i) = k_0 + \omega(i) \times k_v$. k_0 and k_v are set as 60 and 40, respectively. ω is a shape variation weight and takes the value $\omega(i) = exp(-\sigma_n(i)/\sigma_{n\_max})$. Here, $\sigma_n(i)$ is the normal variance of point p_i, as defined in Eq. (6), and σ_{n_max} is the maximum $\sigma_n(i)$ among all $p_i \in P$. This setting makes NRLC capable of identifying the majority of prominent geometric features.

The second parameter of the NRLC method should be the magnification factor α which is adjustable and defined in Eq. (5). The larger the α value takes, the easier the identification of shallow features will be. Thus, it can be combined with k to identify different scales of features. Commonly, we set $\alpha > 1$ to assist feature identification. If the local shape variation between a neighbor and the central point is smaller than a threshold ϵ, we will set the shape variation weight as zero, which will avoid judging the constant high curvature surfaces as geometric features. We set ϵ as a constant 10^{-4} for all our experimental cases.

Another parameter is the separating factor A that we introduced in Eq. (11). If we set a large value of A, fewer noisy feature points will be excluded, and vice versa. In all our experiments, we set A to be a constant 2. In the process of assimilation, if the normal variance of a neighbor is larger than a specified threshold φ, this neighbor will be assimilated as a convex or concave feature point as the current point. We set φ to be the third quartile of the normal variance distribution for all our experimental cases. In the feature detection stage, λ determines how many points will be selected as the candidate feature points. We set λ as the $80th$ percentile of the length distribution of the feature vectors, as we stated in Section 4.2.1.

Except for k and α, most parameters are assigned constant values. The specific values of these parameters are reported in TABLE 1.

TABLE 1
Parameter setting of NRLC for different tested models. The value φ denotes the percentile of the normal variance distribution. λ takes the percentile of the length distribution of the feature vectors.

Model	Figure	Parameter					
		$k(i)$	α	ϵ	A	φ	λ
SVS1	Fig. 1		2	0.0001	2	0.75	0.80
sculpture	Fig. 2		2	0.0001	2	0.75	0.80
column	Fig. 7	$k_0 + \omega(i) \times k_v$,	2	0.0001	2	0.75	0.80
car	Fig. 8	$k_0 = 60$,	3	0.0001	2	0.75	0.80
SVS2	Fig. 9	$k_v = 40$	2	0.0001	2	0.75	0.80
scene1	Fig. 10		2	0.0001	2	0.75	0.80
scene2	Fig. 11		2	0.0001	2	0.75	0.80

5.2 Qualitative evaluation

We test NRLC on six different kinds of point clouds. These models, containing object-level targets, single view surfaces (SVS), indoor scenes, synthesized shapes, CAD and outdoor LiDAR point clouds, vary considerably in terms of the type, scale, shape feature and noise level.

Both the salient and shallow features that appear on object-level models (the sculpture in Fig. 2 and Fig. 3, the column in Fig. 7 and the car in Fig. 8) are identified by the NRLC method. Since a single view surface often does not form a complete object and includes many object parts, surface boundary features with a considerable proportion

will appear in SVS. In Fig. 1 and Fig. 9, NRLC identifies those surface boundary points as well as the convex and concave features concurrently.

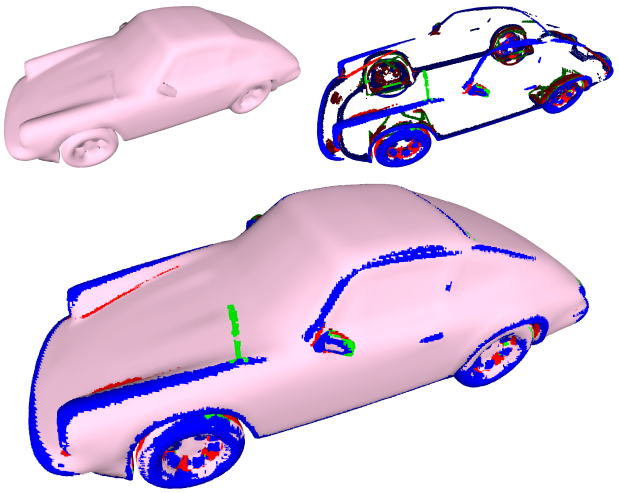


Fig. 8. The experiment on a car model. Top-left: the input point cloud. Bottom: NRLC identified result. Top-right: corresponding feature sets.

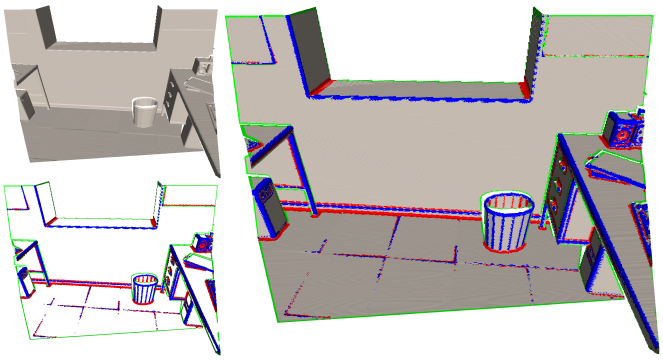


Fig. 9. Example of the second single view surface (SVS2). The top-left, right and bottom-left are the input point cloud, NRLC identified result and the feature sets, respectively.

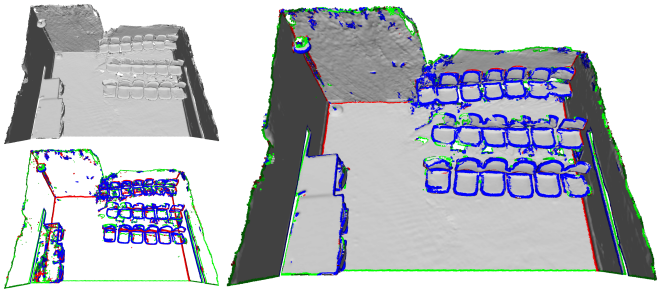


Fig. 10. Scene 1: a classroom point cloud with heavy noise. Top-left: the input model. Right: NRLC identified result. Bottom-left: feature sets.

