Computer-Aided Design 164 (2023) 103592

Contents lists available at ScienceDirect

Computer-Aided Design

journal homepage: www.elsevier.com/locate/cad

Robust and Accurate Feature Detection on Point Clouds

Zheng Liu [a], Xiaopeng Xin [b], Zheng Xu [b], Weijie Zhou [a], Chunxue Wang [c], Renjie Chen [d,*], Ying He [e]

[a] School of Computer Science, China University of Geosciences, Wuhan 430074, China
[b] National Engineering Research Center of Geographic Information System, China University of Geosciences, Wuhan 430074, China
[c] School of Sciences, Hangzhou Dianzi University, Hangzhou 310018, China
[d] School of Mathematical Sciences, University of Science and Technology of China, Hefei 230026, China
[e] School of Computer Science and Engineering, Nanyang Technological University, 639798, Singapore

ARTICLE INFO

Article history:
Received 30 March 2023
Received in revised form 28 June 2023
Accepted 9 July 2023

Keywords:
Feature detection
Point clouds
Tensor voting
Segmentation
Surface reconstruction
Feature line extraction

ABSTRACT

Geometric feature detection on surfaces is a crucial task for the characterization and understanding of geometry shapes. In this paper, we present a robust and reliable approach for accurately capturing local surface variations at different feature sizes within point clouds. To this end, we define a bilateral weighted centroid projection-based metric to quantify surface deviations. Based on the metric, we propose a structure-to-detail feature perception algorithm to accurately locate geometric features of varying sizes. Additionally, we use tensor analysis to extract boundary features. We evaluate our method on various object- and scene-level point clouds, demonstrating its superiority and versatility over existing techniques. Computational results show that our method is capable of identifying a wide range of geometric characteristics within point clouds, including complicated structures, rich textures, fine details, shallow curves, and geometric boundaries. We also validate the effectiveness of our approach on several downstream applications, including segmentation, surface reconstruction, and feature line extraction.

1. Introduction

With the increasing availability of 3D scanning technology, such as LiDAR and consumer-level depth sensors, point clouds have become a common representation for 3D data in a wide range of applications [1], including computer graphics [2,3], computer vision [4,5], autonomous driving [6], robotics [7], and 3D mapping [8,9], just name a few. However, processing and analyzing point clouds can be challenging due to their large size and unordered nature. A critical problem in point cloud processing is detecting geometric features, which are essential for understanding shape geometry and have a wide range of downstream applications, such as line and contour extraction [10,11], segmentation [12], reconstruction [13,14]. Although there are existing methods for feature detection, they often have limitations in terms of accuracy, robustness, and efficiency.

When processing a point cloud, humans tend to focus their attention on regions with noticeable geometric variations, such as sharp edges or curves, which are commonly referred to as geometric features. This is because the human perceptual system can abstract the surface shape using the features identified [15], allowing for a higher-level understanding of the entire surface [16]. For example, feature detection can be used to extract the boundaries of objects or parts in a point cloud, allowing for more accurate segmentation and reconstruction of the surface. As such, point cloud feature detection has become an essential and increasingly important task in various communities, including computer graphics and computer vision.

Over the past decade, point cloud feature detection has been extensively studied, and many methods have been proposed to detect geometric features on sampled surfaces. Although a considerable amount of progress has been made, it remains difficult to single out one method that can reliably and accurately identify all types of geometric features, especially in the presence of noise, complex structures, or fine details. For example, the method of Park et al. [17] works well on CAD models, but struggles to extract small-scale details and mid-scale features on non-CAD surfaces and is sensitive to noise. Chen et al. [18] proposed a multi-scale feature detection method that can robustly extract and refine features on CAD and manufactured surfaces in the presence of noise, but often fails to identify fine-scale and transition features on certain complex non-CAD objects. Nie [19] proposed a method that produces satisfactory results on both CAD and non-CAD surfaces, but tends to induce false features when dealing with noisy surfaces, especially in the case of large noise. The method of Liu et al. [20] relies on high-quality normals, making their results sensitive to the user-specified size of the geometric

* Corresponding author.
E-mail address: renjiec@ustc.edu.cn (R. Chen).

https://doi.org/10.1016/j.cad.2023.103592
0010-4485/

neighborhood. These limitations highlight the need for a new method that can accurately identify a wide range of geometric characteristics, including those that are challenging to detect with existing methods.

To address the aforementioned limitations of existing methods for point cloud feature detection, we propose a new and robust method that can locate various types of geometric features accurately and reliably. Our method is based on a novel saliency metric, which is formulated to be robust to noise and independent of region size, allowing for accurate measurement of the degree of surface variations within a given region. To efficiently identify features, we present a structure-to-detail feature perception algorithm that iteratively splits the given neighborhood region into multiple subregions, selects the subregion with the largest variation based on the proposed saliency metric, and checks the stopping criteria before further splitting the subregion if necessary. We also present a tensor analysis to extract boundary points. Our iterative approach can accurately locate features of varying sizes and complexities on point clouds, making it more versatile and effective than existing methods. We demonstrate the effectiveness of the proposed method on both synthetic and scanned data, showing that it outperforms existing methods in terms of accuracy and robustness. Finally, we demonstrate the effectiveness of our method on several downstream applications, including segmentation, surface reconstruction, and feature line extraction.

Our main contributions can be summarized as follows:

- We propose a structure-to-detail feature perception algorithm that employs a neighborhood splitting and selection strategy to accurately locate geometric features of varying sizes.
- We design a geometric saliency metric, in terms of centroid projection weighted by both point and normal deviations within a given neighborhood. This metric is robust against noise and flexible to the neighborhood size.
- We introduce a simple and effective tensor analysis to extract boundary features completely.

2. Related work

Identifying features on point clouds is a traditional yet essential problem that has been extensively studied. Due to the abundance of literature, reviewing all existing feature detection methods is beyond the scope of this paper. We focus on three categories: geometry-, statistic-, and learning-based methods. Our feature detection method belongs to the first category, on which we will concentrate, although we will also mention other relevant approaches.

Geometry-based methods. Many remarkable feature detection methods have been proposed for polygonal meshes. Hildebrandt et al. [21] used discrete differential operators on piecewise linear meshes to speed up the computation of high-order surface derivatives and filter them to improve the quality of extracted feature lines. Ohtake et al. [22] effectively extracted ridge-valley lines on meshes by combining multi-level implicit surface fitting and finite difference approximations. Clarenz et al. [23] proposed to use moment analysis of local surface patch neighborhood to extract feature regions. Liu et al. [24,25] introduced a total variation-based metric to detect geometric discontinuities on meshes during the denoising process.

Identifying features on point clouds is challenging due to the lack of connectivity and their unorganized nature. Pauly et al. [26] proposed a feature classification framework that can perform discrete surface analysis at multiple scales using a multiscale classification operator. Using the covariance matrices of Voronoi cells to estimate curvature and feature information on sampled surfaces, Mérigot et al. [27] defined a Voronoi covariance measure (VCM). Kalogerakis et al. [28] presented a novel method for estimating curvatures from point clouds and extracting lines of principal curvatures, which combines local surface variation measurements and curvature analysis to achieve robust and accurate results. Park et al. [29] proposed a multi-scale tensor voting method to identify sharp features on sampled surfaces. Liu et al. [1] detected features on point clouds using a bi-tensor voting scheme that combines normal- and point-tensor voting and is robust against noise. Some researchers directly applied point coordinates and normals to identify features. For example, Demarsin et al. [30] segmented the input point cloud into different regions based on the angle difference between the normals of these regions and extracted candidate features between them as feature points. Bazezian et al. [31] analyzed the eigenvalues of the covariance matrix and then applied clustering for fast feature detection. Béarzi et al. [32] explored a local frequency framework that can amplify the details of shapes by regional frequency analysis for identifying features. Guo et al. [33] used the local binary pattern (LBP) to exclude non-feature points from potential feature points. Chen et al. [18] presented a joint metric (PFR-LSV) based on the optimal fitting plane to exclude potential non-feature points while retaining correct features. Nie [19] proposed a local surface variation measurement, called the smooth shrink index (SSI), to combine with the Laplace smoothing and thinning technology to extract feature points accurately. Liu et al. [20] introduced a neighbor reweighted local centroid (NRLC) computational approach to detect geometric features on point clouds.

Statistic-based methods. Researchers have also attempted to describe geometric features from the statistical perspective [34–37]. Rusu et al. [34] proposed constructing 16D feature histograms with three descriptors to identify features by using the property that the feature histograms are unaffected by the location, orientation, and density of the point cloud. Guo et al. [35] presented RoPS, a local feature descriptor based on rotational projection statistics, to describe the local surface shape. This local descriptor is designed by rotating and projecting the neighbor points onto 2D planes and then calculating the low-order moments and entropy of the 2D distribution matrix on these planes. Yang et al. [36] developed the local feature statistics histogram (LFSH) to characterize matching feature points for point cloud registration. Zhang et al. [37] applied the region-growing method based on the Poisson distribution to extract features adaptively. Although the statistics-based approaches can produce remarkable results on sharply curved surfaces, these approaches may miss certain features when the sampled surfaces are corrupted by noise or contain rich geometric features.

Learning-based methods. Recently, several learning-based methods [38–42] have attracted increasing attention, which have the advantage of being parameter adjust-free. EC-Net proposed by Yu et al. [38] is the first work that identifies sharp edges on noisy point clouds with the neural network. PIE-Net, presented by Wang et al. [39], is designed for estimating parametric feature curves in an end-to-end manner, which is particularly effective in detecting features on CAD and man-made surfaces. Himeur et al. [40] proposed a lightweight neural network named PCEDNet, which can produce satisfactory feature detection results on more general surfaces, including scene data. Matveev et al. [41] proposed DEF for predicting sharp geometric features on CAD surfaces. Zhao et al. [42] proposed a deep learning approach that can learn to detect sharp feature points on raw point clouds with good noise resilience. The success of these learning-based methods crucially relies on a large amount of annotated data. However, manually labeling features on sampled surfaces is labor-intensive, especially for non-CAD surfaces and large-scale scenes.

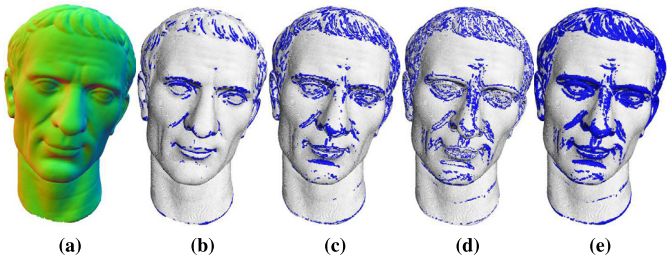


Fig. 1. Structure-to-detail feature detection results using varying neighborhood sizes. (a) Raw input (colormap based on the normals of input). (b), (c), and (d) show the detected structural, mesoscale, and detailed features using large-, mid-, and small-scale neighborhoods, respectively. (e) shows the union of those detected features from (b), (c), and (d). The feature points are plotted in blue, while the non-feature points are plotted in gray. (For interpretation of the references to color in this figure legend, the reader is referred to the web version of this article.)

