



Feature-based point cloud simplification method: an effective solution for balancing accuracy and efficiency

Jiangsheng Wu¹ · Xiaoming Lai² · Xingliang Chai¹ · Kai Yang² · Tianming Wang² ·
Haibo Liu¹ · Yongqing Wang¹

Accepted: 21 February 2024 / Published online: 16 March 2024

© The Author(s), under exclusive licence to Springer Science+Business Media, LLC, part of Springer Nature
2024

Abstract

Traditional point cloud simplification methods are slow to process large point clouds and prone to losing small features, which leads to a large loss of point cloud accuracy. In this paper, a new point cloud simplification method using a three-step strategy is proposed, which realizes efficient reduction of large point clouds while preserving fine features through point cloud down-sampling, normal vector calibration, and feature extraction based on the proposed feature descriptors and neighborhood subdivision strategy. In this paper, we validate the method using measured point clouds of large co-bottomed component surfaces, visualize the errors, and compare it with other methods. The results demonstrate that this method is well-suited for efficiently reducing large point clouds, even those on the order of ten million points, while maintaining high accuracy in feature retention, refinement precision, efficiency, and robustness to noise.

Keywords Point cloud simplification · Feature descriptor · Neighborhood subdivision strategy · Normal vector calibration

1 Introduction

Large and complex curved components find extensive use in fields such as aviation, aerospace, and shipbuilding. The digitization and processing of these components often rely on high-precision sensors to obtain dense point cloud data. Three-dimensional laser scanning technology has rapidly developed and has been applied in large-scale complex surface measurements due to its non-contact, high precision,

✉ Xingliang Chai
chaixingliang@mail.dlut.edu.cn

¹ State Key Laboratory of High-Performance Precision Manufacturing, Dalian University of Technology, Dalian 116024, China

² Beijing Satellite Manufacturing Factory, Beijing 100086, China

digitization, and automation capabilities [1]. This technology overcomes the limitations of traditional measurement methods in terms of accuracy and efficiency. However, due to the high-frequency data acquisition characteristics of scanners, the captured 3D point cloud data often suffers from issues such as large data size, scattered distribution, and redundant points, posing significant challenges for subsequent tasks such as defect detection, reverse modeling, and tool path generation. Point cloud simplification is an effective approach to address these issues, although it can potentially remove important feature points, resulting in incomplete representation and subpar results.

Existing point cloud simplification methods can be classified into two main categories: point-based and mesh-based [2–4]. While mesh-based simplification has been studied earlier [5] and is highly efficient for point clouds with small volumes, it is not suitable for large-scale point clouds due to the lengthy grid reconstruction process and the substantial computational power required. Therefore, this paper primarily focuses on point-based simplification. Point-based simplification methods are mainly divided into the following categories: sampling-based point cloud simplification methods, feature-based point cloud simplification methods, and deep learning-based point cloud simplification methods [6].

The sampling-based point cloud simplification method involves resampling or down-sampling the original point cloud to reduce redundant points. While this method offers high efficiency and simplicity, it may lead to feature loss and increased errors. Common sampling-based point cloud simplification methods include random sampling, uniform voxel sampling, non-uniform voxel sampling, and curvature sampling [7].

To address the above issues, scholars have proposed feature-based point cloud simplification methods. For example, Han [8]. use the ratio between the difference in neighboring points on both sides of a projection plane and the total number of neighboring points to detect edge points. For non-edge points, they calculate importance values based on the normal vector, delete the least important points, iteratively update the normal vectors and importance values of affected points until the simplification rate is achieved. Wu [9]. calculate the angle between the average normal vector of points within a bounding box and the normal vectors of individual points. They use this angle to determine whether the bounding box needs further subdivision using octree partitioning. Then, they use a quadratic surface fitting method to approximate the point cloud and calculate the principal curvatures of each point. Feature points are extracted and retained based on the Hausdorff distance of the principal curvatures. Feature point extraction based on normal vectors is sensitive to noise and may overlook regions with relatively small curvature. Therefore, Chen [10]. use four parameters (curvature at any point, average angle between normals of a point and its neighbors, average distance between a point and its neighbors, and distance between a point and the centroid of its neighbors) to extract feature points and employ an octree voxel grid to simplify non-feature points. However, Chen et al.'s method only uses Principal Component Analysis (PCA) to calculate normal vectors, leading to inaccurate results, and feature recognition is still susceptible to noise. In addition, Zhang [11]. define scale-preserving simplification entropy, contour-preserving entropy, and curve-preserving entropy to extract geometric features.

Feature points are obtained based on simplification entropy, and neighborhoods are established between feature points. Special processing is applied to these neighborhoods to effectively preserve surface features. However, this method requires a significant amount of time for surface feature detection and shape matching, resulting in low algorithm efficiency. Lv [12], propose an Approximate Intrinsic Voxel Structure (AIVS) point cloud simplification framework, consisting of two core components: global voxel structure and Local Farthest Point Sampling (Local FPS). This framework can meet diverse simplification requirements, such as curvature-sensitive sampling and sharp feature preservation. However, this method has low efficiency in simplifying large point cloud data and does not consider the influence of noise. These methods calculate the sharpness of various regions by computing point cloud features such as density, curvature, normals, etc. [13]. Different simplification methods are applied to different regions, making these algorithms complex. Although they can retain some feature point information, they need to balance issues such as overall integrity and efficiency.

Point cloud simplification methods based on deep learning, such as the approach proposed by Potamias. [14], utilize trained graph neural network architectures to learn and identify feature points in point clouds, achieving point cloud simplification. Qi Charles [15, 16], on the other hand, efficiently and robustly learn features from point sets using the PointNet and PointNet++ networks, enabling effective point cloud simplification. However, deep learning methods require a large amount of training data to achieve high accuracy, making them less suitable for large point cloud data.

In this paper, a point cloud with a surface area of the part to be measured larger than 5m^2 and the number of measured point clouds larger than 5 million is defined as a large point cloud. Existing point cloud simplification algorithms rely on traditional parameters and are susceptible to noise, leading to inaccurate parameters and inefficient simplification of large point clouds while retaining features. Therefore, this paper proposes a new point cloud simplification method. The second section introduces the proposed point cloud simplification method, the third section applies the method to actual point clouds and compares it with other simplification methods, and the fourth section presents the conclusion of this paper.

2 Method

This paper addresses large-scale point clouds and proposes a novel point cloud simplification method employing a three-step refinement strategy. In the first step, considering feature dimensions, rapid initial coarse simplification of large point cloud data is achieved through octree down-sampling, preserving accuracy while enhancing the speed of the subsequent fine simplification. This step yields two sets of point cloud data: the first set consists of appropriately spaced retained points for fine simplification, and the second set comprises higher refinement level points. In the second step, based on the second set of data, rapid calibration of normal vectors for the first set is accomplished with isosurface constraints, significantly improving efficiency. In the third step, feature descriptors are computed, distinguishing feature

