An Efficient Feature Point Extraction Algorithm for Noisy Point Clouds

Nanhua Huang [1], Ming Chen[2], Zhengqin Zhang[3] and Shenglian Lu[4]

[1]School of Computer and Engineering, Guangxi Normal University, hnh0774@163.com
[2]School of Computer and Engineering, Guangxi Normal University, hustcm@hotmail.com
[3]School of Computer and Engineering, Guangxi Normal University, 1565493549@qq.com
[4]School of Computer and Engineering, Guangxi Normal University, lsl@gxnu.edu.cn

Corresponding author: Ming Chen, hustcm@hotmail.com

Abstract. Point feature extraction is an important step for point cloud data processing, which includes denoising, matching, segmentation and recognition. The quality of point cloud feature extraction has a significant impact on the outcomes of subsequent point cloud data processing. This paper proposes a point cloud feature extraction algorithm that combines the Smooth Shrink Index (SSI)[11] and Point Density Index (PDI). Voxelization is also used to speed up the algorithm. More specifically, the proposed algorithm is divided into two stages: first, a density evaluation is used to quickly filter most non-feature points, and then a combined feature extraction function is defined that takes into account both SSI and PDI and is used to identify the final feature points among the candidate feature points after the filtering step. The experimental results show that the proposed method has a good anti-noise ability and can extract feature points more completely than three commonly used methods, namely PCA [13], SSI[11], and the method in[17] . Although the feature evaluation function is partially based on the SSI method, the proposed algorithm is 30-40% faster and more correct feature points can be extracted than the SSI method.

Keywords: Point cloud；Feature extraction；Point cloud processing；Reverse engineering
DOI: https://doi.org/10.14733/cadaps.2023.936-945

1 INTRODUCTION

Point cloud models have been widely used in cultural relic restoration, reverse engineering, inspection and measurement, automatic driving, remote sensing measurement, virtual reality, and other fields [19][20]. Generally, acquired point cloud data contain noise and are non-uniformly sampled. The feature extraction of point cloud is critical for point cloud processing such as denoising, matching, segmentation and recognition[8][14][16][21]. How to quickly and accurately extract features from noisy point cloud data is a hot topic and challenging.

Nie [11] recently used Smooth Shrink Index (SSI) to measure the degree of surface change and selected feature points based on the absolute SSI value of each point. This method has good anti-noise properties and can extract sharp feature points with significant surface changes. We discover that repeated scanning will be usually performed at feature regions of interests to increase the acquisition density such that more accurate feature point can be obtained. Some colleagues have used this advantage and attempted to identify feature regions using the density factor [10], but the anti-noise ability is frustrating. This paper combines the above two ideas in the proposed feature extraction algorithm on point cloud data and employs the well-known voxelization method to efficiently achieve better results.

2 RELATED WORK

The feature extraction methods on point cloud are classified into four types in this paper:

1) Curvature and normal vectors-based methods. Altantsetseg *et al*. [1] used the Fourier series to approximate the major direction and evaluated the curvature to detect feature locations. Zhou *et al*. [23] computed the normal vector and curvature for each point before using the entropy function to adaptively determine the neighborhood radius. Ma [10]introduced a point cloud feature extraction approach based on normal vector and density, in which normal vectors are evaluated using principal component analysis. This type of approaches is noise-sensitive.

2) Surface fitting method. Daniels *et al*. [4] first identified potential feature points through a robust moving least squares method, projected the potential feature points onto the intersection of multiple surfaces, and then used principal component analysis to smooth characteristic curves. This method can extract a relatively smooth characteristic curve, but is time-consuming. Kim *et al*. [7] also used moving least squares to fit local planes to obtain curvature and derivatives. Then, the neighborhood information is constructed through the Voronoi diagram to connect the valley feature points. In this work, ridge feature points are identified using the zero-crossings of the extremum of the Delaunay edge curvature derivatives. Generally, the effectiveness and robustness of the surface fitting methods mainly depend on the accuracy of the surface fitting procedure [6], which will take a long time.

