Journal of Rock Mechanics and Geotechnical Engineering 18 (2026) 3722–3741

Contents lists available at ScienceDirect

Journal of Rock Mechanics and Geotechnical Engineering

journal homepage: www.jrmge.cn

Full Length Article

Normal vector estimation method for rock mass point clouds with sharp feature preservation via local geometric adjustment

Mingming Ren [a,b], Manchao He [a], Jie Hu [a,*], Hongru Li [c], Yuxiang Ding [a,b], Xinhao Miao [a,b], Hongyi Zhang [a,b]

[a] State Key Laboratory for Tunnel Engineering, China University of Mining and Technology (Beijing), Beijing, 100083, China
[b] School of Mechanics and Civil Engineering, China University of Mining and Technology (Beijing), Beijing, 100083, China
[c] China Academy of Safety Science and Technology, Beijing, 100012, China

ARTICLE INFO

Article history:
Received 17 July 2025
Received in revised form
28 December 2025
Accepted 5 January 2026
Available online 20 January 2026

Keywords:
Rock mass point cloud
Normal vector estimation
Sharp feature
Rock engineering

ABSTRACT

Accurate extraction of rock mass discontinuity parameters is crucial for stability assessment and engineering safety. High-resolution remote sensing facilitates automated extraction, but its effectiveness relies heavily on precise normal estimation to ensure geometric reliability. Conventional methods struggle to preserve sharp features such as edges and corners, thereby reducing accuracy. To address this, we propose a normal estimation method based on local geometric adjustment that enhances feature extraction while maintaining sharp geometries. The approach consists of four steps: (1) classifying points, (2) applying normal and axial projections, (3) fitting segmentation lines via least squares, and (4) refining normals by optimizing local neighborhoods. The proposed method was evaluated on computer-aided design (CAD) models, real objects, and rock mass point clouds, and benchmarked against eight representative algorithms, including principal component analysis (PCA), 2-Jet PCA, Voronoi-based PCA, PCPNet, neural gradient function (NeuralGF), low rank representation (LRR), normal estimation via shifted neighborhood (NSN) and pair consistency voting (PCV). Experimental results demonstrate that our method achieves superior accuracy and efficiency, significantly improving structural plane extraction and ensuring better preservation of sharp geometric features.

1. Introduction

The structural surfaces of rock masses play a critical role in ensuring the safety and stability of rock engineering projects, with accurate measurement and characterization serving as the foundation for reliable rock stability analysis (Jaboyedoff et al., 2009). According to Barton et al. (1974), key parameters such as orientation, spacing, trace length, roughness, aperture, infill, seepage, set number, and block size are recommended to describe the geometric, mechanical, and hydraulic properties of structural surfaces. Consequently, precise extraction of these parameters is essential for stability analysis, as parameter accuracy directly affects the reliability of stability assessments. Traditionally, rock mass structural surface parameters have been measured using geological compasses and tape measures, employing line scanning or window sampling techniques. However, these fully manual methods suffer from inherent limitations, including measurement bias, restricted accessibility to hazardous or hard-to-reach areas, and low efficiency (Slob et al., 2005). The rapid advancement of remote sensing technologies has provided innovative solutions to overcome these challenges. Techniques such as light detection and ranging (LiDAR), drones integrated with structure-from-motion (SfM), and photogrammetry now enable the acquisition of high-precision three-dimensional (3D) point cloud data, offering efficiency, environmental adaptability, and widespread applicability in geological investigations. With these technological advancements, semi-automatic and automatic methods for extracting structural surface parameters based on remote sensing, including trace extraction (Zhang et al., 2020, 2024; Leng et al., 2021; Sun et al., 2021; Chen et al., 2022) and structural surface extraction (He et al., 2020; Kong et al., 2020; Chen et al., 2021; Singh et al.,

* Corresponding author.
E-mail address: hujie@cumtb.edu.cn (J. Hu).

Peer review under responsibility of Institute of Rock and Soil Mechanics, Chinese Academy of Sciences.

https://doi.org/10.1016/j.jrmge.2026.01.002

2021; Peng et al., 2024), are gradually replacing traditional approaches, shaping the future direction of rock mass stability analysis. These methods typically utilize normal vector information as prior knowledge, as normal vectors describe local geometric properties in point clouds and provide essential data for the automated extraction of structural surfaces. Accurate normal vector estimation thus enhances the precision of extracted structural parameters, significantly improving the reliability of rock stability assessments.

Normal vector estimation has emerged as a critical research focus in computer vision due to challenges such as varying sampling densities, noise, outliers, and the necessity of preserving sharp geometric features. This process is fundamental to tasks including segmentation (Ben-Shabat et al., 2018), classification (Charles et al., 2017), and surface reconstruction (Guerrero et al., 2018). Traditional approaches, such as principal component analysis (PCA) (Hoppe et al., 1992), estimate normal vectors by defining a local neighborhood and fitting a plane. While numerous PCA-based variants (Pauly et al., 2002; Mitra et al., 2003; Huang et al., 2009) have been proposed to improve accuracy and robustness, PCA inherently acts as a low-pass filter, making it inadequate for preserving sharp features. To address this limitation, the *n*-jet method (Cazals and Pouget, 2005) fits higher-order surfaces using truncated Taylor expansions, whereas Voronoi-based methods (Amenta and Bernt, 1998) employ point cloud triangulation for normal vector computation. Although both methods outperform PCA on smooth surfaces, they remain highly sensitive to noise and struggle to retain sharp features. The rise of deep learning has introduced novel solutions, such as HoughCNN (Boulch and Marlet, 2016), which applies the Hough transform (HF) to project 3D point clouds into two-dimensional (2D) space before estimating normal vectors using convolutional neural networks (CNNs). To further mitigate spatial information loss, normal estimation net (Nesti-Net) (Ben-Shabat et al., 2019) incorporates a parameterized local neighborhood distribution and a mixture of expert networks to enhance normal estimation. PCPNet (Guerrero et al., 2018), built upon the PointNet (Charles et al., 2017) framework, improves estimation accuracy by extracting local geometric features, while neural gradient function (NeuralGF) (Li et al., 2023) employs an unsupervised learning strategy, leveraging latent feature representations of point clouds to circumvent the need for ground-truth training data. In terms of post-processing techniques, Zhang et al. (2013) applied low rank representation (LRR) for subspace clustering, segmenting neighborhoods into distinct regions and selecting the optimal subset for normal estimation. Cao et al. (2018) introduced the normal estimation via shifted neighborhood (NSN) method, which enhances accuracy by shifting local neighborhoods toward regions with clearer geometric features. Additionally, Zhang et al. (2019) proposed PCV to identify neighborhoods within the same subspace, enabling accurate normal estimation while effectively preserving sharp features. Collectively, these advancements mark substantial progress in improving the precision, robustness, and applicability of normal vector estimation techniques.

Normal vector estimation plays a pivotal role in the intelligent extraction of rock mass structural surface parameters. Riquelme et al. (2014) enhanced the accuracy of normal estimation by incorporating local neighborhood coplanarity into PCA calculations and refining neighborhood selection through threshold filtering. Similarly, researchers such as Wang et al. (2017), Ge et al. (2022), Guo et al. (2017), Zhang et al. (2024), and Sun et al. (2021) leveraged PCA for normal vector and curvature estimation, utilizing these results as inputs for subsequent algorithms. In parallel, Umili et al. (2013), Singh et al. (2021, 2022), Chen et al. (2016), and Zhang et al. (2018a, b) employed point cloud triangulation combined with Voronoi-based normal estimation methods, further integrating these estimations into downstream processing. However, despite these advancements, preserving sharp geometric features during normal vector estimation remains an underexplored challenge. Kong et al. (2020) addressed this issue by employing the iterative reweighted plane fitting (IRPF) algorithm (Wang et al., 2013) to estimate normal vectors while retaining sharp features in rock mass point clouds. Furthermore, the application of deep learning for normal estimation in rock mass point clouds remains in its infancy within rock engineering, highlighting a promising direction for future research.

To address the challenge of preserving sharp features, this paper proposes a normal vector estimation method specifically designed to retain sharp features in rock mass point clouds. The method integrates bidirectional projection and segmentation techniques into a four-step framework: classification, projection, segmentation, and correction. This approach not only improves the efficiency and accuracy of normal vector estimation but also effectively preserves sharp features in rock mass structures. To evaluate the performance of the proposed algorithm, three sets of experiments were conducted, comparing eight normal vector estimation methods, including traditional techniques, deep learning-based approaches, and post-processing strategies. A comprehensive analysis and discussion of the experimental results are presented. The remainder of this paper is structured as follows. Section 2 details the proposed method, Section 3 presents and analyzes the experimental results, and Section 4 concludes the study.

2. Methodology

2.1. Overview

The PCA-based normal vector estimation method has been widely adopted in various fields due to its computational efficiency and robustness, and it is particularly suitable for the intelligent extraction of structural surface parameters in rock masses (Klasing et al., 2009). However, in rock masses, discontinuities and sharp geometric features – such as edges and corners – play a dual role: they serve as stress concentration zones where failure is likely to initiate, and define the orientations of structural planes, both of which are critical for strength evaluation and stability analyses. Due to its inherent low-pass filtering property, PCA struggles to preserve these sharp features, which limits its effectiveness in accurately characterizing rock mass structures. As illustrated in Fig. 1a, normal vector estimations near sharp features tend to exhibit a smooth transition, forming an arc-like pattern that blurs essential edge and corner features. This smoothing effect leads to misalignment between the estimated normal and the true geometry (Fig. 1b), thereby compromising the precise representation of abrupt features within the rock mass and failing to accurately retain these sharp characteristics.

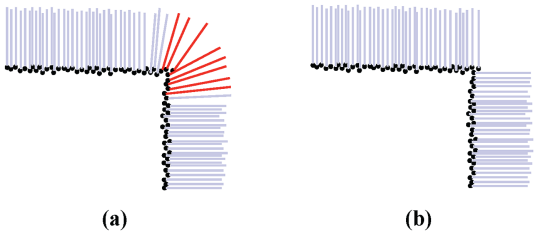

Fig. 1. Illustration of normal vectors at sharp features: (a) Diagram showing sharp features without preservation; and (b) Diagram showing preserved sharp features.

To enhance the accuracy of normal vector estimation while effectively preserving sharp features, this paper proposes a novel method tailored for sharp feature retention in rock mass point clouds. The core principle is to segment the nearest neighbor set of sharp feature points, extracting an optimal subset of neighbors that reside on the same plane as the target point. Normal vector estimation is then performed based on this refined subset. As illustrated in Fig. 2, the proposed method consists of four primary steps: classification, projection, segmentation, and correction, which are detailed as follows:

(1) Classification. The input point cloud data are processed by categorizing points into planar and sharp feature points.
(2) Projection. For sharp feature points, normal and axial projection operations are conducted. First, the central point and its nearest neighbor set are projected along the current normal direction, transforming the projection result into 2D coordinates and extracting two sets of coplanar feature points. Then, the central point and its nearest neighbors are projected along the principal axis (the direction corresponding to the largest eigenvalue) onto a 2D plane, and its central line is extracted.
(3) Segmentation. Based on the distribution of planar feature points along the central line, the least squares method is applied to fit two lines and determine their intersection angle bisector as the segmentation boundary. The points on the same side are retained as the target point to construct the optimal nearest neighbor set.
(4) Correction. The target point is identified as an edge point or a corner point. If it is an edge point, the optimal nearest neighbor set is directly utilized to refine the normal vector. If it is a corner point, a heuristic clustering method is applied for further adjustment, obtaining the final corrected normal vector.

2.2. Mechanism

2.2.1. Projection mechanism

Fig. 3 illustrates a point-to-plane projection in 3D space. In this figure, Fig. 3a and b represents two scenarios, where the projected point lies on the same side as the plane normal and on the opposite side, respectively. Here, O denotes the origin of the plane, $\vec{n}(n_x, n_y, n_z)$ is the plane's normal vector with $\|\vec{n}\| = 1$, and v_i and v_i' represent the original point and its projected counterpart, respectively.

As illustrated in Fig. 3a, according to the conditions that $\vec{n}$ is parallel to $\overrightarrow{v_i'v_i}$ and perpendicular to $\overrightarrow{Ov_i'}$, we can deduce

$$\begin{aligned}
&\left(1-n_x^2\right)x_i - n_x n_y y_i - n_x n_z z_i + n_x^2 o_x + n_x n_y o_y + n_x n_z o_z = x_i' \\
&-n_x n_y x_i + \left(1-n_y^2\right)y_i - n_y n_z z_i + n_x n_y o_x + n_y^2 o_y + n_y n_z o_z = y_i' \\
&-n_x n_z x_i - n_y n_z y_i + \left(1-n_z^2\right)z_i + n_x n_z o_x + n_y n_z o_y + n_z^2 o_z = z_i'
\end{aligned} \tag{1}$$

The coordinate $\left(x_i', y_i', z_i'\right)$ represents the location of the projected point. Similarly, for the case illustrated in Fig. 3b, an analogous result can be derived. To improve computational efficiency, Eq. (1) is vectorized. Let $\boldsymbol{V}$ denote the set of points to be projected and $\boldsymbol{V}'$ the corresponding set of projected points, then they are formulated as follows:

$$\boldsymbol{V} = \begin{bmatrix} x_0 & y_0 & z_0 \\ x_1 & y_1 & z_1 \\ \vdots & \vdots & \vdots \\ x_m & y_m & z_m \end{bmatrix}, \boldsymbol{V}' = \begin{bmatrix} x_0' & y_0' & z_0' \\ x_1' & y_1' & z_1' \\ \vdots & \vdots & \vdots \\ x_m' & y_m' & z_m' \end{bmatrix} \tag{2}$$

where m is the number of elements in the point set. Based on Eq. (1), the vectorized projection formula can be derived as follows:

$$\boldsymbol{V} + \left(\boldsymbol{I}_m \otimes \boldsymbol{O}^{\mathrm{T}} - \boldsymbol{V}\right)\vec{n}\vec{n}^{\mathrm{T}} = \boldsymbol{V}' \tag{3}$$

where $\boldsymbol{I}_m = [1 \quad 1 \quad \cdots \quad 1]^{\mathrm{T}}$ is an m-dimensional unit column vector, and $\boldsymbol{I}_m \otimes \boldsymbol{O}^{\mathrm{T}}$ is the Kronecker product.

2.2.2. Coordinate transformation

Fig. 4 illustrates basis transformation and coordinate transformation into 3D space, where p represents a point in 3D space.

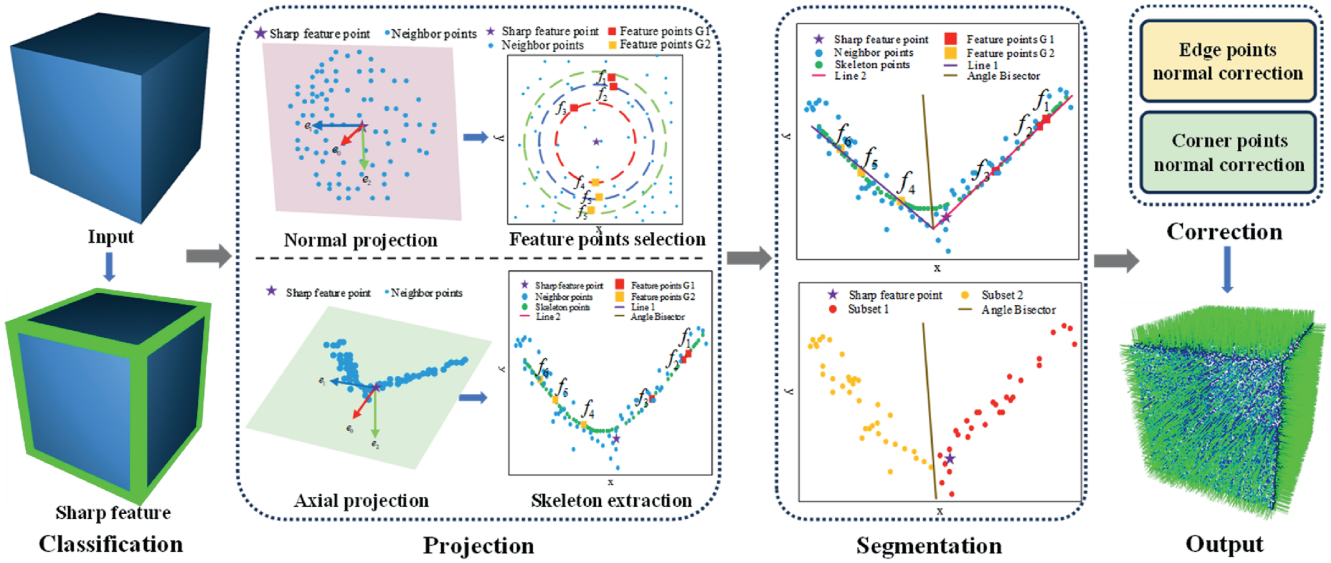

Fig. 2. Overview of proposed method.

