Anisotropic neighborhood searching for point cloud with sharp feature

Xiaocui Yuan, Baoling Liu and Yongli Ma

Abstract
The k-nearest neighborhoods (kNN) of feature points of complex surface model are usually isotropic, which may lead to sharp feature blurring during data processing, such as noise removal and surface reconstruction. To address this issue, a new method was proposed to search the anisotropic neighborhood for point cloud with sharp feature. Constructing KD tree and calculating kNN for point cloud data, the principal component analysis method was employed to detect feature points and estimate normal vectors of points. Moreover, improved bilateral normal filter was used to refine the normal vector of feature point to obtain more accurate normal vector. The isotropic kNN of feature point were segmented by mapping the kNN into Gaussian sphere to form different data-clusters, with the hierarchical clustering method used to separate the data in Gaussian sphere into different clusters. The optimal anisotropic neighborhoods of feature point corresponded to the cluster data with the maximum point number. To validate the effectiveness of our method, the anisotropic neighbors are applied to point data processing, such as normal estimation and point cloud denoising. Experimental results demonstrate that the proposed algorithm in the work is more time-consuming, but provides a more accurate result for point cloud processing by comparing with other kNN searching methods. The anisotropic neighborhood searched by our method can be used to normal estimation, denoising, surface fitting and reconstruction et al. for point cloud with sharp feature, and our method can provide more accurate result comparing with isotropic neighborhood.

Keywords
k nearest neighbors, neighborhood searching, hierarchical clustering, normal estimation

Date received: 28 April 2020; accepted: 16 September 2020

Introduction
In the field of reverse engineering application, generally, two methods including contact and non-contact scanning devices are used to acquire 3D point cloud of objects, but the collected data is 3D point cloud commonly. It can realize surface reconstruction and product remanufacturing by processing the point cloud. However, there is not any topological structure for scattered point cloud only containing 3D coordinate information.

Most point cloud processing methods depend on the neighborhood structure of point. Efficiency and accuracy of neighborhood searching method directly affect the speed and result of point cloud processing including normal estimation, point cloud registration, denoising and surface reconstruction. For example, estimating normal usually needs to search k nearest neighborhood (kNN) of each point for building the local tangent plane; for most of point cloud denoising and reconstruction algorithms, the normal vector and the kNN are the indispensable input parameter. Thus, neighborhood searching is of great significance to point cloud processing and its application.

The most commonly used neighborhood in point cloud processing is kNN. It is defined as follows: Given a point cloud \( p \) including \( n \) points in \( \mathbb{R}^d \) and a positive integer \( k \leq n - 1 \), compute the kNN of each point in \( p \).

More formally, let \( p = \{p_1, p_2, ..., p_n\} \) be a point cloud in \( \mathbb{R}^d \), where \( d \leq 3 \); for each \( p_i \in P \), let \( Nb_k(p_i) \) be the \( k \) points in \( P \) closest to \( p_i \).

In recent year, many kNN searching methods have been put forward. The most typical method is to directly calculate the Euclidean distance between current query point \( p \) and any other points one by one, sorting these points in ascending order according to the distance. Then the closest \( k \) points are chosen for kNN.
This method, although time-consuming and poor-efficiency, is easy to be implemented. To address this problem, many neighborhood searching methods are proposed to improve the computational efficiency. These algorithms fall into two categories broadly: data reorganization (DR) algorithm and spatial partition (SP) algorithm.

DR algorithms divide point cloud into multi-layered subspaces that are employed to construct tree nodes based upon different splitting rules. In these algorithms, the point cloud dataset is regarded as multi-children whose tree nodes are different from each other because of splitting rules. Thus, splitting rules can decide searching efficiency. For instance, KD_tree is a widely used nearest neighborhood searching method which is a space-partitioning data structure for organizing points in a k-dimensional space. It employs tree nodes to store space range of partition data. Other tree data structure, such as, Cell_tree and VP_tree are also employed for nearest neighborhood searching. Cell_tree searches nearest neighbors by dividing the data space into cubes of equal size. VP_tree builds a binary tree using a spherical with distance from the selected advantageous location that is different from Cell_tree building tree employing cubic cell. The KD_tree is most widely used kNN searching method for point cloud.

SP algorithm is also called 3D grid partition method. It partitions the bounding box of point cloud into lots of cells. When searching the nearest neighborhood of a point, SP algorithm first locates the cell of a current query point, calculates the distance of each point in the cell to the current query point, and sorts the point in ascending order by distance. If points in the cell are enough, and the k-th shortest distance is smaller than the distance between the current query point, then stop searching. Otherwise, it continues to search more cells until meet conditions. SP algorithm splits the point cloud dataset into a series of cells, and searching the nearest neighborhood in the neighbor cells, thus reducing the searching scope. It is not only searching nearest neighborhood fast, but also with satisfactory accuracy.