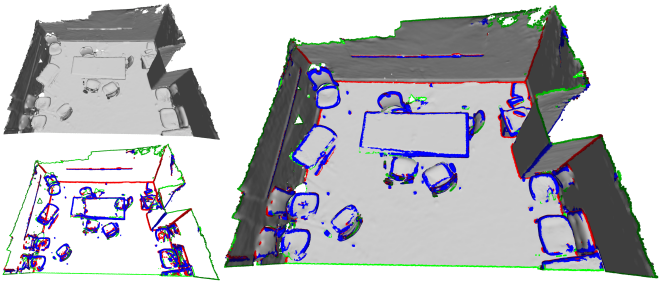


Fig. 11. Scene 2: another classroom model. Top-left: the input point cloud. Right: NRLC identified result. Bottom-left: feature sets.

Two indoor scenes (Fig. 10 and Fig. 11) contain different extents of noise. When a noisy point is viewed as the considered central point, it is prone to be incorrectly judged as a feature point by NRLC due to its inaccurate position and normal. As a neighbor point, since NRLC just employs the position statistics of the local neighbors, a noisy neighbor generally does not change the position relationship of the central point and its corresponding centroid. Hence, the NRLC method can smoothly treat indoor scenes in the presence of noise. In Fig. 10 and Fig. 11, although some noisy features are found incorrectly, the main patterns of the scene structure and the shape variations of the objects are all correctly identified as the corresponding geometric features.

In Fig. 12, we test F3D [42], PCA [32] and NRLC on an outdoor LiDAR building point cloud. F3D [42] just produces the linear edges. PCA detects the surface bending features and has no response to surface boundary features. Only NRLC identifies the complete scene structure with convex, concave and surface boundary features. In addition to these reported point cloud cases, some additional experiments are also given in the supplementary material.

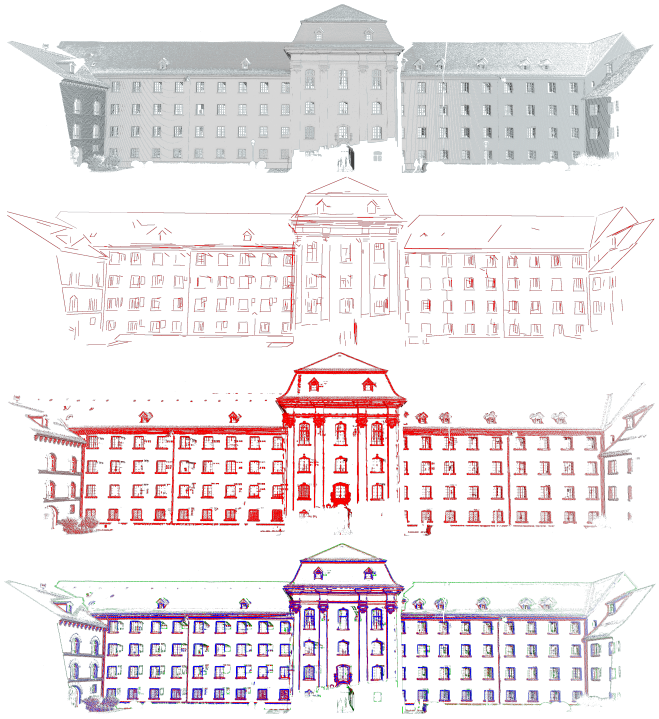


Fig. 12. Comparison experiment between F3D [42], PCA [32] and NRLC on the first outdoor building LiDAR point cloud. From top to bottom, these figures are the input model, feature sets of F3D, PCA and NRLC.

5.3 Quantitative evaluation

To verify the validity of the NRLC method, we conduct several groups of comparison experiments from different viewpoints.

5.3.1 Indirect comparisons

For most real reconstructed objects and scanned scenes, it is not easy to obtain their practical geometric features as the reference for evaluating the NRLC method. Hence, we

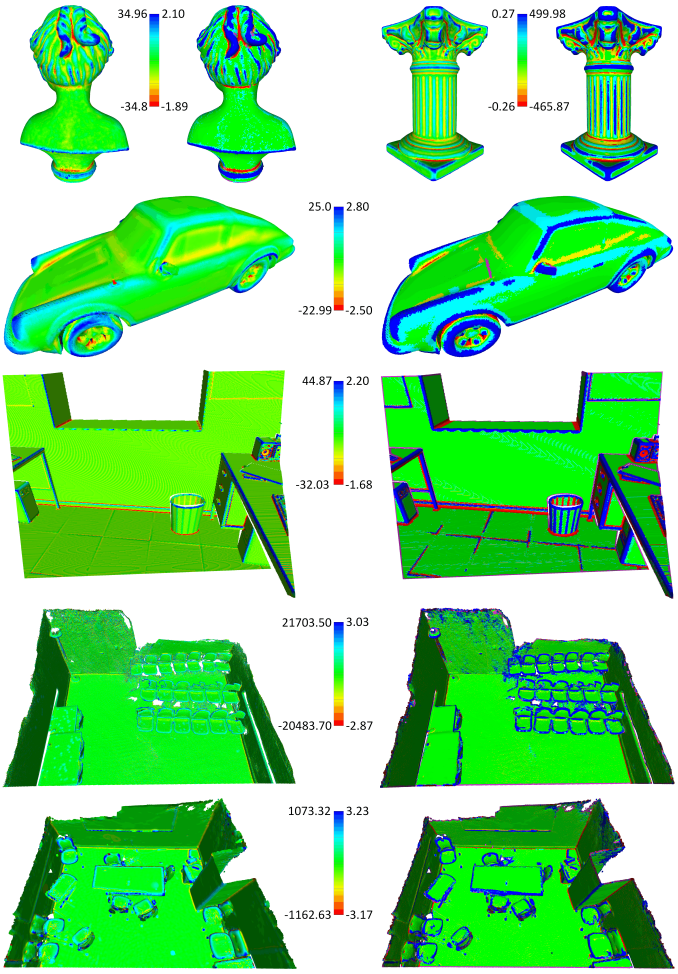


Fig. 13. Comparison of the feature descriptive power. In each case, the left result is produced by APSS, and the right is the result of NRLC. All models are colored from blue to red corresponding to the shape variation between the maximum convexity and maximum concavity. Boundary points in those models produced by NRLC are colored with the purple.

conduct an indirect comparison by analyzing the geometric features produced by NRLC to illustrate its capability.