3) Cluster-based segmentation method. Demarsin *et al*. [5]first calculated the normal vectors of points and neighborhoods, then performed cluster segmentation according to the normal vector angle threshold, later segmented the positions of sharp edge points as candidates feature points, and finally edge extraction is performed on candidate points by means of image processing. This method is suitable for extracting closed sharp feature lines. Zhang [22] used the normal angle as the local feature detection operator, used Poisson distribution to grow the boundary region, and then performed feature extraction. This method can adapt to the threshold value, and has a better feature extraction effect for non-uniformly sampled 3D point cloud models. However, the robustness to noise depends on the accuracy of the normal vector calculation. Wang *et al*. [18] firstly segmented the point cloud into different sub-regions through a clustering algorithm combining social particle swarm-fuzzy C-means, and then used the angle criterion for edge detection on the segmented sub-regions, and the average curvature was used to detect sharp Feature points. Finally, the minimum spanning tree method was used to generate the characteristic lines. This method has better noise robustness and better extraction effect on non-uniformly sampled point cloud. However, the extraction effect is poor in places with dense feature lines. The method is based on clustering and segmentation cluster and segment the points with the same geometric properties; the segmentation has a significant influence on the extraction result.

4) Others. Cao *et al*. [2] first extracted potential feature points by calculating the weighted average position of each point's neighborhood, calculated the displacement between the point and the average position, and then used principal component analysis to calculate the normal vector of the point. The displacement was next projected to the normal direction, and its extreme values were used to pick out final feature points. This method has a certain anti-noise ability for extracting sharp feature points, but there are many incorrect feature points extracted for non-sharp feature points.

Nie *et al*. [11]sed the Laplacian smooth shrinkage and refinement method to extract features by calculating the smooth shrink index and distinguishing surface concavity and convexity, and finally extracted feature lines with Laplacian refinement. This method has good noise robustness and can distinguish feature lines with short distances, but it often misses the extraction of features with gentle surface changes (see Figure 1), and has poor extraction effect on feature lines with short distances that are convex or concave at the same time. In view of the problem of incomplete feature extraction by a single method, Wang *et al*. [17]extracted point cloud features using a combination of neighborhood average normal vector angle, curvature and density. Chen *et al*. [3] proposed a multi-discriminant method to extract 3D point cloud features. This method identifies point cloud features by combining four parameters, such as curvature and normal vector angle. In the method, many parameters should be determined and much time will be taken, thus it is difficult to determine the proper parameters to achieve an ideal extraction effect.

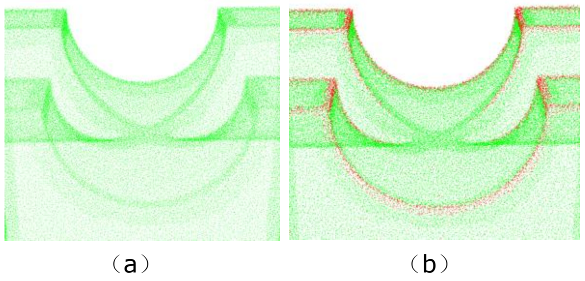

Figure 1: The features at the bottom positions of intersecting lines cannot be fully extracted by SSI method, as these places are of gentle surface changes.

3 METHOD OUTLINE

Inspired by the study [2][11]and [10][15], the proposed method in the paper's feature-evaluation function $F(P_i)$ for are defined using a weighted average of SSI and point density index short for PDI, and the function $F(P_i)$ is used to identify the feature points. In order to improve the efficiency of feature extraction, the well-known voxelization idea is used to accelerate the algorithm. The outline flowchart of the proposed algorithm is shown in Figure 2.

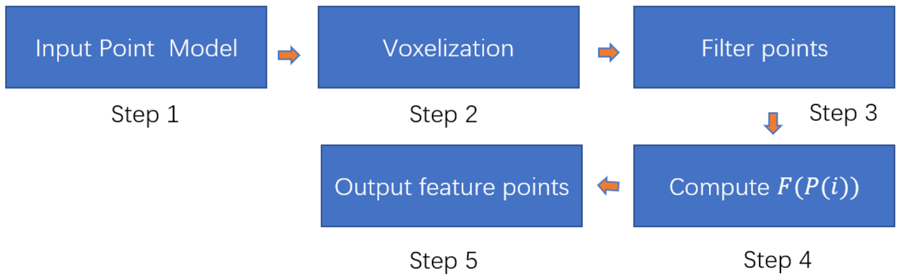

Figure 2: The schematic flow chart of the proposed algorithm.