Although there are a lot of kNN searching algorithms, most of them mainly focus on how to improve the computational efficiency without considering the accuracy of the neighborhood for point cloud with sharp feature. When the point cloud model is smooth everywhere, kNN can be used to describe local surface information; however, when the point cloud model contains sharp feature, the kNN of feature point are anisotropic, which involves points locating at more than one surface patch. Point data processing based on the isotropic neighbors may lead to feature blurring or large processing error. To better preserve the original geometry during data processing, such as normal estimation, denoising and surface reconstruction, the neighbors of point should be anisotropic, located at one continuous surface patch. For feature preserving surface mesh denoising, Wang et al. searched the anisotropic neighborhoods for each vertex by constructing a weighted dual graph, over which biquadratic Bezier surface patches are fitted and projected. Similarly, Zhu et al. segmented the isotropic neighborhood to some anisotropic sub-neighborhoods via local spectral clustering. Although these methods can calculate the anisotropic neighborhood for mesh surface, they could not be directly used for point cloud for neighborhood searching. If the neighbors of points searching by SP and DR are directly applied to point cloud with sharp feature for processing, such as normal estimation and point denoising, it will lead to serious processing error. The motivation of the work was to search appropriate neighborhood differing from the most of algorithms which are reducing searching scope and computational time. Thus, an anisotropic neighborhood searching method for point cloud with sharp feature was proposed. In this work, the KD tree was used to search the kNN, and the isotropic kNN are segmented by mapping the neighbors to Gaussian sphere to form a series of sub-neighbors which is anisotropic.

The rest of the work is organized as follows. Section 2 gives the details of our novel algorithm for neighborhood searching. The experimental results and discussions are presented in Section 3, and the conclusions are drawn in Section 4.

Our method

Normal estimation and feature point detection

The normals of points are estimated and used to map the isotropic neighbors into Gaussian image. Many normal estimation methods can be classified into two classes: local surface fitting (LSF) and Delaunay-based methods. LSF is commonly used method to estimated normal vector for point cloud. It is suppose that the surface of point cloud is smooth everywhere. As mentioned in our previous work, the kNN of point are usually employed for fitting the least square tangent plane and calculating the covariance matrix of local surface. The normal vector of a point is the eigenvector corresponding to the smallest eigenvalue of the covariance matrix. LSF method is also called the PCA method. The estimated normals based on PCA method are accurate in flat regions, but normals of feature points are not accurate enough since neighbors of feature point are isotropic. To accurately estimate the normal vector, many algorithms have been reported but these algorithm are more time-consuming comparing to the PCA method. Compromising efficiency and accuracy, the PCA method is used for normal estimation for point cloud.

The PCA method for normal estimation and feature detection is as follows: For a point set \( P = \{p_1, p_2, \ldots, p_n\} \) where \( n \) is the total number of a point cloud. Constructing KD tree for point cloud data \( P \), the kNN of point \( p_i \) is represented as \( Nb_{k}(p_i) \). Let \( u \) be the
The least eigenvalue $\lambda_0$ is to define the surface variation along the surface normal. The estimated normal vector based on PCA method is $n_{i}$, the refined normal $n_{i}$ of $f_{i}$ is defined as:

$$n_i = \sum_{j \in NB_{k}(p_i)} \omega_{i,j} \left( ||c_i - c_j|| \right) / \sum_{j \in NB_{k}(p_i)} \omega_{i,j} \left( ||c_i - c_j|| \right)$$  \hspace{1cm} (3)$$

Where $N(i) = \{ j : ||c_i - c_j|| < \rho = 2\sigma_d \}$ is the neighboring face set of $n_{i}$, and the corresponding unit normal of the neighboring face $f_j$ in $N(i)$ is $n_j$.

Similarly, in our previous work, we proposed an improved bilateral normal filtering to refine the normal of point cloud as follows:

$$n_i = \frac{\sum_{j \in NB_{k}(p_i)} \omega_{i,j} \left( ||p_j - p_i|| \right) W_{n_i} \left( ||n_j - n_i|| \right) n_j}{\sum_{j \in NB_{k}(p_i)} \omega_{i,j} \left( ||p_j - p_i|| \right) W_{n_i} \left( ||n_j - n_i|| \right)}$$  \hspace{1cm} (4)$$