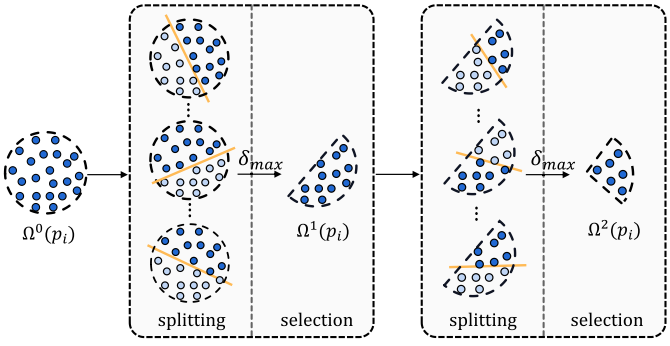


Fig. 2. Combining the neighborhood splitting and selection, we can progressively locate geometric features varying from large scale to small scale.

3. Method

We define features as surface characteristics, including complex structures, rich textures, fine details, shallow curves, and geometric boundaries. Features on sampled surfaces are usually of various scales, necessitating varying neighborhood sizes when detecting features ranging from large to small. Given a point cloud, we can directly extract its large-scale structures using the initial neighborhood, as shown in Fig. 1(b). However, using the original (large-scale) neighborhood, we cannot accurately identify mesoscale and small-scale features. To address this limitation, we propose splitting the initial neighborhood into multiple sub-neighborhoods and selecting the sub-neighborhood with the highest local surface variation, as depicted in Fig. 1(c). To further detect finer details, we continue to split the sub-neighborhood into more refined ones and then search for the one with the highest local variation, as illustrated in Fig. 1(d). As Fig. 1 shows, the size of the neighborhood has a significant impact on feature detection, ranging from structural to detailed features. Therefore, to detect structure-to-detail features as accurately as possible, we iteratively perform neighborhood splitting and selection.

3.1. Structure-to-detail feature perception

Let $\mathbf{P} = \{p_i\}_{i=1}^{M}$ be an unstructured point cloud sampled from a 2-manifold in $\mathbb{R}^3$, where M is the number of samples. Let $\mathbf{F}$ be the feature point set of the point cloud, and $\mathbf{V}$ be a potential feature point set initialized as $\mathbf{V} = \mathbf{P}$. Given a point $p_i \in \mathbf{P}$, we define its multiscale neighborhood as $\Omega^z(p_i)$. When $z = 0$, we get the original neighborhood of p_i as $\Omega^0(p_i) = \{p_j | \ \|p_j - p_i\| < r\}$, where r is the neighborhood radius. When z increases ($z \geq 1$), $\Omega^z(p_i)$ represents the refined sub-neighborhood of p_i, which allows locating the local shape variation more accurately. To identify structure-to-detail features on the point cloud, we iteratively perform the following steps:

(1) For each point $p_i \in \mathbf{V}$, we calculate its variation metric value δ_i by (1) using the current neighborhood $\Omega^z(p_i)$. Then, we put these variation values into the set φ and sort them in descending order.
(2) We move the points with larger variation values (the top β of the points in the set φ) into the current feature point set P_F and move the points with smaller variation values (the bottom γ in φ) into the current non-feature point set P_{nonF}, and then identify the remaining points as the updated potential feature point set $\mathbf{V}$. β and γ are percentage coefficients.
(3) For each point $p_i \in \mathbf{V}$, we split its current neighborhood $\Omega^z(p_i)$ into multiple sub-neighborhoods using the neighborhood splitting operation, which will be elaborated on in the following. Then, we choose the sub-neighborhood with the largest variation (δ_{max}) as the refined neighborhood $\Omega^{z+1}(p_i)$ of p_i. See Fig. 2 for a 2D illustration of our neighborhood splitting and selection strategy.

Algorithm 1 outlines the above procedure. Based on neighborhood splitting and selection, this algorithm iteratively performs the above three steps. When either the proportion of the identified feature points exceeds the percentage coefficient α, or the maximum iteration number is reached, the algorithm terminates and returns the feature point set $\mathbf{F}$. In most cases, three neighborhood updates are enough to produce satisfactory feature detection results. Thus, we set the maximum iteration number as 3 to balance computational complexity and feature identification accuracy. Fig. 1 shows an example.

Algorithm 1: Structure-to-detail feature detection

Input: point cloud $\mathbf{P}$
Output: feature point set $\mathbf{F}$
initialization: $\mathbf{F} = P_F = P_{nonF} = \varphi = \phi$, $\mathbf{V} = \mathbf{P}$, $z = 0$;
while ($|\mathbf{F}| < \alpha\,|\mathbf{P}|$ *or* $z < 3$) **do**
  (1) Calculate variation metric values for potential feature points
  for *each* $p_i \in \mathbf{V}$ **do**
    Compute δ_i using (1);
    $\varphi = \varphi \ \cup \{\delta_i\}$;
  end
  Sort φ in descending order;
  (2) Update the potential feature point, feature point, and non-feature point set based on the variation values
  $P_F \leftarrow$ extract the top β points from φ;
  $P_{nonF} \leftarrow$ extract the bottom γ points from φ;
  $\mathbf{V} = \mathbf{V} \setminus (P_F \cup P_{nonF})$, $\mathbf{F} = \mathbf{F} \cup P_F$;
  $P_F \leftarrow \phi$, $P_{nonF} \leftarrow \phi$, $\varphi \leftarrow \phi$;
  (3) Neighborhood splitting and selection for potential feature points
  for *each* $p_i \in \mathbf{V}$ **do**
    $\Omega^{z+1}(p_i) \leftarrow$ splitting $\Omega^z(p_i)$ and then selection;
  end
end
return F.

Neighborhood splitting operation. Given the current neighborhood $\Omega^z(p_i)$, we would like to refine it to match the shape of the curved region. Hence we design a splitting operation that repeatedly divides the current neighborhood into a series of sub-neighborhoods. Specifically, given a point p_i, we first compute the

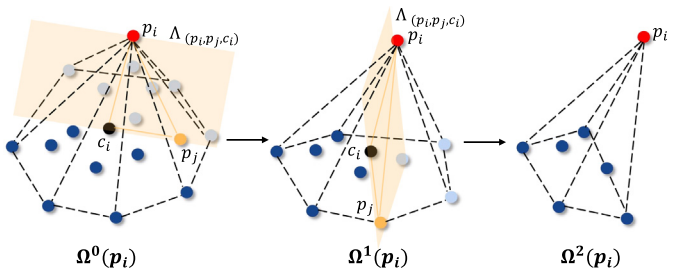


Fig. 3. The construction of multiscale neighborhoods $\Omega^0(p_i)$, $\Omega^1(p_i)$, and $\Omega^2(p_i)$ for a given point p_i. The partitioning plane $\Lambda_{(p_i,p_j,c_i)}$ is constructed from p_i, a neighbor point p_j of p_i, and the centroid of $\Omega^z(p_i)$, which is used to refine the current neighborhood.

centroid c_i of the current neighborhood as $c_i = \frac{1}{|\Omega^z(p_i)|}\sum_{j\in\Omega^z(p_i)} p_j$. Then for each neighbor point p_j, we can construct a plane $\Lambda_{(p_i,p_j,c_i)}$ that divides $\Omega^z(p_i)$ into two sub-neighborhoods. By repeatedly performing the splitting operation on the current neighborhood $\Omega^z(p_i)$, we can obtain $2K$ sub-neighborhoods with $K = |\Omega^z(p_i)|$. Then, for the next round of splitting, we can select the sub-neighborhood with the largest δ as $\Omega^{z+1}(p_i)$. The aforementioned descriptions are illustrated in Fig. 3.

3.2. Weighted Centroid Projection Metric

In order to robustly and flexibly quantify the geometric saliency of a given point p_i with different neighborhood sizes, we define the weighted centroid projection distance

$$\delta_i = k_i \cdot D_i, \tag{1}$$

where k_i is a bilateral weight that measures the overall flatness within a larger neighborhood around p_i, and the term D_i quantifies the geometric variation of a given neighborhood of p_i.

Centroid projection term. To measure the local curved variation of p_i within the current neighborhood $\Omega^z(p_i)$, we propose the centroid projection term as

$$D_i = |(c_i - p_i)^T n_i|, \tag{2}$$

where n_i is the normal of p_i estimated using local PCA, and c_i is the centroid of $\Omega^z(p_i)$. This term is formulated as the projection of $\overrightarrow{p_ic_i}$ onto normal n_i; see Fig. 4 for an illustration. The centroid projection distance increases as the local shape of the current neighborhood becomes more curved, and vice versa; see Figs. 4(a) and 4(b). With term (2), we can effectively identify feature points at varying scales of $\Omega^z(p_i)$. However, when the sampled surface is corrupted by noise, many noise points may be falsely identified as features if term (2) alone is used to detect features; see Fig. 5(b). The reason is that the projection of $\overrightarrow{p_ic_i}$ becomes significantly longer due to noise interference, causing p_i to be incorrectly identified as a feature point even though p_i actually lies in a flat region, as Fig. 4(c) illustrates.

It is known that discretizing the Laplace–Beltrami operator on triangular meshes via the cotangent formula yields the discrete mean curvature normal. On point clouds, we note that our local term D_i shares some similarities with the Laplace–Beltrami operator and that the local term is related to the mean curvature.

Bilateral weights. However, relying solely on mean curvature as a local metric can result in noisy points being misidentified as features. To counteract the impact of noise in feature detection, we propose a bilateral weight that incorporates the centroid projection term (2) in order to contrast feature points and noisy points, thereby making it easier to identify feature points correctly. This nonlocal weighting scheme is a novelty of our metric

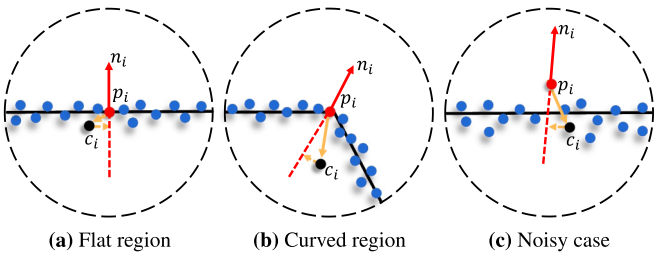


Fig. 4. Illustration of the centroid projection distance of $\overrightarrow{p_ic_i}$ onto n_i in (a) a flat region, (b) a curved region, and (c) a flat region corrupted by noise.

