Available online at www.ijpe-online.com

International Journal of Performability Engineering

vol. 15, no. 3, March 2019, pp. 782-791
DOI: 10.23940/ijpe.19.03.p7.782791

Algorithm for Point Cloud Compression based on Geometrical Features

Shiquan Qiao, Kun Zhang*, and Kai Gao

School of Information Science and Engineering, Hebei University of Science and Technology, Shijiazhuang, 050018, China

Abstract

As a common and important form, point cloud data exists in computer graphics, especially for 3D visualization. However, with the development of 3D scanning technology, huge data sets have become a main burden in the data processing of point clouds. Therefore, the technology of point cloud compressing is a key content in data pre-processing. This paper provides a new algorithm to compress the point cloud data set. The compressing algorithm can be carried out based on the feature of measure objects. In order to find the data feature, we firstly introduce a point cloud compressing model based on conicoid according to the measure objects. Secondly, for the comparison of the features between the model and the point cloud, we provide a shape operator and a contour operator based on the estimation of geometrical features. Then, according to the value of the shape operator and the contour operator, we provide a matching model. The compressing data algorithm can be created through the matching computation of geometrical features. At last, we use the experiment to prove the feasibility of compressing algorithm, and compare the result of the proposed algorithm and the result of other algorithms in terms of the running time and the compressing effect.

Keywords: point cloud compressing; conicoid; geometrical features; contour feature; matching model

(Submitted on October 23, 2018; Revised on November 24, 2018; Accepted on December 27, 2018)

1. Introduction

With the development of 3D scanning technology, many portable devices, such as Microsoft's Kinect, HoloLens, and Intel's RealSense, have emerged. The dense point cloud of objects can be obtained at a lower cost. Taking Kinect (V2) as an example, data can be collected at 12MB per second. Obviously, dense data shows more details on the surface of measure objects. However, the dense data produces a huge data volume. The massive amounts of data produce unimaginable pressure on 3D processing, such as reverse reconstruction of objects. Therefore, the compression of point cloud data is necessary for point cloud pre-processing.

In order to compress the data volume, many algorithms have been provided so far. Among these algorithms, compressing the data by the voxel method is the most widely used, especially in engineering. The voxel algorithm has a compelling advantage for efficiency. Unfortunately, the voxel algorithm can introduce an influence on the quality of the data. Therefore, in order to obtain higher quality with shorter running time, research on the point cloud compressing algorithm has been carried out in recent years.

This paper provides a new compressing data algorithm based on surface feature estimation. The contributions are as follows: (1) the geometrical features of point data are described through the model, and then the theory that can estimate the value of the geometrical features is presented. (2) A compressing data model is created based on geometrical features of point data, and a compressing data algorithm is provided. (3) Experiments are provided to prove the feasibility of the compressing algorithm, and a comparison is made with other algorithms for the running time and the effect.

The rest of the paper is organized as follows. In Section 2, we introduce the point cloud compressing algorithms and the algorithms classified into four categories. In Section 3, we present the related theory of point cloud compressing models based on geometrical features. In Section 4, according to the features of different point cloud compressing models, a

* Corresponding author.
E-mail address: euphkun@163.com

compressing data algorithm is provided. Section 5 provides experiments to compare the algorithm proposed in this paper with other algorithms. We conclude the whole paper in Section 6.

2. Related Work

Facing the problem of huge point cloud volume, researchers have provided many methods to compress the data in order to increase the data processing speed. However, in order to protect the features of objects measured and reduce the redundant data, the feature extraction technique is applied in the data compressing algorithm. In this paper, according to the data attributes that the compressing algorithms rely on, the compressing algorithms are classified into four categories.