Curvature-based methods can distinguish convexity and concavity. Unlike most existing curvature operators defined on meshes, APSS [39] can directly work on point clouds and compute the discrete mean curvature for each point. Therefore, we compare NRLC with APSS (which fits an algebraic sphere to obtain the local curvature radius for each point) to show their feature description capabilities. For six different models, the surface bending extents measured by both the mean curvature (APSS) and the parallel component of the feature vector (NRLC) are visualized in Fig. 13. The indirect comparisons in TABLE 2 are computed based on these two quantities. Note that for NRLC, the convex features have negative parallel components, and the concave features have positive parallel components. It is just opposite with APSS . For comparison convenience, we exchanged the signs of the positive and negative values for NRLC on the right side of each color bar of Fig. 13.

In TABLE 2, we use two indicators to evaluate the feature distinguishing capability of APSS and NRLC. The first is the coefficient of variation (CoV), which is defined as the standard deviation σ divided by the mean μ for the specified quantity, $CoV = \sigma/\mu$. It is a relative indicator and reflects the whole dispersion degree of the considered quantity. Compared to APSS, NRLC has larger CoVs on the tested models. It is beneficial for identifying different point classes from a holistic perspective. The other is the relative interclass distance (RICD), $RICD = |\mu_f - \mu_{Nf}|/\mu_{Nf}$. Here, μ_f and μ_{Nf} are the means of the feature and nonfeature classes, respectively. RICD represents the separability of two point classes. NRLC also presents higher RICDs between convex/concave features and nonfeature points in TABLE 2. Both CoV and RICD verify that NRLC shows higher feature contrast and separable capability. This is also qualitatively verified by the comparisons in Fig. 13.

Indoor scene 2 is dirty, and scene 1 involves even heavier noise on the wall. APSS has an abnormally large interval of the mean curvature (see the values beside the color bar in Fig. 13), which leads to order of magnitude differences in CoV and RICD (last 2 rows in TABLE 2). Thus, the CoVs and RICDs of APSS for the two scenes are unreliable. The noisy points with extremely high mean curvatures suppress the practical geometric features (last two rows of Fig. 13). Especially for scene 1 (the 4th row of Fig. 13), even no real feature points are demonstrated in the result of APSS. In contrast, the NRLC results show relatively stable CoVs and RICDs and highlight the practical geometric features.

TABLE 2
Evaluation of the feature description for APSS and NRLC. CoV and RICD are computed based on the mean curvature (APSS) and the signed length of the parallel component of feature vector (NRLC), respectively. APSS and NRLC take an equal number of the feature points to estimate the RICD.

Model	CoV		RICD(conc.)		RICD(conv.)	
	APSS	NRLC	APSS	NRLC	APSS	NRLC
sculpture	0.2106	0.2305	0.4403	0.4609	0.4165	0.5244
column	0.2591	0.2611	0.5093	0.5597	0.5147	0.5990
car	0.2134	0.2455	0.4565	0.5708	0.4239	0.5097
SVS2	0.2257	0.2384	0.4076	0.5481	0.5309	0.6129
scene1	0.0108	0.2180	0.0170	0.4422	0.0102	0.3949
scene2	0.0239	0.2006	0.0404	0.3983	0.0400	0.3708

5.3.2 *Direct comparisons*

For CAD models with explicit crease features, we directly compare the results produced by different methods with the manually labeled ground truth feature sets to evaluate their performances. Specifically, we implement PCA [32], APSS [39] and NRLC on three (the Fandisk, a Box and a Thorn) models with different levels of noise to identify surface bending features. The AC [48] and NRLC are also implemented on deficient point clouds to detect surface boundaries.

With the ground truth features, we use the *precision*, *recall rate* and the *F1-score* to evaluate the performance of feature identification for each method. *Precision* and *recall rate* are defined as $Pre = TP/(TP + FP)$ and $Rec = TP/(TP + FN)$, respectively. Here, TP is the acronym of true positives representing the number of correctly detected

points, FP denotes false positives representing the number of wrongly detected points, and FN stands for false negatives representing the number of false rejections. The *F1-score* is defined as $F1 = 2 \times \frac{Pre \times Rec}{Pre+Rec}$. The *precision* shows the percentage of the correctly detected number of feature points over the number of totally detected points. The *recall rate* reflects the complete extent of the feature points in the identified result. The *F1-score* is a balance between the *precision* and the *recall rate*.

In Fig. 14, we compare the surface bending features produced by PCA, APSS and NRLC with the ground truth feature sets. We combine the convex and concave feature points into one type of surface bending feature since PCA does not distinguish convexity and concavity. In this experiment, each result is generated with the same scale parameter (k = 10 well matches with PCA) and contains the same number of feature points (equal to the number of ground truth feature points). The related evaluations are given in TABLE 3. NRLC achieves the highest precision for each model.

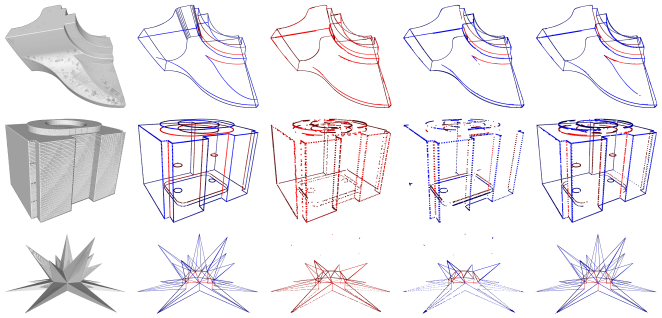


Fig. 14. Geometric feature sets of three CAD models produced by PCA, APSS and NRLC. The three methods take the same scale of nearest neighbor number ($k = 10$) and the same number of feature points (equal to the number of ground truth feature points). For each row, from left to right are the input model, the ground truth feature sets, and the results of PCA, APSS and NRLC, respectively.

TABLE 3
The *precision* of feature-identified results for PCA, APSS and NRLC methods on the original Fandisk, Box and Thorn models. Since each result takes the same number of the feature points as the ground truth feature sets, the *recall rates* actually equal the *precisions* so that we just list the *precisions* for each method. (Pre: *precision*)

Model	Pre		
	PCA	APSS	NRLC
Fandisk	0.7060	0.7084	**0.7479**
Box	0.4344	0.4096	**0.7694**
Thorn	0.4140	0.3429	**0.9827**

To compare the robustness, we implement PCA, APSS and NRLC on these models with four different levels of noise. In this group, we tune the parameters to generate the best *F1-scores* for each method under different noise levels. Note that the numbers of feature points corresponding to different methods vary and no longer equal the ground truth feature points. The results of the feature sets are shown in Fig. 15. The quantitative evaluations (*precisions*, *recall rates* and *F1-scores*) are reported in TABLE 4.