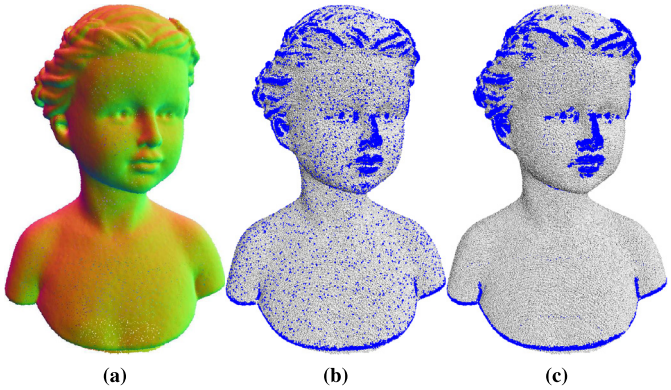


Fig. 5. Feature identification on noisy input using the centroid projection term (2). From left to right: noisy input, identification results without and with the bilateral weight (3), respectively. The combination of the projection term (2) and the weight (3) allows us to identify multiscale features while keeping noise points excluded.

and sets it apart from traditional curvature-based methods. This weight is determined by two factors

$$k_i = \kappa_i^n \cdot \kappa_i^p. \tag{3}$$

Weight (3) consists of two parts: κ_i^n is a monotonically increasing exponential function w.r.t. the overall normal difference within the neighborhood $\Omega^0(p_i)$, and κ_i^p is the same function w.r.t. the overall distance difference. Specifically, we have κ_i^n as

$$\kappa_i^n = e^{\rho_i/\mu_n}, \quad \rho_i = \sqrt{\frac{1}{|\Omega^0(p_i)|}\sum_{j\in\Omega^0(p_i)}(\theta_{ij} - \bar{\theta}_i)^2}, \tag{4}$$

where ρ_i is the standard deviation of the normal within $\Omega^0(p_i)$, μ_n is a scaling factor, $\theta_{ij} = \arccos\left(\frac{n_i}{||n_i||}\cdot\frac{n_j}{||n_j||}\right)$ is the angle difference between normal n_i and n_j, and $\bar{\theta}_i$ denotes the mean angle $\bar{\theta}_i = \frac{1}{|\Omega^0(p_i)|}\sum_{j\in\Omega^0(p_i)}\theta_{ij}$. Meanwhile, we have κ_i^p as

$$\kappa_i^p = e^{\pi_i/\mu_p}, \quad \pi_i = \sqrt{\frac{1}{|\Omega^0(p_i)|}\sum_{j\in\Omega^0(p_i)}(d_{ij} - \bar{d}_i)^2}, \tag{5}$$

where π_i is the standard deviation of the distances within $\Omega^0(p_i)$, μ_p is a scaling factor, d_{ij} is the distance from the neighbor point p_j to the least-squares fitting plane η_i, which is the plane with the minimum sum of squared distances to the neighbors of p_i. $\bar{d}_i$ is the mean distance $\bar{d}_i = \frac{1}{|\Omega^0(p_i)|}\sum_{j\in\Omega^0(p_i)} d_{ij}$. By multiplying term κ_i^n by κ_i^p, we can lessen the effect of incorrectly estimated normals on the bilateral weight (3) and make the computation more stable. Fig. 6 shows an example.

The bilateral weight (3) allows us to measure the overall flatness within a larger neighborhood ($\Omega^0(p_i)$) around p_i. It is worth

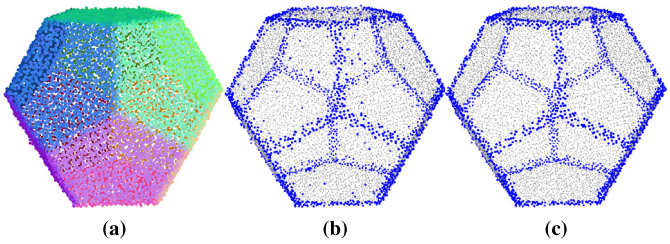


Fig. 6. (a) Noisy input. (b) Feature identified with $k_i = \kappa_i^n$. (c) Feature identified with $k_i = \kappa_i^n \cdot \kappa_i^p$.

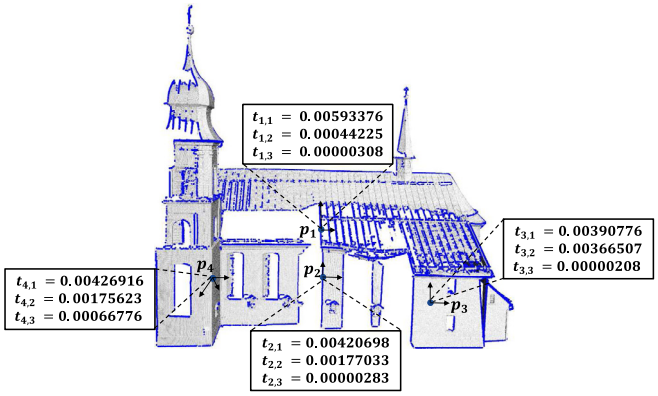


Fig. 7. Eigenvalues of the voting tensor for different points. p_1 and p_2 are boundary points with the eigenvalue distribution $t_{i,1} > t_{i,2} \gg t_{i,3} \approx 0$. p_3 is the point lying on the plane with $t_{3,1} \approx t_{3,2} \gg t_{3,3}$. p_4 is the feature point on the sharp edge with $t_{4,1} > t_{4,2} > t_{4,3}$.

noting that using a larger neighborhood can effectively reduce the impact of local noise on the overall measurement result. Weight (3) reduces the projection distance (2) for non-feature points while increasing it for true feature points, which makes it crucial for excluding non-feature points (e.g., noise points) in the identification result. Figs. 5(b) and 5(c) show its effect. As we can see, without this weight, many noise points are falsely identified as features. In contrast, the result with this weight is superior.

3.3. Detection of boundary features

In the previous subsections, we have identified geometric features on the sampled surface except for boundary feature points. Here, we apply a simple but effective tensor analysis to extract boundary features.

Given a point p_i, its voting tensor is constructed as the weighted sum of the coordinates covariance matrices of the neighboring points,

$$T_i = \frac{\sum_{j\in N(p_i)} w(p_i, p_j)(p_j - \bar{p}_i) \otimes (p_j - \bar{p}_i)}{\sum_{j\in N(p_i)} w(p_i, p_j)}, \tag{6}$$

where the weight $w(p_i, p_j) = e^{-\|p_i - p_j\|^2 / r_b^2}$ is a Gaussian function that decreases as the distance between the two points increases, $N(p_i) = \{p_j \mid \|p_j - p_i\| < r_b\}$ is the neighboring point set, and $\bar{p}_i = \frac{1}{|N(p_i)|}\sum_{j\in N(p_i)} p_j$. The neighborhood radius is set as $r_b = 2r$ to ensure the following tensor analysis is reliable. Symbol $\otimes$ denotes the outer product $(p_j - \bar{p}_i)(p_j - \bar{p}_i)^T$. As tensor (6) is symmetric and positive semidefinite, it can be diagonalized with eigenvalue decomposition,

$$T_i = \sum_{m=1}^{3} t_{i,m} e_{i,m} \otimes e_{i,m}, \tag{7}$$

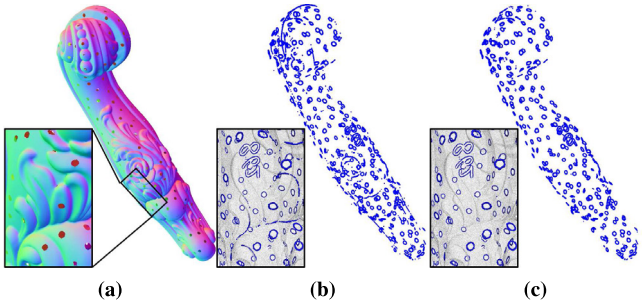


Fig. 8. Boundary detection results on a scanned surface. From left to right: raw surface, boundary detection results without and with $R_{i,2}$, respectively. With the addition of $R_{i,2}$, non-boundary points can be excluded effectively.

Table 1
Values of $R_{i,1}$, $R_{i,2}$, and B_i for points marked in Fig. 7.

	$R_{i,1}$	$R_{i,2}$	B_i
p_1	0.860861	0.986071	0.848870
p_2	0.407457	0.996803	0.406154
p_3	0.032039	0.998865	0.032003
p_4	0.375448	0.239553	0.089940

where $t_{i,m}$ and $e_{i,m}$ are the corresponding eigenvalues and eigenvectors with the eigenvalues assumed to be sorted in decreasing order $t_{i,1} \geq t_{i,2} \geq t_{i,3} \geq 0$.

In order to correctly extract boundary points, we analyze the relationship between the eigenvalues of the voting tensor (6) and propose the boundary detection operator on p_i as

$$B_i = R_{i,1} \cdot R_{i,2} = \frac{t_{i,1} - t_{i,2}}{t_{i,1} + t_{i,2} + t_{i,3}} \cdot \frac{t_{i,2} - 2t_{i,3}}{t_{i,2}}. \tag{8}$$

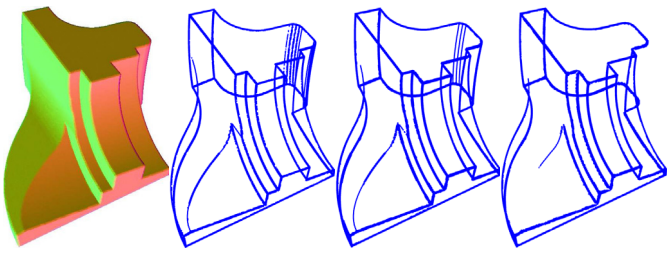
Term $R_{i,1}$ refers to a linear shape factor [11,43], which can be used to detect boundary points with the neighborhood forming a linear shape and exclude the non-boundary points lying on smoothly curved regions.

Table 1 lists the values of $R_{i,1}$ for the examples in Fig. 7. As we can see, p_1 is a boundary point with surrounding points lying on a line in the direction of the eigenvector $t_{1,1}$, and its $R_{i,1}$ value is closer to 1. In contrast, p_4 is a plane point with $R_{i,1}$ value closer to 0. However, using only factor $R_{i,1}$, we cannot distinguish the boundary point p_2 that lacks partial neighborhood from the feature point p_4 lying on the sharp edge; see $R_{i,1}$ values of p_2 and p_4 in Table 1.

Through carefully examining the eigenvalue distribution of boundary points, we formulate a rule for the boundary points: $t_{i,2} \gg t_{i,3} \approx 0$. Based on this rule, we introduce $R_{i,2}$ in (8), which allows us to clearly distinguish boundary and non-boundary points, since boundary points have much larger B_i values than non-boundary points, as the B_i column in Table 1 shows. Fig. 8 shows the extracted boundary features from a scanned surface. The result obtained using only $R_{i,1}$ is a little messy as some edge points other than boundary points are detected; see Fig. 8(b). With the addition of $R_{i,2}$, the boundary and non-boundary points can be clearly distinguished; see Fig. 8(c). Thus, we can classify boundary points as $\mathbf{F}_b = \{p_i \in \mathbf{P} \mid B_i > \epsilon_b\}$, where ϵ_b is a threshold in the range of [0.3, 0.6].

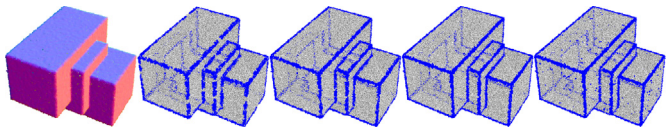
4. Experiments and discussions

In this section, we present experimental results on a variety of point clouds visually and numerically, including CAD, non-CAD, raw scanning surfaces, and scene data. The tested surfaces are corrupted by either synthetic or raw noise. The synthetic noise



(a) Input (b) $r = 5\bar{r}$ (c) $r = 10\bar{r}$ (d) $r = 20\bar{r}$

Fig. 9. Feature detection results for varying r. We keep the same number of feature points in each result. From left to right: clean input, and results with increasing r.



(a) Input (b) $\mu_n = 0.01$ (c) $\mu_n = 0.07$ (d) $\mu_n = 0.25$ (e) $\mu_n = 1.0$

Fig. 10. Feature detection results for varying μ_n. From left to right: noisy input (1% noise), and results with increasing μ_n.

is generated by a zero-mean Gaussian function with the standard deviation proportional to the diagonal of the axis-aligned bounding box of the ground truth. We compare our feature detection method with the state-of-the-arts, including tensor voting (TV) [29], smooth shrink index (SSI) [19], edge-aware point set consolidation network (EC-Net) [38], neighbor reweighted local centroid (NRLC) [20], and local surface variation and plane fitting residual (PFR-LSV). For PFR-LSV and EC-Net, we run the codes provided by the authors. For TV, SSI, and NRLC, we implemented them in C++ based on the description in the published articles.