where $NB_{k}(p_i)$ represent the $k$-nearest neighbors for point $p_i$, the value of $k_{i}$ ranging from 16 to 32; $n_i$ and $n_j$ are the normal vectors of $p_i$, and $p_j$, respectively, and normals are normalized and adjusted to the same direction; equations (3) and (4) both consider the same Gaussian function $\omega_{i,j}(d)$, and the difference lies on the definition of $W_{n_i}$ and $W_{n_j}$ in equations (3) and (4). Therefore, we analyze $W_{n_i}$ and $W_{n_j}$, that is,

$$\begin{align*}
W_{n_i}(n) &= \exp \left(-\frac{||n \cdot (n - n_i)||^2}{2\sigma_n^2} \right) \\
W_{n_j}(n) &= \exp \left(-\frac{||n_j - n_i||^2}{2\sigma_n^2} \right)
\end{align*}$$  \hspace{1cm} (5)$$

$W_{n_i}$ and $W_{n_j}$ are the Gaussian functions of the normal vector in equations (3) and (4) respectively, and the $\delta_n$ is the standard deviation. Consider the unit normal vector $n_i$ as the y-axis direction under $xoy$ coordinate system, and the distribution of $W_{n_i}$ and $W_{n_j}$ in term of $n_i$ are shown in Figure 2. From Figure 3 it can be know that, $W_{n_j}$ is greater than $W_{n_i}$ with the same $n_j$, which means that $n_j$ under $W_{n_j}$ has more weights on the refining result $n_i$ of $n_j$ than that under $W_{n_i}$. As a result, equation (3) is more sensitive to noise and feature point compared with equation (4). In addition, the weight that is, $W_{n_i}$ and $W_{n_j}$ both decrease as the normal vector difference of $n_i$ and $n_j$ increases, therefore, a larger the deviation between $n_i$ and $n_j$, the less effect on the refining result.

Figure 3 is the schematic of point cloud and normal vector before refining in two surface patches, where the red point represents the feature point. The kNN of feature points are located at two surface patches, and the blue points in surface patch $S_2$ can be regard as noise or outlier for point in surface $S_1$ when normal vector refining. Although the kNN of feature point locate at two surface patches at least, most of neighbors of feature points are non-feature points. Normal vectors of
non-feature points are similar to each other because some of neighbor points are located at one surface patch. When neighbor points are located at different surface patches, normals of these points are deviated from each other; the surface is sharper; the angle of normal vector between different points is larger.

Even though equation (4) decreases the effect of noise point, it still exists with a certain value. Consequently, more iteration times are required to refine the normal vector. To tackle this problem, we optimize equation (4) as follows:

$$\mathbf{n}_i = \frac{\sum_{j \in \mathcal{N}_k(i)} \omega_d(\mathbf{p}_j - \mathbf{p}_i) \mathbf{W}_{n_j}(\mathbf{n}_j \cdot \mathbf{n}_i) \mathbf{n}_j}{\sum_{j \in \mathcal{N}_k(i)} \omega_d(\mathbf{p}_j - \mathbf{p}_i) \mathbf{W}_{n_j}(\mathbf{n}_j \cdot \mathbf{n}_i)}$$

(6)

where $W_{n_j}(\mathbf{n})$ is represented by:

$$W_{n_j}(\mathbf{n}_j, \mathbf{n}_i) = \begin{cases} 0 & \text{if } (\mathbf{n}_j, \mathbf{n}_i) \mathbf{n}_j \perp \mathbf{u}_1 \\ \left[\mathbf{n}_j \cdot \mathbf{n}_i - \mathbf{u}_1\right]^2 & \text{otherwise} \end{cases}$$

(7)

where $\mathbf{u}_1 = \frac{\sum_{j \in \mathcal{N}_k(i)} [\mathbf{n}_j \cdot \mathbf{n}_i] \mathbf{n}_j}{\sum_{j \in \mathcal{N}_k(i)} [\mathbf{n}_j \cdot \mathbf{n}_i]}$.

Essentially, we truncate the normal vectors if the difference between them and $\mathbf{n}_i$ are greater than the average normal vector difference $\mathbf{u}_1$. From Figure 2, the weight $W_{n_j}$ is always smaller than $W_{n_i}(\mathbf{n})$ and $W_{n_i}(\mathbf{n})$ with the same $\mathbf{n}_i$. Thus, our method is less sensitive to noise and outlier.

In order to improve the computational efficiency, only normals of feature points are refined. Figure 4 is the normal refine results for two different point cloud model. It can be seen that the refined normal vector is nearly perpendicular to the corresponding local surface. For comparison, the refined normal based on our method and our previous work is shown in Figure 5. Three iterations are performed for both methods when normal refining. The detailed comparison result can be seen the black rectangular area. It can be seen that our method can obtain more precision results for normal refining with the same iteration.