(1) The first kind of compressing algorithm is to compress data via the division of measure space. Compared to other kinds of compressing algorithms, this kind of method is adopted frequently. Reference [1] provides an algorithm to compress data volume by creating the index of point cloud. In order to create the data index, the Octree algorithm has been introduced. The scale of the data is chosen interactively by the user. However, the interactive operation reduces the algorithm efficiency. Similar to in reference [1], data using the partition space is reduced in [2-3]. The Octree algorithm is the classical subspace division method in computer vision; based on the Octree algorithm, the measure space is divided into many voxels, and then the data is reduced in each voxel. However, the Octree compressing algorithm does not take the object's surface features into account. The features of the object are lost during compression. Another method can obtain the division of measure space using the cluster algorithm. The sub-spaces are created using the point cloud cluster, and then the compressing algorithm can be implemented in each cluster [4-8]. Reference [6] divides the sub-space using the K-means algorithm according to the distance of raw point cloud. Then, the points with low enough distance values are chosen and deleted. Reference [7] reduces point cloud by k-means based on directed Hausdorff distance, and the surface edges are protected well. The cluster process is complex and time-consuming, and this influences the efficiency of the compressing algorithm.

(2) The second one is to compress data with the help of location information. This kind of compressing algorithm is usually used in the remote sensing field. Using the auxiliary GPS information, the volume of data can be compressed [9-11]. However, using these compressing algorithms, the retrieval of location information would produce extra costs.

(3) The third and most important method compresses data according to the geometric feature. Using these methods, the better effect can be achieved in the reverse engineering field, so it has drawn more attention from researchers in recent years. Reference [12] provides point cloud compressing methods on the basis of panorama images. The data is projected into different geometry structures, and the data volume can be compressed a lot based on the limited geometry. Reference [13] proposes a new approach to solve the compressing problem by restricting the control points and obtain a more intuitive location of the control points through Bezier curves with respect to the least squares norm. Reference [14] proposes a 3D lossy compression system based on plane extraction, which represents the points of each scene plane as Delaunay triangulation and a set of points/area information. Reference [15] presents a method for incremental planar segmentation of a gradually expanding point cloud map and another method for efficient triangulation texturing of planar surface segments, and then it uses the segment methods to exploit the richness information of planar surfaces. However, the computing of the geometric feature is time-consuming, and this influences the system efficiency of point cloud processing. Reference [16] reduces point cloud by the energy function of curvature.

(4) The last kind of method is to compress data using the extra attributes with the processing of point cloud data retrieval; for example, find the similar information according to the color of point cloud or the intensity of reflected light, and reduce the similar and redundant information in the raw point cloud [17-20].

3. The Model of Point Data Compressing based on Geometrical Features

In the process of reverse reconstruction of the point cloud, because of the scatter data, it is difficult to find the relationship between the data sets and estimate the geometric feature of the point cloud. In different application fields, there are different focuses on geometrical features. Therefore, there are many descriptions on the geometrical features for point cloud, for example, normal, curvature, flat, peak, cylinder, minimal surface and ridge, etc. Among them, the normal and the curvature are used to measure the direction and curve of the surface. Local geometric feature estimation is an important technology. The accuracy of geometric feature directly determines the effect of point cloud reverse reconstruction and the precision of the measuring result.

In order to find the point cloud data feature during the process of data compressing, this paper provides a data compressing model based on conicoid. By matching the computation between the raw point cloud and the data compressing model, the data volume can be reduced by a certain geometrical feature, while the effective data feature can be reserved.

Because of the accuracy and the control ability of conicoid, we create the data compressing model based on conicoid. Assume that there is a point data set $P(x, y, z)$, and $P(x, y, z)$ belongs to the raw point cloud *points*, that is,

$$f(x, y, z) = C_0 + C_1x + C_2y + C_3z + C_4x^2 + C_5xy + C_6xzC_7y^2 + C_8yz + C_9z^2 \tag{1}$$

If the point *P* can satisfy Equation (1), the point *P* is considered to be one element in the data compressing model *CM*.