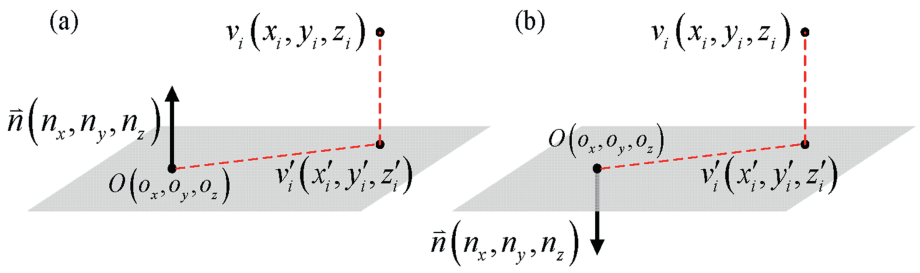

Fig. 3. Projection diagram. (a) Same-side case; and (b) Opposite-side case.

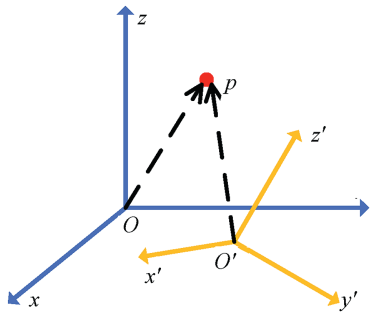

Fig. 4. Basis transformation and coordinate transformation.

Assume that the basis of the 3D coordinate system xyz is $\boldsymbol{A} = [\alpha_1 \quad \alpha_2 \quad \alpha_3]$, with $\alpha_i \in \mathbb{R}^3$, and that the basis of coordinate system $x'y'z'$ is $\boldsymbol{B} = [\beta_1 \quad \beta_2 \quad \beta_3]$, with $\beta_i \in \mathbb{R}^3$. Let the coordinates of point p under basis $\boldsymbol{A}$ be (x_1, y_1, z_1).

For each vector β_i in basis $\boldsymbol{B}$, its coordinates in basis $\boldsymbol{A}$ are obtained. This yields the transition matrix from basis $\boldsymbol{A}$ to $\boldsymbol{B}$, denoted as

$$\boldsymbol{P} = (\boldsymbol{P}_1, \boldsymbol{P}_2, \boldsymbol{P}_3) = \begin{bmatrix} p_{11} & p_{21} & p_{31} \\ p_{12} & p_{22} & p_{32} \\ p_{13} & p_{22} & p_{33} \end{bmatrix} \tag{4}$$

Then, the coordinates of point p in the coordinate system $x'y'z'$ are given by

$$\begin{bmatrix} x_1' \\ y_1' \\ z_1' \end{bmatrix} = \boldsymbol{P}^{-1} \begin{bmatrix} x_1 \\ y_1 \\ z_1 \end{bmatrix} \tag{5}$$

2.3. Classification

We begin by preprocessing the input point cloud data, classifying them into planar and sharp points. For the input point cloud be $\boldsymbol{P} = \{p_1, p_2, \ldots, p_N\}$, where N is the total number of points in the dataset, a point $p = (x, y, z)$ is selected from the set $\boldsymbol{P}$ and its nearest neighbor set $\boldsymbol{N} = \{p_1, p_2, \ldots, p_k\}$ is determined. Assuming that $\overline{p} = \sum_{i=1}^{k} p_i / k$ denotes the centroid of this neighbor set, the covariance matrix $\boldsymbol{C}_{3\times3}$ of the neighbor set $\boldsymbol{P}$ is then given by

$$\boldsymbol{C}_{3\times3} = \frac{1}{k} \begin{bmatrix} p_1 - \overline{p} \\ \vdots \\ p_k - \overline{p} \end{bmatrix}^{\mathrm{T}} \begin{bmatrix} p_1 - \overline{p} \\ \vdots \\ p_k - \overline{p} \end{bmatrix} \tag{6}$$

Using PCA, we compute the eigenvalues $\lambda_0 \geq \lambda_1 \geq \lambda_2$ of the covariance matrix $\boldsymbol{C}$, along with corresponding eigenvectors $\vec{e}_0$, $\vec{e}_1$ and $\vec{e}_2$. As illustrated in Fig. 5, the blue pentagram represents the target point p, while the red dots indicate its neighboring points N of p. The eigenvector $\vec{e}_2$ associated with the smallest eigenvalue is defined as the normal vector $\vec{n}'$ at point p. The red and green planes represent planes passing through p, with normal vectors $\vec{e}_2$ and $\vec{e}_0$, respectively.

The curvature $\sigma(p)$ at point p is defined as follows (Pauly et al., 2003):

$$\sigma(p) = \frac{\lambda_2}{\lambda_0 + \lambda_1 + \lambda_2} \tag{7}$$

A threshold Ω_{f} is set, and points are classified according to the following rules:

$$\left.\begin{aligned} \boldsymbol{P}_{\mathrm{f}} &= \left\{ p_i \in \boldsymbol{P} \middle| \sigma(p_i) \leq \Omega_{\mathrm{f}} \right\} \\ \boldsymbol{P}_{\mathrm{s}} &= \left\{ p_i \in \boldsymbol{P} \middle| \sigma(p_i) > \Omega_{\mathrm{f}} \right\} \end{aligned}\right\} \tag{8}$$

where $\boldsymbol{P}_{\mathrm{f}}$ is the set of planar feature points, and $\boldsymbol{P}_{\mathrm{s}}$ is the set of sharp feature points.

2.4. Projection

After classification, the point cloud is divided into two sets: a plane feature point set $\boldsymbol{P}_{\mathrm{f}}$ and a sharp feature set $\boldsymbol{P}_{\mathrm{s}}$. For the plane feature points, their normal vectors can be directly calculated using PCA. However, for sharp feature points, additional processing is

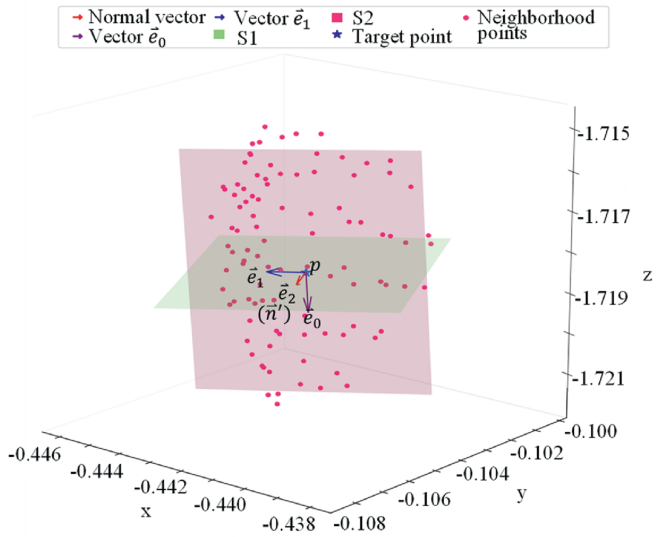

Fig. 5. Diagram of PCA computation results.

required. As shown in Fig. 6, the blue pentagram marker indicates the sharp feature point $p_i \in \boldsymbol{P}_s$, while the red point cloud indicates its nearest neighbor set, denoted as $\boldsymbol{N}_i = \{p_{i0}, p_{i1}, \ldots, p_{in}\}$. The red arrow illustrates the normal vector $\vec{n}'$ estimated using PCA. It is evident that the points in $\boldsymbol{N}_i$ are distributed across two adjacent planes, introducing bias in the normal vector calculation for the sharp feature point p_i when PCA is applied. Therefore, it becomes necessary to correct the normal vector for the sharp feature point. The following sections will provide a detailed explanation of the normal vector projection method and the axial projection method employed in this study.

2.4.1. Normal projection

First, a normal vector projection is performed on the sharp feature point p_i and its neighboring points $\boldsymbol{N}_i$, projecting them onto a plane that passes through point p_i with a normal vector $\vec{n}'$. The coordinates of all points in $\boldsymbol{N}_i$ are expressed as follows:

$$\boldsymbol{V}_{N_i} = \begin{bmatrix} x_{i0} & y_{i0} & z_{i0} \\ \vdots & \vdots & \vdots \\ x_{ij} & y_{ij} & z_{ij} \\ \vdots & \vdots & \vdots \\ x_{in} & y_{in} & z_{in} \end{bmatrix} \tag{9}$$

After applying the projection formula in Eq. (3), the projected coordinates are obtained as

$$\boldsymbol{V}'_{N_i} = \begin{bmatrix} x'_{i0} & y'_{i0} & z'_{i0} \\ \vdots & \vdots & \vdots \\ x'_{ij} & y'_{ij} & z'_{ij} \\ \vdots & \vdots & \vdots \\ x'_{in} & y'_{in} & z'_{in} \end{bmatrix} \tag{10}$$

As shown in Fig. 7a, the red points represent the nearest neighbor set after normal vector projection. These projected coordinates are then transformed into a 2D coordinate system with the target point p_i as the origin. Clearly, the original basis $\boldsymbol{A}$ is the identity matrix $\boldsymbol{I}$, and the transformed basis $\boldsymbol{B}$ is $\begin{bmatrix} \vec{e}_2 & \vec{e}_1 & \vec{e}_0 \end{bmatrix}$, resulting in the transition matrix $\boldsymbol{B}^{-1}$. Using Eq. (5), the transformed coordinates can be computed. By ignoring the z-axis component of the transformed coordinates, the final 2D coordinates $\boldsymbol{V}''_{N_i} = \left\{ \left(x''_{ij}, y''_{ij} \right) | j = 1, \ldots, n \right\}$ can be obtained. As shown in Fig. 7b, the red point represents the transformed nearest neighbor set, denoted as $\boldsymbol{N}_i^{2D} \subset \mathbb{R}^2$.

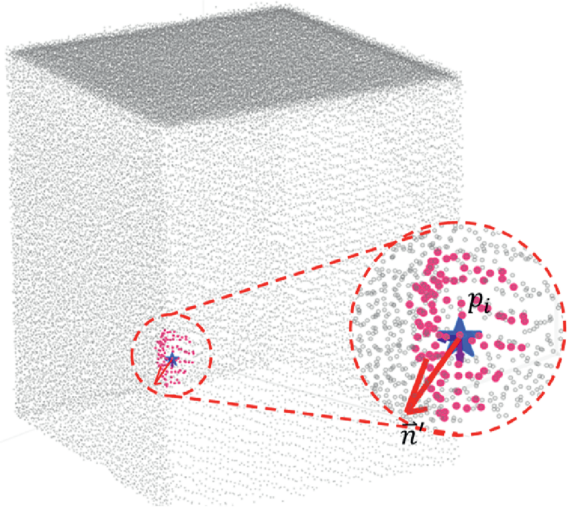

Fig. 6. Diagram of feature points.

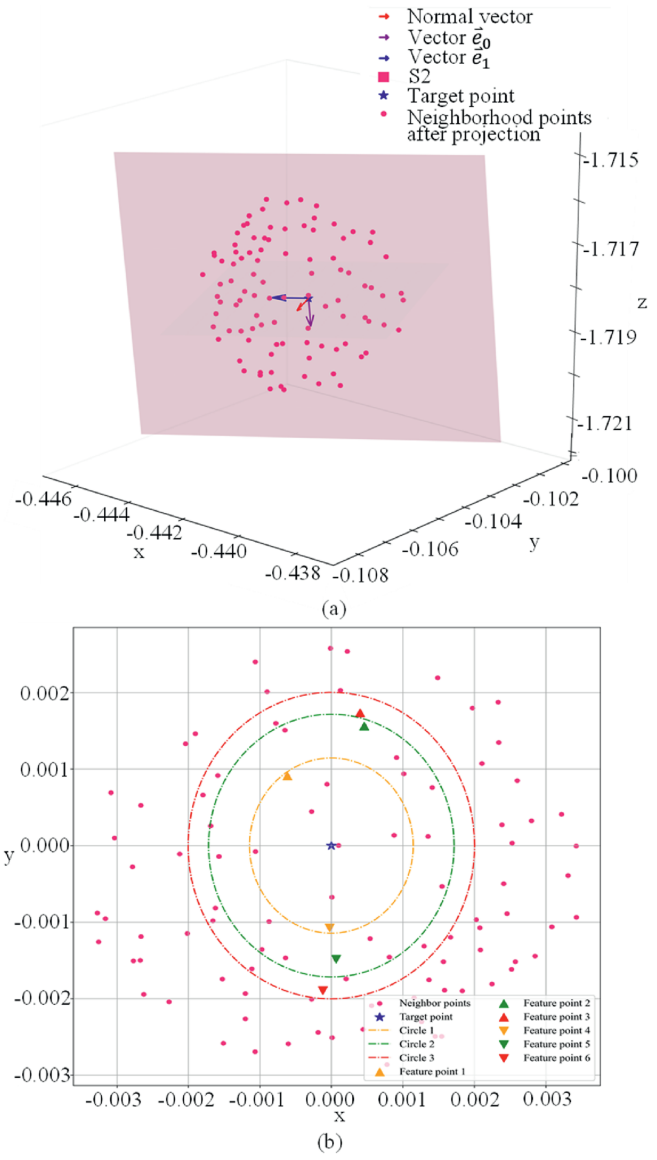

Fig. 7. Normal projection and feature selection: (a) Neighboring points after normal projection; and (b) Neighboring points after coordinate transformation and feature point extraction.

The coplanar feature point sets are extracted from $\boldsymbol{N}_i^{2D}$, where each coplanar feature set represents a collection of points lying within the same plane. Based on empirical knowledge, each coplanar feature point set is configured to contain three points to ensure sufficient geometric constraints for subsequent calculations. The extraction rules for coplanar feature points are as follows:

(1) Calculate the geometric information of $\boldsymbol{N}_i^{2D}$ using the following formula:

$$d = \max\left\{ \left| y_{ik} \right| \left| y_{ik} \in v_k(x_{ik}, y_{ik}), v_k \in \boldsymbol{V}''_{N_i} \right. \right\} \tag{11}$$

where d represents the distance to the farthest point along the y-axis.

(2) Define three coefficients r_0, r_1, and r_2, each with the range (0, 1], and multiply them by d to obtain three distinct radii. Using point p_i as the center, search within each radius to identify the points with the maximum and minimum y values, and store their indices in the coplanar feature index sets $\boldsymbol{Idx}_{\max}$ and $\boldsymbol{Idx}_{\min}$, respectively.

As depicted in Fig. 7b, the points marked by upward and downward triangles correspond to the feature point indices in sets $\boldsymbol{Idx}_{\max}$ and $\boldsymbol{Idx}_{\min}$ respectively.

2.4.2. Axial projection

For the sharp feature point p_i and its neighboring point $\boldsymbol{N}_i$, the axial projection method described above is applied, projecting them onto a plane with a normal vector $\vec{e}_0$, corresponding to the largest eigenvalue and passing through point p_i. After coordinate transformation, the resulting 2D coordinate point set is denoted as $\tilde{\boldsymbol{N}}_i^{2D} \subset \mathbb{R}^2$. As illustrated in Fig. 8, the red points in Fig. 8a represent the nearest neighbor set after axial projection, and in Fig. 8b represents the set transformed to 2D coordinates.