In Step 1, a point cloud model sampled through a 3D scanner are input, which is of noise. In the scanning process, the scanning angle is adjusted or the interested regions are scanned multiple times, ensuing these featured regions can obtain denser sampled points. The sampled point model is denoted as $\Phi=\{P_i\}$, where P_i is point coordinate. Φ contains a certain amount of noise points, and it is hoped to obtain the feature point set $\Omega=\{Q_i\}$, where $Q_i \in \Phi$. In step 2, voxelization processing is performed to transform Φ into a lattice model C. In step 3, counting the point number for each lattice and filtering points according to the density comparison between each lattice and its adjacent lattices, speeding up the algorithm. All points in the kept lattices will be added to the candidate feature point set Ψ; Step4 calculates the value of $F(\Psi(i))$ for all points in Ψ, and all points of larger $F(\Psi(i))$ than the preset threshold will be regarded as the point of set Ω for an output.

3.1 Feature Point Evaluation Function F(P_i)

$F(P_i)$ consists of two parts, the Smooth Shrink Index (SSI), denoted as $L(P_i)$, and the Point Density Index $D(P_i)$, which are defined as follows:

$$F(P_i) = \alpha L(P_i) + (1-\alpha) D(P_i) \tag{1}$$

Where α is a preset parameter that can be set based on the noise level of the input model: if the noise level is high, α can be set to a relative larger value to improve anti-noise ability. The range of alpha varies from 0.5 to 1.

3.1.1 *SSI* $L(P_i)$

The definition of SSI is proposed in the literature [11]. The study evaluates the distance of one point and its neighboring weighted average point, then projects the distance along its normal vector, and the projected distance is regarded as SSI value, i.e., $L(P_i)$ and it can be calculated in Eq. (2) as below:

$$L(P_i) = ||(P_i - P_i^w) \bullet n_i|| \tag{2}$$

Where P_i^w is the weighted gravity center of the P_i's neighborhood, and n_i is the normal vector at P_i'. n_i is the eigenvector corresponding to the minimum eigenvalue of the covariance matrix E of the k-nearest neighborhood of P_i via the principal component analysis method[9]. The matrix E can be expressed as:

$$E_{3\times3} = \frac{1}{k}\sum_{j=1}^{k}(P_j - \overline{P})(P_j - \overline{P})^T \tag{3}$$

Where $\overline{P} = \frac{1}{k}\sum_{j=1}^{k} P_j$, P_j is one of P_i's k- nearest neighboring points.

P_i^w is evaluated in Eq. (4) as follows:

$$P_i^w = \frac{\sum_{j=1}^{k} g(i)m(i)P_j}{\sum_{j=1}^{k} g(i)m(i)} \tag{4}$$

$$g(i) = e^{-\left(\frac{||P_j - P_i||}{r}\right)^2} \tag{5}$$

Considering the density difference of the point cloud, P_i in Eq. (2) can be replaced by its average point $\bar{P}_i$ in Eq. (6) to improve the anti-noise ability:

$$\overline{P_i} = \frac{\sum_{j=1}^{k} m(j)P_j}{\sum_{j=1}^{k} m(j)} \tag{6}$$

$$m(i) = \frac{1}{k}\sum_{j=1}^{k} |P_j - P_i| \tag{7}$$

where $m(i)$ is the average distance of the k (k=5 in the paper) neighbor points of point P_i. Finally, $L(i)$ can be evaluated as:

$$L(i) = ||(\overline{P}_i - P_i^w) \bullet n_i|| \tag{8}$$

3.1.2 *Density function* $D(i)$

There are two types of methods for calculating point cloud density: counting the point number in a unit radius and calculating the average distance between each pair of neighboring points[10]. In the first type of methods, the larger the number of points in one unit volume, the larger the point density is; in the second methods, the smaller the average distance, the points are denser. As $L(i)$ and $D(i)$ should be combined in this paper, the second density evaluation method is adopted in the paper. Because the average distance has been calculated in Eq. (7) and saved in implementation, Density is calculated simultaneously with the mean distance in Eq. (7). its inverse value is used directly as $D(i)$ in Eq. (9):

$$D(i) = \frac{1}{\left(\frac{1}{m}\sum_{j=1}^{m}||p_i - p_j||\right)} \tag{9}$$

where *m* is set to be 15.