Because of the geometrical feature by which we estimate the local data sets, the whole conicoid model is too large. The matching operation between model and data sets would consume more time. Therefore, we revise and simplify the *CM*. For convenience of algorithm implementation, in this paper, we simplify the data compressing model based on Nature Point Cloud Model (NPM).

The NPM model includes ball, ellipsoid, cylinder, and cone. The plane is an important and frequently used model in reconstruction. The plane satisfies Equation (2):

$$\begin{aligned} &f(x, y, z) = ax + by + cz + d \\ &\Rightarrow f(x, y, z) = C_0 + C_1x + C_2y + C_3z \end{aligned} \tag{2}$$

Therefore, in this paper, the plane and the NPM can be considered the basic models in the data compressing model, *CM*. Using the NPM and the plane, the *CM* can be simplified considerably. Then, the *CM* becomes the feasible for the compressing data algorithm. Based on the *CM*, the similar and the redundant data can be retrieved, and then the compressing algorithm can be implemented by reducing the data with similar geometry features. Next, for exploiting the points/area with similar geometry feature, the matching operator has been provided.

3.1. The Matching Operator-based on the Shape Feature

In order to measure the similarity between the model and data sets in geometric features, we provide a matching operator in the process of compressing. In this paper, we compute the similarity through the comparison of the shape and outlier between the model and data sets. Therefore, the definitions of the shape operator and outlier operator have been provided.

Definition 1 (shape operator) Denote the point cloud data sets as points and the data compressing model as *CM*, and then the shape measurement between points and *CM* can be computed as shown in Equation (3):

$$shape(CM, points) = \frac{1}{\sqrt{2\pi}\sigma} \exp\left(-\frac{\left\|\phi(points) - \phi(CM)\right\|^2}{2\sigma^2}\right) \tag{3}$$

Where σ is the scale of data sets and ϕ is the function for the estimation value of shape features.

According to the shape operator, we can compute the value of similarity between the data sets and *CM*. In other words, the point, which is on the surface of *CM*, is necessary for shape operator computing. In Definition 1, the function ϕ, the estimation value of the shape feature, is important. The function shows the shape information, and the value of $\emptyset$ is proportional to the shape feature.

The normal vector is an important parameter describing the direction of the tangent plane, and the direction of the curved tangent plane is very important to describe the shape of the object. In this paper, the three orthogonal directions can be used to analyze the normal vector, and the value of the normal vector has been decomposed into three orthogonal directions that can be used to quantify the shape feature, as shown in Equation (4):

$$\phi = \sum (n(x) \cdot n(N_x) + n(y) \cdot n(N_y) + n(z) \cdot n(N_z)) \tag{4}$$

In Equation (4), *n* represents the normal estimation of point; *x*, *y*, and *z* are the three orthogonal directions; and $N_i (i = x, y, z)$ is the neighbor of data sets in the i^{th} direction.

3.2. The Matching Operator based on the Contour Feature

In order to compute the similarity of contour feature of data sets, we estimate the contour feature by the value of curvature. In this paper, the shape feature has been considered by the shape operator. In order to find the most similarity model, we compare the similarity through the other prominent feature in terms of the contour of the data. We will analyze the contour feature using effective project data. For the accuracy results, we have to find an effective project direction, and then the projective data can provide the contour feature similar to the measured object. Firstly, we find the suitable local data sets. According to the PCA, the principal component direction vector $(V_1, V_2, V_3)^T$ can be computed. The covariance matrix C is created by the point cloud P, and the center of P is $\bar{p}$. Then, the C is decomposed by eigendecomposition, and the principal component direction vector $(V_1, V_2, V_3)^T$ can be estimated.