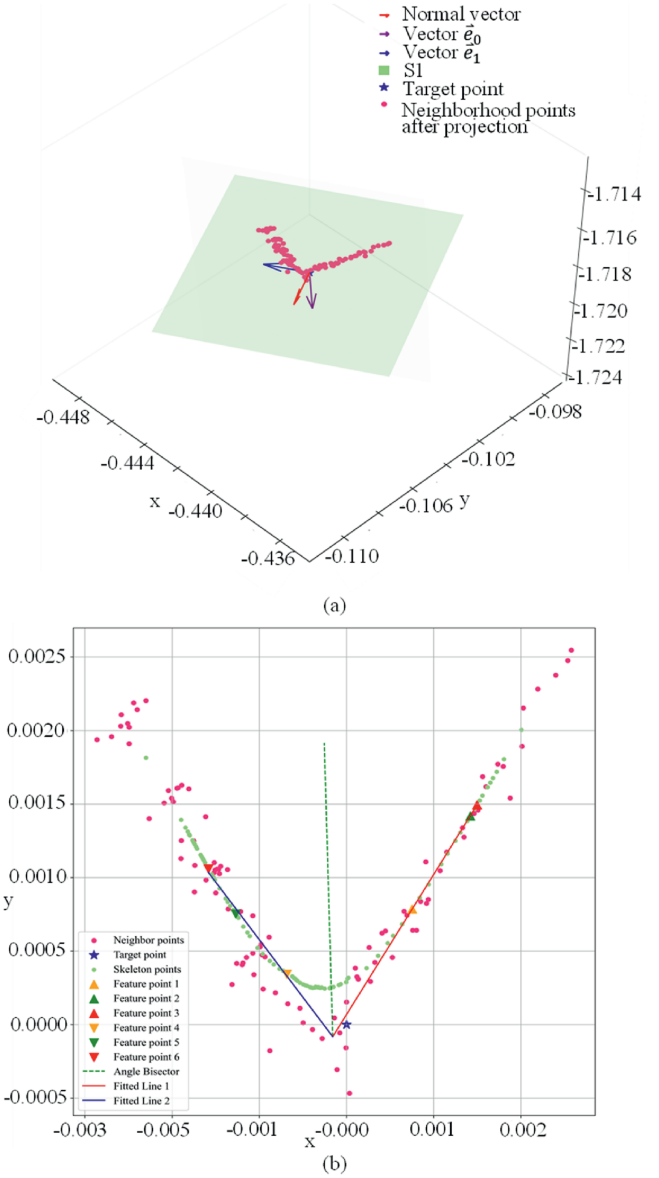

Fig. 8. Axial projection and skeleton extraction: (a) Spatial positions of points after axial projection; and (b) Positions of points transformed into a 2D coordinate system.

The centerline of $\tilde{\boldsymbol{N}}_i^{2D}$ is extracted using the L_1-medial skeletonization algorithm (Huang et al., 2013), with the following calculation steps:

(1) Let $\boldsymbol{X} = \tilde{\boldsymbol{N}}_i^{2D}$, and establish the optimization objective function as follows:

$$\underset{x}{\operatorname{argmin}} \sum_{j=1}^{n} \sum_{l=1}^{n} \theta\left(\left\|x_j - v_l\right\|\right) + \sum_{j \in \boldsymbol{J}} \gamma_j \sum_{j' \in \boldsymbol{J} \setminus \{j\}} \frac{\theta\left(\left\|x_j - x_{j'}\right\|\right)}{\sigma_j \left\|x_j - x_{j'}\right\|} \tag{12}$$

where $\boldsymbol{J} = \{1, 2, \ldots, n\}$, $\boldsymbol{J}\setminus\{j\}$ denotes the subset of $\boldsymbol{J}$ obtained by excluding the element j, $x_j \in \boldsymbol{X}$, $v_l \in \tilde{\boldsymbol{N}}_i^{2D}$, $\theta(r) = \mathrm{e}^{-r^2/[(h/2)]^2}$ is the weighting function, h represents the range, $\left\{\gamma_j\right\}_{j \in \boldsymbol{J}}$ is the balancing coefficient, and $\sigma_j = \lambda_0^j / \left(\lambda_0^j + \lambda_1^j + \lambda_2^j\right)$ denotes the linearity of the location of the point x_j. The goal of this optimization is to bring the points in $\boldsymbol{X}$ as close as possible to the regions in $\tilde{\boldsymbol{N}}_i^{2D}$ with higher linearity, while maintaining a certain distance between the points in $\boldsymbol{X}$.

(2) Solving this objective function yields the update formula for the points in $\boldsymbol{X}$ as follows:

$$x_j^{k+1} = \frac{\sum_{l \in \boldsymbol{L}} v_l \alpha_{jl}^k}{\sum_{l \in \boldsymbol{L}} \alpha_{jl}^k} + \mu \sigma_j \frac{\sum_{j' \in \boldsymbol{J} \setminus \{j\}} \left(x_j^k - x_{j'}^k\right) \beta_{jj'}^k}{\sum_{j' \in \boldsymbol{J} \setminus \{j\}} \beta_{jj'}^k} \tag{13}$$

where $\boldsymbol{L} = \{1, 2, \ldots, n\}$, $\alpha_{jl}^k = \dfrac{\theta\left(\left\|x_j^k - v_l\right\|\right)}{\left\|x_j^k - v_l\right\|}$ $(l \in \boldsymbol{L})$, $\beta_{jj'}^k = \dfrac{\theta\left(\left\|x_j^k - x_{j'}^k\right\|\right)}{\left\|x_j^k - x_{j'}^k\right\|^2}$ $(j' \in \boldsymbol{J}\setminus\{j\})$ and $\mu = \dfrac{\gamma_j \sum_{j' \in \boldsymbol{J} \setminus \{j\}} \beta_{jj'}^k}{\sigma_j \sum_{l \in \Lambda} \alpha_{jl}^k}$ are constants. After completing the iterations, the centerline of $\tilde{\boldsymbol{N}}_i^{2D}$ is obtained.

2.5. Segmentation

As shown in Fig. 8b, the green points represent the centerline of $\tilde{\boldsymbol{N}}_i^{2D}$. Using the feature indices in the coplanar feature index sets $\boldsymbol{Idx}_{\max}$ and $\boldsymbol{Idx}_{\min}$, the positions of the coplanar feature points on the centerline are identified. These points are then used to construct the feature set $\boldsymbol{F}_{\max}$ and $\boldsymbol{F}_{\min}$, as indicated by the upward and downward triangles in Fig. 8b.

Using the least squares fitting method, linear fitting is performed separately for the points in $\boldsymbol{F}_{\max}$ and $\boldsymbol{F}_{\min}$. The calculation formula is as follows:

$$m, c = \underset{m,c}{\operatorname{argmin}} \sum_{i=1}^{3} \left[y_i - (m x_i + c)\right]^2 \tag{14}$$

where m and c are the parameters of the linear equation. The goal of the least squares fitting method is to minimize the sum of the squares of the perpendicular distances from the given characteristic points to the line. y_i represents the y-coordinate of the ith characteristic point, and x_i represents the x-coordinate of the ith characteristic point, resulting in m_1, c_1, m_2 and c_2. Based on the expressions of the first and second linear equations, the intersection point $p_I(x_{\mathrm{int}}, y_{\mathrm{int}})$ of the two lines can be calculated using the

formula:

$$\left.\begin{aligned} x_{\text{int}} &= \frac{c_2 - c_1}{m_1 - m_2} \\ y_{\text{int}} &= m_1 x_{\text{int}} + c_1 \end{aligned}\right\} \tag{15}$$

Based on the intersection point p_{I} of the two lines, the bisector of the angle between them is calculated. To do this, we select a point $p_{\boldsymbol{F}_{\max}}$ from set $\boldsymbol{F}_{\max}$ and another point $p_{\boldsymbol{F}_{\min}}$ from set $\boldsymbol{F}_{\min}$. Using these points and the intersection point p_{I}, we construct two vectors, $\overrightarrow{vec_1} = \frac{\overrightarrow{p_{\text{I}} p_{\boldsymbol{F}_{\max}}}}{\|\overrightarrow{p_{\text{I}} p_{\boldsymbol{F}_{\max}}}\|}$ and $\overrightarrow{vec_2} = \frac{\overrightarrow{p_{\text{I}} p_{\boldsymbol{F}_{\min}}}}{\|\overrightarrow{p_{\text{I}} p_{\boldsymbol{F}_{\min}}}\|}$. The direction vector of the angle bisector can then be calculated using the following method:

$$\overrightarrow{vec_{bi}} = \overrightarrow{vec_1} + \overrightarrow{vec_2} = (v_x, v_y) \tag{16}$$

Then we can calculate the slope m_{bi} and intercept c_{bi} of the straight line where the angular bisector vector is located:

$$\left.\begin{aligned} m_{bi} &= \frac{v_y}{v_x} \\ c_{bi} &= y_{\text{int}} - m_{bi} x_{\text{int}} \end{aligned}\right\} \tag{17}$$

As shown in Fig. 8b, the green dashed line is the calculated angle bisector.

The angle bisector can divide the neighboring point set $\tilde{\boldsymbol{N}}_i^{\text{2D}}$ into two parts, upper and lower. We designate the points that lie in the same part as the target point p_i as co-planar neighbors of p_i. The formula for determining the relative position of any point with respect to the angle bisector is as follows:

$$loc_{ind} = y_{ind} - (m_{bi} x_{ind} - c_{bi}) \tag{18}$$

where $(x_{ind}, y_{ind}) \in \tilde{\boldsymbol{N}}_i^{\text{2D}}$ $(ind = 1, 2, \ldots, n)$ denotes the index of a neighboring point. When $loc_{ind} > 0$, the point is positioned above the angle bisector, whereas $loc_{ind} < 0$ signifies that the point is below it. Using Eq. (18), we determine the positional relationship loc_{p_i} between the target point and the angle bisector, which enables us to construct a new index set of neighboring points $\boldsymbol{Idx}_i^{\text{sub}} \subset \mathbb{R}$ (e.g. the indices corresponding to the purple points in Fig. 9). Using the index set $\boldsymbol{Idx}_i^{\text{sub}}$, the corresponding neighboring points from $\boldsymbol{N}_i$ are retrieved to construct a new neighboring point set $\boldsymbol{N}_i^{\text{sub}}$.

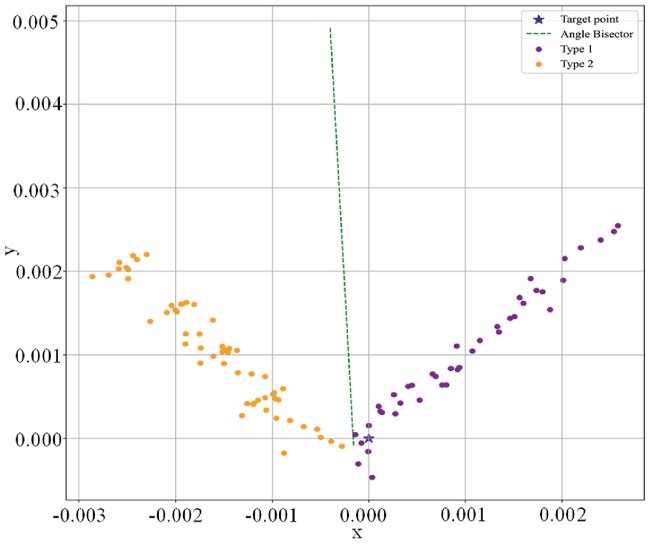

Fig. 9. Nearest neighbor set segmentation.

2.6. Correction

In the classification of sharp feature points, we initially only distinguished between planar and sharp points. However, sharp feature points can be further subdivided into two categories: edge points (as shown in Fig. 10a), whose nearest neighbors span only two planes, and corner points (as shown in Fig. 10b), whose nearest neighbors span more than two planes. For edge points, coplanarity between $\boldsymbol{N}_i^{\text{sub}}$ and the target point p_i can be maintained, whereas for corner points, this coplanarity cannot be guaranteed. Thus, it is necessary to further classify the sharp feature points and apply special handling to the corner points. The subsection is structured into three parts: subclassification, edge point normal vector correction, and corner point normal vector correction.

2.6.1. Subclassification

We utilize the nearest neighbor set $\boldsymbol{N}_i^{\text{sub}}$ to further classify sharp feature points into edge and corner points. The core principle of this subclassification is described as follows. if the current target point p_i is an edge point, then the points within its nearest neighbor set $\boldsymbol{N}_i^{\text{sub}}$ exhibit a high degree of coplanarity; whereas if p_i is a corner point, the coplanarity among points in the set $\boldsymbol{N}_i^{\text{sub}}$ is significantly lower. Using this criterion, we establish the rules for detailed classification.

Let $\boldsymbol{S} = \left\{ s_k = \sigma(p_k) \middle| p_k \in \boldsymbol{P}_f, k = 1, \ldots, K \right\}$ denote the curvature values of all points in the planar feature point set $\boldsymbol{P}_{\text{f}}$. A threshold value $\theta_f = \frac{1}{K} \sum_{k=1}^{K} s_k$ is defined, and the curvature $\sigma_{\text{sub}}(p_i)$ of $\boldsymbol{N}_i^{\text{sub}}$ is then computed using Eqs. (6) and (7). We define $\alpha \in \mathbb{R}$ as a balancing coefficient. If $\sigma_{\text{sub}}(p_i) > \alpha\theta_{\text{f}}$, the current target point p_i is classified as a corner point; otherwise, it is classified as an edge point. As illustrated in Fig. 11, the red points in Fig. 11a represent the sharp feature points obtained from the initial classification, whereas the red points in Fig. 11b represent the corner points identified through the detailed subclassification.

2.6.2. Edge point normal vector correction

The nearest neighbor set $\boldsymbol{N}_i^{\text{sub}}$ of edge feature points exhibits a high degree of coplanarity, which allows for accurate normal vector estimation through PCA. However, the segmented nearest neighbor set $\boldsymbol{N}_i^{\text{sub}}$ may sometimes contain an insufficient number of points, which could undermine the robustness of the algorithm. To enhance robustness, this study proposes an adaptive expansion rule for the nearest neighbor set. As illustrated in Fig. 12, this

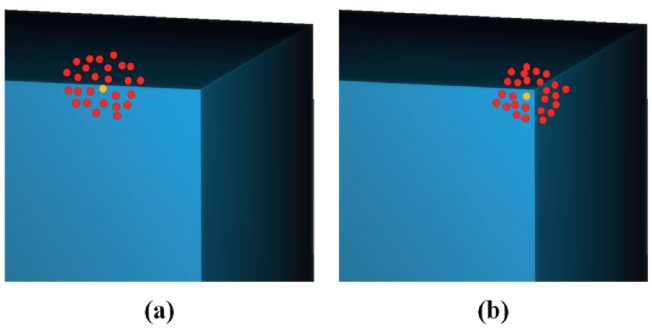

Fig. 10. Types of sharp feature points: (a) An edge point; and (b) A corner point.

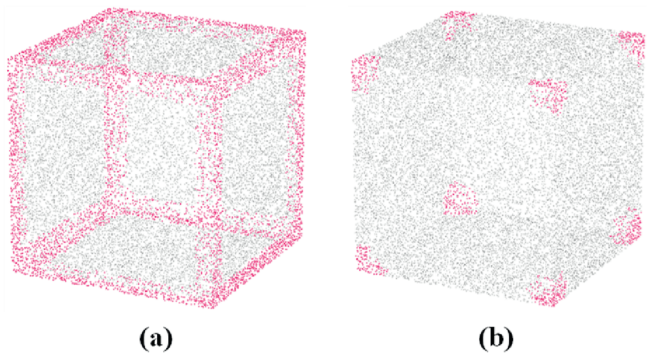

Fig. 11. Sharp feature points and corner points: (a) A sharp feature point represented by the red points; and (b) A corner point represented by the red points.

diagram depicts the expansion of the nearest neighbor set.