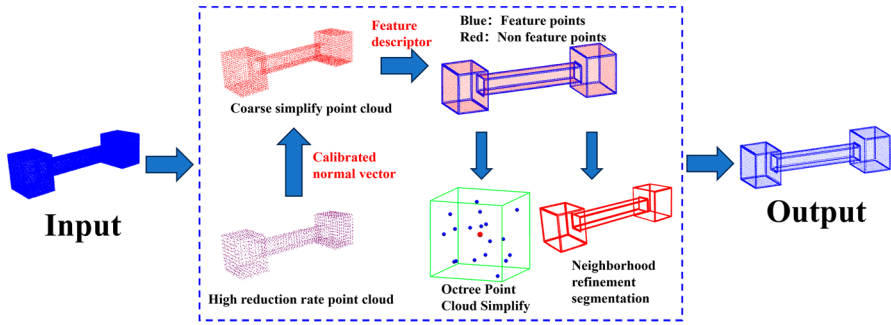


Fig.1 Workflow of the Point Cloud Simplification Method Proposed in this Paper

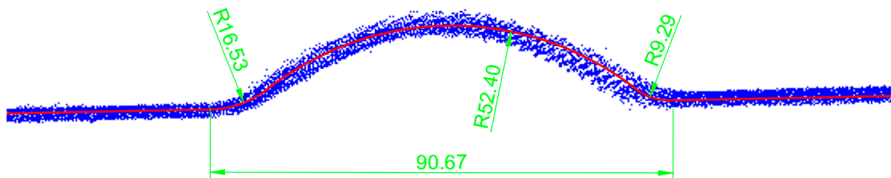


Fig.2 A Characteristic Dimension on the Common-Bottom Component Surface

points from non-feature points based on descriptor values. Through neighborhood subdivision strategy, potential feature points within feature points are further identified. For feature points, based on the simplification rate, refinement is performed according to the descriptor values, with the number of feature points typically not exceeding half of the total points after simplification. For non-feature points, fast simplification is achieved through octree down-sampling. Finally, the refined results of feature and non-feature points are merged to obtain the ultimate fine point cloud simplification result. The algorithm flow is illustrated in Fig. 1.

2.1 Point cloud down-sampling

Due to the high resolution and measurement rate of the utilized scanner, the point cloud dataset reaches a substantial 15 million points, with an average spacing of approximately 0.55 mm between points. Figure 2 illustrates a schematic representation of a feature size on the Common-Bottom Component surface. The average distance between point clouds is too dense relative to the features in the point cloud and the size of the Common-Bottom Component surface. To address this issue, this paper initiates point cloud down-sampling as the first step, reducing the point cloud density while retaining essential feature information, thereby enhancing the computational efficiency of feature recognition in the subsequent step.

Voxel grid down-sampling using an octree structure is a common method for point cloud simplification [17], and its octree structure is depicted in Fig. 3.

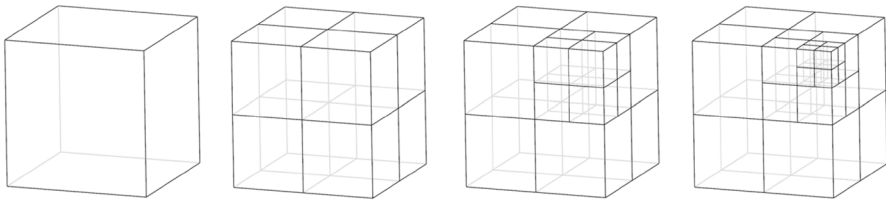


Fig.3 Octree Node Diagram

Although this method does not inherently identify and preserve features, it boasts high efficiency in reducing redundant measurements, making it suitable for eliminating a substantial amount of redundant points during the coarse simplification phase.

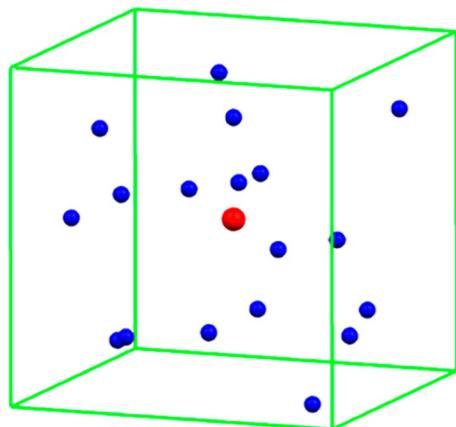
This paper employs an improved octree algorithm, specifically using the point closest to the voxel centroid instead of all points within the voxel [18]. This ensures that the sampled points are all part of the original point cloud, as schematically illustrated in Fig. 4.

The PCL (Point Cloud Library) is a robust library for point cloud processing algorithms [19]. Its octree library allows for the creation of octree data structures, loading point cloud data into octrees, and obtaining the coordinates of the centroid of the voxels occupied in the octree. The KdTreeFLANN (Kd-tree based on the FLANN library) is used for nearest neighbor searches, facilitating the identification of points closest to the voxel centroid. The result of octree voxel grid partitioning is illustrated in Fig. 5.

2.2 Normal vector computation and calibration

In this paper, the Principal Component Analysis (PCA) method is employed for normal vector calculation [20], and the results of solving the normal vector are shown in Fig. 6a, and the results of taking the inverse direction for the solved normal vector

Fig.4 Example of Voxel Grid Simplification Principle



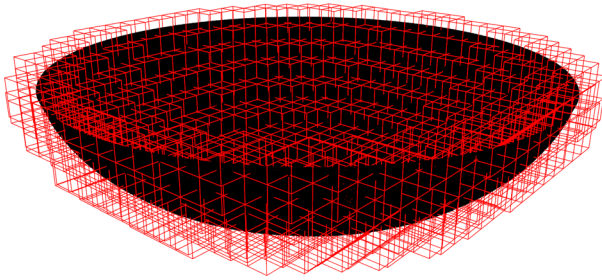


Fig. 5 Schematic Diagram of Octree Voxel Grid

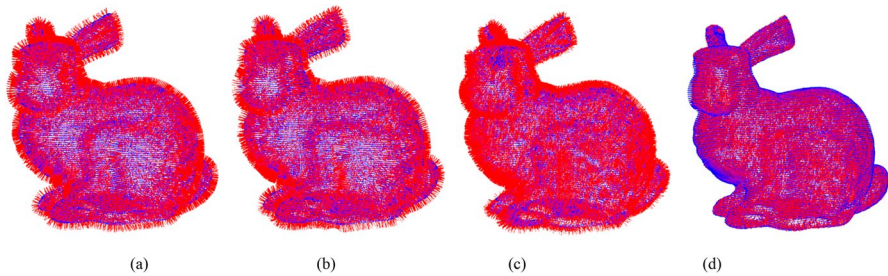


Fig.6 Normal Vectors Obtained by PCA and Calibration Results, **a** PCA Normal Vectors, **b** PCA Normal Vectors Reversed, **c** Proposed Method, **d** Proposed Method Reversed

are shown in Fig. 6b. Comparing the two pictures we can see that the normal vector has duality, the normal vector obtained by the PCA method does not have global consistency, and the result of solving the normal vector of the sharp region is not accurate enough, so in this paper we refer to the optimization formula for the calibration of the normal vector.

This paper references the optimization formula proposed by Dong [21], achieving global consistency of normal vectors through constraints such as discrete Poisson equation, isosurface constraints, and local consistency constraints. The overall objective function is formulated as follows:

$$E(x, n) = E_{iso}(x) + \alpha E_{Poi}(x, n) + \beta E_{loc}(n) \tag{1}$$

where: α and β are parameters adjusted for different shapes and noise intensities; $E_{iso}(x) = ||Ux - \frac{1}{2}1||^2$ is the isosurface constraint; $E_{Poi}(x, n) = ||Ax - Bn||^2$ is the Poisson equation constraint; $E_{loc}(n) = E_D(n) + E_{COD}(n) + n^T n$ is the local consistency constraint.

The indicator function is 0 outside the surface, 0.5 on the surface, and 1 inside the surface. Therefore, the overall objective function must be minimized under Dirichlet boundary conditions:

$$\begin{cases} F = \min_{x,n} E(x, n) \\ s.t. x_i = 0, i \in \zeta \end{cases} \tag{2}$$

According to the Poisson equation, the variation of the indicator function is primarily caused by the vector field formed by point normals. As shown in Fig. 7, if the indicator function is 0 on the bounding box and 0.5 on the surface (indicated by the blue contour), we can predict the normal direction perpendicular to the circle and pointing inward based on the change in indicator values. The indicator function transitions from 0 to 0.5, similar to "encircling" the surface from the bounding box.