**Anisotropic neighborhood searching**

To get the anisotropic neighborhood, the isotropic neighborhood is divided into several sub-neighborhoods by mapping the isotropic kNN into Gaussian image.

Let $T = T(u, v) \subset \mathbb{R}^3$ be a regular surface with consistent and normalized normal vectors in a three dimensions space. The map $G : T \rightarrow T^2$, and $T^2$ represents a unit normal of a point $p (p \in T)$ in the unit sphere,

$$T^2 = \{(x, y, z) \subset \mathbb{R}^3 | x^2 + y^2 + z^2 = 1\}$$

where

The resulting map $G : T \rightarrow T^2$ is defined as the Gaussian map of the surface $T$. The Gaussian image can be denoted by $G(T) = \{n(p) | p \in T\}$. The unit sphere $T^2$ can be called the Gaussian sphere.
The Gaussian image has very important property: If a connected surface is a plane, the Gaussian image of the point on that surface is a point; Conversely, if the Gaussian image of a connected surface is a point, all the points on that surface are planar points. However, there is always error between estimated normal and ground-truth normal. Therefore, the Gaussian image of point cloud dataset in a planar surface display a cluster where data is in dense distribution. Figure 6(a) shows the sketch map of Gaussian image of planar surface.

If the estimated normal vector is approximated to the ground-truth normal, for a point locating at the surface edge area, its kNN points may be from two surface, that the kNN points in Gaussian map displaying two dense clusters; And for a point locating at the surface corner area, its Gaussian map displays several clusters, and the number of clusters equals to that of surface patches. Figure 6(b) shows the schematic plot of Gaussian image of dihedral surface, where data is in dense distribution. Figure 6(a) shows the sketch map of Gaussian image of planar surface.

To segment the isotropic neighborhood of feature points $p_i$, a larger neighbors $N_b, k_2(p_i)$ are selected. For any point $p_i$, its neighbors with size $k_2$ are represented as $p_j \in n_i$ is the normalized normal of point $p_i$; $n_i$ the neighborhood normal. All normal vectors must be normalized and adjusted to the same direction for Gaussian mapping. Then, the Gaussian map of a point $p_i$ can be denoted by $G(p_i) = \{n_{ij}\}, j=0,1,...,k_2-1$. The
clustering algorithm is employed to divide the dataset in Gaussian sphere to appropriate sub-cluster. There are many clustering algorithms have been report in literature, but most of them require knowing the number of clusters to be produced in advance. However, the number of surface patches contained in the isotropic kNN of feature point is unknown, so is the number of clusters in the Gaussian map.

Mean shift algorithm and hierarchical agglomerative algorithm are non-parameter clustering algorithms to cluster data set without specifying the number of cluster, widely used in data mining and machine vision area. However, it is found that the mean shift algorithm usually classifies one cluster data with too many sub-clusters, some of which may just contain one or two points. The hierarchical agglomerative ("bottom-up") clustering method was employed to classify data in the work. It begins with each point as a separate cluster, and merges them successively into larger clusters. For the hierarchical agglomerative clustering algorithm, it is an important step to determine the similarity between two clusters.

There are three classic standards, which are single-linkage, complete-linkage and average-linkage measures, to define the distance between two clusters. The average-linkage standard defining the mean distance between elements of each cluster tends to merge two clusters with small difference. It is used to define the distance between two clusters in the work.

The data set $G(p_i)$ is the input data for hierarchical agglomerative algorithm. Then, the mean distance $D_i$ between elements of each cluster can be defined as

$$D_i(C_1, C_2) = \frac{1}{|C_1||C_2|} \sum_{n_i \in C_1} \sum_{n_j \in C_2} d(n_i, n_j)$$

where $|C_1|$ and $|C_2|$ are the numbers of point data corresponding to clusters $C_1$ and $C_2$, respectively; $d(n_i, n_j)$ is the Euclidean distance between two points $n_i \in C_1$ and $n_j \in C_2$, in the Gaussian map, where $x, y = 0, 1, \ldots, 21$. 

$$d(n_i, n_j) = 2(1 - \langle n_i, n_j \rangle)$$

where $\langle n_i, n_j \rangle$ represents the inner product of $n_i \in C_1$ and $n_j \in C_2$, when the angle between $n_i \in C_1$ and $n_j \in C_2$ is less than $T$. That is, when the distance of two points in Gaussian is less than $2(1 - \cos \theta)$, two points are merged into the one cluster. The largest cluster corresponding to the points are selected as the optimal neighbor for the feature point.

Results and discussion

Parameter selection

There are five parameters influencing the neighborhood searching result, requiring the user to prepare in the algorithm. They are $Thresh$, $t$, $k$, $k_2$, and $\theta$, respectively.