4.1. Parameter setting

Our feature detection method has a few parameters that should be tuned in order to produce satisfactory results. In the following, we will discuss the parameter roles of our method.

Parameter r is the radius of neighborhood $\Omega^0(p_i)$, which affects the accuracy of feature extraction. As Fig. 9 demonstrates, the detected shallow edges gradually disappear as r increases. Furthermore, Figs. 9(b) and 9(c) show that satisfactory results can be produced for r in the range $[5\bar{r}, 10\bar{r}]$, where $\bar{r} = \frac{1}{|\mathbf{P}|}\sum_{p_i\in\mathbf{P}}(\min_{p_j\in\mathbf{P}\setminus\{p_i\}}\|p_i - p_j\|)$ is the average distance between each point and its nearest neighbor. This demonstrates that our method is insensitive to the perturbation of r.

Parameter μ_n is a scaling factor for controlling bilateral weight (3) to exclude non-feature points (e.g., noise points) in detection results. For any sampled surface, there exists a range of μ_n that can yield satisfactory results ($[0.07, 0.25]$ in this example); see Figs. 10(c) and 10(d). Too small μ_n leads to excessively large weight (3), causing many features to be missed; see Fig. 10(b). In contrast, too large μ_n causes underweighting, resulting in misidentified noise points; see Fig. 10(e). Similar to μ_n, μ_p also influences weight (3) for excluding non-feature points. μ_p plays an important role when the estimated normals are inaccurate, and its value is suggested to be in the range of $[0.2\bar{r}, 0.8\bar{r}]$.

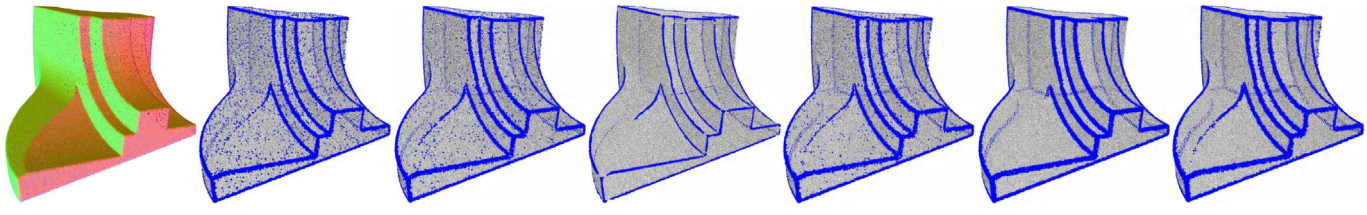
Parameters α, β, and γ are percentage coefficients used in Algorithm 1 for feature extraction. α determines the specific percentile extracted as features during iteration. A small α produces extraction results containing more large-scale structures and fewer small-scale details, while a large α tends to extract more fine details. However, when the surface is severely corrupted by noise, large α may cause noise points to be misidentified as features. β is the extracting percentage coefficient in each iteration. We found that it tends to produce appealing results in most cases by setting $\beta = \frac{|\mathbf{P}|}{2|\varphi|}\alpha$ in the first iteration and $\beta = \frac{|\mathbf{P}|}{4|\varphi|}\alpha$ in the following iterations. γ is the excluding percentage coefficient in each iteration, whose value is suggested to be in the range of $[0.15, 0.25]$. Too large γ may cause some detailed features to be classified as non-feature points, while too small γ will increase the algorithm runtime.

4.2. Qualitative comparison

Feature detection on CAD surfaces. In Fig. 11, we compare the feature detection results for a noisy CAD surface containing sharp and shallow features. Because both features and noise are high-frequency signals, TV, SSI, and NRLC are unable to distinguish them properly, causing many noise points being misidentified as features and extracted incorrectly; see Figs. 11(b), 11(c), and 11(e). In contrast, EC-Net, PFR-LSV, and our method are more robust against noise perturbation. PFR-LSV uses non-local surface variance to reduce noise impact, allowing it to identify most features even in noisy environments. However, the non-local property of PFR-LSV may blur shallow features, causing them to be misidentified as non-feature points; see Fig. 11(f). Although EC-Net produces more delicate feature extraction results, it ignores many important geometric features (e.g., corners) and causes discontinuity in sharp edges; see Fig. 11(d). As Fig. 11(g) shows, our method produces visually the best result with the most geometric features being extracted effectively.

Fig. 12 shows a comparison on a CAD surface contaminated by considerable noise. Once again, TV, SSI, and NRLC produce results with excessive noise points. The result of EC-Net exhibits feature discontinuities and residual noise. As shown in Fig. 12(e), PFR-LSV appears to miss some key features on sharp edges and corners, and mistakes many non-feature points on smooth regions. Our method outperforms the compared methods in extracting sharp edges and corners completely while preventing the introduction of artifacts, as shown in Fig. 12(g).

Fig. 13 illustrates the results on Kinect scanned data. TV and SSI extract false features in bumpy regions, whereas EC-Net produces a degraded result with feature discontinuities. Although PFR-LSV can effectively identify features while excluding noise



(a) Input (b) TV (c) SSI (d) EC-Net (e) NRLC (f) PFR-LSV (g) Ours

Fig. 11. Comparison of feature detection results on Fandisk (0.3% noise). Our method is able to detect shallow edges while reducing noise interference.

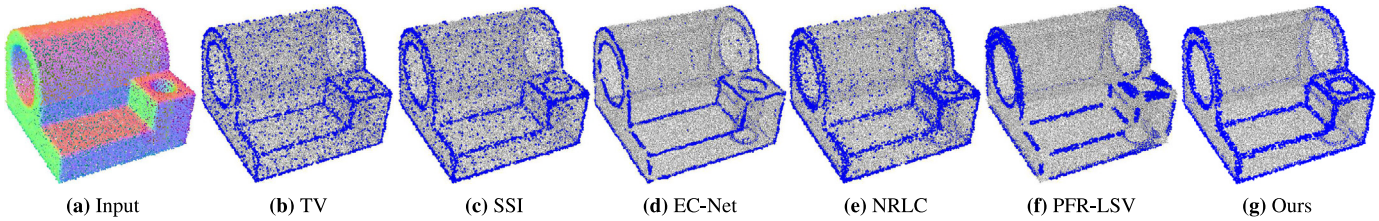


Fig. 12. Comparison of feature detection results on Joint (1.5% noise). Our method is able to completely detect features on the surface corrupted by considerable noise.

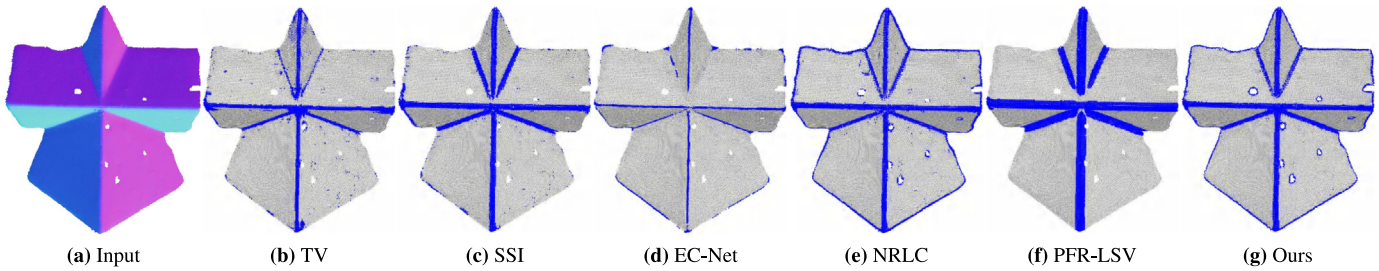


Fig. 13. Comparison of feature detection results on data acquired by Kinect. Our method is able to detect boundary points and avoid redundant extraction results.

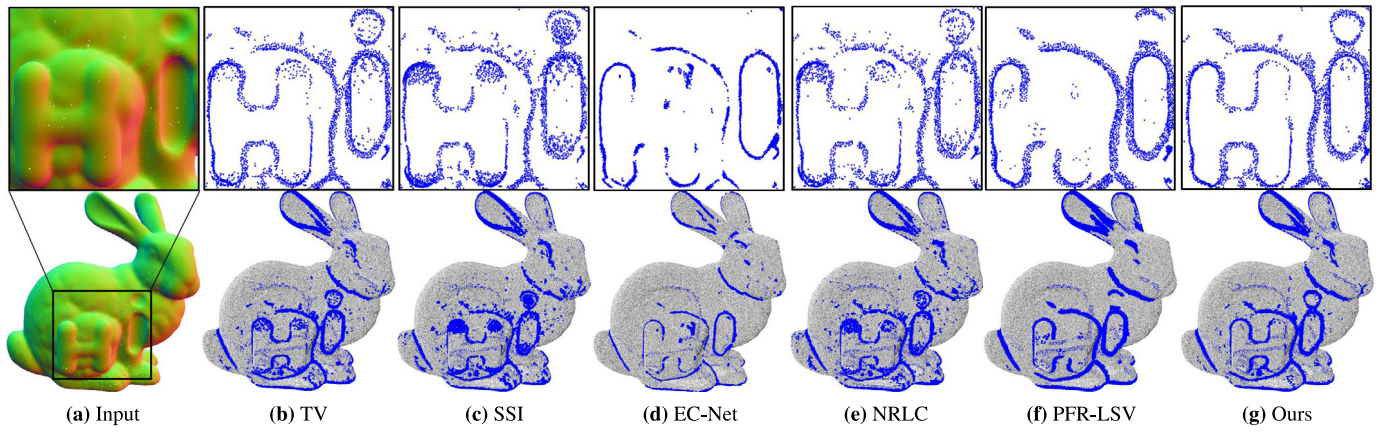


Fig. 14. Comparison of feature detection results on Bunny-Hi (clean). Our method extracts more faithful features with the least noticeable artifacts.

points, it is prone to detecting redundant points around underlying sharp edges, as seen in Fig. 13(f). Except for our method and NRLC, the others fail to extract boundary features. However, the result of NLRC is slightly messy with bumps, and the extracted boundaries of holes are incomplete, as shown in Fig. 13(e). In contrast, our method produces a result that is free of visible artifacts and contains all the features on the underlying surface.

Feature detection on non-CAD surfaces. Fig. 14 shows the results of feature detection on a non-CAD surface with rich geometric details. While TV, SSI, and NRLC can detect different levels of features, they misidentify many non-feature points on smoothly curved regions to varying degrees. EC-Net and PFR-LSV ignore small-scale features and fine details, which is particularly severe for EC-Net, as shown in Fig. 14(d). In contrast, our method produces a visually superior result with more faithful features, demonstrating its effectiveness in detecting features on non-CAD surfaces.

Fig. 15 compares the results of feature detection on a noisy surface with multi-scale features. As expected, TV, SSI, and NRLC fail to distinguish between noise and feature points. EC-Net and PFR-LSV can detect large-scale structures but ignore small-scale textures, as shown in Figs. 15(d) and 15(f). In contrast, our method yields visually superior results containing more textures than other methods, as demonstrated in the close-up views in Fig. 15(g).