3.2 Acceleration Strategy

For dense point cloud models, $F(\mathrm{i})$ will consume a large amount of computing resources, which is the algorithm's efficient bottleneck. Given that the majority of the points are non-feature points, one method should be devised to quickly filter out invalid points in order to speed up the algorithm. In the paper, the input model is voxelized as one lattice model C, i.e., voxel model, and the non-feature points can be quickly filtered through the density difference in the paper.

3.2.1 *The construction of Voxel model.*

In the paper, the construction of *C* from an input point cloud model consists four steps, which is introduced as follows:

Step 1: For the input point cloud model Φ, calculate its bounding box's upper right corner point (X_max, Y_max, Z_max), the lower left corner point (X_min, Y_min, Z_min) and the bounding box's center can be represented as Eq.(10).

$$\begin{cases} X_0 = 0.5 * (X\_max + X\_min) \\ Y_0 = 0.5 * (Y\_max + Y\_min) \\ Z_0 = 0.5 * (Z\_max + Z\_min) \end{cases} \tag{10}$$

Step 2: Calculate the average distance d of each pair of neighboring points in Φ, and set the lattice side length $a = \beta * d$, where β is a preset constant, and set to 2.5 in the paper.

Step 3: Calculate the lattice number in the *X*-axis, *Y*-axis and *Z*-axis directions by Eq. (11):

$$\begin{cases} N_x = [(X\_max - X\_min)\,/a] \\ N_y = [(Y\_max - Y\_min)\,/a] \\ N_z = [(Z\_max - Z\_min)\,/a] \end{cases} \tag{11}$$

The total lattice number is $N_{total} = N_x N_y N_z$

Step 4: Calculate the center coordinates of each lattice using Eq. (12)

$$\begin{cases} X\_center = X\_min + (i + 0.5) * a \\ Y\_center = Y\_min + (j + 0.5) * a \\ Z\_center = Z\_min + (k + 0.5) * a \end{cases} \tag{12}$$

Where i, j and k are the index numbers of the lattice in the *X*-axis, *Y*-axis and *Z*-axis directions, respectively.

3.2.2 *Filter valid points*

After voxelization, the input model Φ is transformed into be a lattice model denoted as C. The paper first traverses all the lattices of C, and counts the point number for each lattice. For one lattice $C(i)$, if any of its 26 neighboring lattice has fewer points than $C(i)$ by a preset ratio, the points in $C(i)$ will be added to the candidate feature point set Ψ; in order to avoid missing valid feature points, all the points in $C(i)$'s 26-neighboring lattices will be also added to Ψ.

Filter_Algorithm

```
Input: Point cloud model $\Phi$
Output: Candidate feature point set $\Psi$

Step 1: Use Eq.(10)-Eq.(12) to construct a voxel model $C$ for $\Phi$
Step 2:   FOR ($i$ =0; $i< N_{total}$;$i$++)
              Count the point number of $C(i)$ and the number is denoted as $N_{C(i)}$
                 Search $C(i)$'s neighboring voxel $C(j)$
              FOR each $C(j)$,
                     IF $C(j)$ is not flagged and its point number is not counted.
                          count the point number and denote it as $N_{C(j)}$
                     END IF
                     IF $(N_{C(i)} - N_{C(j)})/ N_{C(j)} > \varepsilon$
                        Add all points in $C(j)$ to $\Psi$.
                        Add $C(i)$'s all neighboring voxels' points to $\Psi$
                        Assign a flag to all $C(j)$s
                        Break;
                  END IF
              END FOR
          END FOR
Step 3:   Output all points in $\Psi$ as candidate feature points
```

Since **Filter_Algorithm** does not involve any numerical calculation, most invalid points can be quickly filtered by means of point density. Finally, $F(\Psi(i))$ is calculated for all points in Ψ, once $F(\Psi(i))$ is larger than a threshold, point P_i will be selected as one feature point.