(1) Feature point detection threshold $Thresh$ requires the user to prepare for selecting the feature point where neighbors should be segmented, and normal needs to be modified. The sharpness threshold is related to the “sharpness” of the features of the point cloud. If the surface of point cloud contains very sharp features, it should be a small value; otherwise, it is assigned with a relatively high value. Based on the experimental, the value of $Thresh$ can be 0.001 to 0.1.

(2) $t$ is the iteration time to refining the normal vector. We have the setting in our implementation: $t: [1–5]$.

(3) $k$ is the size of nearest neighborhood, representing the KD tree neighborhood of point data for user to define. After establishing KD tree for point cloud, the $k$NN of the point cloud are selected by user. When a point is non-feature, the $k$NN is the final neighbor.

(4) $k_2$ is the size of nearest neighborhood of feature point, with neighbors mapped into Gaussian image. Usually, the $k_2$-nearest neighbors are isotropic, segmented into several sub-neighbors that are anisotropic. The value of $k_2$ is associated with $k$. Since the feature point is close to the area of intersection of two or more surface patches, we need to consider the surface complexity to set the value of $k_2$, thus making the neighbor size of feature point close to the neighbor of regular point as much as possible.

(5) For example, if the feature points are the intersection of two or more surface patches, it can be set as $k_2 = 2k$; if the feature points are the intersection of three or more surface patches, then it can set as $k_2 = 3k$. It is assumed $k_2 = 3k$, and $k = 16$ in default in this work.

(6) Parameter $\theta$ is used in hierarchical clustering algorithm that denotes the angle threshold of two normal vectors. That is, when the distance of two points in Gaussian is less than $2(1 - \cos \theta)$, two points are merged into the one cluster. When points are located at one continuous surface patch, the normal vector of point between each other is nearly parallel to each other in local area, and the angle between normal vectors is small. However, when the distance between two points in Gaussian sphere is large, these points are from two different surface patches.

The sharper surface patch leads to the larger angle between normal vectors of points in different surface patch. $\theta$ is an important threshold, if $\theta$ is set to a big value, the segmented neighbors is still isotropic; if set to a small value, the isotropic neighbors are segmented into many anisotropic sub-neighbors. Previous researches have measured the accuracy of
estimated normal vector with angle threshold $\theta = 10^\circ$. In other words, when the angle between the estimated normal vector and ground-truth normal vector is less than $10^\circ$, the estimated normal is regarded as accurate, and the corresponding point is a flat point; otherwise, the point is non-flat or bad. The threshold $\theta$ is similar to normal accuracy measurement, so we set $\theta = 10^\circ$ according to the suggestion in the research.\textsuperscript{25,26,32} Namely when the distance between two points in Gaussian image is less than 0.173, two points or clusters are merged into one large cluster.

**Applications of point cloud neighborhood**

In order to verify the validity of our algorithm, the anisotropic neighbors of point cloud are applied to point data processing. We compare our proposed method with the state-of-the-art neighborhood searching algorithms, such as KD tree\textsuperscript{14} and 3D grid\textsuperscript{13} kNN searching method. These algorithms are implemented using Microsoft vision studio 2017 C++, and OpenGL. Since it is hard to visualize neighbors for every point, the neighbors of point are applied for data processing, for example, normal estimation based on local surface fitting, and point data denoising. In the comparison, all the parameters are same except for neighbors.

**Normal estimation based on local surface fitting.** Normal vector is an important property of a 3D surface. Many effective point cloud processing algorithms depend on the accurate normal as input to produce a precise processing result. The normal estimation can be applied in point cloud denoising, simplification, registration and surface reconstruction. LSF method estimates the normal vector by fitting the least square tangent plane using the local neighborhoods. If neighborhoods of point are anisotropic, the estimated normal is nearly perpendicular to the local tangent plane which means normal are accurate. LSF method is used to estimated normal vector with the KD tree, 3D grid and our anisotropic neighbors as input.

Figures 7 and 8 shows the visualization results of normal vectors based on different neighborhoods as input. Octahedral and fandisk model consists of many continuous surface patches, and feature boundaries can be observed clearly. Normals are normalized and adjusted to the consistent direction for displaying. It can be known that the normal vector of point is nearly perpendicular to the local surface in flat area for 3D grid and KD tree neighbor as input. However, the normal vector of feature point is smoothed, deviated from the local surface because the neighbors of feature point are from at least two surface patches. The estimated

![Figure 7. Normal vector estimation results for octahedral model based on different neighbors as input: (a) Triangulated model of octahedral point cloud; (b) KD tree neighbor; (c) 3D grid neighbor; (d) Our neighbor.](image)

![Figure 8. Normal vector estimation results for fandisk model based on different neighbors as input: (a) Triangulated model of fandisk point cloud; (b) KD tree neighbor; (c) 3D grid neighbor; (d) Our neighbor.](image)
normal vectors based on our neighbors are accurate in flat area and feature points because neighbors of feature points are anisotropic, only located at one continuous surface patch.