Feature detection on scene data. To demonstrate the versatility of our method, we test it on scene data. As Fig. 16 shows, the competing methods exhibit the aforementioned issues that degrade the quality of feature extraction results. In contrast, our method produces a more appealing result with faithful structures and details, while having few artifacts, as seen in the close-up view in Fig. 16(g).

4.3. Quantitative evaluation

For the quantitative evaluation, we compare our method to TV, SSI, NRLC, and PFR-LSV, and present the results in Table 2. Due to the lack of ground truth, we manually label features of

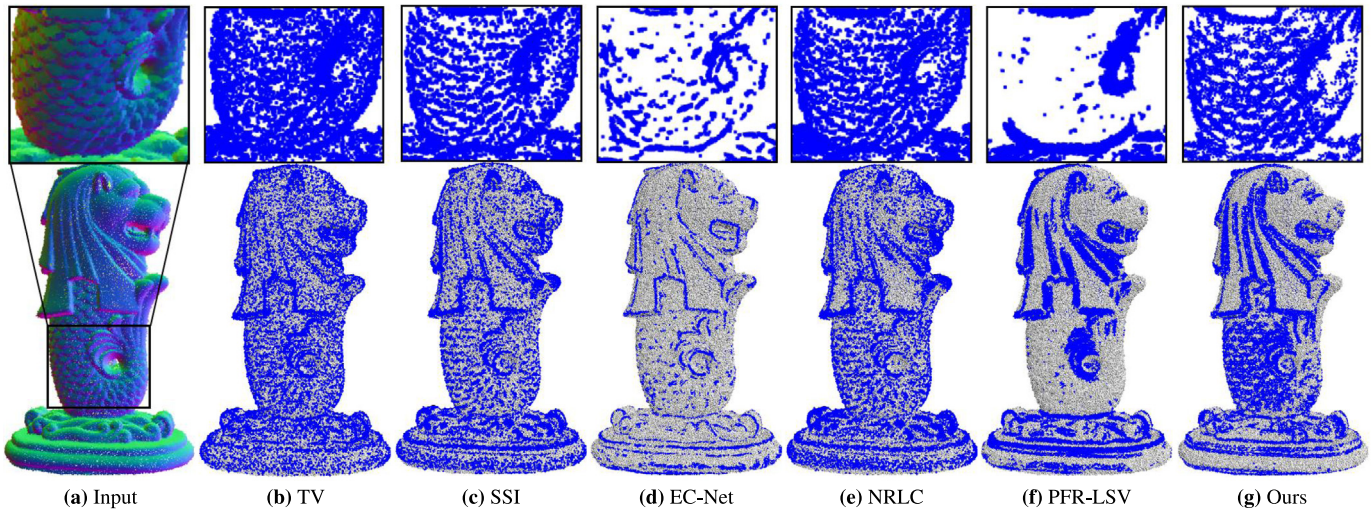


Fig. 15. Comparison of feature detection results on Merlion (0.6% noise). Our method successfully detects large-scale structures and small-scale textures simultaneously on this noisy input.

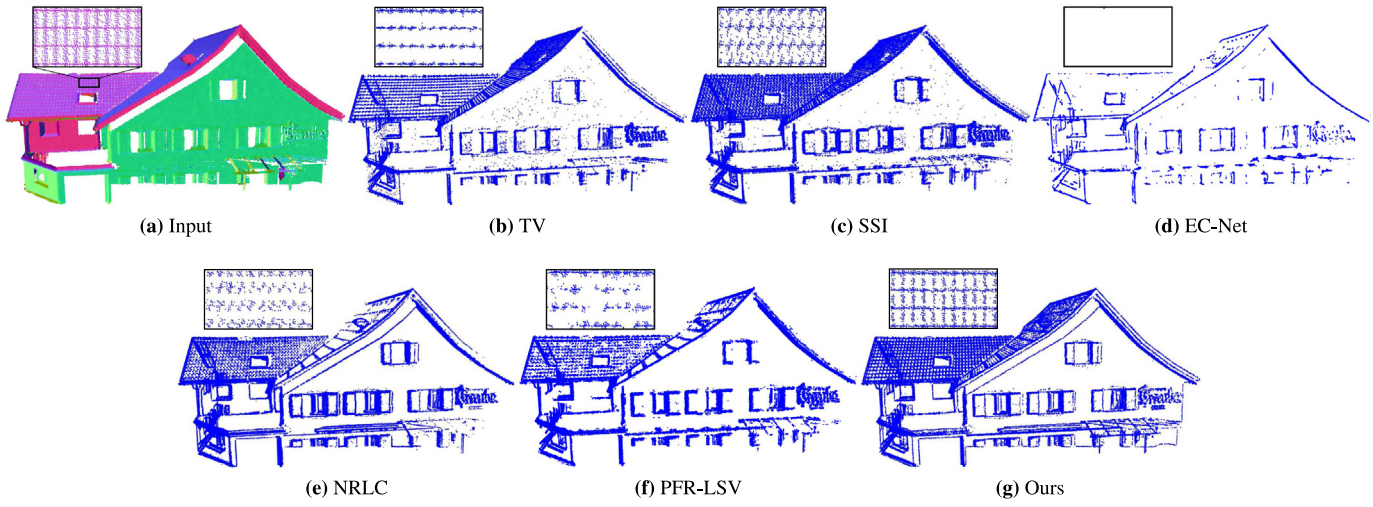


Fig. 16. Comparison of feature detection results on Building. Our method successfully extracts structural, detailed, and boundary features simultaneously in this input scene, demonstrating its ability to detect features of varying types and sizes with high accuracy.

the surfaces in Figs. 11, 12, 13, and 16 for numerical comparisons. We first employ different state-of-the-art feature detection approaches (e.g., NRLC, PFR-LSV, etc.) to identify feature points on the input surface, which can construct different feature sets. Then, we combine all the feature sets to construct a new feature set and eliminate those same feature points in this feature set. Finally, we manually refine the feature set by removing redundant feature points and adding missing feature points. We use four metrics to evaluate the performance of the methods: precision (*Pre*), recall rate (*Rec*), intersection over union score (*IoU*), and F1-score ($F1$). *Pre*, *Rec*, and *IoU* are defined as $Pre = \frac{TP}{TP+FP}$, $Rec = \frac{TP}{TP+FN}$, and $IoU = \frac{TP}{TP+FP+FN}$, respectively. *TP* (True Positives), *FP* (False Positives), and *FN* (False Negatives) represent the number of correctly detected points, incorrectly detected points, and falsely rejected points, respectively. *Pre* indicates the proportion of correctly detected features in all detected points. *Rec* measures the completeness of correctly detected features. *IoU* represents the similarity between the extraction result and ground truth. The F1-score is defined as $F1 = 2 \times \frac{Pre \times Rec}{Pre+Rec}$, which denotes the trade-off between precision and recall rate. We refer the interested reader to [40,44] for more details on these metrics.

As Table 2 shows, our method achieves the best results in the *Rec*, *IoU*, and $F1$ metrics, demonstrating its superior accuracy in detecting geometric features compared to the existing methods. Specifically, the higher *Rec* and *IoU* scores indicate better completeness of our results. Although our *Pre* scores on Fandisk and Building are not the best, our method achieves the highest $F1$ scores in all tested situations, showing our results are closer to the ground truth and outperform the others.

We also record the running time of all the tested methods in Table 3. As we can see, TV is the fastest method, while EC-Net is the slowest one. Due to the structure-to-detail feature perception mechanism, our method is more time-consuming than TV, SSI, and NRLC in most cases, which is the expense for better results in terms of visual quality and error metrics.

Table 2
Quantitative comparison of our method with state-of-the-arts for results in Figs. 11, 12, 13, and 16 (*Pre*: precision; *Rec*: recall rate; *IoU*: intersection over union score; *F*1: F1-score; ↑: the higher score, the better).

Surfaces (M)	Methods	Metrics			
		Pre ↑	*Rec* ↑	*IoU* ↑	*F*1 ↑
Fandisk (200.0K)	TV	0.6882	0.6876	0.5243	0.6879
	SSI	0.7125	0.7083	0.5509	0.7104
	EC-Net	0.6757	0.7010	0.5245	0.6881
	NRLC	**0.7199**	0.6846	0.5406	0.7018
	PFR-LSV	0.6316	0.6802	0.4870	0.6550
	Ours	0.6872	**0.7485**	**0.5583**	**0.7166**
Joint (67.0K)	TV	0.4992	0.5358	0.3485	0.5169
	SSI	0.5853	0.6305	0.4358	0.6070
	EC-Net	0.6592	0.7339	0.5320	0.6945
	NRLC	0.6253	0.6736	0.4799	0.6485
	PFR-LSV	0.6509	0.6809	0.4825	0.6509
	Ours	**0.7092**	**0.7637**	**0.5816**	**0.7354**
Pyramid (42.9K)	TV	0.8528	0.5442	0.4975	0.6644
	SSI	0.8155	0.7067	0.6093	0.7572
	EC-Net	0.6258	0.6743	0.4805	0.6491
	NRLC	0.8795	0.8983	0.7999	0.8888
	PFR-LSV	0.5201	0.6510	0.4067	0.5782
	Ours	**0.9150**	**0.9225**	**0.8496**	**0.9187**
Building (335.2K)	TV	**0.9609**	0.5169	0.5063	0.6722
	SSI	0.7878	0.5167	0.4536	0.6241
	EC-Net	0.7212	0.5027	0.4209	0.5925
	NRLC	0.7554	0.4888	0.4220	0.5936
	PFR-LSV	0.8012	0.5111	0.4536	0.6241
	Ours	0.9212	**0.6127**	**0.5822**	**0.7359**

Table 3
Running time of each tested method for the results in Figs. 11, 12, 13, 14, 15, and 16.

Surfaces (M)	Time (in seconds)					
	TV	SSI	EC-Net	NRLC	PFR-LSV	Ours
Fandisk (200.0K)	2.96	4.17	51.70	4.30	9.37	8.43
Joint (67.0K)	1.41	2.73	17.38	2.83	7.09	4.17
Pyramid (42.9K)	0.92	1.31	9.66	1.40	3.11	2.87
Bunny-Hi (100.0K)	0.81	1.14	25.21	1.21	3.52	2.75
Merlion (189.0.0K)	4.59	7.12	42.78	7.69	14.23	13.85
Building (335.2K)	4.34	9.20	79.39	9.94	18.76	16.67

4.4. Comparison with SGLBP and PCEDNet

To further demonstrate the effectiveness of our method, we compare it to two state-of-the-art methods including the subgraph-based local binary patterns (SGLBP) [33] and PCED-Net [40]. As depicted in Fig. 17, our method produces a visually superior result for the CAD surface corrupted by noise. SGLBP shows poor reliability as it incorrectly identifies some points on planar regions as features. Similarly, PCEDNet has also misidentified features in the presence of noise. For non-CAD surfaces with rich features, both SGLBP and PCEDNet are capable of detecting features to some extent. However, our method outperforms them by accurately identifying a broader range of details. In the case of the outdoor scene, our method and PCEDNet demonstrate the capability to detect small-scale features on the roof. Compared to PCEDNet, our method provides more complete and regular feature identification results. As we can see in Table 4, except for the recall rate, our method exhibits superior performance compared to the competing methods (SGLBP and PCEDNet) in terms of *Pre*, *IoU*, and *F*1. Table 4 also records the running time for the three methods. We can see that PCEDNet is the fastest method, while SGLBP is the slowest. Overall, our method produces results with much better visual quality and numerical metrics in most cases.