3.3 Implementation and Results

The proposed algorithm is implemented with C++ in Visual Studio C++ 2019. The test computer has a 3.20GHz Intel processor and 8GB DDR4 memory. In order to verify the proposed algorithm's effectiveness and noise robustness, Gaussian noise with the mean zero and the standard deviation of 50% of the average distance between points[12][12]are added to three test point cloud models, i.e., Model I 、Model II and Model III. The ModelI, ModelI and Model III are synthetic. In order to compare the accuracy of feature extraction, an evaluation index P_r is proposed in the paper, which is defined as follows:

$$P_r = N_e/N_g \tag{13}$$

Where N_e is the number of exacted feature points and N_g is the ground truth.

The second evaluation index P_t is defined as below:

$$P_t = N_{re}/N_r \tag{14}$$

where, N_{re} is the number of points extracted, which lie in a preset radius r neighborhood of the corresponding theoretical feature points, and N_r is the theoretical number of points within r-neighborhood of the theoretical feature point r. In this paper, the average distance of 1 to 1.3 times is taken as the radius r.

In the test, the neighborhood search radius is set to 5 times the average distance, the *k*-nearest neighborhood value in normal vector calculation is set to 15, and the k-nearest neighborhood value in density calculation is set to 15. PCA [13], SSI [11] and the methods in [17] are selected as benchmarks. The outcomes are depicted in Figure 3, and the statistic are listed in Table 1.