Let M denote the number of points in the nearest neighbor set $\boldsymbol{N}_i^{\text{sub}}$, and define a threshold value θ_M. If $M \geq \theta_M$, $\boldsymbol{N}_i^{\text{sub}}$ is directly used for PCA to obtain the corrected normal vector $\vec{n}_i$. If $M < \theta_M$, the point count is deemed insufficient for accurate normal vector estimation, necessitating the expansion of $\boldsymbol{N}_i^{\text{sub}}$. Using the indices from $\boldsymbol{Idx}_i^{\text{sub}}$, the corresponding points in $\boldsymbol{N}_i^{\text{2D}}$ are located to construct the set $\boldsymbol{N}_i^{\text{Sub2D}}$. The point in $\boldsymbol{N}_i^{\text{Sub2D}}$ with the maximum absolute y-coordinate value is identified, denoted as R, and its corresponding point in $\boldsymbol{N}_i^{\text{sub}}$ is designated as the seed point p_{seed}. Using p_{seed} as the center and R as the radius, an expanded nearest neighbor set $\boldsymbol{N}_{\text{seed}}$ is obtained. The final expanded set $\boldsymbol{N}_i^{\text{e}}$ is formed by merging $\boldsymbol{N}_i^{\text{sub}}$ and $\boldsymbol{N}_{\text{seed}}$ while removing duplicates. PCA is then performed on $\boldsymbol{N}_i^{\text{e}}$ to determine the corrected normal vector $\vec{n}_i$. Fig. 13 depicts the corrected normal vector for edge points.

2.6.3. Corner point normal vector correction

The correction of normal vectors for corner points is carried out after completing all edge point normal vector corrections. Since the neighbor set $\boldsymbol{N}_i^{\text{sub}}$ associated with a corner point does not guarantee coplanarity, a clustering method is employed for normal vector correction. Fig. 14 depicts the process of correcting the normal vector for corner points.

When the target point p_i is classified as a corner point, the neighboring points can be categorized into three types: planar points, edge points, and corner points. Through the correction of edge point normal vectors, the normal vector information for both

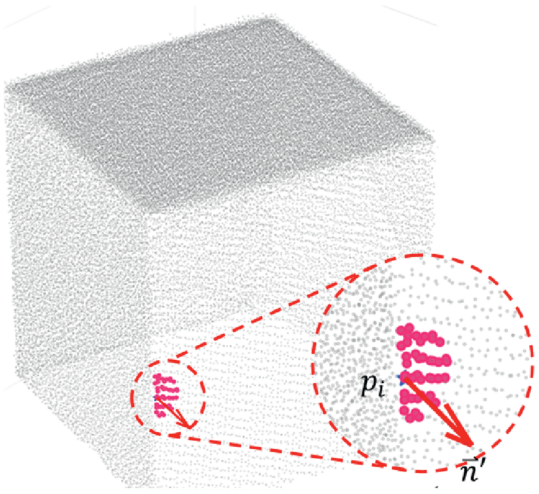

Fig. 13. Corrected normal vector of edge points.

planar and edge points is obtained. For correcting corner point normal vectors, the core idea is to effectively leverage the known normal vector information within its neighborhood and classify different types of normal vectors through clustering analysis. Subsequently, the initial normal vector of the target point is compared with the normal vectors within each cluster, and the cluster center with the smallest deviation is selected as the final estimation of the target point's normal vector.

The similarity measure between normal vectors is defined as

$$Similarity = \frac{\langle \vec{a}, \vec{b} \rangle}{\|\vec{a}\| \|\vec{b}\|} \tag{19}$$

where $\langle \vec{a}, \vec{b} \rangle$ represents the inner product of two normal vectors $\vec{a}$ and $\vec{b}$. A threshold value $\theta_{simalrity}$ is defined such that if $Similarity \leq \theta_{similarity}$, the two normal vectors are considered part of the same cluster. Using this similarity measure, clustering is performed on the neighboring points of p_i, resulting in normal vector clusters ($\boldsymbol{C}_1, \boldsymbol{C}_2, \ldots,$ and $\boldsymbol{C}_k$). The mean normal vector of each cluster is computed as $\vec{n}_{C_1}$, $\vec{n}_{C_2}$, …, and $\vec{n}_{C_k}$. Subsequently, the similarity measure between the current normal vector $\vec{n}'$ of point p_i and each cluster is calculated using Eq. (20). The normal vector with the highest similarity is then selected as the corrected normal

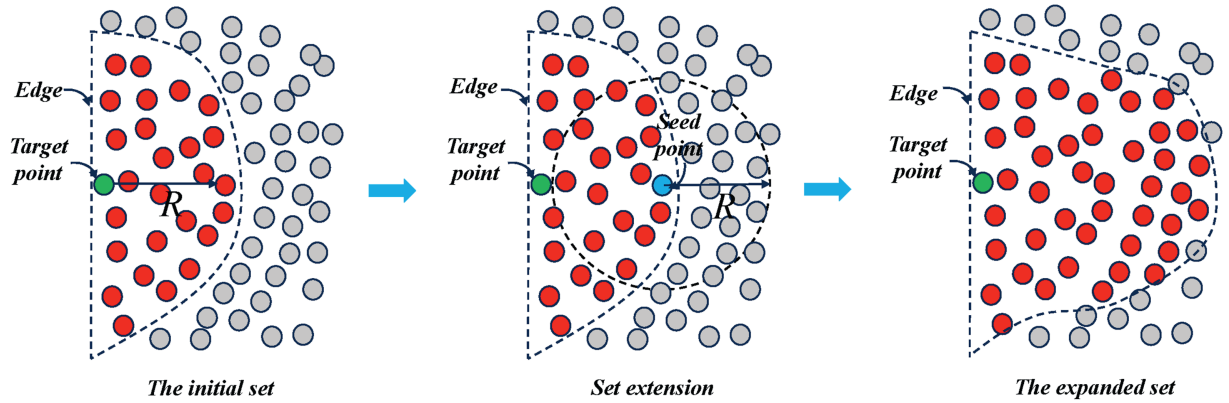

Fig. 12. Diagram of extended neighboring point set.

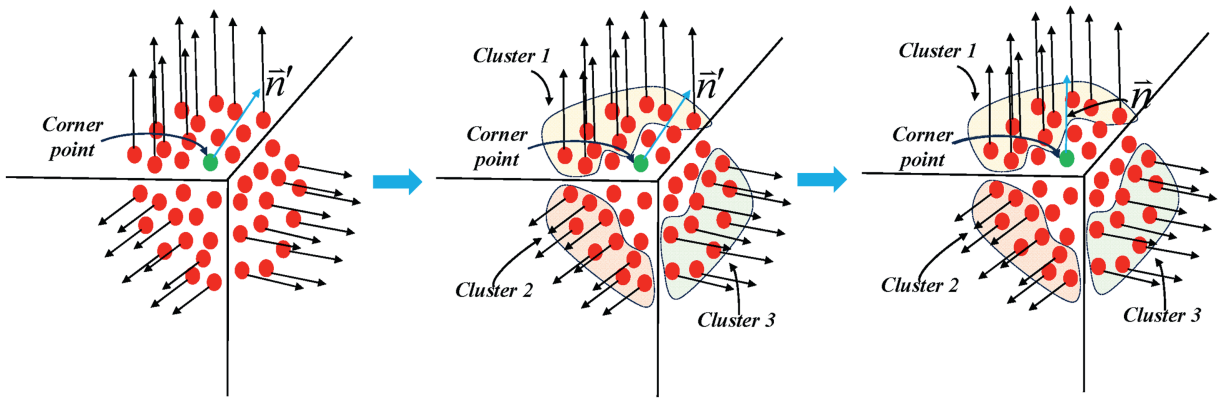

Fig. 14. Diagram of corner point normal vector correction.

vector of p_i, denoted by $\vec{n}$. Fig. 15a shows the normal vectors of corner points before correction, while Fig. 15b presents the corrected normal vectors.

3. Experiment results and discussion

Based on the outlined workflow, the algorithm was implemented in Python within the Visual Studio (VS) code development environment. To evaluate the performance of the proposed method, three sets of experiments were conducted on a Lenovo laptop (2.20 GHz Intel(R) Core(TM) i9-14900HX, 32.0 GB RAM). The first set of experiments focused on computer-aided design (CAD) models, including cubes and octahedra, for both qualitative and quantitative analysis. For comparison, eight normal vector estimation methods were selected (as shown in Table 1), encompassing traditional, deep learning, and post-processing approaches. The traditional methods included PCA, 2-jet, and Voronoi. The deep learning category consisted of the supervised method PCPNet and the unsupervised method NeuralGF. The post-processing category comprised LRR, NSN, and PCV. The second set of experiments utilized real-world point clouds of cubes and square pyramids for qualitative analysis. The third set investigated the algorithm's effectiveness in processing actual rock mass point clouds.

To facilitate a quantitative comparison of the algorithms, this study uses two evaluation metrics: root mean square error (RMSE), denoted as RMS_τ, and the number of outliers, denoted as *NBP*. The definition of RMSE is as follows:

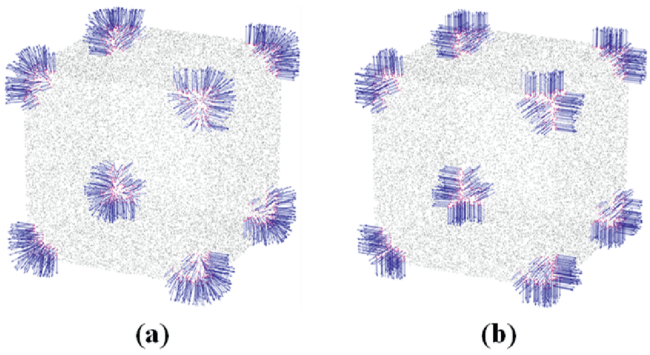

Fig. 15. Corner point normal vector calculation: (a) Normal vector calculation results without corner point processing; and (b) Normal vector results calculated using the proposed method.

$$\left.\begin{aligned} RMS_\tau &= \sqrt{\frac{1}{N}\sum_{p\in \boldsymbol{P}} f(n_p,\tilde{n}_p)^2} \\ f(n_p,\tilde{n}_p) &= \begin{cases} \widehat{n_p\tilde{n}_p} & (\widehat{n_p\tilde{n}_p} < \tau) \\ \pi/2 & (\text{otherwise}) \end{cases} \end{aligned}\right\} \tag{20}$$

where $\boldsymbol{P}$ represents the point cloud set, and N is the number of points within $\boldsymbol{P}$; and n_p and $\tilde{n}_p$ denote the true and the estimated normal vectors of p ($p \in \boldsymbol{P}$), respectively. The angle between these two vectors is denoted as $\widehat{n_p\tilde{n}_p}$, while τ is a constant, set to 10°. If the angle between the estimated and the true normal vectors exceeds a threshold τ, the point is classified as an outlier. The total number of outliers in set $\boldsymbol{P}$ is represented by *NBP*.

3.1. CAD model

CAD models are highly precise, 3D digital representations generated using specialized software, widely utilized in engineering design, architecture, and manufacturing. These models accurately encode an object's geometry, dimensions, and structural details, enabling high-fidelity simulations of complex structures. By extracting point cloud data from CAD models, we can obtain ground-truth normal vectors, providing a reliable basis for both qualitative and quantitative evaluations of the proposed algorithm.

3.1.1. Cube

Fig. 16 illustrates the cube CAD model, where Fig. 16a depicts the original CAD model, and Fig. 16b presents the corresponding point cloud extracted from it. The data processing workflow begins with utilizing CloudCompare software to extract normal vectors from the cube CAD model as ground truth. Next, point cloud data are randomly sampled from the CAD model while preserving the corresponding ground-truth normal vectors for each point, resulting in a total of 20,000 extracted points. Finally, Gaussian noise is introduced at varying levels of 10 %, 30 %, 50 %, 70 %, and 100 % to the extracted cube point cloud, forming the experimental dataset.

Fig. 17 compares the normal vector estimation results of different algorithms under noisy conditions. The traditional PCA and 2-jet algorithms produce smooth transitions at sharp features, leading to limited feature preservation. The Voronoi algorithm demonstrates an improvement over PCA and 2-jet under low-noise conditions; however, its robustness is inadequate, and its

Table 1
Code sources and parameters of the compared methods.

Method	Code source/Implementation	Parameters
PCA	built-in plugins of CloudCompare	$k = 50$
2-jet	built-in plugins of CloudCompare	$k = 50, \text{radius} = 0.5$
Voronoi	built-in plugins of CloudCompare	$k = 50, \text{radius} = 0.5$
PCPNet	https://github.com/paulguerrero/pcpnet	Pre-trained model
NeuralGF	https://github.com/LeoQLi/NeuralGF	Parameters require training
LRR	https://github.com/jjcao/sf-pcd2013	$S = 20, S^* = 60, k = 10, r = 4$
NSN	https://github.com/AnkaChan/NormalEstimatePatchShift	$\Omega_c = 0.35, K_0 = 60, S = 100$
PCV	https://github.com/jjcao/pcv	$k = 10, \alpha = 4, \beta = 2$

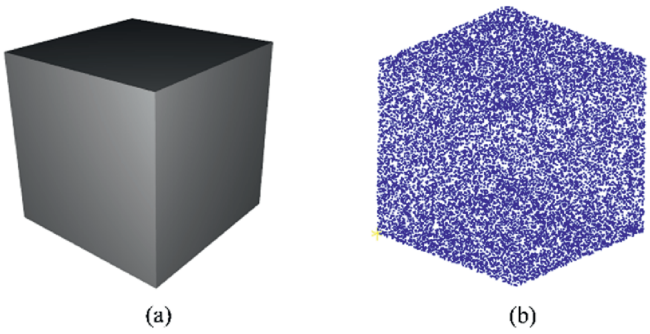

Fig. 16. The cube CAD model: (a) CAD model; and (b) Point cloud extracted from (a).

performance deteriorates significantly as noise increases. The supervised learning-based PCPNet model exhibits poor generalization capability – while it slightly outperforms traditional methods in feature preservation, its results remain inconsistent. The unsupervised learning algorithm NeuralGF achieves higher accuracy than PCPNet and traditional methods, but suffers from weak noise resistance. In contrast, feature-preserving algorithms such as LRR, NSN, and PCV offer significant advantages. Among them, LRR and PCV exhibit comparable performance in preserving sharp features, whereas the NSN algorithm introduces minor errors in feature regions. The proposed algorithm in this study achieves feature preservation on par with LRR and PCV, effectively retaining sharp features in the point cloud. To provide a more intuitive visualization of normal vector estimation results, we project the normal vectors of the dataset with a 50 % noise level onto the unit sphere, as illustrated in Fig. 18.