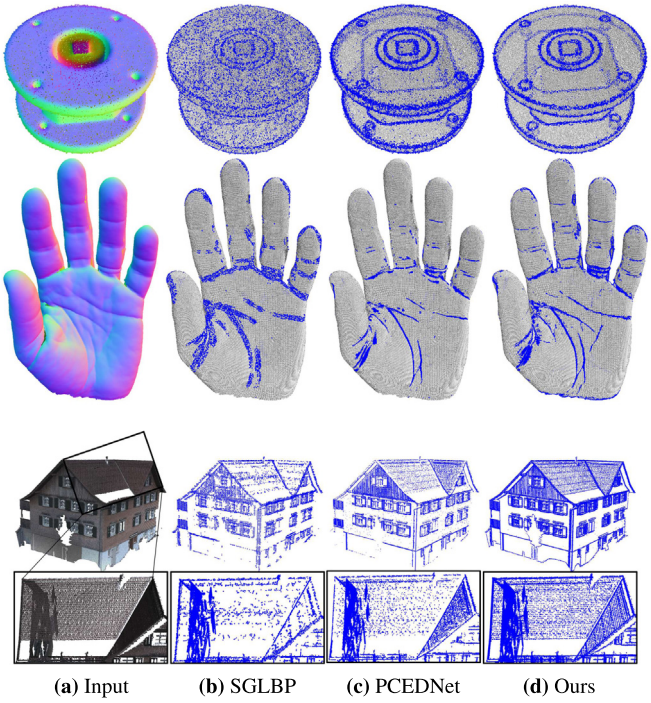


Fig. 17. Comparison with SGLBP and PCEDNet on Bearing (1% noise), Hand, and Bildstein. The zoomed-in views highlight that our method provides more complete small-scale and structure features.

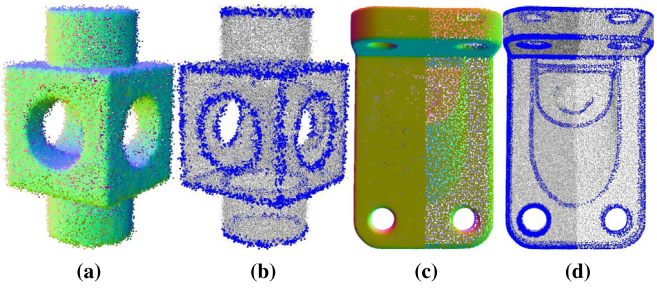


Fig. 18. Robustness tests on noisy and non-uniform sampling point cloud inputs (2.5% noise).

4.5. Robustness tests

We conduct robustness tests to evaluate the performance of our method under challenging conditions.

Large noise & irregular sampling. Figs. 18(a) and 18(b) show that our method can accurately identify salient features even in the presence of high noise. Irregular sampling is a common issue in point clouds. We demonstrate the robustness of our method against non-uniform sampling in Figs. 18(c) and 18(d). As shown in the figures, our method is capable of producing high-quality detection results despite the varying density distributions of the tested surface.

Complex scenes. To further validate the versatility of our method, we test it on outdoor, indoor, and desktop scenes. In Fig. 19, we demonstrate the effectiveness of our method on an outdoor LiDAR scan. Our method successfully extracts scene structures and boundary points while excluding most false features introduced by the scanning process. We also apply our method to an indoor scene acquired by RGB-D sensors, as shown in Fig. 20. Despite having a few artifacts, our method yields a visually convincing result containing most of the structural edges

Fig. 19. Result for a LiDAR scan of a building. From left to right, the images display a photograph of the building, the raw scan, and the extraction result. Our method successfully identifies the scene structures and boundary points while excluding most false features introduced by the scanning process.

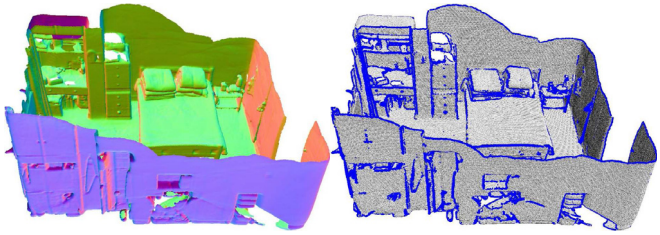


Fig. 20. Result for an indoor scene. The left image displays the low-quality input, while the right image shows the extraction result, which contains a few false features.

Table 4
Quantitative comparison of our method with state-of-the-art methods (SGLBP and PCEDNet) on results in Fig. 17 (*Pre*: precision; *Rec*: recall rate; *IoU*: intersection over union score; *F*1: F1-score; ↑: the higher score, the better).

Surfaces (M)	Methods	Metrics				Time
		Pre ↑	*Rec* ↑	*IoU* ↑	*F*1 ↑	
Bearing (121.0K)	SGLBP	0.3641	0.4166	0.2411	0.3886	58.72
	PCEDNet	0.5519	**0.8662**	0.5086	0.6742	6.51
	Ours	**0.7201**	0.7085	**0.5555**	**0.7143**	11.52
Bildstein (212.0K)	SGLBP	0.2560	0.0524	0.0454	0.0869	310.94
	PCEDNet	0.7034	0.4067	0.3471	0.5154	7.23
	Ours	**0.7792**	**0.8065**	**0.6565**	**0.7926**	9.10

and some fine details. In Fig. 21, we present our result for a desktop scene. Our method not only clearly extracts the boundary features but also detects salient features (desktop structures) and shallow features (thin books), demonstrating the capability of our approach in detecting features of varying sizes and types.

Generalization. We employ different datasets and repositories to verify our method and demonstrate its generality. Fig. 22 shows our results of feature detection for CAD shapes on the ABC dataset [45]. Fig. 23 illustrates our results for non-CAD shapes on Stanford scanning repository and AIM@SHAPE repository. We also verify our method on datasets of outdoor and indoor scenes (Semantic3D and S3DIS), as demonstrated in Figs. 24 and 25. We believe that the results on the tested datasets have fully demonstrated that our proposed method has great generalization ability for detecting rich geometric features on object-level shapes and large-scale scenes.

4.6. Impact of normal estimation

In our method, the raw normals are estimated over the point cloud simply based on Principal Component Analysis (PCA), followed by normal orientation propagation using the minimal spanning tree scheme (MST) proposed by Hoppe et al. [46].

PCA-based normal estimation is known to be sensitive to the presence of large noise. To examine the impact of the input normals against noise, we compare our feature detection results produced with different raw normals as inputs. As illustrated in Fig. 26, using better raw normals estimated by bilateral normal smoothing [47] as input, we can obtain more accurate feature extraction results. Due to the difficulty in estimating reliable normals from noisy point clouds, we use normals estimated by PCA to keep the input simple and more general, despite that this may degrade the performance of our method in the presence of high noise.

Fig. 21. Result for a desktop scene derived from 60 photos. The left image displays the photographs of the desktop, while the top-right image shows the input point cloud. The bottom-right image shows the extraction result, which clearly highlights the boundary features as well as the salient and shallow structures of the desktop.

The normal orientation also plays an important role in our method. The oriented normals clearly indicate whether the normals of the underlying surface are facing outward or inward, providing a globally consistent orientation that helps depict the geometric structure of the surface. When the estimated normals have globally inconsistent orientations, it may result in a number of misidentified features; see Fig. 27 for an example.

4.7. Limitations

Our method exhibits superiority in feature detection compared to the state-of-the-art. However, it has some limitations. Like most traditional methods, our method contains a few parameters that need to be tuned in order to produce satisfactory results. Furthermore, due to the heuristic nature of our structure-to-detail perception algorithm, it is challenging to perform theoretical analysis on the algorithm. Besides, our method is unable to produce good results when the input surfaces are corrupted by heavy noise.

5. Applications

Segmentation is a widely used technique in various applications, such as object recognition [48], scene understanding [49], and 3D reconstruction [50]. Our proposed feature detection method can be used to extract geometric features from point clouds, which can then be used to partition the point cloud into multiple regions based on region growing [51]. Fig. 28 demonstrates the effectiveness of our method for point cloud segmentation on both an object-level surface and a scene-level surface. The former is a CAD model with complex geometry, while the latter is a point cloud of an indoor environment. Using our method, we are able to simultaneously detect sharp and shallow features, enabling us to accurately segment the CAD surface into multiple regions, as shown in the top of Fig. 28. For the indoor environment, our method partitions the point cloud into different regions corresponding to the different objects in the scene, such as walls, floors, doors, windows, and furniture, as illustrated in the bottom of Fig. 28. These segmentation results demonstrate the effectiveness of our feature detection method in

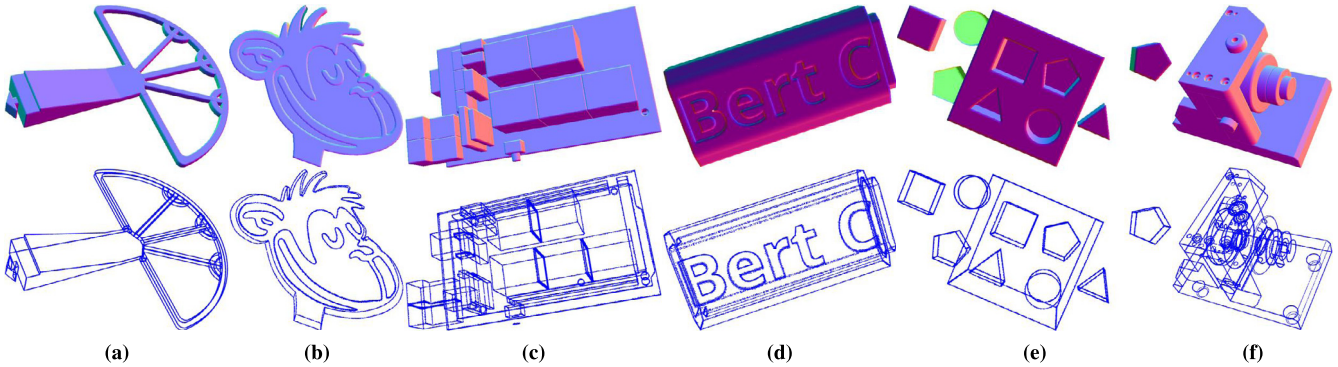


Fig. 22. Results of feature detection on ABC dataset for CAD shapes.