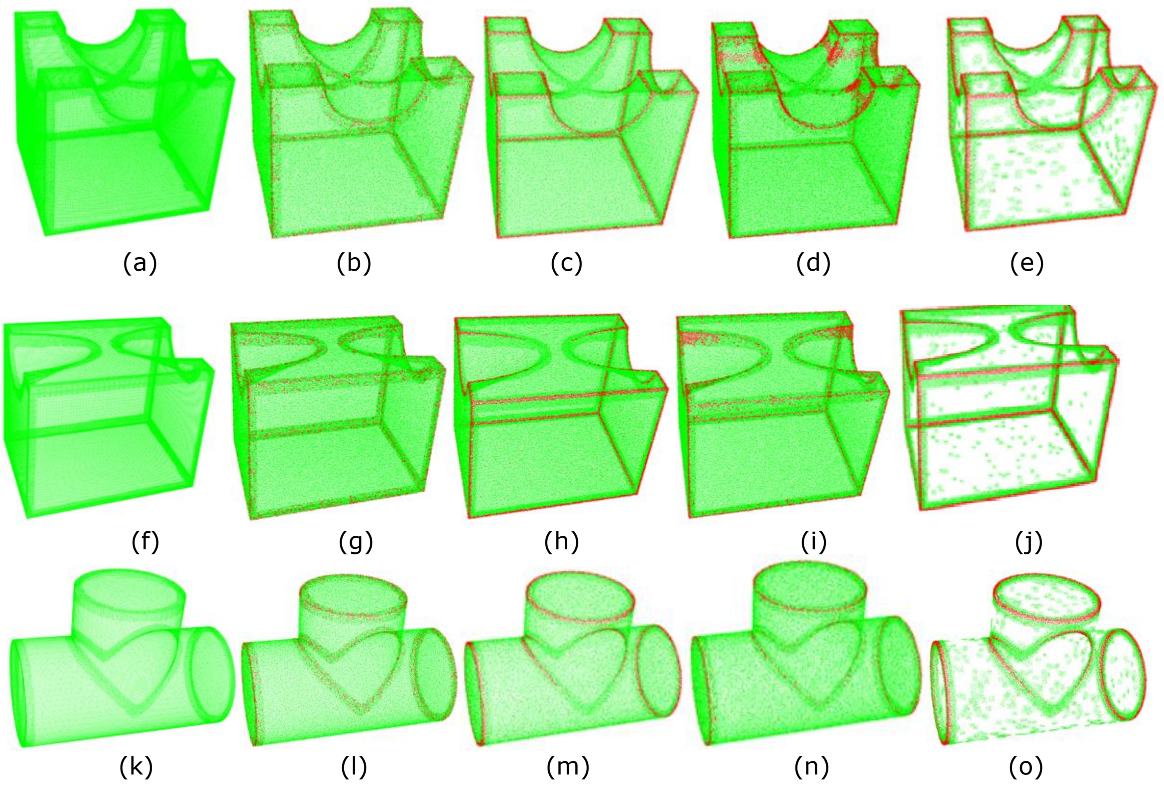

Figure 3: The results of Model I, Model II and Model III by different methods: (a)The input Model I; (b) The result by PCA; (c) The result by SSI; (d) The result by the method in [17]; (e) The result by the proposed algorithm; (f) The input Model II; (g) The result by PCA; (h) The result by SSI; (i) The result by the method in [17]; (j) The result by the proposed algorithm. (f) The input Model III; (g) The result by PCA; (h) The result by SSI; (i) The result by the method in [17]; (j) The result by the proposed algorithm; (k) The input Model III; (l) The result by PCA; (m) The result by SSI; (n) The result by the method in [17]; (0) The result by the proposed algorithm.

From Figure 3 and Table 1, we can find that the PCA method extracts many incorrect points around the correct feature points, with only 61.16% of feature points being exacted. For the SSI method, the feature points of small surface changes, such as Model I’s middle valley place and Model II’s middle places on the top flat plane, cannot be extracted. The above situation also occurs in the intersecting part of the middle surface of Model III. The approach in [17] considers the density factor as well as the differences of normal angles and curvatures, however, incorrect points are also identified as feature points. Compared to the above three methods, the proposed algorithm can detect the largest number of feature points, and it is superior than SSI in terms of smooth feature extraction. The results by the proposed algorithm performs best in terms of P_r and P_t. Essentially, the proposed method need to calculate both the SSI value and the density value, but as the voxelization idea is used to accelerate the calculation, acceptable efficiency can also be achieved. From Table 1, we can find that the proposed algorithm is 28.64% 、 31.55% and 34.63 faster than SSI in Model I 、 Model II and Model III, respectively, but can obtain more correct feature points.

Test model	Methods	Point Number	P_r (%)	P_t (%)	Time (sec.)
Model I	PCA	170K	61.16	21.44	1.637
	SSI		86.32	33.93	4.479
	The method in [17]		86.78	32.85	1.961
	Ours		91.41	36.7	3.196
Model II	PCA	220K	57.58	21.19	2.128
	SSI		84.46	34.44	5.744
	The method in [17]		85.24	33.66	2.468
	Ours		90.16	36.91	3.932
Model III	PCA	130K	62.56	17.58	1.287
	SSI		81.35	24.02	3.413
	The method in [17]		82.17	22.2	1.464
	Ours		85.52	25.37	2.231

Table 1: The statistics data on the test models, i.e., Model I , Model II and Model III.

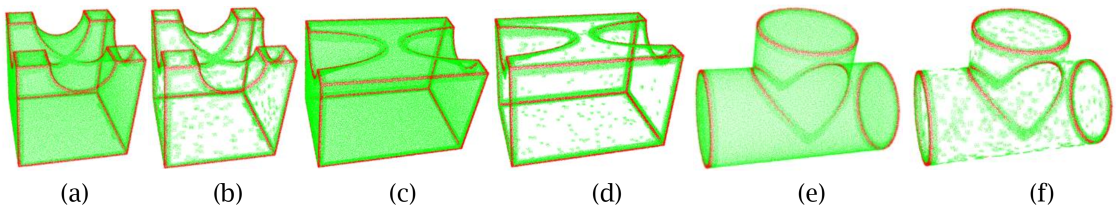
Figure 4: The results before and after acceleration using voxelization: (a) The result of Model I without acceleration; (b) The result of Model I with acceleration; (c) The result of Model II without acceleration; (d) The result of Model II with acceleration; (e) The result of Model III without acceleration; (f) The result of Model III with acceleration.

The proposed algorithm is sped up by employing voxelization and density function to filter out non feature points. This step makes senses. Figure 4 and Table 2 shows the comparison results before and after acceleration. Table 2 shows that the acceleration impact is evident, as the time for processing Mode I , Model II and Model III has been reduced by 33.95% , 34.64% and 35.98%, respectively. However, the extraction accuracy can be kept nearly the same (91.41% V.S 91.54% ; 90.17% V.S 90.24% and 85.55% V.S85.52%).

Test Model	Time(sec.) No acceleration	P_r (%) No acceleration	Pt (%) No acceleration	Time(sec.) Acceleration	P_r (%) Acceleration	Pt (%) Acceleration
Model I	4.839	91.54	36.71	3.196	91.41	36.7
Model II	6.016	90.24	37.11	3.932	90.17	36.91
Model III	3.485	85.55	25.37	2.231	85.52	25.31

Table 2: The comparing data of time cost and P_r 、 P_t with and without acceleration.