**Point cloud denoising.** With the development of 3D scanner, it is very convenient to acquire 3D point cloud. However, the collected point cloud data inevitably contains noise due to a variety of internal and external factors interference. Denoising is necessary step to improve the data quality for further data processing. The main problem of point cloud denoising is to remove noise while preserving sharp feature. Bilateral filtering algorithm\(^3\) can be used to denoise for point cloud and mesh surface while preserving the sharp feature; however, it needs the accurate normal and neighbors as input. Bilateral filtering denoising for point cloud is given as follows: Let \( p \in P \) be a point of input point data set \( P \), the bilateral filter denoising result for \( p \) is \( p' \):

\[
p' = p + \alpha \mathbf{n}
\]

where \( \alpha \) is the filter coefficient.

\[
\alpha = \frac{\sum_{p_j \in Nb_K(p)} W_c(\|p - p_j\|) W_s(\| \langle p - p_j, \mathbf{n} \rangle \|) \langle p - p_j, \mathbf{n} \rangle}{\sum_{p_j \in Nb_K(p)} W_c(\|p - p_j\|) W_s(\| \langle p - p_j, \mathbf{n} \rangle \|)}
\]

Where \( \mathbf{n} \) is the normal vector of point \( p; \langle p - p_j, \mathbf{n} \rangle \) is the inner product of \( p - p_j \) and \( \mathbf{n}; \) \( Nb_K(p) \) is the \( K \) neighbors of point \( p \), with \( p_j \in Nb_K(p); \) \( W_c \) and \( W_s \) are weighting functions, where 

\[
W_c(x) = \exp(-x^2/2\sigma_c^2);
\]

\[
W_s(x) = \exp(-x^2/2\sigma_s^2); \sigma_c = \sigma_s = 0.2.
\]

To highlight the influence of neighborhood on denoising results, normal of point is estimated by method introduced in the Literature.\(^{26}\) It can accurately estimate normal vector and robust to noise. Two point cloud which contain sharp feature are employed to test denoising results.

Figures 9 and 10 show the denoising results based on 3D grid, KD tree and our neighbors as input \( Nb_K(p) \). In order to clearly observe the denoising results, the point cloud models are triangulated for displaying. The deviation including maximum error (ME), average error (AE) between denoising model and noise-free model are measured using Geomagic Studio software.\(^{34}\) The values of ME and AE for mounting-foot and gear model are shown in Table 1, and the deviation results are shown in Figures 9 and 10. Figures 9 and 10 and Table 1 show that our neighbors as input for denoising can preserve sharp feature well and generate the least ME and AE values.

**Computational time analysis**

Our proposed method needs to fit a local tangent plane to estimate normal vector and detect feature points, and normal vectors of these points are iteratively refined. After hierarchical clustering in Gaussian sphere, isotropic neighbors are classified into several anisotropic sub-clusters besides constructing KD tree. Therefore, it is more computational than the neighborhood searching method by 3D grid and KD tree. Test computations are implemented on a PC with a 2.5 GHz processor.
Intel Core processor and 16 GB RAM. Table 2 shows the results of computational time for different point cloud. Different computers take different amounts of time to execute the algorithm, thus, the computational time in Table 2 just for reference. According to the computational complexity and Table 2, it is obvious that our method is more time-consuming that several times larger than that required by the method of 3D grid and KD tree.

### Conclusion

The neighborhood searching of point data is very important and a basic work for point cloud processing. A large amount of point data processing algorithms need neighbors as input to produce precise processing result, such as normal estimation, denoising, and reconstruction. Since the kNN of feature points is isotropic that may contain points belonging to more than two surface patches, sharp feature of point cloud model are over-smoothed with these neighbors as input for point processing. An anisotropic neighborhood searching method was proposed in the work.