For large point cloud data and to improve solution speed, this optimization problem is transformed into an unconstrained least squares problem: $\nabla_x E(x, n) = 0$, $\nabla_n E(x, n) = 0$, which is equivalent to solving a linear system:

$$\tilde{A}\tilde{x} = \tilde{b} \tag{3}$$

where: $\tilde{A} = \begin{pmatrix} U^T U + \alpha A^T A & -\alpha A^T B \\ -\alpha B^T A & \alpha B^T B + \beta M \end{pmatrix}$, $\tilde{x} = \begin{pmatrix} x \\ n \end{pmatrix}$, $\tilde{b} = \begin{pmatrix} U^T \frac{1}{2} \vec{1} \\ 0 \end{pmatrix}$, Matrices U, A, B, and M are all sparse matrices.

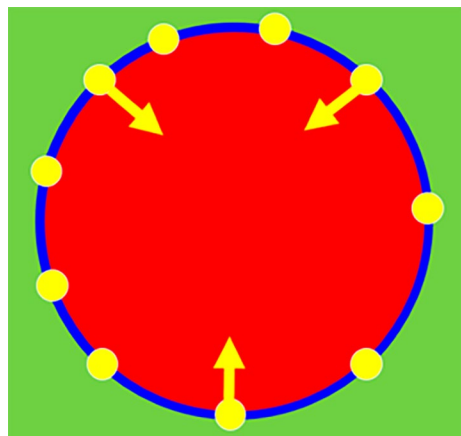
Finally, the Conjugate Gradient (CG) method is employed to solve this linear system [22]. By transforming the problem into a linear system and solving it, the algorithm's runtime is significantly improved. For point clouds on the order of tens of millions, it takes only around 2 min on an ordinary laptop (i7-12700H 1.5 GHz 16 GB). The results of normal vector calibration are shown in Fig. 6c, and the results after taking the opposite direction for the calibrated normal vectors are shown in Fig. 6d. After calibration, the normal vectors exhibit global consistency.

2.3 Feature descriptor computation

To accurately identify feature points, this paper introduces a new feature measurement parameter: the feature descriptor, to quantify surface variations. It is defined as:

$$\delta_i = k_i D_i \tag{4}$$

Fig.7 Dirichlet Boundary Conditions



where $D_i = |(c_i - p_i)^T n_i|$ is the point-to-centroid projection distance, representing the projected distance along the normal vector of the current point between the line connecting the current point and the centroid of its neighborhood P. Here, c_i is the centroid of neighborhood P, p_i is the current point, and n_i is the estimated normal vector for each point as estimated in Sect. 2.2. $k_i = \kappa_i^n \kappa_i^p \kappa_i^o$ is the weight.

2.3.1 Point-to-centroid projection distance

The purpose of D_i is to measure the local surface normal vector variation within the current neighborhood P. Its magnitude increases as the local shape of the current neighborhood becomes more curved, and vice versa. As shown in Fig. 8a–c, by calculating the local point-to-centroid projection distance D_i , feature points can be effectively identified. However, when the sampled surface is affected by noise, using only D_i to detect features may lead to errors. In the presence of noise, the projection distance between noise points and the centroid increases along the normal vector, falsely identifying the point as a feature point, as shown in Fig. 8d. The centroid projection distance exceeds the set threshold, leading to misclassification as a feature point, while in reality, it is a noise point. Additionally, relying solely on the calculation of D_i cannot identify boundary points.

2.3.2 Weighting calculation

To minimize the impact of noise on feature point detection and to effectively identify boundary features, this paper proposes a new weight composed of three parts: κ_i^n , κ_i^p and κ_i^o . Here, κ_i^n is a monotonically increasing exponential function of the overall normal direction difference within the neighborhood, making the boundary between feature and non-feature points more distinct. κ_i^p is a monotonically increasing exponential function of the overall distance difference within the neighborhood, shielding the impact of noise on feature point detection. κ_i^o is a monotonically increasing exponential function of the distance difference between the current point and the centroid, aiding in boundary point identification. The specific formulas are as follows:

$$\kappa_i^n = e^{\mu_n \rho_i}, \quad \kappa_i^p = e^{\mu_p \pi_i}, \quad \kappa_i^o = e^{\mu_o \lambda_i} \tag{5}$$

Where: $\rho_i = \sqrt{\frac{1}{|\Omega^0(p_i)|} \sum_{j \in \Omega^0(p_i)} (\theta_{ij} - \bar{\theta}_i)^2}$ is the standard deviation of normal angles within the neighborhood; $\pi_i = \sqrt{\frac{1}{|\Omega^0(p_i)|} \sum_{j \in \Omega^0(p_i)} (d_{ij} - \bar{d}_i)^2}$ is the standard deviation

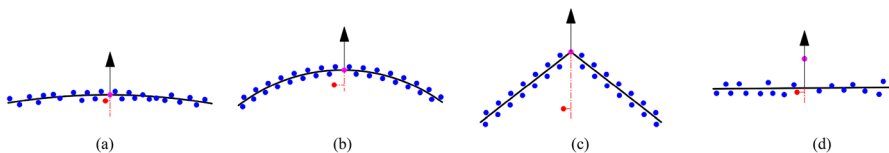


Fig. 8 Point-to-Centroid Projection Distance, **a** Smooth Region of Small Curvature without Noise, **b** Smooth Region of Large Curvature without Noise, **c** Characteristic Area without Noise, **d** With Noise

of distances within the neighborhood; $\lambda_i = d_{i0}$ is the difference in distance between the current point and the centroid within the neighborhood; $\theta_{ij} = \arccos(\frac{n_i}{\|n_i\|} \cdot \frac{n_j}{\|n_j\|})$ represents the angle between the normal vector at the current point and the normal vectors of other points within the neighborhood; $\bar{\theta}_i = \frac{1}{|\Omega^0(p_i)|} \sum_{j \in \Omega^0(p_i)} \theta_{ij}$ is the average angle between the normal vectors within the neighborhood.; $\bar{d}_i = \frac{1}{|\Omega^0(p_i)|} \sum_{j \in \Omega^0(p_i)} d_{ij}$ is the average distance from all points within the neighborhood to the least squares fitting plane; d_{ij} is the distance from any point within the neighborhood to the least squares fitting plane of that neighborhood, where the distance is positive on the side of the least squares fitting plane's normal vector and negative on the opposite side; $d_{i0} = \sqrt{(p_{ix} - \bar{p}_{ox})^2 + (p_{iy} - \bar{p}_{oy})^2 + (p_{iz} - \bar{p}_{oz})^2}$ is the distance from the point to the centroid;

$\bar{p}_o = \frac{1}{k} \sum_{i=1}^k p_i$ is the centroid of the current point's neighborhood; μ_n, μ_p, μ_o are proportionality factors.

By multiplying the local point-to-centroid projection distance D_i with parameters $\kappa_i^p, \kappa_i^n, \kappa_i^o$, the surface change rate descriptor is obtained to evaluate feature points. This significantly reduces the impact of noise, resulting in more stable and reliable computation. The following section will specifically demonstrate the effects of κ_i^p, κ_i^n and κ_i^o .

κ_i^n parameter: As shown in Fig. 9a, the surface change rate is small in non-feature regions, with small angles between adjacent normal vectors, resulting in a small value for the computed parameter. Conversely, as shown in Fig. 9b, the feature region exhibits the opposite behavior. Through the parameter adjustment, feature points have a distinct value, enhancing the differentiation between feature and non-feature points.