In Fig. 16, we compare the convex, concave features (produced by APSS and NRLC) and surface boundary features (produced by AC and NRLC) with their corresponding ground truth feature sets. The corresponding *precisions*, *recall rates* and *F1-scores* are listed in TABLE 5. The last three columns shown in Fig. 16 are the results with the best *F1-scores* that we could produce via the AC, APSS and NRLC methods. NRLC obtains both a higher *precision* and *recall rate* than APSS on convex and concave features. In the Fandisk model, in addition to the sharp edges, the prismatic

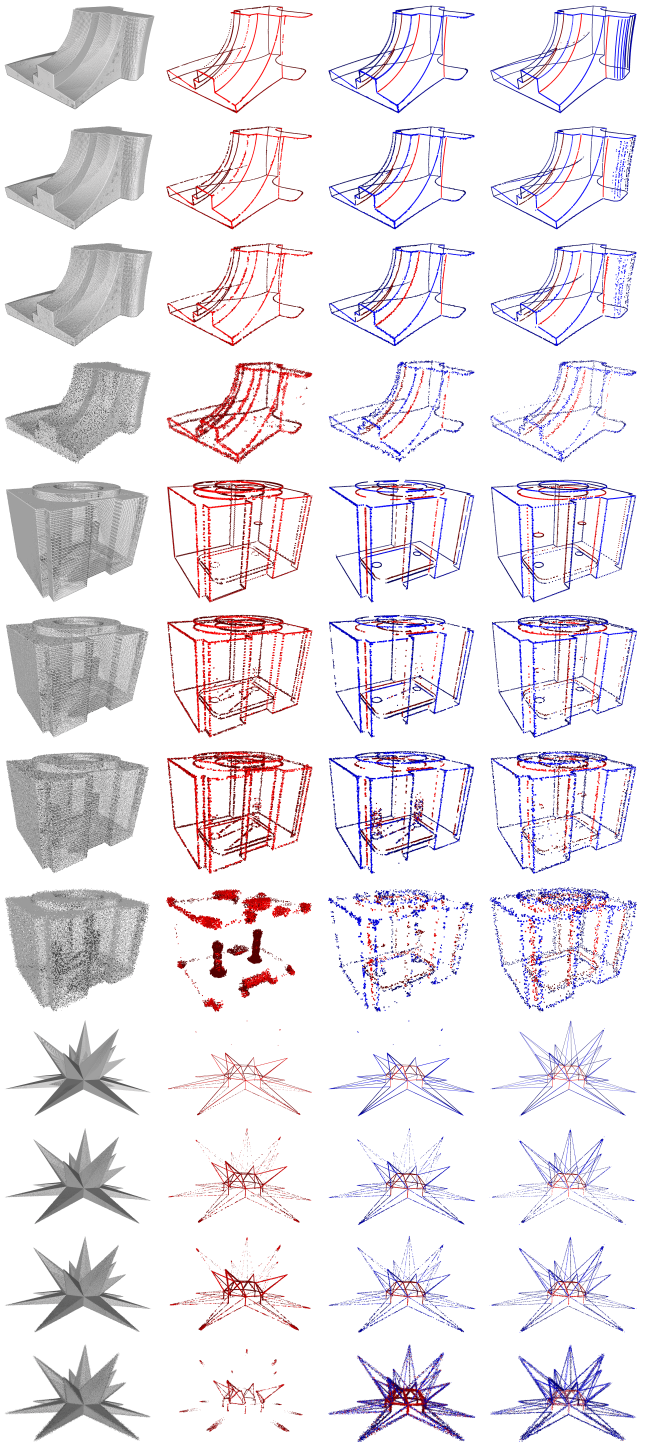


Fig. 15. The geometric feature-identified results on three CAD (Fandisk, Box and Thorn) models with four levels of noise. For each group, from top to bottom, the standard deviation σ of the Gaussian noise is 0.00, 0.08, 0.15 and 0.30. From left to right of each row, the point cloud and the feature sets produced by PCA, APSS and NRLC are shown.

TABLE 4
The *precisions*, *recall rates* and *F1-scores* of PCA, APSS and NRLC methods on the Fandisk, Box and Thorn point clouds with four different noise levels. The three indicators are counted on the surface bending points compared with the corresponding ground truth feature sets. The results do not distinguish the convexity and concavity. (NL: noise level; Pre: *precision*; Rec: *recall rate*; F1: *F1-score*)

Model	NL	Pre			Rec			F1		
		PCA	APSS	NRLC	PCA	APSS	NRLC	PCA	APSS	NRLC
Fandisk	0.00	0.8160	0.7507	**0.9743**	0.6574	0.7132	**0.9874**	0.7281	0.7315	**0.9808**
	0.08	0.7597	0.7379	**0.9504**	0.6528	0.7070	**0.8085**	0.7022	0.7222	**0.8738**
	0.15	0.6382	0.7000	**0.8927**	0.6088	0.6904	**0.7350**	0.6231	0.6951	**0.8062**
	0.30	0.1661	0.2786	**0.4941**	**0.5352**	0.3287	0.3810	0.2535	0.3016	**0.4303**
Box	0.00	0.4588	0.5556	**0.8334**	0.7709	0.8138	**0.8312**	0.5752	0.6604	**0.8323**
	0.08	0.2948	0.4267	**0.7277**	0.7515	**0.7870**	0.7322	0.4235	0.5534	**0.7299**
	0.15	0.2089	0.2989	**0.5807**	0.7415	**0.7712**	0.6492	0.3260	0.4309	**0.6130**
	0.30	0.0445	0.1887	**0.2796**	0.2152	0.2110	**0.3343**	0.0737	0.1993	**0.3045**
Thorn	0.00	0.4941	0.3779	**0.9820**	0.6448	0.7429	**0.9844**	0.5595	0.5009	**0.9832**
	0.08	0.1372	0.2703	**0.5632**	0.5850	**0.5950**	0.5726	0.2222	0.3718	**0.5676**
	0.15	0.0877	0.2074	**0.3574**	0.3739	0.4565	**0.5092**	0.1420	0.2853	**0.4200**
	0.30	0.0335	0.0610	**0.1008**	0.4289	**0.4475**	0.3482	0.0622	0.1074	**0.1563**

part also contains several edges formed by dihedrals with large angles. APSS neglects these shallow features (blue vertical lines), while NRLC finds them; see the comparison in Fig. 16. APSS does not respond to the surface boundaries. AC specifically designed for surface boundaries reaches high *recall rates* on Fandisk and Box models. It also obtains high precision on the quality point cloud (Thorn), while once the quality of the model is reduced (Box), its performance degrades. NRLC obtains *F1-scores* comparable to AC for surface boundaries on these three models.