---

**Table 1.** Comparison of error (mm).

| Point model | Error | KD tree | 3D grid | Our method |
|-------------|-------|---------|---------|------------|
| Mount-foot  | AE    | 0.025   | 0.020   | 0.007      |
|             | ME    | 2.415   | 2.430   | 1.362      |
| Gear        | AE    | 0.002   | 0.002   | 0.001      |
|             | ME    | 0.159   | 0.153   | 0.072      |

**Table 2.** Comparison of computational time.

| Point model    | Number of point cloud | Computational time (s) |
|----------------|-----------------------|------------------------|
|                |                       | KD tree | 3D grid | Our method |
| Fandisk        | 11230                 | 0.21    | 0.19    | 0.41        |
| Octahedron     | 32219                 | 0.58    | 0.51    | 1.06        |
| Mount-foot     | 110985                | 1.92    | 1.85    | 4.07        |
| Gear           | 105309                | 1.88    | 1.78    | 3.89        |

Our method divides a big isotropic kNN of feature points into several anisotropic sub-neighborhoods by mapping the big isotropic kNN neighbors into a Gaussian sphere. The points on Gaussian image formed several sub-clusters corresponding to the anisotropic sub-neighbors. The agglomerative hierarchical clustering algorithm was used to separate the data in Gaussian image into appropriate cluster. The biggest sub-neighborhood corresponding to the biggest sub-cluster was selected as the optimal anisotropic neighbors. Since a large kNN of feature point was segmented into a series of sub-neighbors, the number of neighbors of point is not same everywhere. Neighbors of points are applied to estimate normal and bilateral filtering denoising.

K nearest neighborhood usually is isotropic for point cloud with sharp feature, different from the other k nearest neighborhood search methods, this paper emphasizes on the anisotropic neighborhood searching instead of computational efficiency. Point cloud normal estimation and denoising based on 3D grid, KD tree...
and our neighbors as input, our method can obtain more precision result for point cloud with sharp feature. The anisotropic neighborhood searching based on our method can be used for point cloud normal estimation, denoising, surface fitting, and reconstruction, et al. It is important to point out that the method in the work is more computational and complex than DR and SP neighborhood searching method. The future work will improve the computational efficiency and reduce complexity for seeing anisotropic neighborhood.

Acknowledgement
The authors would like to thank Jiangxi Province Education Department, Jiangxi provincial department of science and technology, National Natural Science Foundation of China and our previous work to support this work.

Author contributions
Xiaocui Yuan proposed the idea of this algorithm, and wrote programs to implement our algorithm and other correlation algorithm. Baoling Liu designed the framework of this paper. Yongli Ma was responsible for data collection and our previous work to support this work.

Declaration of Conflicting Interests
The author(s) declared no potential conflicts of interest with respect to the research, authorship, and/or publication of this article.

Funding
The author(s) disclosed receipt of the following financial support for the research, authorship, and/or publication of this article: This work is support by Jiangxi Province Education Department (Grant No. GJJ161122, GJJ161104, GJJ161110), Jiangxi Provincial Department of Science and Technology (Grant No. 20171BAB206037) and National Natural Science Foundation of China (Grant No. 61903176, 62001202).