The data center of data set $\bar{p}$ is given as Equation (5):

$$\bar{p} = \frac{1}{n}\sum_n p_i \tag{5}$$

The matrix C is the eigenvectors of the data set p_i, as shown in Equation (6):

$$C = \frac{1}{k}\sum_{i=1}^{k}(p_i - \bar{p})(p_i - \bar{p})^T \tag{6}$$

By the singular value, matrix C can be decomposed as shown in Equation (7):

$$C = [V_1, V_2, V_3]\begin{bmatrix}\lambda_1 & & \\ & \lambda_2 & \\ & & \lambda_3\end{bmatrix}\begin{bmatrix}V_1 \\ V_2 \\ V_3\end{bmatrix} \tag{7}$$

Assume that the local data sets have three effective directions. Based on the PCA algorithm, the three principal component direction vectors can be described as $(V_1, V_2, V_3)^T$, as shown in Figure 1.

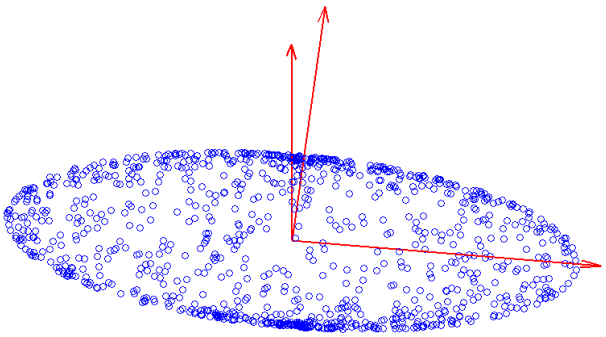


Figure 1. The vector directions of the three principal components

Then, the data set can be projected on the three principal directions as

$$\lambda p_j = (x_j \cdot V_i, y_j \cdot V_i, z_j \cdot V_i)^T,\ i = 1,2,3 \tag{8}$$

In Equation (8), p_j represents the point $p_j(x_j, y_j, z_j)$ and V_i represents the principal directions.

Definition 3 (contour operator) $outline(points, M)$. Denote point cloud data sets as points and the data compressing model as *CM*, and then the contour measurement between points and *CM* can be computed as follows.

$$outline(M, points) = \left\| K(\lambda P) - K(CM') \right\|_i, \ i = 1,2,3 \tag{9}$$

In Equation (9), CM' is considered as the effective projection of *CM*, λP as the feature points in the original data set points, *K* as the curvature function, and *i* as the index of the three directions.

Inspired by the Scale Invariant Feature Transform (SIFT), we evaluate the projection operation feature throng the local extreme. The scale space of the projection data can be described as Equation (10):

$$G(\lambda P_{jx}, \lambda P_{jy}, \sigma) = \frac{1}{2\pi\sigma^2} \exp\left(-\frac{\lambda P_{jx}^2 + \lambda P_{jy}^2}{2\sigma^2} \right) \tag{10}$$

In Equation (10), the variable σ represents the scale that you have chosen.

The outlier feature can be computed by the differential operation as shown in Equation (11):

$$\begin{aligned} D(\lambda P_{jx}, \lambda P_{jy}, \sigma) &= (G(\lambda P_{jx}, \lambda P_{jy}, k\sigma) - G(\lambda P_{jx}, \lambda P_{jy}, \sigma)) * I(\lambda P_{jx}, \lambda P_{jy}) \\ &= L(\lambda P_{jx}, \lambda P_{jy}, k\sigma) - L(\lambda P_{jx}, \lambda P_{jy}, \sigma) \end{aligned} \tag{11}$$

The operation of convolution is described as *. In Equation (11), the origin projection data set can be presented as $I(\lambda P_{jx}, \lambda P_{jy})$.

Definition 4 (matching operator) Denote the model as $M = \{m_1, m_2, m_3, m_4, m_5\}$, the data set as $D = \{d_1, \cdots, d_n\}$, the candidate set as $\psi = \{d_1, \cdots, d_m\}$, $\psi \subseteq D$, and *s* as the scale parameter to the data set. Then, the data set satisfying the following rules can compose the point cloud matching model.

- Pattern $M = \{\psi\}$, and the candidate set is not an empty set;
- Pattern $M \in CM(s)$, when the scales parameter has the certain value, the cloud matching model is a finite set;
- $shape(\psi, \text{Pattern } M) = \max(shape(M, D))$;
- $outline(\psi, \text{Pattern } M) = \min(outline(M, D))$.