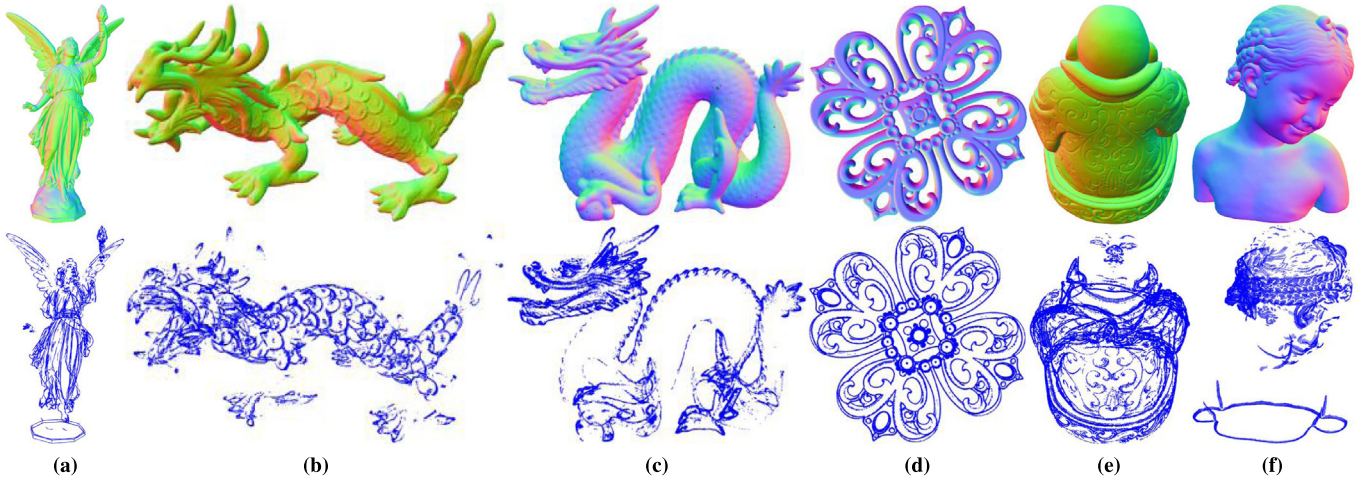


Fig. 23. Results of feature detection on the Stanford 3D Scanning Repository and the AIM@SHAPE Repository for non-CAD shapes.

Fig. 24. Results of feature detection on the Semantic3D dataset of outdoor scenes.

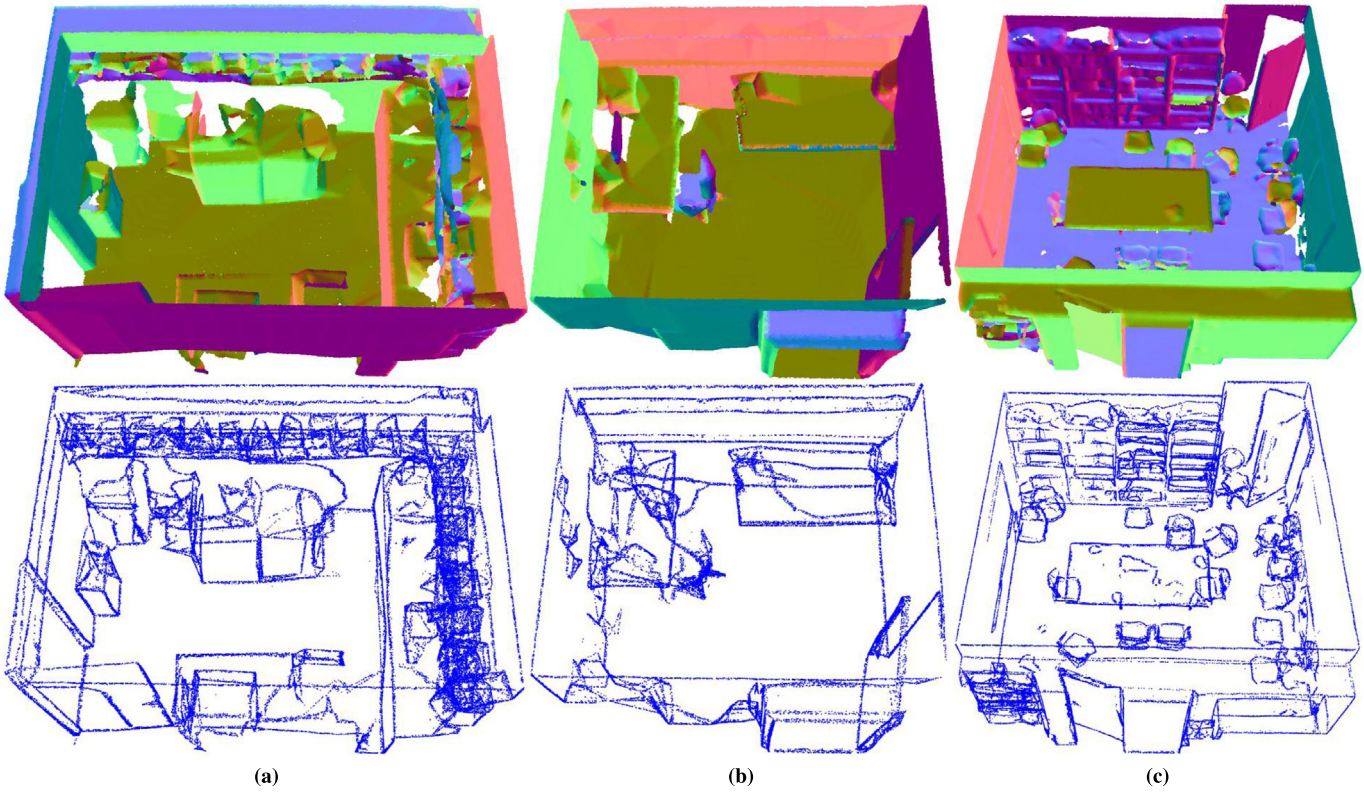


Fig. 25. Results of feature detection on the S3DIS dataset of indoor scenes.

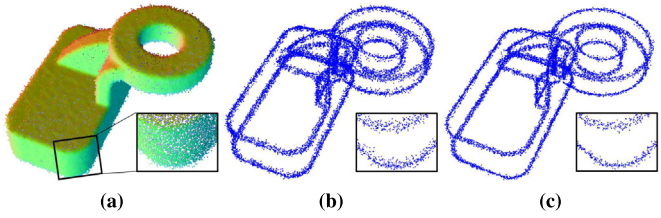


Fig. 26. (a) Noisy input (1.5% noise). (b) and (c) are feature detection results using raw normals estimated by PCA and bilateral normal smoothing accordingly. The zoomed-in views show that using better raw normals as input, our method can produce more accurate identification features.

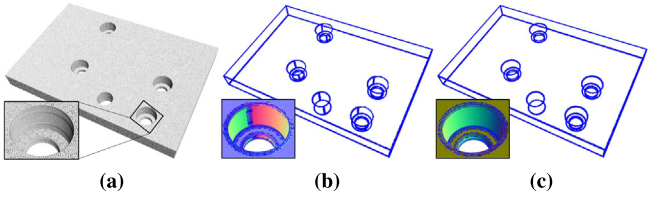


Fig. 27. (a) Input. (b) and (c) show feature detection results using unoriented and oriented normals as inputs accordingly. The zoomed-in views highlight that using unoriented normals as input, it may lead to misidentified features in regions with inconsistent normals.

identifying and partitioning point clouds into meaningful regions for downstream analysis and applications.

Surface reconstruction. Our feature detection results, which correspond to intersections of various smoothly curved regions, can represent the geometric structures of underlying surfaces intuitively. We use our feature detection results as the input to the iterative Poisson surface reconstruction (iPSR) method [13].

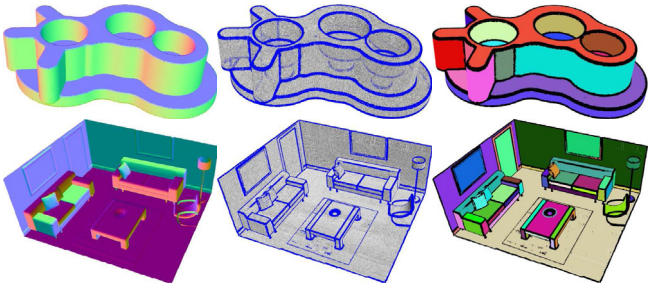


Fig. 28. Segmentation results of our method. From left to right: input data, feature detection results, and segmentation results based on the extraction features.

When the input feature points cannot fully describe the underlying surface structure, the reconstructed mesh obtained from iPSR is incomplete and has missing parts, as the zoomed-in views of Figs. 29(b) and 29(c) show. In contrast, Fig. 29(d) shows that we can reconstruct high-quality watertight surfaces from sparse but structurally complete points.

Feature line extraction. Features detected by our method can be used to generate accurate feature lines, which can sketch the intended shape with several curves. Extracting lines from point clouds is important in a variety of applications, such as reconstruction [52], registration [53], and object detection [54]. Fig. 30 shows that we can obtain feature lines through three steps. We begin by applying our method to the input surface to detect feature points, which are then refined and downsampled using the approach in [19]. Finally, following [26], we connect the resampled feature points into lines using the minimum spanning tree.

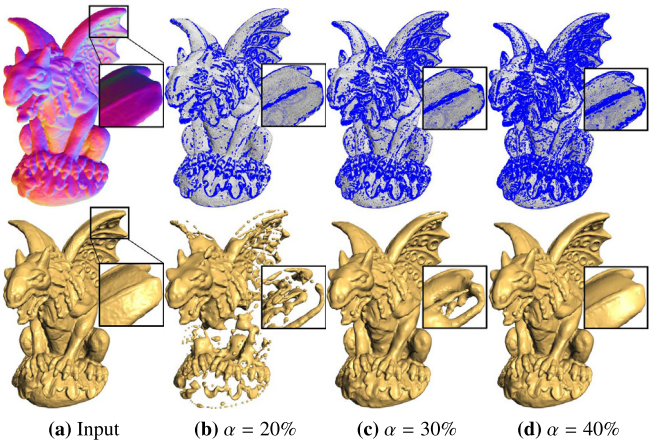


Fig. 29. Surface reconstruction based on our feature detection results. Top row: the input and the detected features with varying values of α, which is the percentage of feature points in the point cloud. Bottom row: the corresponding reconstruction results.

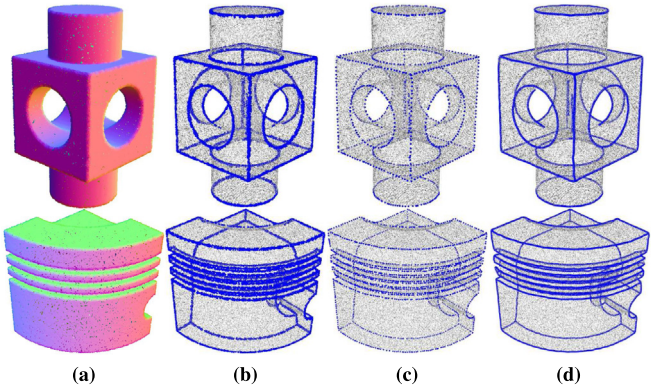


Fig. 30. Feature line extraction based on our feature detection results. (a) Input surfaces. (b) Feature detection results of (a). (c) Refined feature points from (b). (d) Connecting the refined feature points into feature lines.

6. Conclusion

We presented a novel approach for feature detection on point clouds. Our approach is based on a new saliency metric that takes into account both position and orientation variations within a local neighborhood. With bilateral centroid projection-based weighting, the saliency metric can effectively characterize local features. Using this metric, we propose a novel structure-to-detail feature perception algorithm that accurately locates geometric features of varying sizes by employing a neighborhood splitting and selection strategy. We believe our method makes a significant contribution to the field of point cloud processing, by providing a reliable and accurate way to detect geometric features on a wide range of surfaces, including those with noise, complex geometries, rich textures, shallow curves, and boundaries. We demonstrated the effectiveness and superiority of our feature detection method through several downstream applications, including segmentation, surface reconstruction, and feature line extraction.

Our feature detection approach has the potential for further extensions. First, pattern similarity features on underlying surfaces may exist, which can provide valuable insights into the intrinsic properties of the surface. This type of geometric characteristic needs to be identified globally, which is worth further investigation. Second, for large-scale scene data, our method can be computationally intensive. However, since our feature detection is independently executed on each point, it is therefore possible to accelerate the process using CPU or GPU parallelization. Finally, we look forward to applying our feature detection method to applications in various fields, such as autonomous driving, robotics, and urban architecture modeling.