To quantitatively evaluate the performance of the algorithms, we conducted a statistical analysis of their results under varying noise levels, as shown in Table 2 and Fig. 19. The analysis indicates that the PCA and 2-jet algorithms exhibit comparable computational accuracy (*RMS* in the range of 0.64–0.68) and a relatively high number of bad points (*NBP* ranging from 3296 to 3730). While these methods demonstrate strong robustness, they fail to effectively preserve sharp features. The Voronoi method surpass PCA and 2-jet at a 10 % noise level, achieving an *RMS* of 0.4696 and an *NBP* of 1763. However, its robustness is limited, as both *RMS* and *NBP* increase sharply when the noise level rises from 10 % to 100 %. PCPNet, due to its insufficient model generalization, exhibits the poorest computational accuracy, highlighting the strong

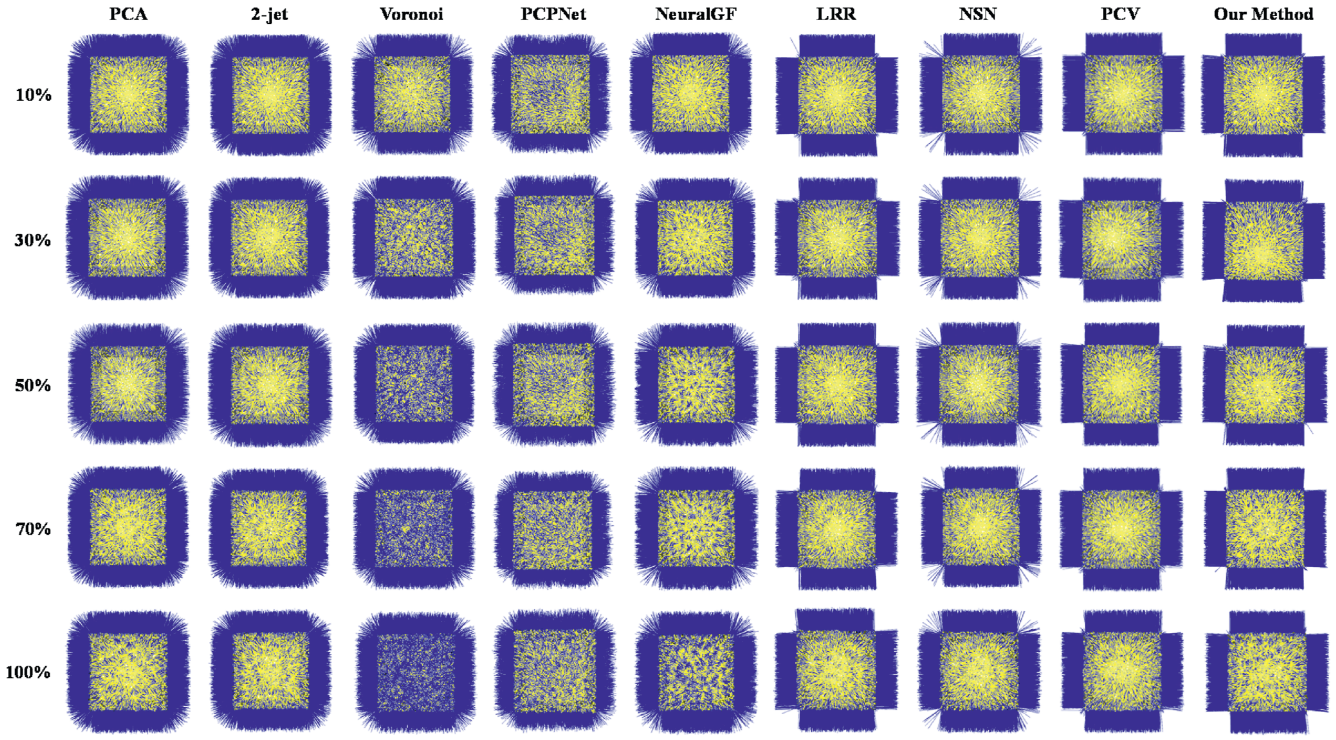

Fig. 17. Results of the cube.

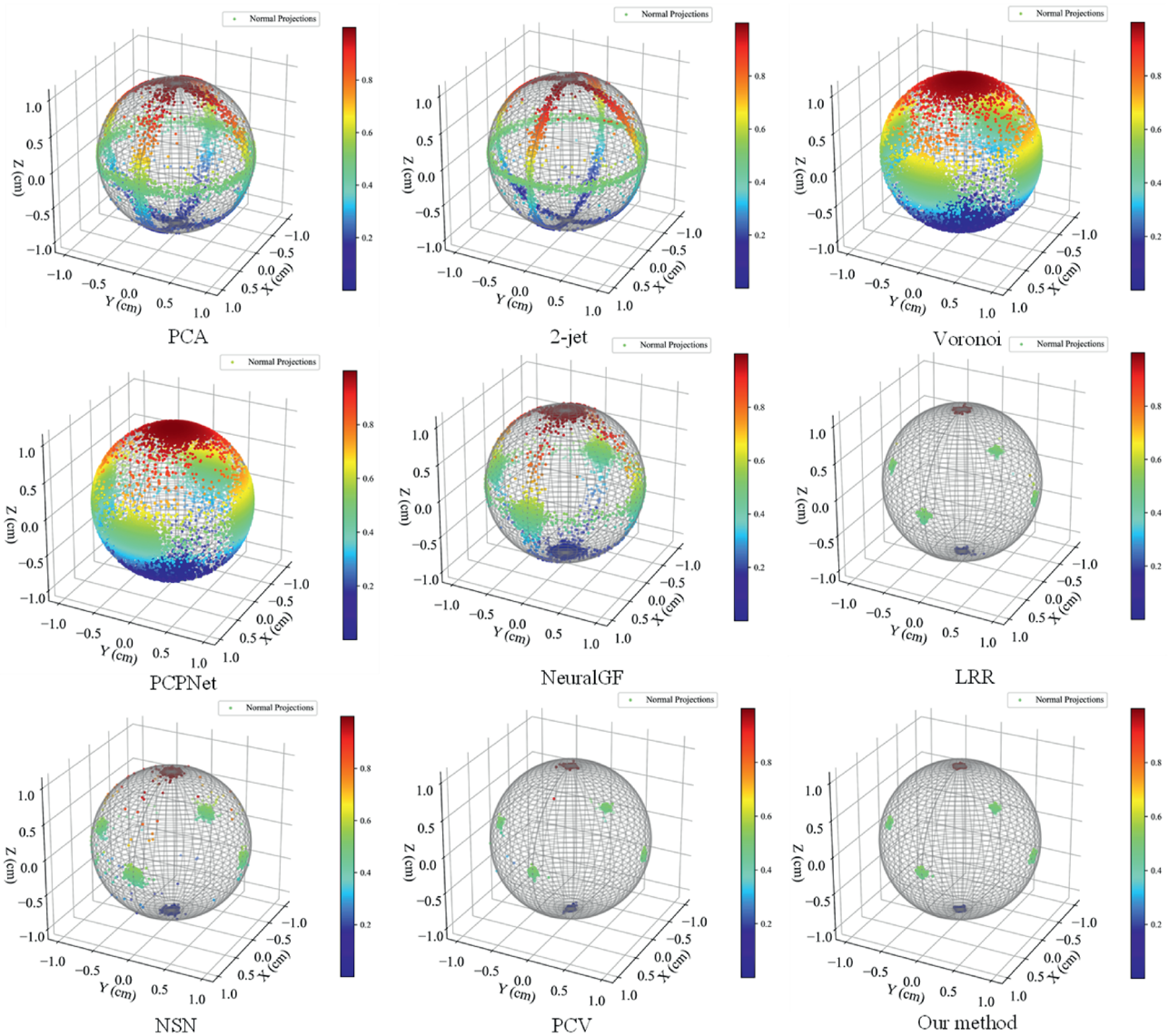

Fig. 18. The projection of normal vector calculation results of different algorithms on the unit sphere at a 50 % noise level.

Table 2
Results of the cube.

Method	10 %		30 %		50 %		70 %		100 %	
	NBP	RMS	NBP	RMS	NBP	RMS	NBP	RMS	NBP	RMS
PCA	3296	0.6382	3311	0.6398	4342	0.733	3401	0.6492	3472	0.6569
2-jet	3559	0.6631	3561	0.6634	3394	0.648	3554	0.6635	3730	0.6801
Voronoi	1763	0.4696	6463	0.8967	11680	1.2025	14845	1.3545	17180	1.4564
PCPNet	20001	1.5707	19999	1.5706	19990	1.5703	19985	1.5701	19976	1.5691
NeuralGF	1044	0.3603	1173	0.3846	2025	0.5065	4789	0.7737	9276	1.0728
LRR	154	0.1384	212	0.1625	285	0.1889	408	0.2265	625	0.2814
NSN	274	0.1841	412	0.2263	502	0.2509	730	0.303	1408	0.4201
PCV	87	0.1041	189	0.1533	249	0.1765	346	0.2084	500	0.2512
Proposed	132	0.1321	191	0.1558	233	0.1716	330	0.2054	485	0.2505

dependence of supervised learning methods on training data quality and model robustness. NeuralGF outperforms traditional methods in the 10 %–50 % noise range (*RMS* ranging from 0.36 to 0.51, and *NBP* in a range of 1044–2025), yet its robustness remains limited, with a significant increase in *NBP* at a 100 % noise level. Feature-preserving algorithms such as LRR, NSN, and PCV demonstrate notable advantages over both traditional and deep learning-based approaches. Among them, NSN performs slightly worse than LRR and PCV (*RMS* = 0.1841–0.4201, and *NBP* = 274–1408), whereas LRR and PCV achieve superior results (*RMS* = 0.1041–0.2814, and *NBP* = 87–625). The proposed method achieves outstanding overall performance (*RMS* = 0.1321–0.2505,

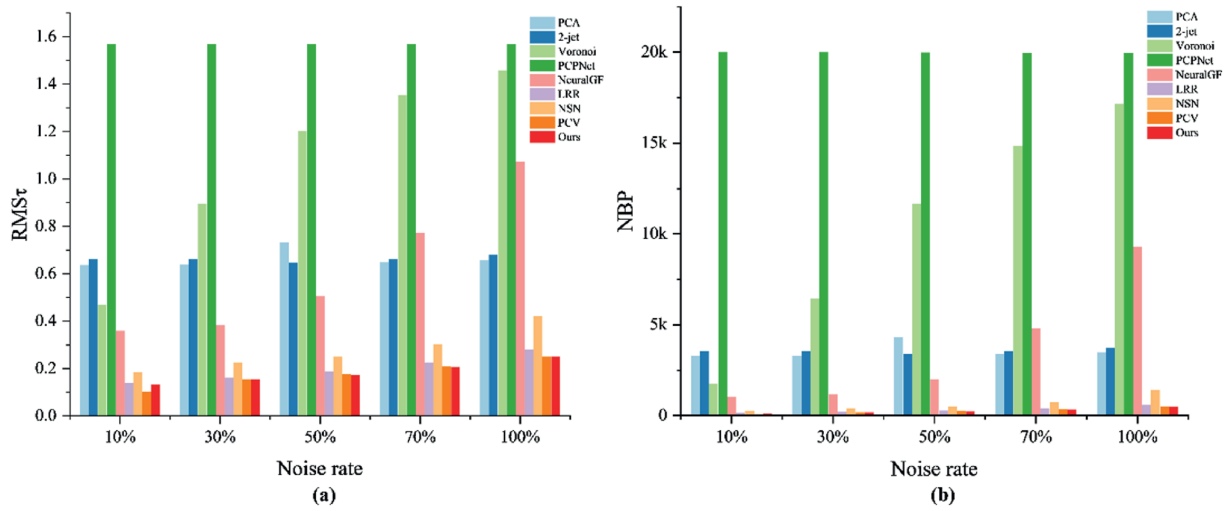

Fig. 19. Quantitative analysis results: (a) *RMS* statistical results; and (b) *NBP* statistical results.

and *NBP* = 132–485), striking a balance between high robustness and strong computational accuracy.

3.1.2. Octahedron

Fig. 20 illustrates the octahedron CAD model, where Fig. 20a depicts the original CAD model, and Fig. 20b presents the point cloud extracted from it. The data processing procedure follows the same steps as previously described, with a total of 20,000 points extracted. Subsequently, Gaussian noise is introduced at levels of 10 %, 30 %, 50 %, 70 %, and 100 % to the extracted octahedron point cloud, serving as the experimental dataset.

Fig. 21 presents a comparative analysis of the normal vector estimation results across different algorithms at various noise levels. The PCA algorithm maintains consistent performance across various noise levels, but it struggles to effectively preserve sharp features. The 2-jet algorithm performs better than PCA under low noise conditions and retains sharp features to some extent; however, as noise increases, its performance gradually converges with that of PCA. The Voronoi algorithm demonstrates superior performance to PCA at low noise levels but exhibits higher sensitivity to noise. Among deep learning methods, the unsupervised learning algorithm NeuralGF significantly outperforms the supervised learning-based PCPNet, which is constrained by the quality of training data and the model limitations. The LRR and PCV algorithms perform similarly, both excelling in preserving sharp features, whereas the NSN algorithm exhibits slightly inferior performance compared to the two. The proposed algorithm achieves results comparable to LRR and PCV, demonstrating strong capability in preserving sharp features within the point cloud. To enhance the intuitive visualization of normal vector estimation results, the normal vectors corresponding to the data with 50 % noise level were projected onto the unit sphere, as depicted in Fig. 22.

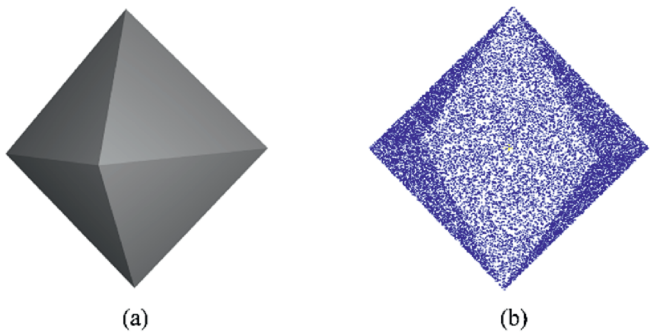

Fig. 20. The octahedron CAD model: (a) CAD model; and (b) Point cloud extracted from (a).

The quantitative analysis results based on the octahedron data are presented in Table 3 and Fig. 23. The *RMS* values for traditional algorithms range from 0.4883 to 1.4321, while the *NBP* values vary between 1901 and 16,605. Among them, the PCA algorithm exhibits an *RMS* of 0.6733–0.6875 and an *NBP* of 3665–3805. Although its accuracy is relatively low, it demonstrates strong robustness. The 2-jet and Voronoi algorithms outperform PCA at low noise levels but exhibit weaker robustness. The supervised learning-based PCPNet yields the worst performance due to its insufficient model generalization capability. In contrast, the unsupervised learning algorithm NeuralGF performs exceptionally well at noise levels between 10 % and 50 % (*RMS* = 0.3725–0.4687, and *NBP* = 1117–1741) but degrades to the level of traditional algorithms when the noise level increases to 70 %–100 %. The LRR, NSN, and PCV algorithms generally perform well, with NSN (*RMS* = 0.1926–0.4694, and *NBP* = 298–1762) slightly underperforming compared to LRR and PCV. The proposed algorithm (*RMS* = 0.1332–0.2894, and *NBP* = 138–652) achieves performance comparable to LRR and PCV, exhibiting optimal accuracy at noise levels between 10 % and 70 % and demonstrating excellent robustness.

3.1.3. Computational efficiency

Both qualitative and quantitative analyses indicate that while the PCA algorithm cannot preserve sharp features, it exhibits strong resistance to noise. The 2-jet and Voronoi algorithms outperform PCA at low noise levels but demonstrate poor noise robustness, with accuracy deteriorating to levels even lower than PCA as noise increases. The accuracy of the supervised learning-based PCPNet is constrained by the quality of training data and the model, whereas the unsupervised learning algorithm NeuralGF, despite outperforming traditional methods, suffers from insufficient noise robustness. The NSN algorithm performs well in preserving features but is slightly inferior to the LRR and PCV algorithms. The proposed algorithm surpasses LRR in accuracy and performs comparably to PCV.