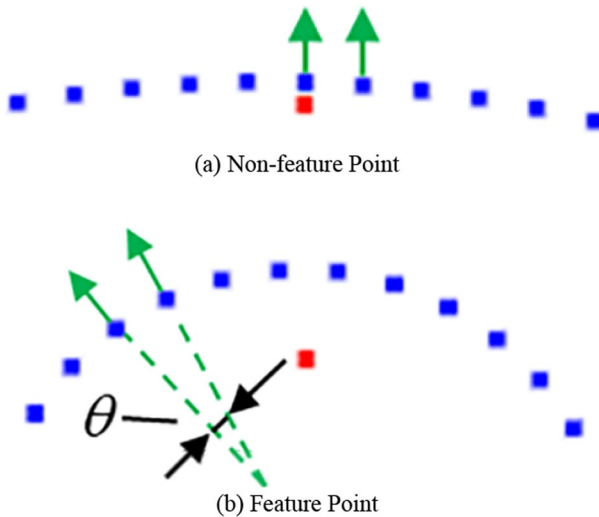


Fig.9 κ_i^n Principle

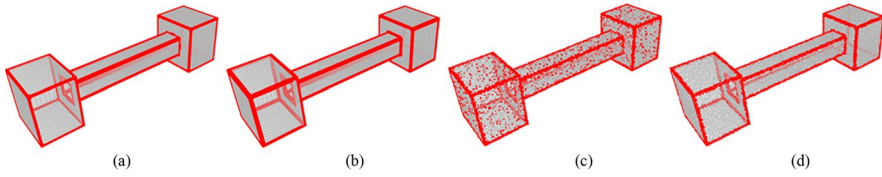


Fig.10 Impact of Different Parameters, **a** No Noise and $\delta_i = \kappa_i^n \kappa_i^o D_i$, **b** No Noise and $\delta_i = \kappa_i^n \kappa_i^o \kappa_i^p D_i$, **c** Adding Gaussian Noise and $\delta_i = \kappa_i^n \kappa_i^o D_i$, **d** Adding Gaussian Noise and $\delta_i = \kappa_i^n \kappa_i^o \kappa_i^p D_i$

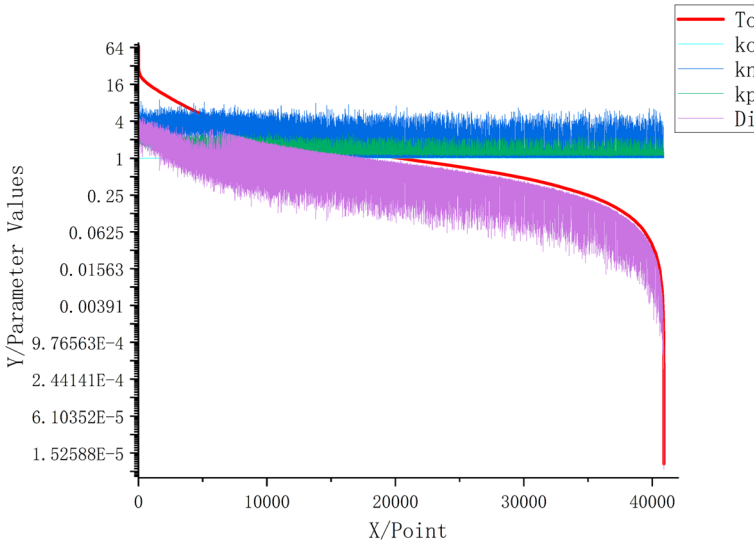


Fig. 11 Variation of Feature Descriptor Values in the Case (d) of Fig. 10

κ_i^p Parameter: By calculating the standard deviation of distances within the neighborhood, the κ_i^p parameter is designed to ensure that the computed κ_i^p value for noise points is smaller than that for feature points, thereby suppressing noise. Figure 10 illustrates the noise suppression effect of the κ_i^p parameter, where data in (a) and (b) are noise-free, and data in (c) and (d) contain Gaussian noise. (a) and (c) show feature descriptors without the term, while (b) and (d) include the term. From (a) and (b), it can be observed that the presence or absence of the K term does not significantly affect the feature recognition result. From (a), (b), (c), and (d), it can be seen that, for data with added Gaussian noise, the absence of the term makes it easy to misidentify noise data as feature points, affecting recognition accuracy. Figure 11, under the condition in (d) of Fig. 10, illustrates the variations of various parameter values based on feature descriptor values from high to low, Where "To" represents the feature descriptor sub-value. It can be observed that is sensitive to noise, and the term plays a crucial role in noise suppression.

κ_i^o parameter: The principle of boundary feature point recognition is illustrated in Fig. 12. The red point represents the current point p_i , the purple circle

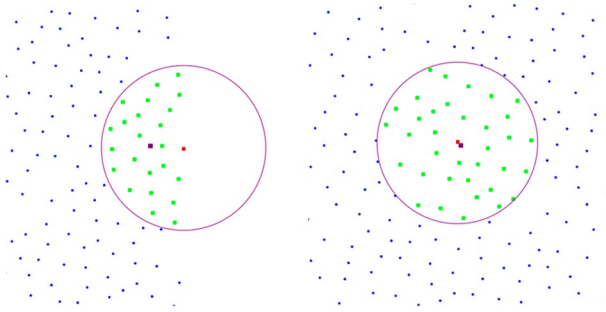


Fig.12 Schematic Diagram of Boundary Recognition Principle

represents a circle with a radius of 5 mm centered at the current point, the green points represent the points in the current point neighborhood, and the purple point represents the centroid of the current point neighborhood \bar{p}_{io} . It is evident from the two images that the distance from boundary points to the neighborhood centroid is much larger than the distance from internal points to the neighborhood centroid. The computed values κ_i^o for a measured large skin point cloud are shown in Figs. 13, and 14 displays the recognized boundary points when an opening is introduced at an arbitrary position in the middle of the point cloud. From the figures, it is apparent that both internal and external boundary points can be accurately identified.

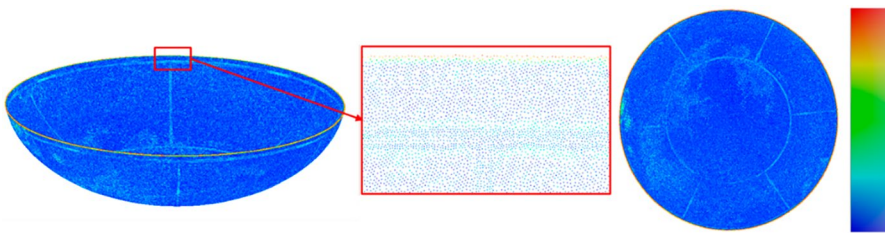


Fig. 13 Cloud Map of κ_i^o Values

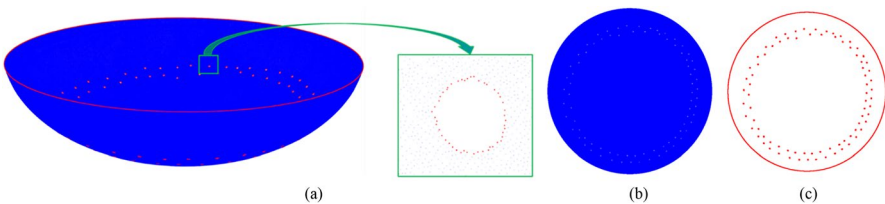


Fig. 14 Boundary Feature Recognition, **a** Main View and Local Magnification View, **b** Top View of the Original Point Cloud, **c** Top View of the Boundary Feature Points

2.4 Neighborhood subdivision strategy

In Sect. 2.3, for the calculation of feature descriptor values, to enhance computational efficiency for large point clouds, a neighborhood radius of 20 mm was chosen. However, a larger neighborhood radius may lead to the loss of subtle surface features, referred to in this paper as potential feature points. As shown in Fig. 15, the maximum diameter at the protrusion is 2.2 mm, and the feature descriptor values calculated for this neighborhood are relatively small, resulting in the loss of features in that region. To address this issue, this paper proposes a neighborhood subdivision strategy. After removing the feature points identified in Sect. 2.3 from the point cloud data, a certain percentage of points with the smallest values of the surface variation rate descriptor are removed, and the remaining points are considered potential feature points. These points undergo neighborhood subdivision, and the surface variation rate descriptor values are recalculated. If the descriptor value for a subdivided neighborhood exceeds a set threshold, the point and the neighborhood it subdivides are selected as a feature region.