4 CONCLUSIONS

A noise-robust point cloud feature extraction algorithm on noisy point cloud is proposed in this paper. The proposed method defines a feature evaluation function which combines the smooth shrink index and the local point density index to extract feature points. In addition, voxelization is used to speed up the algorithm. The test results show that it has a good anti-noise ability, can obtain smooth feature points and detect the largest number of correct feature points compared with three benchmarks, i.e., PCA[13], SSI[11] and the approach in [17]. The time cost is also acceptable and 30-40% faster than SSI. If the cloud model is uniformly sampled and there are no density variations in feature regions, the acceleration step will fail and the whole algorithm will be the same with SSI method. The limits of the paper lie in two points: 1) the input model must have density differences at interested feature regions. More explicitly, the density of feature points near the feature area is required to be higher than that of other non-feature areas. In practical applications, adjusting the 3D camera scanning angles to scan the feature areas multiple times will take a long time; 2) using voxelization for acceleration will result in a loss of effective feature points when the features are tiny or the feature points lie in highly-curved places.

In the future work, how to use parallel computing to calculate each lattice's points' $F(i)$ can be further implemented, and how to automatically determine the parameter $a = \beta * d$ can be also investigated in the future, as this value will significantly affect the algorithm's effectiveness and performance.

5 ACKNOWLEDGEMENTS

The authors of this paper are supported by the funding of Natural Science Foundation of China (No: 61662006, 62062015) and the Innovation Project of School of Computer Science and Information Engineering, Guangxi Normal University under the contract number JXXYYJSCXXM-002 and Guangxi 100 oversea talents plan. The author of the paper is also partially supported by the Guangxi Collaborative Innovation Center of Multi-source Information Integration and Intelligent Processing.

Nanhua Huang, https://orcid.org/0000-0001-9759-6127
Ming Chen, https://orcid.org/0000-0003-0506-5308
Zhengqin Zhang, https://orcid.org/0000-0002-7878-8918
Shenglian Lu, https://orcid.org/0000-0002-4957-9418

REFERENCES

[1] Altantsetseg, E.; Muraki, Y.; Matsuyama, K.: Feature line extraction from unorganized noisy point clouds using truncated Fourier series, The Visual Computer, 29(6), 2013, 617-626.https://doi.org/10.1007/s00371-013-0800-x
[2] Cao, J.; Wushour, S.; Yao, X.; Li, N.; Liang, X.: Sharp feature extraction in point clouds, IET Image Processing 6(7),2012,863-869.https://doi.org/10.1049/iet-ipr.2011.0361
[3] Chen, L.; Cai Y,Zhang,J. S.;Xiang, B. P.: Feature point extraction of scattered point cloud based on multiple parameters hybridization method,Application Research of Computers, 34(9),2017,4. (in Chinese).https://doi.org/10.3969/j.issn.1001-3695.2017.09.067
[4] Daniels, J. I. I.; Ha, L. K.; Ochotta, T.; Silva, C. T.: Robust smooth feature extraction from point clouds, IEEE International Conference on Shape Modeling and Applications 2007 (SMI'07). IEEE, 2007,123-136.https://doi.org/10.1109/SMI.2007.32
[5] Demarsin, K.; Vanderstraeten, D.: Volodine, T.; Roose,D.: Detection of closed sharp edges in point clouds using normal estimation and graph theory, Computer-Aided Design, 39(4), 2007, 276-283.https://doi.org/10.1016/j.cad.2006.12.005

[6] Khameneifar, F.; Feng,H,Y.: Establishing a balanced neighborhood of discrete points for local quadric surface fitting, Computer-Aided Design, 84,2017, 25-38.https://doi.org/10.1016/j.cad.2016.12.001

[7] Kim, S. K.: Extraction of ridge and valley lines from unorganized points, Multimedia tools and applications, 63(1),2013, 265-279.https://doi.org/10.1007/s11042-012-0999-y