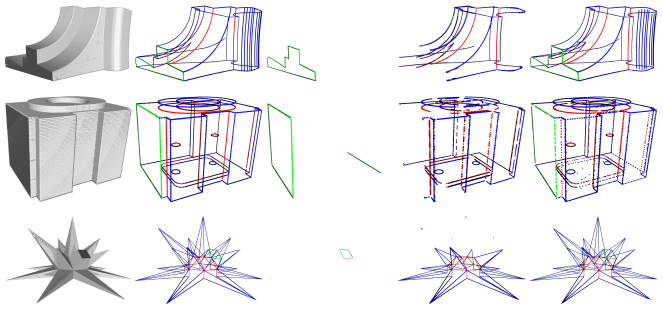


Fig. 16. Comparisons of geometric feature identification on three CAD point clouds with convex, concave and surface boundary feature points. From the left to right columns, these figures are the input models, the ground truth feature sets, the surface boundary features produced by AC, the bending features produced by APSS, and the bending and surface boundary features generated by NRLC.

In addition, we test NRLC with a set of fixed k to show the impacts of the scale parameter. The results and the corresponding evaluations are given in Fig. 17 and TABLE 6, respectively. The Fandisk model contains multiscale geometric features. Under small-scale parameter $k = 5$, NRLC identified shallow edges in the prismatic part due to the exaggerated feature vector (the first result in Fig. 17). As k increases, since those points around the sharp edges will obtain larger parallel components of feature vectors than

TABLE 5
Quantitative comparison of NRLC with AC for surface boundary features and APSS for surface bending (convex and concave) features. (Pre: *precision*; Rec: *recall rate*; F1: *F1-score*)

Model	Method	Convex			Concave			Surface Boundary		
		Pre	Rec	F1	Pre	Rec	F1	Pre	Rec	F1
Fandisk	AC	-	-	-	-	-	-	0.9551	**0.9953**	**0.9748**
	APSS	0.7155	0.6248	0.6670	0.8489	0.9907	0.9143	-	-	-
	NRLC	**0.9790**	**0.9839**	**0.9814**	**0.9621**	**1.0000**	**0.9807**	**0.9586**	0.9227	0.9403
Box	AC	-	-	-	-	-	-	0.7900	**1.0000**	**0.8827**
	APSS	0.6076	0.8652	0.7139	0.4700	0.7364	0.5738	-	-	-
	NRLC	**0.8348**	**0.8669**	**0.8505**	**0.7911**	**0.7781**	**0.7846**	**0.8448**	0.7778	0.8099
Thorn	AC	-	-	-	-	-	-	1.0000	0.8721	0.9317
	APSS	0.3111	0.7007	0.4309	0.9523	0.8503	0.8984	-	-	-
	NRLC	**0.9651**	**0.9861**	**0.9754**	**1.0000**	**0.9826**	**0.9912**	**1.0000**	**0.9186**	**0.9576**

those points located on the shallow edges, the main sharp edges gradually thicken, and the shallow edges disappear.

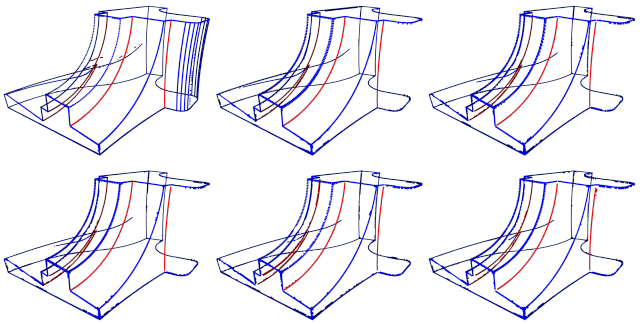


Fig. 17. Geometric feature identification of NRLC on the Fandisk point cloud with increasing nearest neighbor number k. Each result takes the same number of feature points as the ground truth feature sets. From left to right and top to bottom, these results are the feature sets produced by the NRLC method using an increasing nearest neighbor number, namely, $k = 5$, 10, 15, 20, 30 and 40.

TABLE 6
The *precisions* of the results produced by NRLC with increasing k on the Fandisk model.

k	5	10	15	20	30	40
Pre	0.9705	0.7479	0.7342	0.7334	0.7175	0.7156

5.4 Computational efficiency

The NRLC method exploits local statistics to compute the centroid and fully utilizes centroid analysis to identify geometric features. This statistic considers the point (/normal) pairs from a center to its surrounding neighbors so that it does not require the full relationship between all the point (/normal) pairs in a local neighborhood. Moreover, the normal variance can be calculated once in advance and then used in Eqs. (5) and (15). Feature classification, finding the maximum from four probabilities for each point, can be implemented in one loop. Hence, the feature identification algorithm can be implemented readily, and the time cost remains at a reasonably low level.

We compare NRLC with PCA [32] and FPFH [40] on different models to show the performance of the NRLC. The compared results of feature sets on six models are given in

Fig. 18. Since PCA and FPFH do not classify different types of geometric features, to facilitate comparison, three kinds of features recognized by NRLC in Fig. 18 are all colored red. Although PCA and FPFH identify most geometric features on object-level models and scene-level targets, the boundary features are all ignored. The NRLC method identifies the relatively complete structural features of each target, especially for the SVS. TABLE 7 reports the time costs for PCA, APSS, FPFH and NRLC. From TABLE 7, we can see that NRLC has higher efficiency than FPFH for all the tested models in Fig. 18 and that for large-scale point cloud models, NRLC takes remarkably less time than APSS and FPFH.