The specific calculation process of the neighborhood subdivision strategy is shown in Fig. 16, and the procedure is as follows:

1. Extract feature points with relatively large feature descriptor values from the point cloud data after removing the identified feature points in Sect. 2.3, selecting the top 40% of points.
2. Place the potential feature points into a collection and traverse this collection. Starting from the first point p_i , take its neighborhood $P = \{p_{ij}\}$, and iteratively go through each point in the neighborhood. Use the vector between point p_i and neighborhood point p_{ij} , along with the normal vector, to form a plane and partition the neighborhood. Calculate the feature descriptor values for the $2n$ neighborhoods obtained after partitioning. Simultaneously, use the length of the line segment between point p_i and neighborhood point p_{ij} as the diameter of sphere m , and calculate the feature descriptor values within the sphere m neighborhood. Sort the feature descriptor values of the obtained $3n + 1$ neighborhoods, and if they are greater than the specified feature point threshold, consider the point as a feature point, and the subdivided neighborhood as a feature region.
3. The boundary conditions include constraints on the maximum number of feature points, the number of neighborhood subdivisions, and the minimum number of neighborhood points. If the boundary conditions are not met, further subdivision

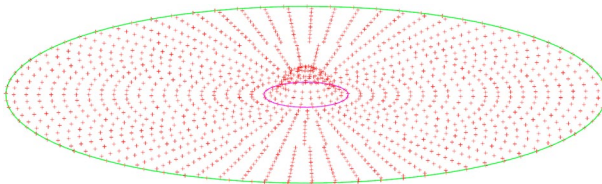


Fig. 15 Neighborhood Parameter Value Calculation

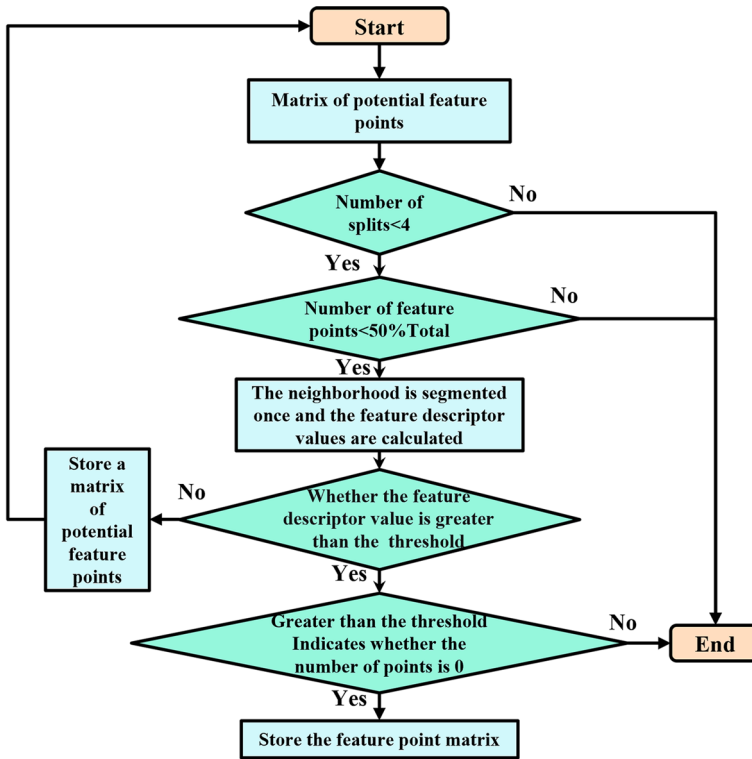
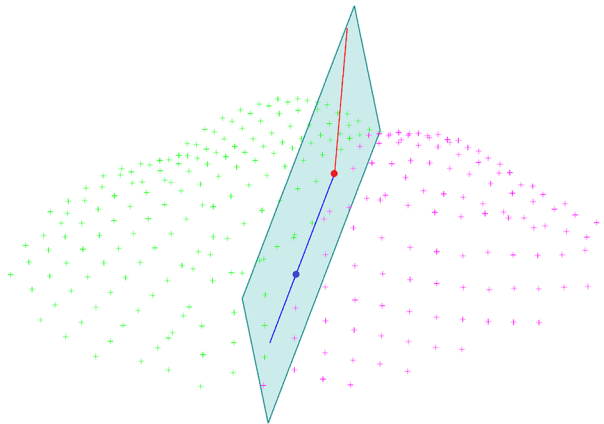


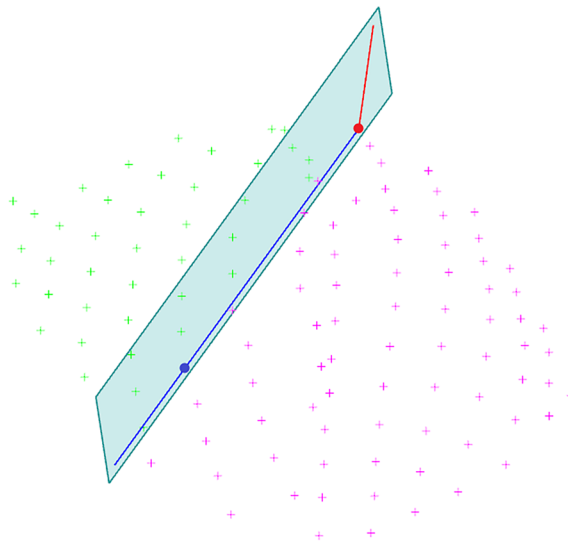
Fig. 16 Neighborhood Subdivision Strategy Flowchart

is applied to the neighborhoods obtained in the first step. If any of the restriction conditions are met, the feature point stops neighborhood subdivision.

The principle of neighborhood subdivision is illustrated in Fig. 17, where (a) represents the first subdivision, with the red point being the current point, blue points being any points in the neighborhood, red lines representing the current point's normal vector, blue lines representing vectors between the current point and neighborhood points, and a cyan plane formed by the normal vector and line vectors that divides the neighborhood into purple and green parts. The feature descriptor values of both parts are calculated for the current point to further determine the feature points' neighborhood. Assuming that the feature descriptor values of the purple part are greater than the threshold and meet the boundary constraints, further subdivision is carried out, as shown in (b). Repeat the above steps until the boundary conditions are satisfied. To balance efficiency and accuracy in neighborhood subdivision, it has been found in experiments that limiting the subdivision to three times produces satisfactory results.



(a) First Subdivision



(b) Second Subdivision

Fig. 17 Neighborhood Subdivision Strategy Principle

3 Experimental verification

The effectiveness of point cloud simplification is influenced by many factors and is generally assessed from the following three perspectives [23]:

- (1) *Simplification Ratio* The ratio of the data removed during the simplification process to the initial data. When selecting the simplification ratio, a reasonable

- analysis and judgment based on actual requirements should be made to ensure that the accuracy after simplification meets the needs of practical applications.
- (2) *Accuracy* The difference between the 3D models obtained before and after simplification. If the error is too large, it may result in the loss of some detailed information (such as sharp corners, pits, edges, etc.), leading to data distortion.
 - (3) *Speed* The time taken during the simplification process. As the scale of point cloud data increases, point cloud simplification algorithms must achieve practical applications and widespread adoption while ensuring both simplification ratio and accuracy, without excessively long processing times.