To further assess the algorithm's performance, a computational

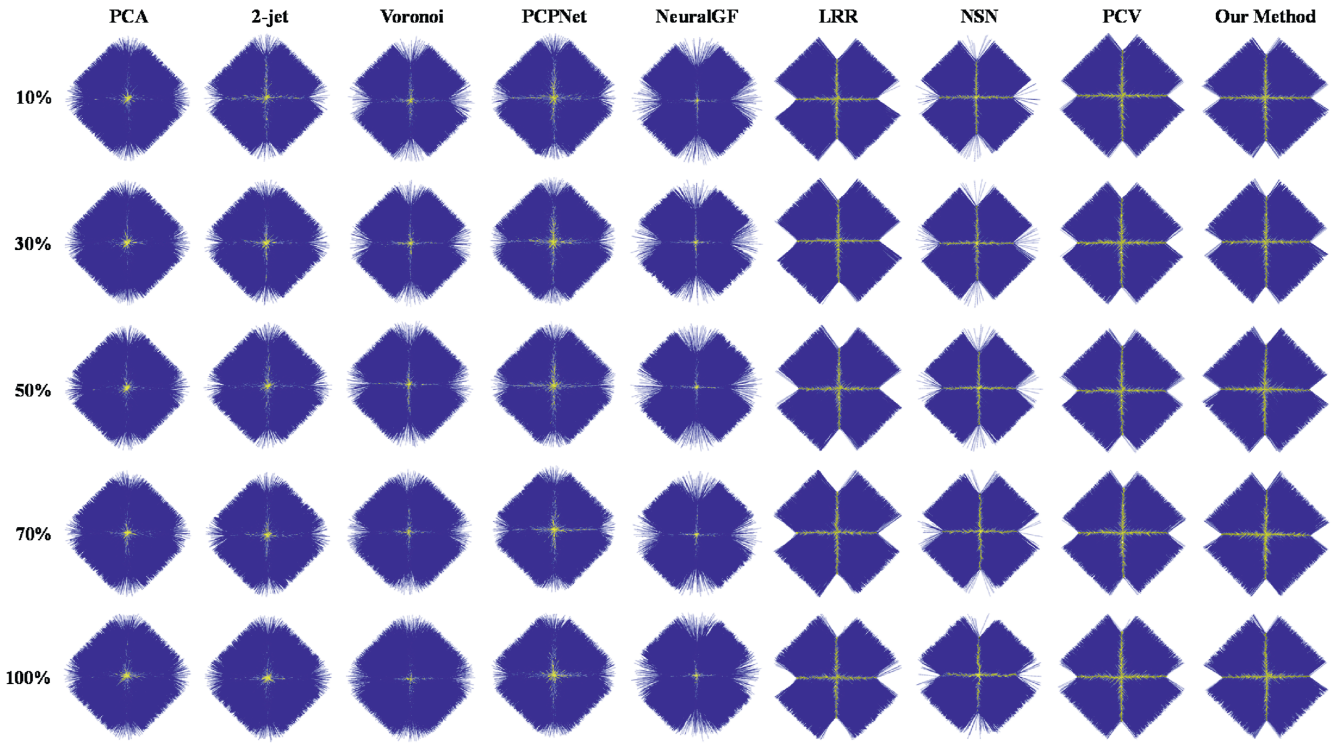

Fig. 21. Results of the octahedron.

efficiency analysis was conducted. Table 4 presents the computation time of various algorithms across different models and noise levels. The results indicate that the algorithms generally fall into two categories: traditional algorithms, which are computationally efficient but cannot preserve sharp features, and feature-preserving algorithms, including NSN, PCV, and the proposed algorithm. According to the data, the proposed algorithm demonstrates superior efficiency compared to both NSN and PCV in terms of average computation time for the cube and octahedron models. Therefore, based on a comprehensive assessment, the proposed algorithm achieves the best overall performance among the compared algorithms.

3.2. Real data

To further validate the performance of the proposed algorithm, normal vector estimation experiments were conducted using real-world data. Since obtaining precise ground-truth normal vectors for real objects is challenging, this section primarily employs qualitative analysis to evaluate the algorithm's effectiveness. The real-world dataset used in this study originates from Riquelme et al. (2014), with cube and square pyramid models selected as experimental data. PCA, NeuralGF, PCV, and the proposed algorithm were chosen for comparative analysis. Fig. 24 presents the point cloud data used in this section, where Fig. 24a depicts a bottomless cube comprising 60,488 points, and Fig. 24b illustrates a triangular prism consisting of 37,226 points. These data were acquired using the Konica Minolta Vivid 9i 3D laser scanner.

Fig. 25 presents the projection distributions of normal vectors estimated using PCA, NeuralGF, PCV, and the proposed method on the unit sphere for the cube point cloud. The experimental results reveal that both PCA and NeuralGF algorithms exhibit significant errors in the sharp feature regions, demonstrating poor feature preservation capabilities. While the PCV algorithm outperforms PCA and NeuralGF, it still introduces two notable errors in the sharp feature areas. In contrast, the proposed method achieves the best performance, substantially surpassing the comparison algorithms in feature preservation.

Fig. 26 illustrates the projection distributions of normal vectors estimated using PCA, NeuralGF, PCV, and the proposed method on the unit sphere for the triangular prism point cloud. The results indicate that both the PCA and NeuralGF algorithms struggle to preserve sharp features effectively. Although both PCV and the proposed method exhibit some erroneous estimations, their overall performance is significantly superior to that of PCA and NeuralGF. Compared to the PCV, the proposed method produces a more concentrated projection distribution, indicating improved feature preservation.

3.3. Rock mass data

To further validate the performance of the proposed algorithm on real rock mass point cloud data, two representative datasets were selected for experimentation. These datasets contain a large number of points, ranging from 400,000 to 800,000, and are characterized by significant noise, outliers, and irregular distribution, posing substantial challenges to the algorithm's robustness and computational efficiency. During the experiment, PCA, NeuralGF, PCV, and the proposed method were employed to estimate the normal vectors of the rock mass point clouds. The estimated normal vectors were subsequently fed into a clustering algorithm to extract discontinuities. Finally, by comparing the clustering results, the accuracy and applicability of each normal vector estimation method were evaluated, thereby verifying the effectiveness of the proposed method in processing real rock mass data.

3.3.1. Case A

The rock mass point cloud for Case A was obtained from the Rockbench repository (Lato et al., 2013). The data were collected from the southwestern part of Ure County in Colorado, USA, using

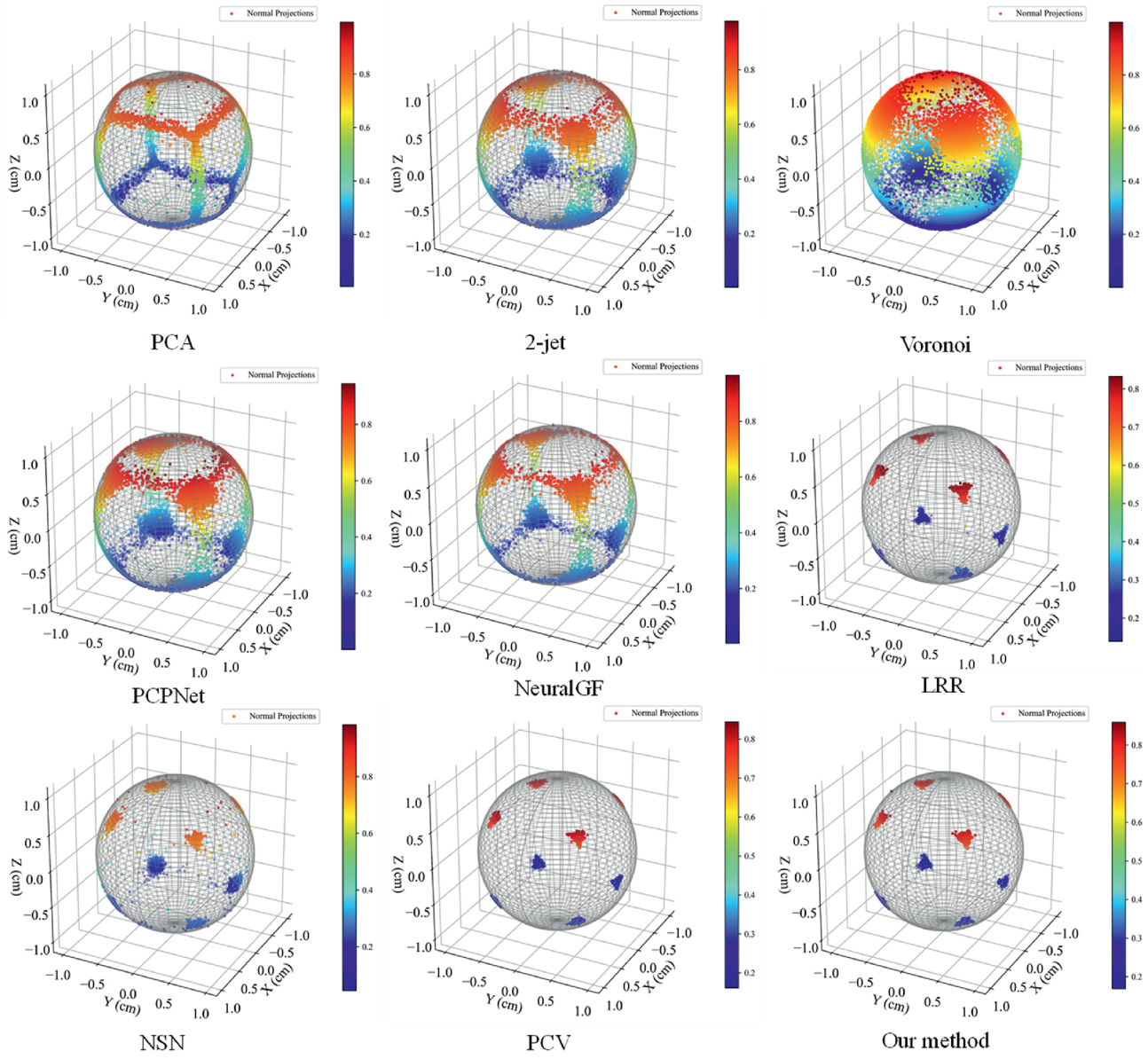

Fig. 22. The projection of normal vector calculation results of different algorithms on the unit sphere at a 50 % noise level.

Table 3
Results of the octahedron.

Method	10 %		30 %		50 %		70 %		100 %	
	NBP	RMS	NBP	RMS	NBP	RMS	NBP	RMS	NBP	RMS
PCA	3671	0.6737	3665	0.6733	3668	0.6739	3746	0.6813	3805	0.6875
2-jet	1901	0.4883	2651	0.5736	3427	0.6565	4391	0.7421	4639	0.761
Voronoi	2221	0.5264	6017	0.8656	11088	1.172	14086	1.3198	16605	1.4321
PCPNet	19877	1.5659	19904	1.567	19920	1.5676	19913	1.5673	19928	1.5679
NeuralGF	1117	0.3725	1351	0.4111	1741	0.4687	3277	0.6416	7752	0.9816
LRR	209	0.1617	359	0.2115	465	0.2416	680	0.292	936	0.3434
NSN	298	0.1926	497	0.2487	626	0.2802	986	0.3517	1762	0.4694
PCV	140	0.1325	241	0.1734	320	0.2003	445	0.2367	641	0.2848
Proposed	138	0.1332	222	0.1676	293	0.1929	413	0.2294	652	0.2894

the Optech 3D laser scanner. The dataset contains a total of 1,515,722 points. Fig. 27a shows the data collection site, with the red box indicating the area studied in this paper. Fig. 27b presents the original point cloud data, which consists of 848,043 points.

To evaluate the performance and practicality of the proposed method in rock mass point cloud processing, we estimated the normal vectors for the Case A point cloud data using PCA, NeuralGF, PCV, and the proposed method. Subsequently, clustering

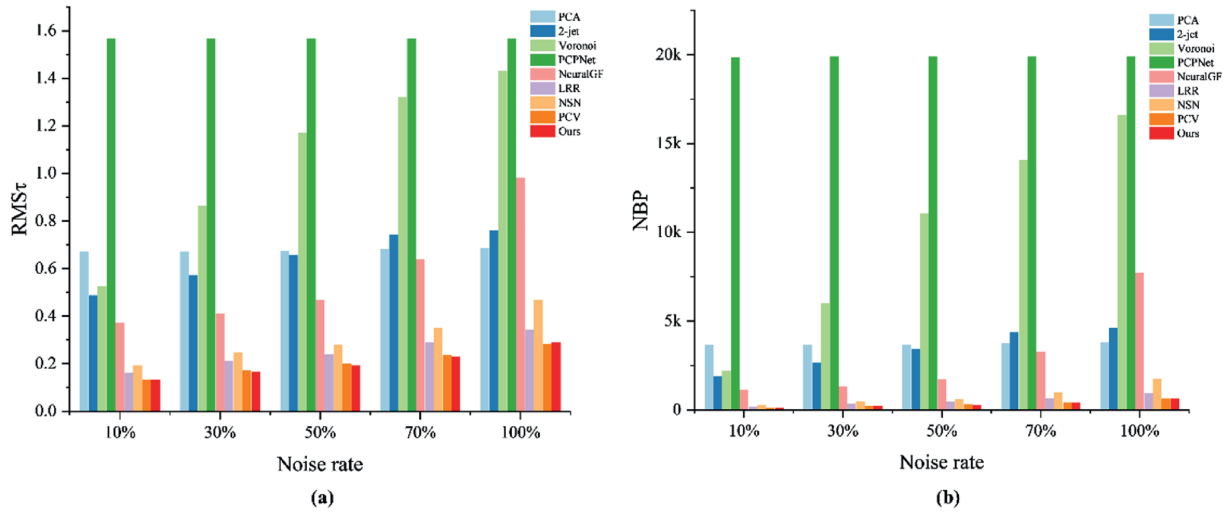

Fig. 23. Quantitative analysis results: (a) *RMS* statistical results; and (b) *NBP* statistical results.

Table 4
Computation time (s).

Method	Cube						Octahdron					
	10 %	30 %	50 %	70 %	100 %	Mean	10 %	30 %	50 %	70 %	100 %	Mean
PCA	0.726	0.718	0.732	0.692	0.715	0.716	0.769	0.764	0.706	0.755	0.733	0.745
2-jet	0.878	1.134	0.984	1.204	1.183	1.076	0.853	0.893	0.904	0.932	0.875	0.891
Voronoi	2.841	1.917	2.617	2.441	2.728	2.508	1.328	1.162	0.989	1.034	1.253	1.153
LRR	727.9	931.2	940.5	515.6	513.1	725.6	542.7	841.4	790.1	720.4	617.5	702.4
NSN	19.02	27.37	19.86	18.87	19.79	20.98	21.7	20.7	20.43	19.48	18.78	20.21
PCV	23.03	24.55	36.64	20.46	20.61	25.06	25.55	22.69	21.82	21.06	20.95	22.41
Proposed	15.44	15.93	21.79	24.23	19.1	19.3	13.91	13.72	15.57	15.83	23.8	16.56

analysis was performed to group the estimated normal vectors and identify distinct discontinuities. Fig. 28 presents the clustering results obtained using different normal vector estimation methods, where different colors represent different discontinuity sets. Fig. 28a illustrates the discontinuities derived from clustering the normal vectors estimated by the PCA method. Since PCA solely relies on the principal direction of the local neighborhood for normal vector estimation, it tends to produce deviations, particularly in edge regions, leading to misclassification in clustering. Fig. 28b shows the clustering results based on the normal vectors estimated by NeuralGF. As an unsupervised deep learning-based approach, NeuralGF improves the accuracy of normal vector estimation in the edge regions to some extent. However, computational biases remain, causing clustering errors. Fig. 28c presents the clustering results obtained using the PCV method. Compared to PCA and NeuralGF, PCV better optimizes normal vector estimation in edge regions, thereby enhancing clustering accuracy. Nonetheless, misclassification persists. Fig. 28d displays the clustering results of the normal vectors estimated by the proposed method. Since the proposed method generates more accurate normal vectors in edge regions, it achieves the best clustering performance, yielding clearer discontinuity divisions and better edge feature preservation.

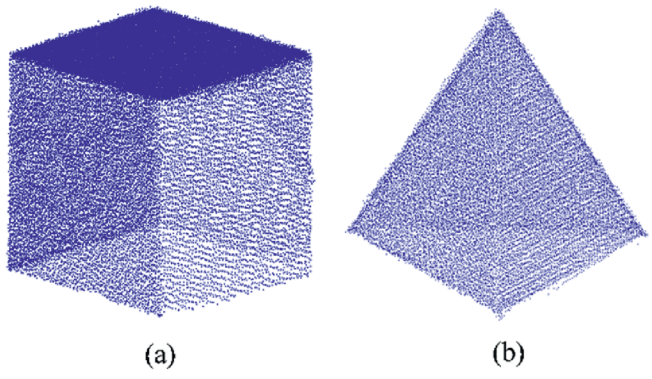

Fig. 24. Real point cloud data: (a) A cube; and (b) An octahedron.