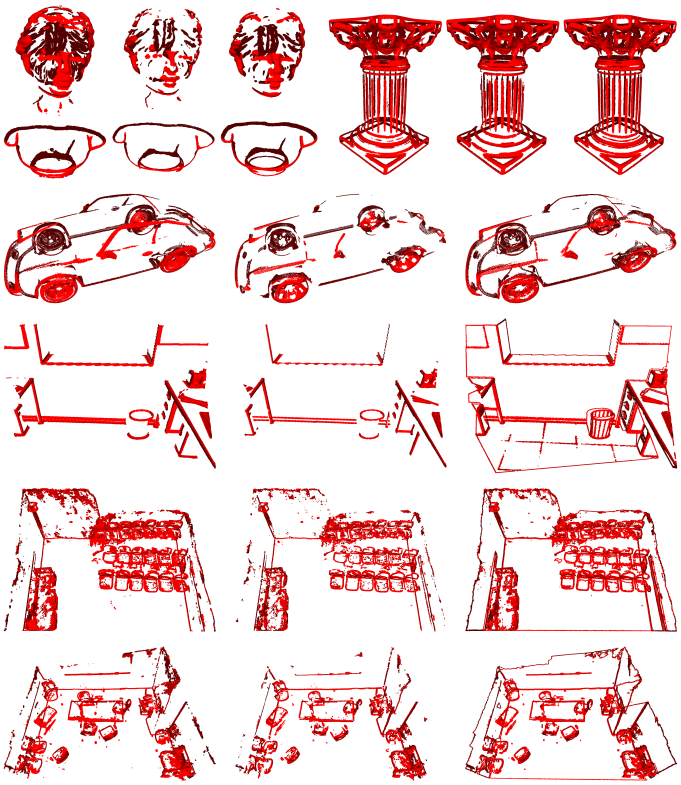


Fig. 18. Feature sets generated by PCA (left), FPFH (middle) and NRLC (right) on six different point cloud models. From left to right and top to bottom, these models are the sculpture, column, and car, SVS2 in Fig. 9, and indoor scenes in Fig. 10 and Fig. 11. Three types of features produced by NRLC are also colored red.

TABLE 7
The point numbers and time costs of PCA, APSS, FPFH and NRLC on different point clouds. The results of NRLC are produced without using any parallel computing techniques.

Model	Figure	Pnum	Time(s)			
			PCA	APSS	FPFH	NRLC
sculpture	Fig. 2	181,651	2.561	1.625	3.644	1.916
column	Fig. 7	262,653	3.508	2.464	8.319	2.975
car	Fig. 8	188,027	2.702	2.691	6.912	2.196
SVS2	Fig. 9	272,322	3.297	4.362	9.705	2.984
scene1	Fig. 10	2,135,724	22.746	189.667	88.872	22.912
scene2	Fig. 11	1,568,722	16.882	96.819	58.058	17.716

6 Discussions

NRLC intentionally increases the contrast of feature vectors just for highlighting and identifying geometric features. It is not an inherent surface measure and cannot precisely describe shape changes as curvature does. GCs resemble an integral invariant, which can reflect the practical surface bending extent. However, GCs show less scalability to identify the meso- and microscale features if we take a large-scale neighborhood (see Fig. 3(c)).

Geometric features are scale-dependent. Therefore, deciding which scale matters will depend on the specific application. NRLC can be readily transferred to a multiscale method by choosing different k values and setting varying factor α. However, in this paper, we concentrate on designing and realizing basic NRLC-based feature identification without paying attention to its multiscale applications.

The quality of the point cloud affects most geometric feature identification methods. Although NRLC possesses certain antinoise capabilities, its performance will inevitably decline as the noise level increases. If the original models contain extremely heavy noise and outliers, the specific filtering, resampling or cleaning operations (such as [4], [5], [53] etc.) should be employed as preprocessing to improve the quality of the input models. In addition, recognizing geometric features from point clouds with varying density is still an open problem, especially on sparse point cloud models. It requires feature-preserved reconstruction or upsampling operations as the preprocessing step before geometric feature identification.

Feature assimilation and dissimilation are very similar to the morphological dilation and erosion operators for images. This pair of operators can practically thin and thicken geometric features. Although feature dissimilation has certain extents of the denoising function, it cannot replace the special denoising processing.

7 Conclusions

We proposed a novel neighbor reweighted local centroid computational algorithm to identify geometric features for point cloud models. The NRLC of a specified point is defined by accumulating its neighboring vectors along two orthogonal directions with different weights. Through adaptively adjusting the weights for each neighbor point, our NRLC method substantially increases the contrast of the feature vectors, which strengthens the feature identification capability for local geometric centroids. Based on the defined NRLC, we designed a new probability set with four elements to identify three essential geometric features concurrently. Moreover, we introduced a pair of feature operators, namely, assimilation and dissimilation, to consolidate the identified geometric features. Many experimental results show that the NRLC method can identify convex, concave and surface boundary features simultaneously and effectively for point cloud models.

In the future, we would like to explore the possibility of improving the feature identification of NRLC for more complicated classes of geometric features, such as saddle surfaces and nonmanifold shapes. The relationship between curvature and local centroid (GC, even NRLC) is another interesting issue deserving to be investigated.

ACKNOWLEDGMENTS

The authors would like to thank the anonymous reviewers for their valuable comments and constructive suggestions. This work was supported by the NSFC (No. 61702422) and the Chinese Universities Scientific Fund (No. 2452018146). Corresponding author: Long Yang.

REFERENCES

[1] Ruimin Wang, Ligang Liu, Zhouwang Yang, Kang Wang, Wen Shan, Jiansong Deng, and Falai Chen. Construction of manifolds via compatible sparse representations. *ACM Transactions on Graphics (TOG)*, 35(2):14, 2016.

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

[3] Richard Lengagne, Olivier Monga, and Pascal Fua. Using crest lines to guide surface reconstruction from stereo. In *Proceedings of 3rd IEEE International Conference on Image Processing*, volume 2, pages 847–850. IEEE, 1996.

[4] A Cengiz Öztireli, Gael Guennebaud, and Markus Gross. Feature preserving point set surfaces based on non-linear kernel regression. In *Computer Graphics Forum*, volume 28, pages 493–501. Wiley Online Library, 2009.

[5] Hui Huang, Shihao Wu, Minglun Gong, Daniel Cohen-Or, Uri Ascher, and Hao Richard Zhang. Edge-aware point set resampling. *ACM Transactions on graphics (TOG)*, 32(1):9, 2013.

[6] Long Yang, Qingan Yan, Yanping Fu, and Chunxia Xiao. Surface reconstruction via fusing sparse-sequence of depth images. *IEEE Transactions on visualization and computer graphics*, 24(2):1190–1203, 2017.

[7] Forrester Cole, Kevin Sanik, Doug DeCarlo, Adam Finkelstein, Thomas Funkhouser, Szymon Rusinkiewicz, and Manish Singh. How well do line drawings depict shape? In *ACM Transactions on Graphics (ToG)*, volume 28, page 28. ACM, 2009.