To validate the practicality and reliability of the proposed method, tests were conducted using measured large-scale Common-Bottom Component point cloud data and measured large-scale storage tank head point cloud data. The measurement device used was the Creaform MetraSCAN 3D optical CMM scanning system, a handheld 3D scanning measurement instrument with a model of MetraSCAN BLACK™ |Elite, achieving a measurement accuracy of up to $\pm 35 \mu\text{m}$. The measured workpieces and data are as follows:

3.35 m rocket Common-Bottom Component, with an approximate skin area of 10.8 m^2 . The scanned point cloud data amounted to 14.82 million points, and the skin is an irregular surface with features such as small depressions, protrusions, and weld seams with a width of approximately 10 mm, as shown in Fig. 18a, b.

3.1 Analysis of normal vector calibration results

The normal vector calculation uses the PCA method described in Sect. 2.2. The direction of the original normal vectors is shown in Fig. 19a, where the red lines represent the normal vectors, and the blue points represent the original point cloud. Figure 19b shows the result of correcting the normal vector direction using the minimum cost tree method. It can be seen that there are still a few points whose normal vector direction is inconsistent with the overall direction. Figure 19c shows the result of refining the normal vectors in the point cloud, achieved through the NormalRefinement function in the PCL library, which can enhance the accuracy and consistency of the normal vectors. However, there are still a few outliers points visible in the figure. Figure 19d shows the result of redirecting the normal vectors using the optimization formula proposed in Sect. 2.2. It can be observed that this method achieves global consistency of normal vectors.

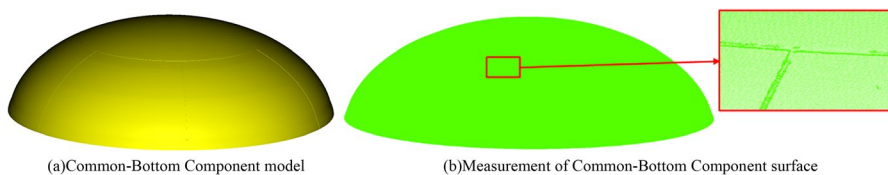


Fig. 18 Measurement Workpiece. **a** Common-Bottom Component model, **b** Measurement of Common-Bottom Component surface

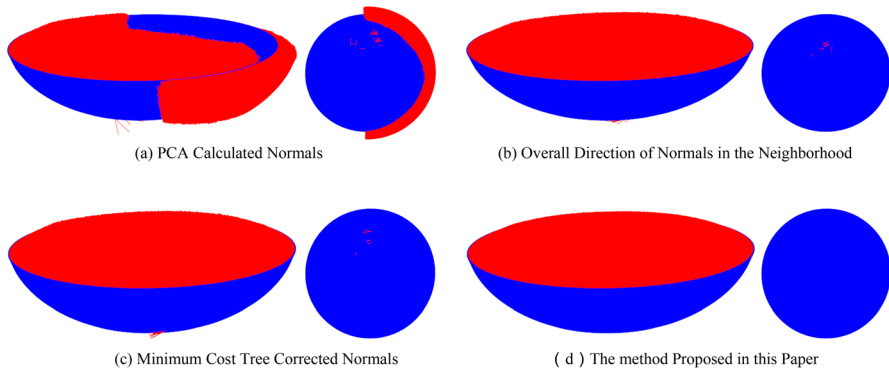


Fig. 19 Normal Vector Calibration Results. **a** PCA Calculated Normals, **b** Overall Direction of Normals in the Neighborhood **c** Minimum Cost Tree Corrected Normals **d** The method Proposed in this Paper

3.2 Analysis of feature recognition results

To demonstrate the specific impact of each parameter on feature recognition, experiments were conducted using the Common-Bottom Component surface point cloud. The results are shown in Fig. 20. Figure 20a displays the feature descriptor values with $\delta_i = D_i$. It can be observed that the features are not prominent due to noise, and boundary features are lost. Figure 20b shows the feature descriptor values with $\delta_i = \kappa_i^n D_i$, where the κ_i^n parameter makes the features more distinct. Figure 20c illustrates the feature descriptor values with $\delta_i = \kappa_i^n \kappa_i^o D_i$, where the κ_i^o parameter accurately identifies boundary features. Figure 20d presents the feature descriptor values with $\delta_i = \kappa_i^n \kappa_i^o \kappa_i^p D_i$, where the κ_i^p parameter effectively suppresses the impact of noise on feature recognition. Figure 21, under the conditions of Fig. 20d, depicts the change in parameter values according to the feature descriptor values (To-red curve) from large to small. It can be seen that, as the feature descriptor values decrease, the parameter values generally follow a decreasing trend, but there are significant local fluctuations. This indicates that the proposed weight Y in this paper effectively suppresses noise, enhances feature recognition accuracy, and retains boundary features.

Figure 22 shows the point cloud feature recognition results. To validate the algorithm’s generality, feature recognition verification was performed on the Stanford Bunny point cloud and a point cloud from the bottom measurement of a large storage box, as shown in Fig. 23. From the figures, it can be observed that the feature descriptors proposed in this paper can accurately identify features.

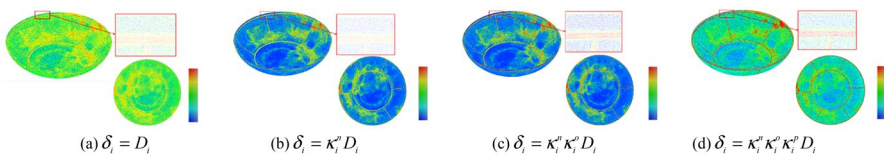


Fig. 20 Impact of Various Parameters on Feature Recognition Results

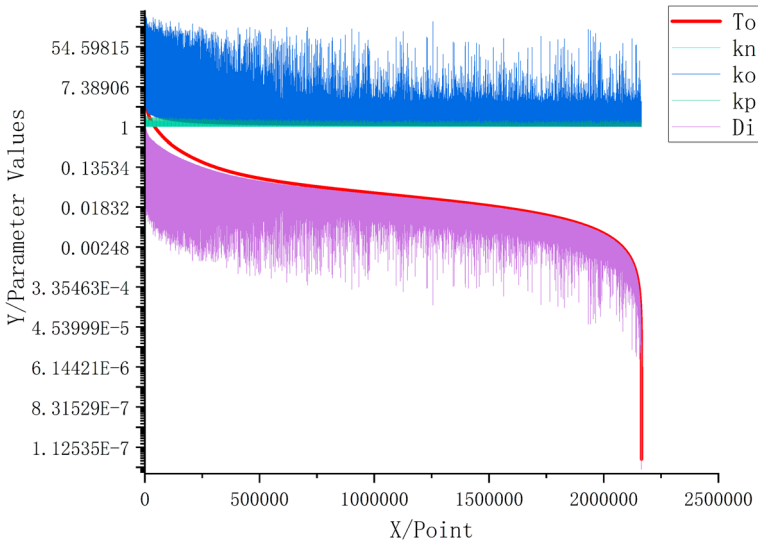


Fig. 21 Variation of Feature Descriptor

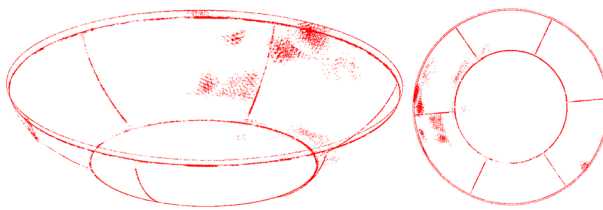


Fig. 22 Common-Bottom Component

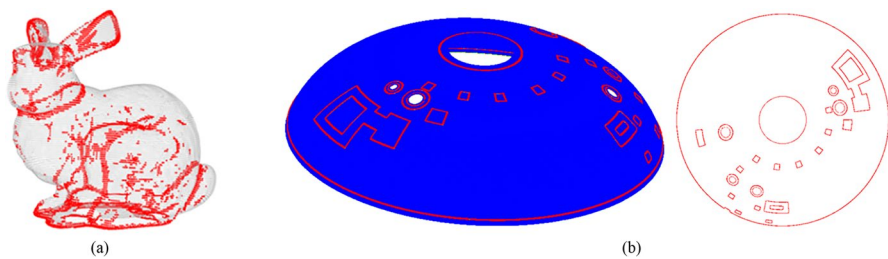


Fig. 23 Feature Recognition Results, **a** Feature Recognition Result for Bunny Point Cloud, **b** Feature Recognition Result for Storage Tank Head