Data availability

Data will be made available on request.

Acknowledgments

This work was supported by National Key R&D Program of China (2022YFB3904100), NSF of China (62072422), NSF of Anhui Province, China (2008085MF195), Collaborative Innovation Center for Natural Resources Planning and Marine Technology of Guangzhou (2023B04J0301), Key-Area Research and Development Program of Guangdong Province (2020B0101130009), Guangdong Enterprise Key Laboratory for Urban Sensing, Monitoring and Early Warning (2020B121202019), and the Ministry of Education, Singapore, under its Academic Research Fund Grants (MOE-T2EP20220-0005 & RT19/22).

References

[1] Liu Z, Xiao X, Zhong S, Wang W, Li Y, Zhang L, et al. A feature-preserving framework for point cloud denoising. Comput-Aided Des 2020;127:102857.
[2] Xu Y, Nan L, Zhou L, Wang J, Wang CCL. HRBF-Fusion: Accurate 3D reconstruction from RGB-D data using on-the-fly implicits. ACM Trans Graph 2022;41(3):1–19.
[3] Chen H, Wei Z, Xu Y, Wei M, Wang J. ImLoveNet: Misaligned image-supported registration network for low-overlap point cloud pairs. In: Proceedings of SIGGRAPH. 2022, p. 1–9.
[4] Zhou H, Chen H, Zhang Y, Wei M, Xie H, Wang J, et al. Refine-Net: Normal refinement neural network for noisy point clouds. IEEE Trans Pattern Anal Mach Intell 2022;45(1):946–63.
[5] Chen H, Wei Z, Li X, Xu Y, Wei M, Wang J. RePCD-Net: Feature-aware recurrent point cloud denoising network. Int J Comput Vis 2022;130(3):615–29.
[6] Huang S, Gojcic Z, Huang J, Wieser A, Schindler K. Dynamic 3D scene analysis by point cloud accumulation. In: Proceedings of the European conference on computer vision. 2022, p. 674–90.
[7] Guo J, Li C, Xia X, Hu R, Liu L. Asynchronous collaborative autoscanning with mode switching for multi-robot scene reconstruction. ACM Trans Graph 2022;41(6):1–13.
[8] Kou R, Yang B, Dong Z, Liang F, Yang S. Mapping the spatio-temporal visibility of global navigation satellites in the urban road areas based on panoramic imagery. Int J Digit Earth 2021;14(7):807–20.
[9] Zhu N, Yang B, Dong Z, Chen C, Huang X, Xiao W. Automatic registration of mobile mapping system lidar points and panoramic-image sequences by relative orientation model. Photogramm Eng Remote Sens 2021;87(12):913–22.
[10] Lin Y, Wang C, Cheng J, Chen B, Jia F, Chen Z, et al. Line segment extraction for large scale unorganized point clouds. ISPRS J Photogramm Remote Sens 2015;102:172–83.
[11] Hackel T, Wegner JD, Schindler K. Joint classification and contour extraction of large 3D point clouds. ISPRS J Photogramm Remote Sens 2017;130:231–45.
[12] Yu E, Arora R, Baerentzen JA, Singh K, Bousseau A. Piecewise-smooth surface fitting onto unstructured 3D sketches. ACM Trans Graph 2022;41(4):1–16.
[13] Hou F, Wang C, Wang W, Qin H, Qian C, He Y. Iterative Poisson surface reconstruction (iPSR) for unoriented points. ACM Trans Graph 2022;41(4):128:1–3.
[14] Huang Z, Carr N, Ju T. Variational implicit point set surfaces. ACM Trans Graph 2019;38(4):1–13.
[15] Todd JT. The visual perception of 3D shape. Trends Cogn Sci 2004;8(3):115–21.
[16] Chua CS, Jarvis R. Point signatures: A new representation for 3D object recognition. Int J Comput Vis 1997;25(1):63–85.
[17] Park MK, Lee SJ, Lee KH. Multi-scale tensor voting for feature extraction from unstructured point clouds. Graph Models 2012;74(4):197–208.

[18] Chen H, Huang Y, Xie Q, Liu Y, Zhang Y, Wei M, et al. Multiscale feature line extraction from raw point clouds based on local surface variation and anisotropic contraction. IEEE Trans Autom Sci Eng 2021;19(2):1003–16.
[19] Nie J. Extracting feature lines from point clouds based on smooth shrink and iterative thinning. Graph Models 2016;84:38–49.
[20] Liu T, Yang Z, Hu S, Zhang Z, Xiao C, Guo X, et al. Neighbor reweighted local centroid for geometric feature identification. IEEE Trans Vis Comput Graphics 2021;29(2):1545–58.
[21] Hildebrandt K, Polthier K, Wardetzky M. Smooth feature lines on surface meshes. In: Proceedings of the Eurographics symposium on geometry processing. 2005, p. 85–90.
[22] Ohtake Y, Belyaev A, Seidel H-P. Ridge-valley lines on meshes via implicit surface fitting. In: Proceedings of SIGGRAPH. 2004, p. 609–12.
[23] Clarenz U, Rumpf M, Telea A. Robust feature detection and local classification for surfaces based on moment analysis. IEEE Trans Vis Comput Graphics 2004;10(5):516–24.
[24] Liu Z, Wang W, Zhong S, Zeng B, Liu J, Wang W. Mesh denoising via a novel Mumford–Shah framework. Comput-Aided Des 2020;126:102858.
[25] Liu Z, Li Y, Wang W, Liu L, Chen R. Mesh total generalized variation for denoising. IEEE Trans Vis Comput Graphics 2021;28(12):4418–33.
[26] Pauly M, Keiser R, Gross M. Multi-scale feature extraction on point-sampled surfaces. Comput Graph Forum 2003;22(3):281–9.
[27] Mérigot Q, Ovsjanikov M, Guibas LJ. Voronoi-based curvature and feature estimation from point clouds. IEEE Trans Vis Comput Graphics 2010;17(6):743–56.
[28] Kalogerakis E, Nowrouzezahrai D, Simari P, Singh K. Extracting lines of curvature from noisy point clouds. Comput-Aided Des 2009;41(4):282–92.
[29] Park MK, Lee SJ, Lee KH. Multi-scale tensor voting for feature extraction from unstructured point clouds. Graph Models 2012;74(4):197–208.
[30] Demarsin K, Vanderstraeten D, Volodine T, Roose D. Detection of closed sharp edges in point clouds using normal estimation and graph theory. Comput-Aided Des 2007;39(4):276–83.
[31] Bazazian D, Casas JR, Ruiz-Hidalgo J. Fast and robust edge extraction in unorganized point clouds. In: International conference on digital image computing: techniques and applications. IEEE; 2015, p. 1–8.
[32] Béarzi Y, Digne J, Chaine R. Wavejets: A local frequency framework for shape details amplification. Comput Graph Forum 2018;37(2):13–24.
[33] Guo B, Zhang Y, Gao J, Li C, Hu Y. SGLBP: Subgraph-based local binary patterns for feature extraction on point clouds. Comput Graph Forum 2022;41(6):51–66.
[34] Rusu RB, Blodow N, Marton ZC, Beetz M. Aligning point cloud views using persistent feature histograms. In: IEEE/RSJ international conference on intelligent robots and systems. 2008, p. 3384–91.
[35] Guo Y, Sohel FA, Bennamoun M, Wan J, Lu M. RoPS : A local feature descriptor for 3D rigid objects based on rotational projection statistics. In: International conference on communications, signal processing, and their applications. 2013, p. 1–6.
[36] Yang J, Cao Z, Zhang Q. A fast and robust local descriptor for 3D point cloud registration. Inform Sci 2016;346:163–79.
[37] Zhang Y, Geng G, Wei X, Zhang S, Li S. A statistical approach for extraction of feature lines from point clouds. Comput Graph 2016;56:31–45.
[38] Yu L, Li X, Fu C-W, Cohen-Or D, Heng P-A. EC-Net: An edge-aware point set consolidation network. In: Proceedings of the European conference on computer vision. 2018, p. 386–402.
[39] Wang X, Xu Y, Xu K, Tagliasacchi A, Zhou B, Mahdavi-Amiri A, et al. PIE-Net: Parametric inference of point cloud edges. In: Advances in neural information processing systems, vol. 33. 2020, p. 20167–78.
[40] Himeur C-E, Lejemble T, Pellegrini T, Paulin M, Barthe L, Mellado N. PCEDNet: A lightweight neural network for fast and interactive edge detection in 3D point clouds. ACM Trans Graph 2022;41(1):1–21.
[41] Matveev A, Rakhimov R, Artemov A, Bobrovskikh G, Egiazarian V, Bogomolov E, et al. DEF: Deep estimation of sharp geometric features in 3D shapes. ACM Trans Graph 2022;41(4):1–22.
[42] Zhao T, Yu M, Alliez P, Lafarge F. Sharp feature consolidation from raw 3D point clouds via displacement learning. Comput Aided Geom Design 2023;103:102204.
[43] Kleppe AL, Tingelstad L, Egeland O. Coarse alignment for model fitting of point clouds using a curvature-based descriptor. IEEE Trans Autom Sci Eng 2018;16(2):811–24.
[44] Chicco D, Jurman G. The advantages of the Matthews correlation coefficient (MCC) over F1 score and accuracy in binary classification evaluation. BMC Genomics 2020;21(6):1–13.
[45] Koch S, Matveev A, Jiang Z, Williams F, Artemov A, Burnaev E, et al. ABC: A big cad model dataset for geometric deep learning. In: Proceedings of the IEEE conference on computer vision and pattern recognition. 2019, p. 9601–11.
[46] Hoppe H, DeRose T, Duchamp T, McDonald J, Stuetzle W. Surface reconstruction from unorganized points. ACM Trans Graph 1992;71–8.
[47] Huang H, Wu S, Gong M, Cohen-Or D, Ascher U, Zhang HR. Edge-aware point set resampling. ACM Trans Graph 2013;32(1):9:1–9:12.
[48] Ning X, Wang Y, Hao W, Zhao M, Sui L, Shi Z. Structure-based object classification and recognition for 3D scenes in point clouds. In: International conference on virtual reality and visualization. 2014, p. 166–73.
[49] Song S, Lichtenberg SP, Xiao J. SUN RGB-D: A RGB-D scene understanding benchmark suite. In: Proceedings of the IEEE conference on computer vision and pattern recognition. 2015, p. 567–76.
[50] Karimi Mahabadi R, Hane C, Pollefeys M. Segment based 3D object shape priors. In: Proceedings of the IEEE conference on computer vision and pattern recognition. 2015, p. 2838–46.
[51] Christoph Stein S, Schoeler M, Papon J, Worgotter F. Object partitioning using local convexity. In: Proceedings Of The IEEE conference on computer vision and pattern recognition. 2014, p. 304–11.
[52] Kazhdan M, Hoppe H. Screened poisson surface reconstruction. ACM Trans Graph 2013;32(3):1–13.
[53] Yang B, Zang Y. Automated registration of dense terrestrial laser-scanning point clouds using curves. ISPRS J Photogramm Remote Sens 2014;95:109–21.
[54] Wang R, Ferrie FP, Macfarlane J. A method for detecting windows from mobile LiDAR data. Photogramm Eng Remote Sens 2012;78(11):1129–40.