[8] Liu, Z.; Xiao, X.; Zhong, S.Wang W.;Xie,z.: A feature-preserving framework for point cloud denoising, Computer-Aided Design, 127, 2020, 102857.https://doi.org/10.1016/j.cad.2020.102857

[9] Lu, J.; Peng, Z. T.; Xia, G. H.: Point cloud registration algorithm based on neighborhood features of multi-scale normal vectors, Journal of Optoelectronics·Laser,26(04),2015,780-787. (in Chinese).https://doi.org/10.16136/j.joel.2015.04.0978

[10] Ma, C. C.; Li, S.; Cao, J. J.; Yu, M.: Feature points extraction of point cloud based on normal vector and density, Computer Applications and Software, 37(5),2020,6. (in Chinese).http://dx.chinadoi.cn/10.3969/j.issn.1000-386x.2020.05.044

[11] Nie, J.H.: Extracting feature lines from point clouds based on smooth shrink and iterative thinning, Graphical Models ,84,2016,38-49.https://doi.org/10.1007/s11704-016-6191-1

[12] Park.; Min, K; Seung, J. L.; Kwan, H. L.: Multi-scale tensor voting for feature extraction from unstructured point clouds, Graphical Models, 74(4),2012, 197-208.https://doi.org/10.1016/j.gmod.2012.04.008

[13] Pauly, M.; Keiser, R.; Gross, M.: Multi‐scale feature extraction on point‐sampled surfaces, Computer graphics forum. Oxford, UK: Blackwell Publishing, Inc, 22(3), 2003, 281-289.https://doi.org/10.1111/1467-8659.00675

[14] Ruan, X.; Liu, B.: Review of 3d point cloud data segmentation methods, International Journal of Advanced Network, Monitoring and Controls, 5(1), 2020, 66-71.https://doi.org/10.21307/ijanmc-2020-010

[15] Sun, D. Z.; Liu, H. D.; Shi, Y.; Li, Y. R.: Boundary Feature Abstraction of Unorganized Points Based on Kernel Density Estimation, Transactions of the Chinese Society for Agricultural Machinery, 44(12), 2013,275-279268. (In Chinese).https://doi.org/10.6041/j.issn.1000-1298.2013.12.046

[16] Wang, G.; Wu, L.; Hu, Y.: Point cloud simplification algorithm based on the feature of adaptive curvature entropy, Measurement Science and Technology, 32(6), 2021, 065004. https://doi.org/10.1088/1361-6501/abd497

[17] Wang, L.; Yuan, B .: Curvature and density based feature point detection for point cloud data, IET International Conference on Wireless. IET, 2011.https://doi.org/10.1049/cp.2010.0694

[18] Wang, X. H.; Chen, H. W.; Wu, L. S.: Feature extraction of point clouds based on region clustering segmentation, Multimedia Tools and Applications, 79(17), 2020, 11861-11889.https://doi.org/10.1007/s11042-019-08512-1

[19] Wang, X. l.; Sun, W. l.; Zhang, J. J.; Huang, Y.;Huang, H. B.: Review on reverse engineering research based on point cloud data , Manufacturing Technology & Machine Tool, 2, 2018,49-53. (in Chinese).https://doi.org/10.19287/j.cnki.1005-2402.2018.02.010

[20] X, Lai.; Yang, J.; Li, Y.: A Building Extraction Approach Based on the Fusion of LiDAR Point Cloud and Elevation Map Texture Features, Remote Sensing, 11(14), 2019, 1636-. https://doi.org/10.3390/rs11141636

[21] Xian, Y.; Xiao, J.; Wang, Y.: A fast registration algorithm of rock point cloud based on spherical projection and feature extraction, Frontiers of Computer Science, 13(1), 2019, 170-182.https://doi.org/10.1007/s11704-016-6191-1

[22] Zhang, Y.; Geng, G.; Wei, X.: A statistical approach for extraction of feature lines from point clouds, Computers & Graphics,56, 2016, 31-45.https://doi.org/10.1016/j.cag.2016.01.004

[23] Zhou, J. Z.; Yan, Y. J.; Chen, C.; Du, W. C.: A two-threshold point cloud feature extraction method with neighborhood adaptive, Intelligent Algorithm, 39(2), 2020,27-33. (in Chinese).https://doi.org/10.19358/j.issn.2096-5133.2020.02.006