3.3 Simplification results analysis

For large point clouds, this paper adopts a high simplification rate strategy to minimize redundant points. The voxel grid side length for coarse simplification of large Common-Bottom Component surface is set to 1.5 mm. During the fine simplification

Table 1 Point Cloud Quantities

Workpiece	Original point cloud quantity	Coarse simplification quantity	Fine simplification quantity		
			Simplification Rate/% ¹	Total quantity	Feature point quantity
Common-bottom component surface	14,820,000	2,164,823	89.0 (25)	1,623,617	81,180
			92.7 (50)	1,082,411	56,432
			96.3 (75)	541,205	41,200
			99.3 (95)	108,241	30,760

¹The simplification rate outside the parentheses is calculated using the point cloud quantity obtained after fine simplification and the original point cloud quantity. The simplification rate within the parentheses is calculated using the point cloud quantity obtained after fine simplification and the point cloud quantity obtained after coarse simplification

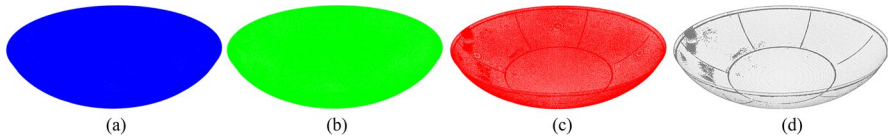


Fig. 24 Results of the Simplification Method Proposed in this Paper, **a** 88.42% Simplification Rate, **b** 92.27% Simplification Rate, **c** 96.25% Simplification Rate, **d** 99.25% Simplification Rate

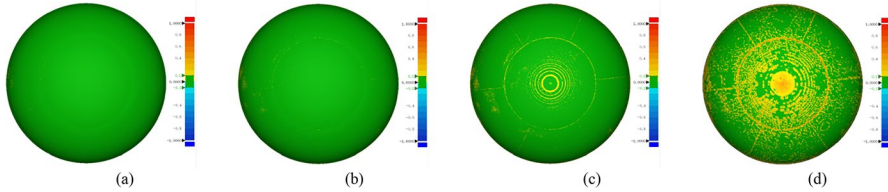


Fig. 25 Error Map of the Simplification Method Proposed in this Paper, **a** 88.42% Simplification Rate, **b** 92.27% Simplification Rate, **c** 96.25% Simplification Rate, **d** 99.25% Simplification Rate

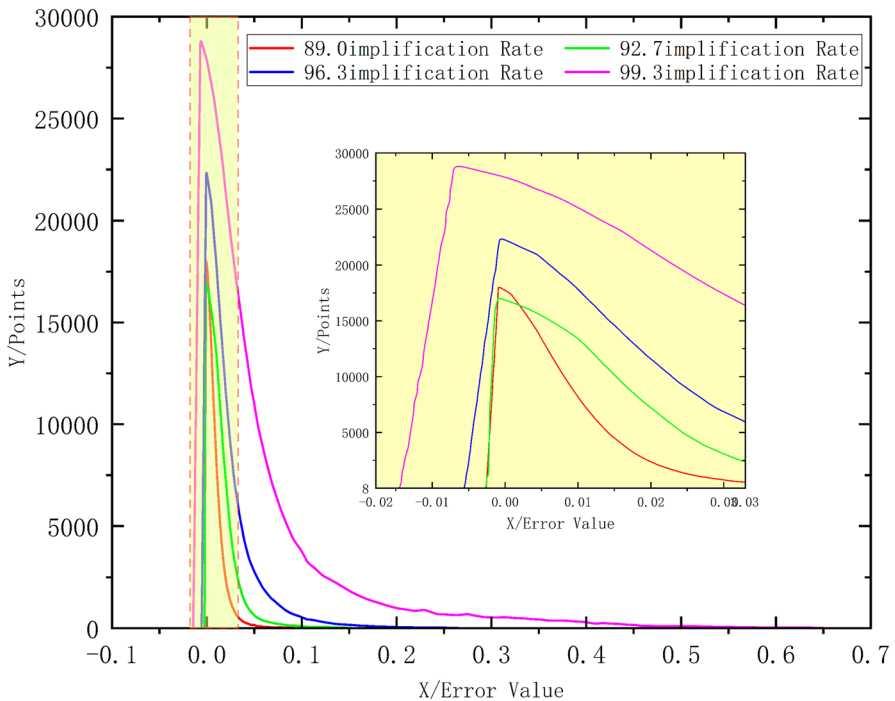


Fig. 26 Error Values of the Method Proposed in this Paper

process, the ratio of feature points to non-feature points is between 0 and 1, and the resulting point cloud quantities are shown in Table 1.

Table 2 Results of Different Methods for Point Cloud Simplification




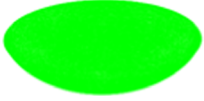

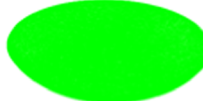






methods	Qi's method	Lv's method	Chen's method
88.42% simplification rate			
92.27% simplification rate			
96.25% simplification rate			
99.25% simplification rate			

Table 3 Error Comparison of Different Simplification Methods on Large Bottom-shared Models

method	simplification rate/%	Max Error/mm	Average Error/mm	Percentage of Points with Error > 0.02 mm /%	Simplification Time/s
Lv' method	88.42	0.5316	0.0619	84.23	1498
	92.27	0.6702	0.0647	76.45	1320
	96.25	0.7930	0.0712	75.02	890
	99.25	1.2784	0.1216	74.90	549
Chen' method	88.42	0.1059	0.0277	26.90	538
	92.27	0.1648	0.0292	24.75	474
	96.25	0.2310	0.0358	23.77	412
	99.25	0.5570	0.0724	20.14	369
Qi' method	88.42	0.3563	0.0619	76.55	695
	92.27	0.4410	0.0647	63.82	633
	96.25	0.6099	0.0712	56.14	582
	99.25	0.8273	0.1216	48.60	519
Our method	88.42	0.0869	0.0106	21.67	563
	92.27	0.1561	0.0169	20.91	492
	96.25	0.2143	0.0277	18.24	437
	99.25	0.5813	0.0703	7.06	352

The error results in this paper are compared based on a data processing software built using VS+Qt+PCL+VTK. The obtained error value is the distance difference between the triangulated mesh models obtained after Poisson reconstruction for the point cloud after fine simplification and the original point cloud after Poisson surface reconstruction. For the measured point cloud data, the results of the simplification method proposed in this paper are shown in Fig. 24. As the simplification rate increases, the feature points are well preserved. The error cloud map is shown in Fig. 25, where the error values increase as the simplification rate rises. The error value change curve is depicted in Fig. 26, showing a continuous decrease in simplification accuracy with the increase in the simplification rate, with the average error increasing from 0.0106 to 0.0703 mm.

To validate the reliability of the point cloud simplification method proposed in this paper, comparisons were made with the methods suggested by Chen [10], Lv [12], and Qi [24]. The simplification results are presented in Table 2, and the error values are shown in Table 3. Our method exhibits relatively small proportions of maximum error, average error, and error values exceeding 0.02 mm. Even with an increase in the simplification level, the simplification error of our method remains smaller compared to other approaches. Qi et al.'s method performs well in smooth regions but poorly in feature areas, leading to potential feature loss. Chen et al.'s method is sensitive to noise and may identify noise as feature points. Lv et al.'s method is time-consuming for large-scale point cloud data and, despite providing solutions for curvature sensitivity and sharp feature preservation, remains insensitive to concave-convex features on large Common-Bottom Component surface. It can be seen that our method balances accuracy and efficiency, and provides an effective means for efficient, high-precision streamlining of large point clouds.