Rock mass edges often contain critical structural surfaces, such as joints, faults, and bedding planes, whose accurate identification is essential for rock mass geometry analysis. Inaccurate normal vector estimation can lead to misidentification of these structural surfaces, thereby affecting the interpretation of rock mass geometric features. Precise normal vector estimation and well-defined clustering results significantly improve the recognition of discontinuity sets, enabling a more accurate characterization of the rock mass structure. Consequently, the accuracy of normal vector estimation directly impacts the clustering results. Compared to other methods, the proposed approach provides superior normal vector estimation in edge regions, thereby enhancing clustering accuracy and improving the recognition of rock mass geometric structure.

3.3.2. Case B

The rock mass point cloud of Case B was collected in the Priorat region of Tarragona province, Catalonia, Spain. The data were acquired using an Optech 3D laser scanner, comprising a total of 1,676,187 points. Fig. 29a illustrates the data collection site, with

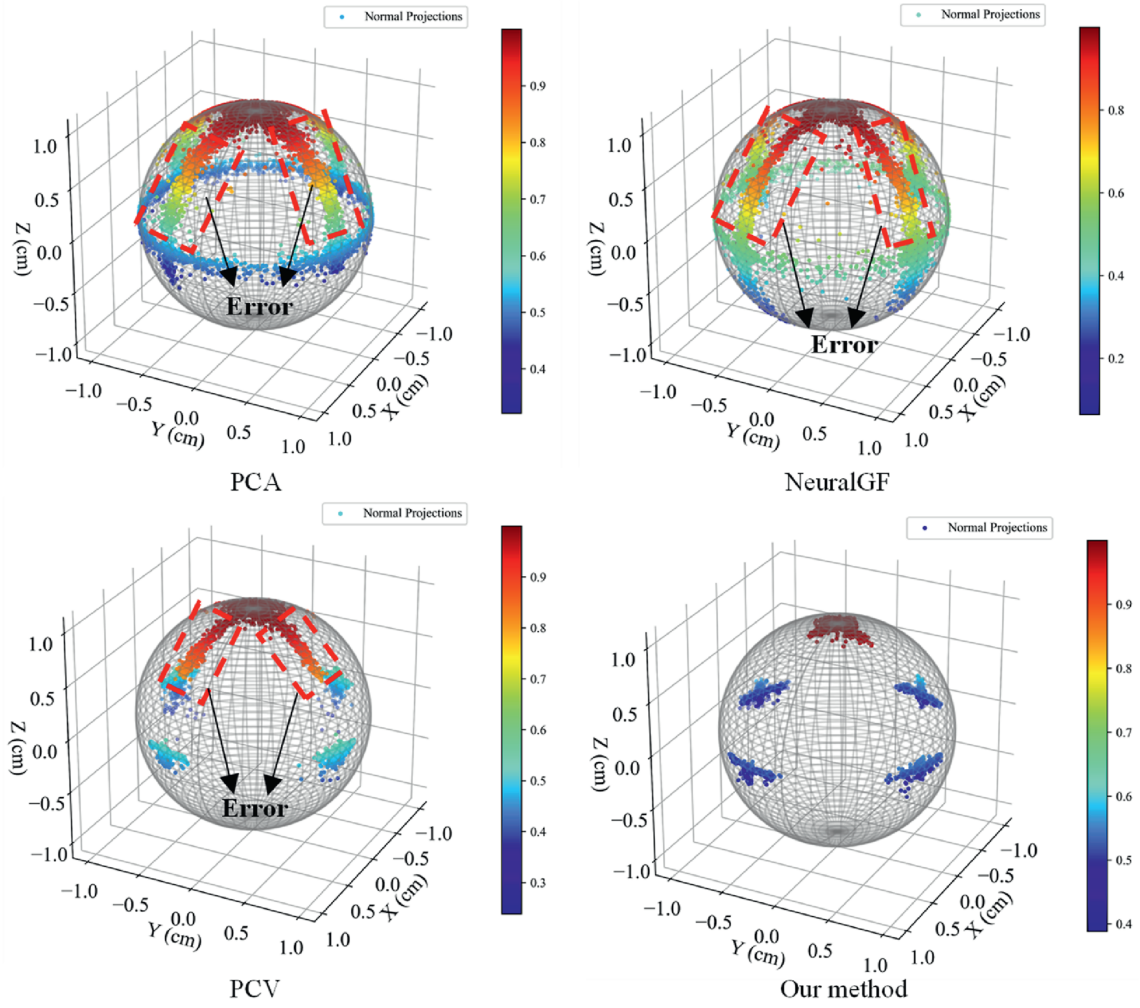

Fig. 25. Results of cube.

the red box highlighting the study area. Fig. 29b presents the original point cloud dataset, which consists of 497,961 points.

To evaluate the performance and practicality of the proposed method for rock mass point cloud processing, we estimated the normal vectors of the Case B point cloud data using PCA, NeuralGF, PCV, and the proposed method. The estimated normal vectors were then grouped using clustering analysis to identify different discontinuity sets. Fig. 30 presents the clustering results obtained using different normal vector estimation methods, where different colors represent distinct discontinuity sets. Fig. 30a shows the clustering results of the normal vectors estimated using the PCA method. Since PCA solely relies on the principal direction of the local neighborhood for normal vector estimation, it tends to exhibit deviations in edge regions, leading to misclassification. Fig. 30b illustrates the clustering result based on normal vectors estimated using the NeuralGF method. As an unsupervised deep learning-based approach, NeuralGF improves the accuracy of normal vector estimation in edge regions to some extent; however, computational biases persist, resulting in misclassification. Fig. 30c presents the clustering results for the normal vectors estimated using the PCV method. Compared to PCA and NeuralGF, this method more effectively optimizes normal vector estimation in edge regions, thereby improving clustering accuracy. Nevertheless, misclassification is not eliminated. Fig. 30d illustrates the clustering results for the normal vectors calculated using the proposed method. Since this method achieves higher accuracy normal vector estimation for edge regions, it produces the best clustering performance, yielding clearer discontinuity set divisions and better retention of edge features.

3.4. Discussion

3.4.1. Capability for processing large-scale data

Our method primarily focuses on processing sharp feature points within the point cloud, while planar points are directly handled using the PCA-estimated normal. Consequently, regardless of the size of the point cloud, the computational cost of the algorithm is mainly concentrated on the treatment of sharp feature points, which constitutes the key factor affecting overall efficiency. In the implementation, we fully exploit the advantages of modern multi-core processors by designing a multi-process framework, which substantially enhances computational efficiency. In addition, data preprocessing through downsampling can further reduce the data volume and improve efficiency. For example, Case A contains 848,043 points, and the computation using the proposed method takes 46.99 s. Case B contains 497,961 points, and the computation requires 20.94 s. These results demonstrate that the proposed method can still achieve satisfactory computational efficiency when applied to large-scale datasets.

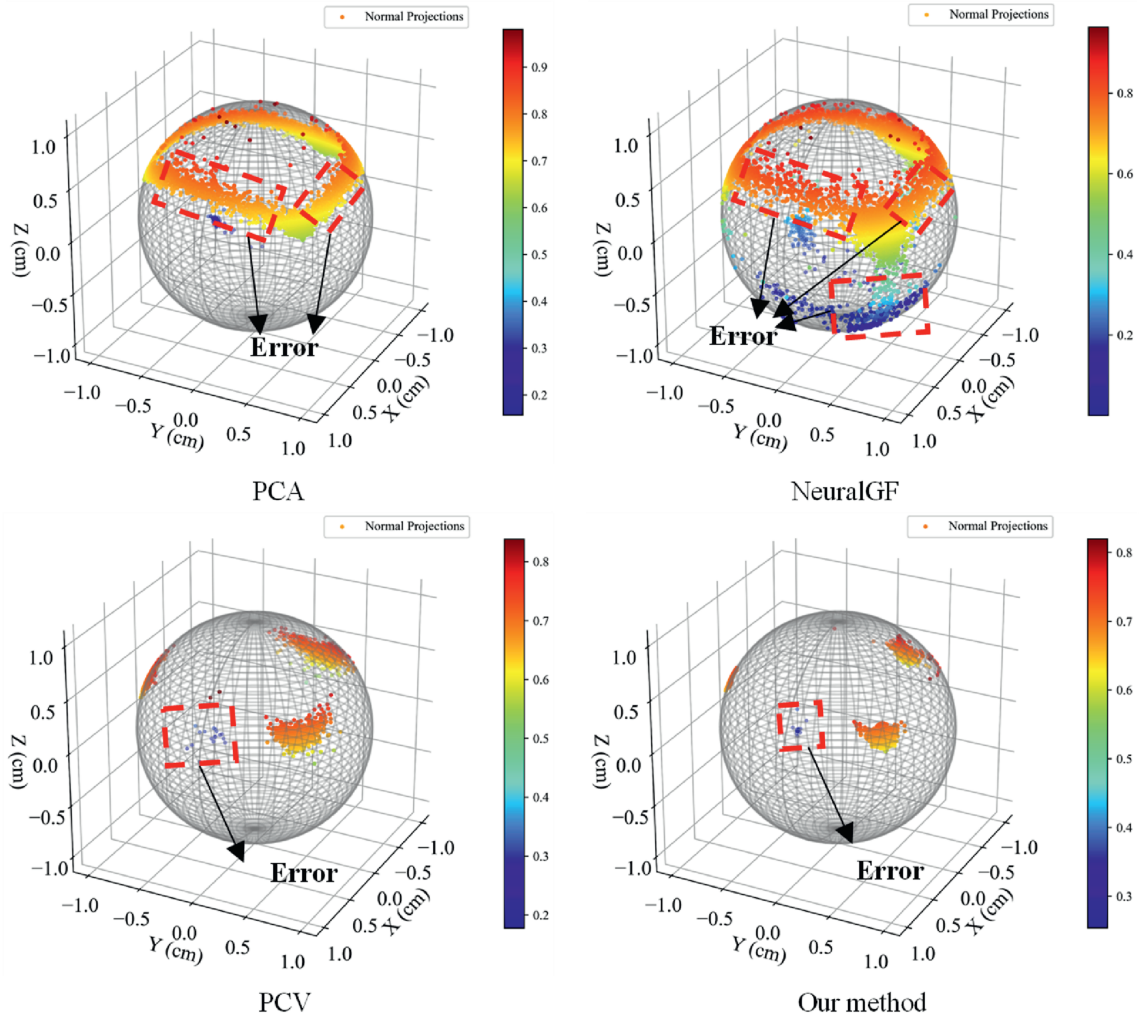

Fig. 26. Results of the octahedron.

3.4.2. Selection of the classification threshold Ω_f

Following the approach of Zhang et al. (2013), the classification threshold Ω_f is automatically determined to partition the point cloud into planar points and sharp feature points. A brief description of this method is provided in the Discussion section. As defined in Eq. (7), the curvature σ_i of point p_i is computed to quantify its proximity to sharp features. The curvature values of all points are aggregated into a histogram, which is treated as a one-dimensional (1D) signal. This signal is subsequently smoothed through a minimization procedure.

$$\hat{f}_\sigma = \underset{\hat{f}_\sigma}{\operatorname{argmin}} \|\hat{f}_\sigma - f_\sigma\|_F - \lambda \|\boldsymbol{D}\hat{f}_\sigma\|_1 \tag{21}$$

where λ is the balancing coefficient, which is fixed at 0.5 in our implementation; f_σ denotes the histogram of $\{\sigma_i\}_{i=1}^n$; $\boldsymbol{D}$ is the first-order difference matrix, and $\|\cdot\|_F$ and $\|\cdot\|_1$ represent the Frobenius norm and the L1-norm, respectively. The smoothed histogram after the first peak exhibits a gradual decline, which is used to determine the threshold Ω_f. Specifically, this point is identified when the difference between its preceding and succeeding bins is less than 0.05 max(f_σ).

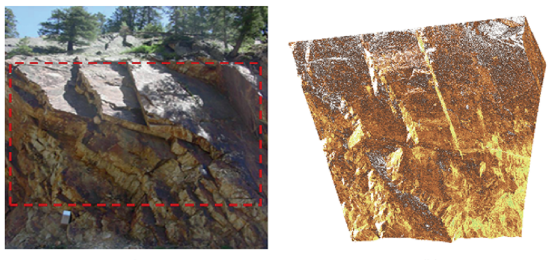

Fig. 27. Case A of the rock mass point cloud: (a) Data collection site map, with the red box indicating the study area; and (b) Raw point cloud data, consisting of 848,043 points.

3.4.3. Selection of the clustering threshold $\theta_{simalrity}$

In Section 2.6.3, we propose using a threshold $\theta_{simalrity}$ to cluster neighboring points. In practice, this threshold is empirically set to 10°, meaning that two points are assigned to the same cluster if the angle between their normal vectors does not exceed 10°.

4. Conclusions

This study addresses the challenge of effectively preserving sharp features in rock mass point cloud normal vector estimation by proposing a novel normal vector estimation method based on

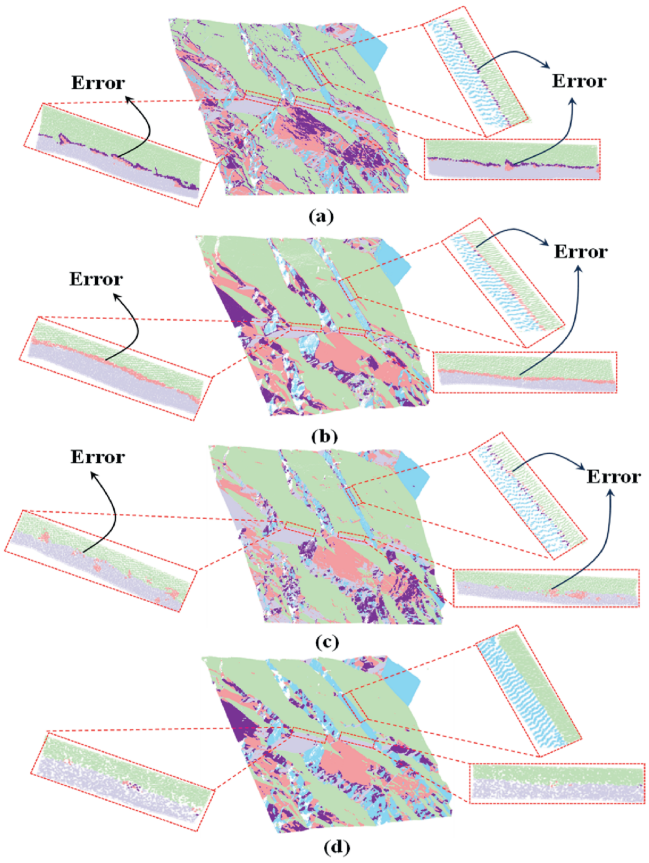

Fig. 28. Clustering results of Case A: The discontinuities after clustering the normal vectors calculated using (a) PCA, (b) NeuralGF, (c) PCV, and (d) the method proposed in this paper.

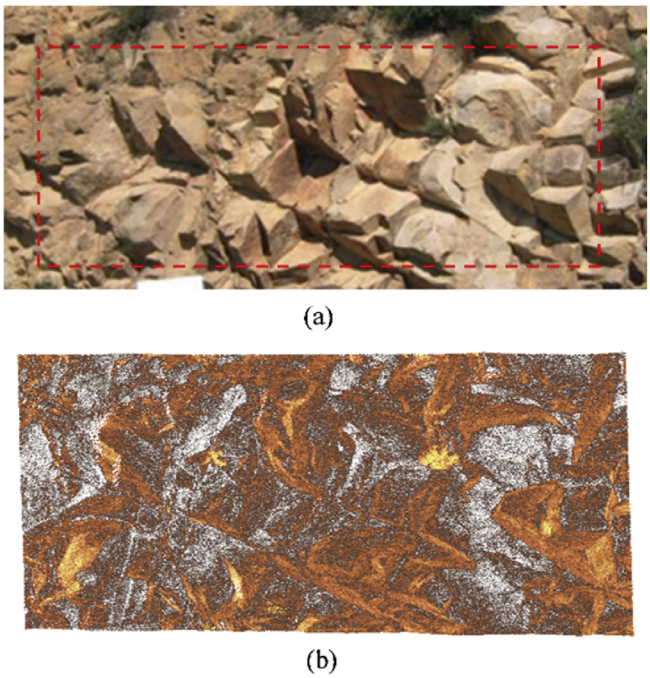

Fig. 29. Case B of the rock mass point cloud: (a) Data collection site map, with the red box indicating the study area; and (b) Raw point cloud data, consisting of 497,961 points.