ORCID iD
Xiaocui Yuan https://orcid.org/0000-0003-4435-5973

References
1. Arvanitis G, Lalos AS and Moustakas K. Denoising of dynamic 3D meshes via low-rank spectral analysis. Comput Graph 2019; 82: 140–151.
2. Cao J, Chen H, Zhang J, et al. Normal estimation via shifted neighborhood for point cloud. J Comput Appl Math 2018; 329: 57–67.
3. Wu L, Wang G and Hu Y. Iterative closest point registration for fast point feature histogram features of a volume density optimization algorithm. Meas Contr 2020; 53(1-2): 29–39.
4. Xing Y, He Y, He L, et al. A dynamic and adaptive scheme for feature-preserving mesh denoising. Graph Model 2020; 110.
5. Gao J, Deng B, Qin Y, et al. Point cloud and 3-d surface reconstruction using cylindrical millimeter-wave holography. IEEE Trans Instrum Meas 2019; 68: 4765–4778.
6. Li Z, Ding G, Li R, et al. A new extracting algorithm of k nearest neighbors searching for point clouds. Pattern Recognit Lett 2014; 49: 162–170.
7. Yen SH, Shih CY, Li TK, et al. Applying multiple kd-trees in high dimensional nearest neighbor searching. Int J Circ Syst Signal Process 2010, 4(4): 153–160.
8. Bentley JL. K-D trees for semidynamic point sets. In: Proceedings of the sixth annual symposium on computational geometry, Berkley, CA, May 1990, pp.187–197. New York, NY: Association for Computing Machinery.
9. Clarkson KL. Fast algorithms for the all nearest neighbors problem. In: 24th annual symposium on foundations of computer science, Tucson, AZ, 7–9 November 1983, pp.226–232. New York: IEEE.
10. Arya S, Mount D, Netanyahu N, et al. An optimal algorithm for approximate nearest neighbor searching. In: SODA ’94: proceedings of the fifth annual ACM-SIAM symposium on Discrete algorithms, Arlington, VA, USA, 23–25 January 1994, pp.573–582. New York: ACM.
11. Yianilos PN. Data structures and algorithms for nearest neighbor search in general metric spaces. In: SODA ’93: proceedings of the fourth ACM-SIAM symposium on discrete algorithms, Austin, TX, 25–27 January 1993, pp.311–321. Philadelphia, PA: Society for Industrial and Applied Mathematics.
12. Behley J, Steinhage V and Cremers AB. Efficient radius neighbor search in three-dimensional point clouds. In: IEEE international conference on robotics and automation, Seattle, WA, 26–30 May 2015, pp.3625–3630. New York: IEEE.
13. Yang J, Lin Y, Zhang R, et al. A fast algorithm for finding k-nearest neighbors of large-scale scattered point cloud. Int J Image Graph 2016; 18(4): 399–406.
14. Piegl LA and Tiller W. Algorithm for finding all k nearest neighbors. Comput Aided Des 2002; 34: 167–172.
15. Wei W, Zhang LY and Zhou LS. Spatial sphere algorithm for searching k-nearest neighbors of massive scattered points. Acta Aeronaut. Astronaut. Sin 2006; 27: 944–948.
16. Ping XL. Novel algorithm for k-nearest neighbors of massive data based on spatial partition. J South China Univ Technol 2007; 35: 65–69.
17. Xiong B and An X. Algorithm for finding `k`-nearest neighbors of scattered points in three dimensions. J Comput Aided Des Comput Graph 2004; 16: 909–908.
18. Ma J, Fang Y, Zhao W, et al. Algorithm for finding k-nearest neighbors based on spatial sub-cubes and dynamic sphere. Geomatics Inf. Sci. Wuhan Univ 2011; 36: 358–362.
19. Min L and Yun M. Study on point cloud management strategy based on Octree-like index. Laser & Optoelectronics Progress, 2020, http://kns.cnki.net/kcms/detail/31.1690.TN.20200114.0735.004.html
20. Yuan X, Peng Q, Wu L, et al. A novel method of normal estimation for 3D surface reconstruction. In: ASME 2015 international design engineering technical conferences and computers and information in engineering conference, 2–5 August 2015, Boston, MA, p.9. New York: American Society of Mechanical Engineers.
21. Wang J, Zhang X and Yu Z. A cascaded approach for feature-preserving surface mesh denoising. *Comput Aided Des* 2012; 44: 597–610.

22. Lei Z, Wei M, Yu J, et al. Course-to-fine normal filtering for feature-preserving mesh denoising based on isotropic sub-neighborhoods. *Comput Graph Forum* 2013; 32: 371–380.

23. Yuan XC, Lu-Shen WU and Chen HW. Normal estimation of scattered point cloud with sharp feature. *Optic Precis Eng* 2016; 34(10): 2581–2588.

24. Hoppe H, DeRose T, Duchamp T, et al. Surface reconstruction from unorganized points. *ACM SIGGRAPH Comput Graph* 1992; 26(2): 71–78.

25. Zhang J, Cao J, Liu X, et al. Point cloud normal estimation via low-rank subspace clustering. *Comput Graph* 2013; 37: 697–706.

26. Wang Y, Feng H-Y, Delorme F-È, et al. An adaptive normal estimation method for scanned point clouds with sharp features. *Comput Aided Des* 2013; 45: 1333–1348.

27. Liu C, Yuan D and Zhao H. 3D point cloud denoising and normal estimation for 3D surface reconstruction. In: *IEEE international conference on robotics and biomimetics*, Zhuhai, 6–9 December 2015, pp.820–825. New York: IEEE.

28. Lu D, Sun Y and Wang J. Deep feature-preserving normal estimation for point cloud filtering. *Comput Aided Des* 2020; 125: 102860.

29. Lee K and Wang W. Feature-preserving mesh denoising via bilateral normal filtering. In: *Proceedings of the ninth international conference on computer aided design and computer graphics*, Hong Kong, 7–10 December 2005, pp.275–280. New York: IEEE.

30. Wang Y, Hao W, Ning X, et al. Automatic segmentation of urban point clouds based on the Gaussian map. *Photogramm Rec* 2013; 28: 342–361.

31. Hastie T, Tibshirani R, Friedman J, et al. The elements of statistical learning: data mining, inference and prediction. *Math Intel* 2005; 27: 83–85.

32. Li B, Schnabel R, Klein R, et al. Robust normal estimation for point clouds with sharp features. *Comput Graph* 2010; 34: 94–106.

33. Fleishman S, Drori I and Cohen-Or D. Bilateral mesh denoising. *ACM Trans Graph* 2003; 22(3): 950–953.

34. https://www.3dsystems.com/press-releases/geomagic/announces-studio-2013.