[8] Doug DeCarlo, Adam Finkelstein, Szymon Rusinkiewicz, and Anthony Santella. Suggestive contours for conveying shape. In *ACM Transactions on Graphics (TOG)*, volume 22, pages 848–855. ACM, 2003.

[9] Doug DeCarlo and Szymon Rusinkiewicz. Highlight lines for conveying shape. In *Proceedings of the 5th international symposium on Non-photorealistic animation and rendering*, pages 63–70. ACM, 2007.

[10] Michael Kolomenkin, Ilan Shimshoni, and Ayellet Tal. Demarcating curves for shape illustration. In *ACM Transactions on Graphics (TOG)*, volume 27, page 157. ACM, 2008.

[11] Haowen Deng, Tolga Birdal, and Slobodan Ilic. Ppfnet: Global context aware local features for robust 3d point matching. In *Proceedings of the IEEE Conference on Computer Vision and Pattern Recognition*, pages 195–205. IEEE, 2018.

[12] J. Park C. Choy and V. Koltun. Fully convolutional geometric features. In *Proceedings of IEEE International Conference on Computer Vision*, pages 8957–8965. IEEE, 2019.

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

[14] Simon Christoph Stein, Markus Schoeler, Jeremie Papon, and Florentin Worgotter. Object partitioning using local convexity. In *Proceedings of the IEEE Conference on Computer Vision and Pattern Recognition*, pages 304–311. IEEE, 2014.

[15] Guo Yulan, Sohel Ferdous, Bennamoun Mohammed, Lu Min, and Wan Jianwei. Rotational projection statistics for 3d local surface description and object recognition. *International Journal of Computer Vision*, 105(1):63–86, 2013.

[16] Guo Yulan, Bennamoun Mohammed, Sohel Ferdous, Lu Min, and Wan Jianwei. 3d object recognition in cluttered scenes with local surface features: A survey. *IEEE Transactions on Pattern Analysis & Machine Intelligence*, 36(11):2270–2287, 2014.

[17] A. Mian, M. Bennamoun, and R. Owens. On the repeatability and quality of keypoints for local feature-based 3d object retrieval from cluttered scenes. *International Journal of Computer Vision*, 89(2-3):348–361, 2010.

[18] Daniel Cohen-Or, Chen Greif, Tao Ju, Niloy J Mitra, Ariel Shamir, Olga Sorkine-Hornung, and Hao Richard Zhang. *A sampler of useful computational tools for applied geometry, computer graphics, and image processing*. AK Peters/CRC Press, 2015.

[19] Christopher W Tyler. *Computer vision: from surfaces to 3D objects*. CRC Press, 2011.

[20] Qian-Yi Zhou, Jaesik Park, and Vladlen Koltun. Open3D: A modern library for 3D data processing. *arXiv:1801.09847*, 2018.

[21] Gabriel Taubin. Estimating the tensor of curvature of a surface from a polyhedral approximation. In *Proceedings of IEEE International Conference on Computer Vision*, pages 902–907. IEEE, 1995.

[22] Yutaka Ohtake, Alexander Belyaev, and Hans-Peter Seidel. Ridge-valley lines on meshes via implicit surface fitting. *ACM Transactions on graphics (TOG)*, 23(3):609–612, 2004.

[23] Andrew E. Johnson and Martial Hebert. Using spin images for efficient object recognition in cluttered 3d scenes. *IEEE Transactions on pattern analysis and machine intelligence*, 21(5):433–449, 1999.

[24] Radu Bogdan Rusu, Nico Blodow, Zoltan Csaba Marton, and Michael Beetz. Aligning point cloud views using persistent feature histograms. In *2008 IEEE/RSJ International Conference on Intelligent Robots and Systems*, pages 3384–3391. IEEE, 2008.

[25] Federico Tombari, Samuele Salti, and Luigi Di Stefano. Unique signatures of histograms for local surface description. In *European conference on computer vision*, pages 356–369. Springer, 2010.

[26] Yulan Guo, Mohammed Bennamoun, Ferdous Sohel, Min Lu, Jianwei Wan, and Ngai Ming Kwok. A comprehensive performance evaluation of 3d local feature descriptors. *International Journal of Computer Vision*, 116(1):66–89, 2016.

[27] Yutaka Ohtake, Alexander Belyaev, Marc Alexa, Greg Turk, and Hans-Peter Seidel. Multi-level partition of unity implicits. In *Acm Siggraph 2005 Courses*, pages 173–es. 2005.

[28] Szymon Rusinkiewicz. Estimating curvatures and their derivatives on triangle meshes. In *Proceedings. 2nd International Symposium on 3D Data Processing, Visualization and Transmission, 2004. 3DPVT 2004.*, pages 486–493. IEEE, 2004.

[29] Evangelos Kalogerakis, Patricio Simari, Derek Nowrouzezahrai, and Karan Singh. Robust statistical estimation of curvature on discretized surfaces. In *Symposium on Geometry Processing*, volume 13, pages 110–114, 2007.

[30] Mark Pauly, Markus Gross, and Leif P Kobbelt. Efficient simplification of point-sampled surfaces. In *Proceedings of the conference on Visualization'02*, pages 163–170. IEEE Computer Society, 2002.

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

[32] Keith Wei Liang Nguyen, A Aprilia, Ahmad Khairyanto, Wee Ching Pang, Gerald Gim Lee Seet, and Shu Beng Tor. Edge detection from point cloud of worn parts. 2018.

[33] Dena Bazazian, Josep R. Casas, and Javier Ruiz-Hidalgo. Fast and robust edge extraction in unorganized point clouds. In *2015 International Conference on Digital Image Computing: Techniques and Applications (DICTA)*, 2015.

[34] Shaobo Xia and Ruisheng Wang. A fast edge extraction method for mobile lidar point clouds. *IEEE Geoscience and Remote Sensing Letters*, 2017.

[35] Olivier Monga and Serge Benayoun. Using partial derivatives of 3d images to extract typical surface features. *Computer vision and image understanding*, 61(2):171–189, 1995.

[36] Helmut Pottmann, Johannes Wallner, Qi Xing Huang, and Yong Liang Yang. Integral invariants for robust geometry processing. *Computer Aided Geometric Design*, 26(1):37–60, 2009.

[37] Yong Liang Yang, Yu Kun Lai, Shi Min Hu, and Helmut Pottmann. Robust principal curvatures on multiple scales. In *Proceedings of the Fourth Eurographics Symposium on Geometry Processing, Cagliari, Sardinia, Italy, June 26-28, 2006*, 2006.