4 Conclusion

This paper presents a novel point cloud simplification method for large-scale point clouds. The method effectively enhances simplification efficiency through coarse and fine simplification steps. By calibrating normal vectors, the method ensures accuracy and global consistency of the normal vectors. The use of feature descriptors helps suppress noise and accurately identify feature points, including boundary features. Additionally, neighborhood subdivision strategy aids in further recognizing potential feature points.

Based on the measured point cloud data of large Common-Bottom Component surface, the results of this method are compared with other reduction algorithms, and the accuracy of this method is intuitively demonstrated through error visualization and error value, which provides a reference for large point cloud reduction to a certain extent.

Acknowledgements This research was supported by the National Key Research and Development Program of China(2022YFB3404700), the Chinese Fundamental Research Funds for the Central Universities under Grant DUT22LAB505, the Changjiang Scholar Program of Chinese Ministry of Education (No. Q2021053, TE2022037).

Author contributions Haibo Liu, Yongqing Wang, and Xiaoming Lai contributed the central idea, Jiangsheng Wu and Xingliang Chai established the theoretical model and wrote the initial draft of the paper. Te Li and Yongqing Wang designed the experiment and analyzed most of the data, Chenglong Wang and JianChi Yu performed the experiment operation.

Data availability All data that support the findings of this study are included within the article (and any supplementary files).

Declarations

Competing interests The authors declare no competing interests.

References

1. Song H, Feng HY (2009) A progressive point cloud simplification algorithm with preserved sharp edge data. *Int J Adv Manuf Technol* 45:583–592. <https://doi.org/10.1007/s00170-009-1980-4>
2. Park IK, Lee SW, Lee SU (2003) Shape-adaptive 3-D mesh simplification based on local optimality measurement. *J Visual Comp Animat* 14:93–104. <https://doi.org/10.1002/vis.308>
3. Huang MC, Tai CC (2000) The pre-processing of data points for curve fitting in reverse engineering. *Int J Adv Manuf Technol* 16:635–642. <https://doi.org/10.1007/s001700070033>
4. C Ji, Y Li, J Fan and S Lan (2019) A Novel Simplification Method for 3D Geometric Point Cloud Based on the Importance of Point *IEEE Access* 7 129029–129042. <https://doi.org/10.1109/ACCESS.2019.2939684>
5. Cignoni P, Montani C, Scopigno R (1998) A comparison of mesh simplification algorithms. *Comput Graph-UK* 22:37–54. [https://doi.org/10.1016/S0097-8493\(97\)00082-4](https://doi.org/10.1016/S0097-8493(97)00082-4)
6. Cheng YQ, Li WL, Jiang C (2022) A novel point cloud simplification method using local conditional information. *Meas Sci Technol* 33:125203. <https://doi.org/10.1088/1361-6501/ac8ac1>
7. He LP, Yan ZM, Hu QJ (2023) Rapid assessment of slope deformation in 3D point cloud considering feature-based simplification and deformed area extraction. *Meas Sci Technol* 34:055201. <https://doi.org/10.1088/1361-6501/acafff>
8. Han HY, Han X, Sun FS, Huang CY (2015) Point cloud simplification with preserved edge based on normal vector. *Optik* 126:2157–2162. <https://doi.org/10.1016/j.ijleo.2015.05.092>
9. Wu JM, Han X, Li DZ (2014) Point cloud simplification based on angle of Inter-normal and hausdorff distance. *Microelectron Comput* 31:52–55. <https://doi.org/10.1930/j.cnki.issn1000-7180.2014.04.013>
10. Chen H, Cui W, Bo C, Ning Y (2023) Point cloud simplification for the boundary preservation based on extracted four features. *Displays* 78:102414. <https://doi.org/10.1016/j.displa.2023.102414>
11. Zhang K, Qiao S, Wang XH, Yang YT, Zhang YQ (2019) Feature-preserved point cloud simplification based on natural quadric shape models. *Appl Sci* 9:10. <https://doi.org/10.3390/app9102130>
12. CL LV, WS Lin and BQ Zhao (2021) Approximate Intrinsic Voxel Structure for Point Cloud Simplification *IEEE Trans. on Image Processing* 30 7241–7255. <https://doi.org/10.1109/TIP.2021.3104174>
13. Tian Y, Song W, Sun S et al (2019) 3D object recognition method with multiple feature extraction from LiDAR point clouds. *J Supercomput* 75:4430–4442. <https://doi.org/10.1007/s11227-019-02830-9>
14. Rolandos Alexandros P, Giorgos B, Stefanos Z (2022) Revisiting point cloud simplification: a learnable feature preserving approach. *Proc Eur Conf Comput Vis* 13662:586–603. https://doi.org/10.1007/978-3-031-20086-1_34
15. Qi CR, Su H, Mo K and Guibas LJ (2017) Pointnet: deep learning on point sets for 3D classification and segmentation *2017 Proc. IEEE Conf. Comput. Vis. Pattern Recog.* pp 77–85. <https://doi.org/10.1109/CVPR.2017.16>

16. Qi CR, Yi L, Su H and Guibas LJ (2017) Pointnet++: deep hierarchical feature learning on point sets in a metric space *Adv Neural Inform Process Syst* 30
17. Yuan H, Feipeng D and Lin T (2017) Research on fast simplification algorithm of point cloud data *5th Int. Conf. Optical and Photonics Engineering* 10449. <https://doi.org/10.1117/12.2270833>
18. Vo AV, Truong-Hong L, Laefer DF, Bertolotto M (2015) Octree-based region growing for point cloud segmentation *ISPRS. J Photogramm Remote Sens* 104:88–100. <https://doi.org/10.1016/j.isprsjprs.2015.01.011>
19. Rusu RB and Cousins S (2011) 3D is here: Point cloud library (PCL) *IEEE Int. Conf. on Robot Autom* (Shanghai, PEOPLES R CHINA) pp 1–4
20. Xu MF, Xin SQ, Tu CH (2018) Towards globally optimal normal orientations for thin surfaces. *Comput Graph-UK* 75:36–43. <https://doi.org/10.1016/j.cag.2018.06.002>
21. Xiao D, Shi Z and Li S (2023) Point normal orientation and surface reconstruction by incorporating iso-value constraints to poisson equation. Ithaca: Cornell University Library ([arXiv:2023.102195](https://arxiv.org/abs/2023.102195)). <https://doi.org/10.1016/j.cagd.2023.102195>
22. Hestenes MR, Stiefel EL (1952) Methods of conjugate gradients for solving linear systems. *J Res Natl Bur Stand* 49:409–436. <https://doi.org/10.6028/jres.049.044>
23. Li W, Liu L, Peng C (2022) A new simplification algorithm for point cloud based on the vertical plane constraint and moving window. *IEEE Access* 10:112555–112564. <https://doi.org/10.1109/ACCESS.2022.3215603>
24. J Qi, W Hu and Z Guo (2019) Feature preserving and uniformity-controllable point cloud simplification on graph, 2019 Proc. IEEE Int. Conf. Multimedia Expo. pp 284–289. <https://doi.org/10.1109/ICME.2019.00057>

Publisher's Note Springer Nature remains neutral with regard to jurisdictional claims in published maps and institutional affiliations.

Springer Nature or its licensor (e.g. a society or other partner) holds exclusive rights to this article under a publishing agreement with the author(s) or other rightsholder(s); author self-archiving of the accepted manuscript version of this article is solely governed by the terms of such publishing agreement and applicable law.