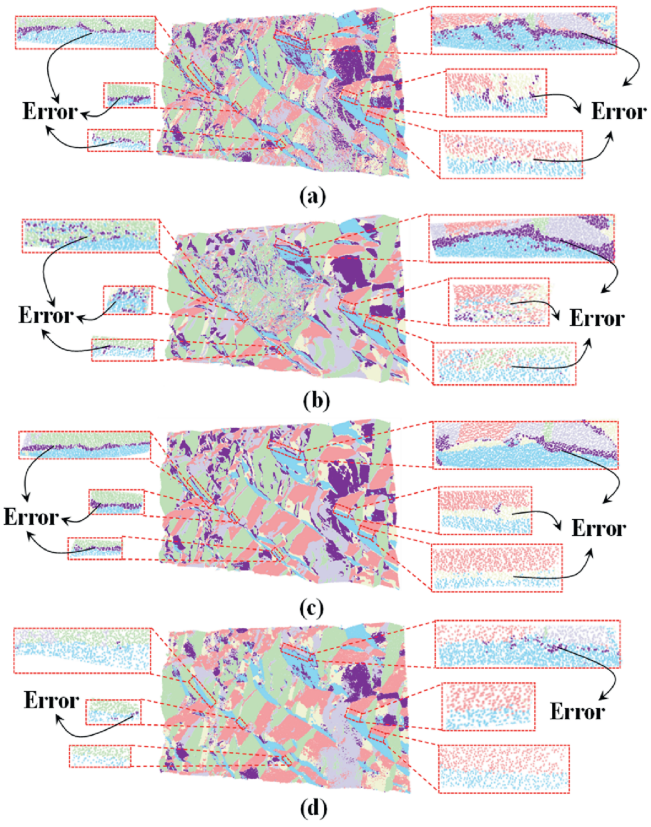

Fig. 30. Clustering results of Case B: The discontinuities after clustering the normal vectors calculated using (a) PCA, (b) NeuralGF, (c) PCV, and (d) the method proposed in this paper.

local geometric adjustment. The proposed method aims to achieve precise normal vector estimation of rock mass point clouds while effectively retaining sharp geometric features. The methodology consists of four main steps:

(1) Classification. The input point cloud is classified into plane points and sharp feature points using a threshold selection technique.
(2) Projection. Normal and axial projections are applied to the sharp feature points to extract a coplanar feature point set and its central axis.
(3) Segmentation. The coplanar feature point set and the central axis are used to fit two lines via the least squares method, thereby determining the segmentation line.
(4) Correction. The segmentation line is used to partition the neighboring point sets, thereby determining the optimal neighboring point set and completing the normal vector correction.

To comprehensively evaluate the performance of the proposed method, comparative experiments were conducted with eight relevant algorithms across three different datasets: CAD models, real-world regular body point clouds, and real rock mass point clouds, followed by an in-depth analysis.

The proposed method offers the following advantages. First, it inherits the robustness and computational efficiency of the PCA-based normal vector estimation method, demonstrating significant competitiveness in stability and computational performance. Additionally, through the integration of projection and segmentation techniques, it effectively enhances the preservation of sharp

geometric features in rock mass point clouds, thereby improving the accuracy of discontinuity set clustering.

However, the proposed method has several limitations. First, the additional processing for sharp feature regions reduces computational efficiency compared to PCA. Second, the accuracy of normal vector estimation relies on the correct classification of sharp feature points, and classification errors can propagate to affect the results. Moreover, specific challenges arise in practical scenarios: (1) in sparse point clouds, insufficient neighboring points may cause unstable normal estimation; (2) at point cloud boundaries, incomplete neighborhoods can introduce bias; and (3) in fractured zones, the coexistence of multiple structural surfaces may compromise clustering and segmentation. Addressing these issues will require further investigation, such as incorporating adaptive neighborhood selection, multi-scale strategies, or statistical constraints, which will be the focus of future work.

Future work will focus on further improving both the accuracy and computational efficiency of normal vector estimation for rock mass point clouds. First, computational efficiency can be enhanced through GPU parallelization and algorithmic acceleration, enabling real-time processing of large-scale point clouds. Second, to reduce the impact of classification errors in sharp feature detection, adaptive strategies that integrate multi-scale analysis, statistical constraints, or deep learning-based feature recognition will be explored to improve robustness and generalization. Finally, building upon accurate normal vector estimation, we aim to develop an automated framework for rock mass discontinuity identification and stability assessment using machine learning and topological analysis, thereby providing a more reliable foundation for engineering safety evaluation in complex rock mass environments.

CRediT authorship contribution statement

Mingming Ren: Writing – original draft, Resources, Methodology, Investigation, Formal analysis. **Manchao He:** Supervision, Project administration, Conceptualization. **Jie Hu:** Writing – review & editing, Supervision, Funding acquisition. **Hongru Li:** Writing – original draft, Resources, Methodology. **Yuxiang Ding:** Visualization, Resources, Data curation. **Xinhao Miao:** Visualization, Investigation. **Hongyi Zhang:** Visualization, Resources.

Declaration of competing interest

The authors declare that they have no known competing financial interests or personal relationships that could have appeared to influence the work reported in this paper.

The author Manchao He is the Vice Editors-in-Chief for this journal and was not involved in the editorial review or the decision to publish this article.

Acknowledgments

This work was supported by the National Natural Science Foundation of China (Grant No. 42507210) and the Fundamental Research Funds for the Central Universities (Grant No. 2025XJSB01).

Data availability

Python was used as the primary programming language in this study. The complete source code is available at: https://github.com/MingmingRen0616/NormalVecotrEstimation_dppca.

References

Amenta, N., Bernt, M., 1998. Surface reconstruction by Voronoi filtering. In: Proceedings of the 14th Annual Symposium on Computational Geometry. Association for Computing Machinery (ACM), New York, USA, pp. 39–48.
Barton, N., Lien, R., Lunde, J., 1974. Engineering classification of rock masses for the design of tunnel support. Rock Mech. 6 (4), 189–236.
Ben-Shabat, Y., Avraham, T., Lindenbaum, M., Fischer, A., 2018. Graph based oversegmentation methods for 3D point clouds. Comput. Vis. Image Underst. 174, 12–23.
Ben-Shabat, Y., Lindenbaum, M., Fischer, A., 2019. Nesti-Net: normal estimation for unstructured 3D point clouds using convolutional neural networks. In: 2019 IEEE/CVF Conference on Computer Vision and Pattern Recognition (CVPR). IEEE, New York, USA.
Boulch, A., Marlet, R., 2016. Deep learning for robust normal estimation in unstructured point clouds. Comput. Graph. Forum 35 (5), 281–290.
Cao, J., Chen, H., Zhang, J., Li, Y., Liu, X., Zou, C., 2018. Normal estimation via shifted neighborhood for point cloud. J. Comput. Appl. Math. 329, 57–67.
Cazals, F., Pouget, M., 2005. Estimating differential quantities using polynomial fitting of osculating jets. Comput. Aided Geomet. Des. 22 (2), 121–146.
Charles, R.Q., Su, H., Kaichun, M., Guibas, L.J., 2017. PointNet: deep learning on point sets for 3D classification and segmentation. In: 2017 IEEE Conference on Computer Vision and Pattern Recognition Workshops. (CVPRW 2017). IEEE, New York, USA, pp. 652–660.
Chen, J., Huang, H., Cohn, A.G., Zhang, D., Zhou, M., 2022. Machine learning-based classification of rock discontinuity trace: SMOTE oversampling integrated with GBT ensemble learning. Int. J. Rock Mech. Min. Sci. 32 (2), 309–322.
Chen, J., Huang, H., Zhou, M., Chaiyasarn, K., 2021. Towards semi-automatic discontinuity characterization in rock tunnel faces using 3D point clouds. Eng. Geol. 291, 106232.
Chen, J., Zhu, H., Li, X., 2016. Automatic extraction of discontinuity orientation from rock mass surface 3D point cloud. Comput. Geosci. 95, 18–31.
Ge, Y., Cao, B., Tang, H., 2022. Rock discontinuities identification from 3D point clouds using artificial neural network. Rock Mech. Rock Eng. 55 (3), 1705–1720.
Guerrero, P., Kleiman, Y., Ovsjanikov, M., Mitra, N.J., 2018. PCPNet learning local shape properties from raw point clouds. Comput. Graph. Forum 37 (2), 75–85.
Guo, J., Liu, S., Zhang, P., Wu, L., Zhou, W., Yu, Y., 2017. Towards semi-automatic rock mass discontinuity orientation and set analysis from 3D point clouds. Comput. Geosci. 103, 164–172.
Hoppe, H., DeRose, T., Duchamp, T., McDonald, J., Stuetzle, W., 1992. Surface reconstruction from unorganized points. In: Proceedings of the 19th Annual Conference on Computer Graphics and Interactive Techniques. ACM, New York, USA, pp. 71–78.
He, X., Xu, H., Sabetamal, H., Sheng, D., 2020. Machine learning aided stochastic reliability analysis of spatially variable slopes. Comput. Geotech. 126, 103711.
Huang, H., Li, D., Zhang, H., Ascher, U., Cohen-Or, D., 2009. Consolidation of unorganized point clouds for surface reconstruction. ACM Trans. Graph. 28 (5), 1–7.
Huang, H., Wu, S., Cohen-Or, D., Gong, M., Zhang, H., Li, G., Chen, B., 2013. L1-medial skeleton of point cloud. ACM Trans. Graph. 32 (4), 1–8.
Jaboyedoff, M., Couture, R., Locat, P., 2009. Structural analysis of Turtle Mountain (Alberta) using digital elevation model: toward a progressive failure. Geomorphology 103 (1), 5–16.
Klasing, K., Althoff, D., Wollherr, D., Buss, M., 2009. Comparison of surface normal estimation methods for range sensing applications. In: 2009 IEEE International Conference on Robotics and Automation. IEEE, New York, USA, pp. 3206–3211.
Kong, D., Wu, F., Saroglou, C., 2020. Automatic identification and characterization of discontinuities in rock masses from 3D point clouds. Eng. Geol. 265, 105442.
Lato, M., Kemeny, J., Harrap, R.M., Bevan, G., 2013. Rock bench: establishing a common repository and standards for assessing rockmass characteristics using LiDAR and photogrammetry. Comput. Geosci. 50, 106–114.
Leng, B., Yang, H., Hou, G., Lyamin, A., 2021. Rock mass trace line identification incorporated with grouping algorithm at tunnel faces. Tunn. Undergr. Space Technol. 110, 103810.
Li, Q., Feng, H., Shi, K., Gao, Y., Fang, Y., Liu, Y.S., Han, Z., 2023. NeuralGF: unsupervised point normal estimation by learning neural gradient function. In: Proceedings of the 37th International Conference on Neural Information Processing Systems (NIPS'23). Curran Associates Inc., Red Hook, USA, pp. 66006–66019.
Mitra, N.J., Nguyen, A., Guibas, L., 2003. Estimating surface normals in noisy point cloud data. In: Proceedings of the Nineteenth Annual Symposium on Computational Geometry (SCG'03). Association for Computing Machinery, New York, USA, pp. 322–328.
Pauly, M., Gross, M., Kobbelt, L.P., 2002. Efficient simplification of point-sampled surfaces. In: IEEE Visualization 2002. IEEE, New York, USA, pp. 163–170.
Pauly, M., Keiser, R., Gross, M., 2003. Multi-scale feature extraction on point–sampled surfaces. Comput. Graph. Forum 22 (3), 281–289.
Peng, X., Lin, P., Xia, Q., Yu, L., Wang, M., 2024. A new method for recognizing discontinuities from 3D point clouds in tunnel construction environments. Tunn. Undergr. Space Technol. 152, 105955.
Riquelme, A.J., Abellán, A., Tomás, R., Jaboyedoff, M., 2014. A new approach for semi-automatic rock mass joints recognition from 3D point clouds. Comput. Geosci. 68, 38–52.
Singh, S.K., Banerjee, B.P., Lato, M.J., Sammut, C., Raval, S., 2022. Automated rock

mass discontinuity set characterisation using amplitude and phase decomposition of point cloud data. Int. J. Rock Mech. Min. Sci. 152, 105072.
Singh, S.K., Raval, S., Banerjee, B.P., 2021. Automated structural discontinuity mapping in a rock face occluded by vegetation using mobile laser scanning. Eng. Geol. 285, 106040.
Slob, S., van Knapen, B., Hack, R., Turner, K., Kemeny, J., 2005. Method for automated discontinuity analysis of rock slopes with three-dimensional laser scanning. Transp. Res. Rec. 187–194, 1913.
Sun, W., Wang, J., Yang, Y., Jin, F., 2021. Rock mass discontinuity extraction method based on multiresolution supervoxel segmentation of point cloud. IEEE J. Sel. Top. Appl. Earth Obs. Remote Sens. 14, 8436–8446.
Umili, G., Ferrero, A., Einstein, H.H., 2013. A new method for automatic discontinuity traces sampling on rock mass 3D model. Comput. Geosci. 51, 182–192.
Wang, X., Zou, L., Shen, X., Ren, Y., Qin, Y., 2017. A region-growing approach for automatic outcrop fracture extraction from a three-dimensional point cloud. Comput. Geosci. 99, 100–106.
Wang, Y., Feng, H.Y., Delorme, F.É., Engin, S., 2013. An adaptive normal estimation method for scanned point clouds with sharp features. Comput. Aided Des. 45 (11), 1333–1348.
Zhang, J., Cao, J., Liu, X., Wang, J., Liu, J., Shi, X., 2013. Point cloud normal estimation via low-rank subspace clustering. Comput. Graph. 37, 697–706.
Zhang, J., Cao, J., Liu, X., Chen, H., Li, B., Liu, L., 2019. Multi-normal estimation via pair consistency voting. IEEE Trans. Vis. Comput. Graph. 25 (4), 1693–1706.
Zhang, K., Wu, W., Liu, Y., Huang, Y., Zhang, M., Zhu, H., 2024. OCM: an intelligent recognition method of rock discontinuity based on optimal color mapping of 3D point cloud via deep learning. Rock Mech. Rock Eng. 57 (7), 4873–4905.
Zhang, K., Wu, W., Zhu, H., Zhang, L., Li, X., Zhang, H., 2020. A modified method of discontinuity trace mapping using three-dimensional point clouds of rock mass surfaces. J. Rock Mech. Geotech. Eng. 12 (3), 571–586.
Zhang, P., Du, K., Tannant, D.D., Zhu, H., Zheng, W., 2018a. Automated method for extracting and analysing the rock discontinuities from point clouds based on digital surface model of rock mass. Eng. Geol. 239, 109–118.
Zhang, P., Li, J., Yang, X., Zhu, H., 2018b. Semi-automatic extraction of rock discontinuities from point clouds using the ISODATA clustering algorithm and deviation from mean elevation. Int. J. Rock Mech. Min. Sci. 110, 76–87.

Mingming Ren received his BSc degree in Software Engineering in 2018 and his MSc degree in Computer Technology in 2023 from North University of China. He is currently a PhD student in Civil Engineering at China University of Mining and Technology (Beijing). His research interests include rock mass stability analysis based on computer vision, geological hazard identification using artificial intelligence algorithms, and decision support systems based on multi-source data.