[38] Yu-Kun Lai, Qian-Yi Zhou, Shi-Min Hu, Johannes Wallner, and Helmut Pottmann. Robust feature classification and editing. *IEEE Transactions on Visualization and Computer Graphics*, 13(1):34–45, 2007.

[39] Gaël Guennebaud and Markus Gross. Algebraic point set surfaces. In *ACM Transactions on Graphics (TOG)*, volume 26, page 23. ACM, 2007.

[40] Radu Bogdan Rusu, Nico Blodow, and Michael Beetz. Fast point feature histograms (fpfh) for 3d registration. In *2009 IEEE International Conference on Robotics and Automation*, pages 3212–3217. IEEE, 2009.

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

[42] Xiaohu Lu, Yahui Liu, and Kai Li. Fast 3d line segment detection from unorganized point cloud. *CoRR*, abs/1901.02532, 2019.

[43] Mingyu Li and Koichi Hashimoto. Curve set feature-based robust and fast pose estimation algorithm. *Sensors (Switzerland)*, 17, 08 2017.

[44] Alexandre Boulch and Renaud Marlet. Deep learning for robust normal estimation in unstructured point clouds. *Computer Graphics Forum*, 35(5):281–290, 2016.

[45] Evangelos Kalogerakis, Derek Nowrouzezahrai, Patricio Simari, James McCrae, Aaron Hertzmann, and Karan Singh. Data-driven curvature for real-time line drawing of dynamic scenes. *ACM Transactions on Graphics (TOG)*, 28(1):11, 2009.

[46] Paul Guerrero, Yanir Kleiman, Maks Ovsjanikov, and Niloy J Mitra. Pcpnet learning local shape properties from raw point clouds. In *Computer Graphics Forum*, volume 37, pages 75–85. Wiley Online Library, 2018.

[47] Lequan Yu, Xianzhi Li, Chi Wing Fu, Daniel Cohen-Or, and Pheng Ann Heng. Ec-net: An edge-aware point set consolidation network. In *European Conference on Computer Vision*, 2018.

[48] Gerhard H Bendels, Ruwen Schnabel, and Rheinhard Klein. Detecting holes in point set surfaces. *Journal of WSCG*, 14:89–98, 2006.

[49] Pavel Chalmoviansk`y and Bert Jüttler. Filling holes in point clouds. In *Mathematics of Surfaces*, pages 196–212. Springer, 2003.

[50] Syeda Mariam Ahmed, Yan Zhi Tan, Chee Meng Chew, Abdullah Al Mamun, and Fook Seng Wong. Edge and corner detection for unorganized 3d point clouds with application to robotic welding. In *2018 IEEE/RSJ International Conference on Intelligent Robots and Systems (IROS)*, 2018.

[51] Evangelos Kalogerakis, Derek Nowrouzezahrai, Patricio Simari, and Karan Singh. Extracting lines of curvature from noisy point clouds. *Computer-Aided Design*, 41(4):282–292, 2009.

[52] Long Yang, Qingan Yan, and Chunxia Xiao. Shape-controllable geometry completion for point cloud models. *The Visual Computer*, 33(3):385–398, 2017.

[53] Yaron Lipman, Daniel Cohen-Or, David Levin, and Hillel Tal-Ezer. Parameterization-free projection for geometry reconstruction. In *ACM SIGGRAPH 2007 Papers*, SIGGRAPH '07, page 22–es, New York, NY, USA, 2007. Association for Computing Machinery.

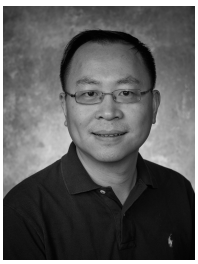
Tong Liu received his BSc degree in electronic commerce from Northwest A&F University, China, in 2019. Currently, he is working toward an MSc degree at the College of Information Engineering, Northwest A&F University. His research interests include computer graphics and machine learning.

Zhenhua Yang is an undergraduate student in computer science at Northwest A&F University. Her research interests are computer graphics and computer vision, and her current research focuses on the segmentation of point cloud models.

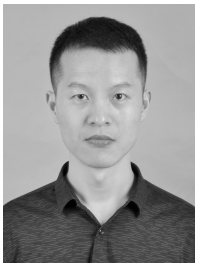
Shaojun Hu is currently an associate professor in the College of Information Engineering at Northwest A&F University. He is engaged in research on computer graphics and computer animation. He received a B.E. and an M.E. at Northwest A&F University, China, and a D.E. at Iwate University, Japan, in 2003, 2006, and 2009, respectively. He worked as a postdoc researcher at TU Wien (2014-2015) and a visiting researcher at the University of Tokyo (2016-2017). He is a member of CCF and ACM.

Zhiyi Zhang is currently a professor in the College of Information Engineering at Northwest A&F University. He is engaged in research on computer graphics and computer-aided design. He received a B.E. in the Department of Chemistry from Shanxi University, China, and M.E. and D.E. in Computer Science from Iwate University, Japan, in 1998, 2004, and 2007, respectively. He joined Northwest A&F University as an Associate Professor in 2007. He worked as a visiting researcher at Iwate University, Japan (2014-2015), supported by CSC.

Chunxia Xiao received BSc and MSc degrees from the Mathematics Department of Hunan Normal University in 1999 and 2002, respectively, and a PhD degree from the State Key Lab of CAD & CG of Zhejiang University in 2006. Currently, he is a professor in the Computer School, Wuhan University, China. From October 2006 to April 2007, he worked as a postdoc in the Department of Computer Science and Engineering, Hong Kong University of Science and Technology, and from February 2012 to February 2013, he visited the University of California-Davis for one year. His main interests include computer graphics, computer vision and machine learning. He is a member of IEEE.



Xiaohu Guo is a full professor of computer science at The University of Texas at Dallas. He obtained his Ph.D. degree in computer science from Stony Brook University, and a B.S. degree in computer science from the University of Science and Technology of China. His research interests include computer graphics, computer vision, medical imaging, and VR/AR, with an emphasis on geometric modeling and processing. He received the prestigious NSF CAREER Award in 2012.



Long Yang received his BSc degree in information and computing science from Chang'an University and his MSc degree in computer science from Northwest A&F University, respectively. After that, he received his PhD degree in computer graphics from Wuhan University in 2017. Currently, he is an associate professor in the College of Information Engineering at Northwest A&F University. His research interests include computer graphics, 3D computer vision and machine learning.