4. The Compressing Algorithm based on the Matching Model

The point cloud compressing algorithm based on matching model is divided into three steps.

Step 1 The first step is the initialization of the point cloud compressing model. Create the projections of the data set on the model *CM* and estimate the scales parameter of the *CM* according to the boundary of projections of the data set.

Step 2 The second step is to compute the matching model Pattern *M*. Create the projection of the data set on the Pattern *M* and compute the matching coefficients of data and Pattern *M* according to the shape operator and contour operator.

Step 3 The third step is to compress the data according to the feature of Pattern *M*.

Figure 2 is the program flow chart of the compressing algorithm based on the matching model. In the algorithm, the shape and contour operations are executed for estimating the similarity, and the variable *coefficients* can store the similarity between the data and the model. The variable *scale* stores the parameters of *CM*.

The shape operator is carried out by updating the parameter of Pattern *M*, improving the similarity of the measure object and the Pattern *M* on shape measure. The algorithm can be described as Algorithm 1. The normal vector of the point cloud data is computed by the PCA algorithm, and then the shape operator between the model and the data set can be calculated using Equation (3). According to the shape operator, the similarity of the measure object and Pattern *M* can be measured. At last, update the parameter of Pattern *M* and improve the similarity of the measure object and Pattern *M*.

Algorithm 1 The shape operation algorithm

Input:
1) viewpoints
2) *points* //original data set
3) Pattern *M* //candidate set

Output: newPattern *M*//the update *M*

1. set Viewpoint;
2. Compute Covariance Matrix(*points*, xyz_centroid,covariance_matrix);//PCA
3. vector (Normal)//compute normal of data set
4. do
5. {shape(*Points*, Pattern *M*); //shape operator
6. while(Pattern $M \neq \phi$){
7. Update(Pattern *M*(*Scales*)); //update the scales parameter
8. $\Delta Pattern\ M$; }
9. }Until(ΔPattern $M < \varepsilon$)
10. Return (Pattern *M* , *Scales*, *Coefficients*)
11. **End**

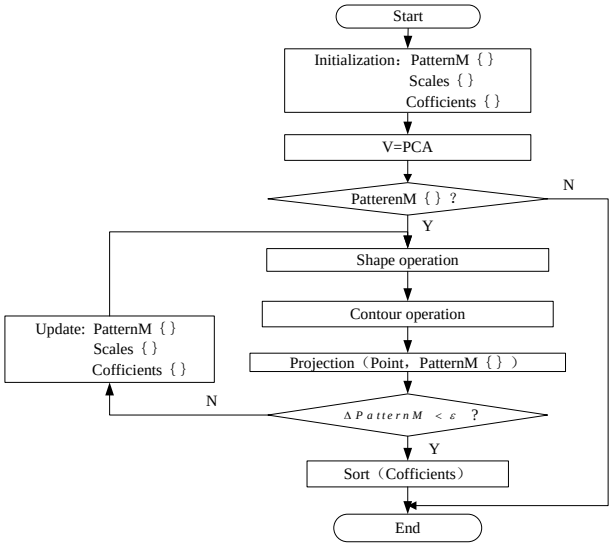


Figure 2. The program flow chart of the compressing algorithm based on the matching model

The contour matching algorithm can be executed using the contour operator based on Equation (9). The compressing model can be found by updating the parameters of the model, to improve the similarity model and the data on the contour. The contour operator can be carried out by PCA and the SIFT algorithm, and then the similarity between the model and the data can be measured. Choose the k-neighbor data in the projection of the data set, compute the curvature of the point cloud and the data model, get the coefficients based on the contour operator, and resort the Pattern *M* according to the coefficient values to find the most similar data model. If you cannot find the similar model, then decompose and update Pattern *M* iteratively until it is found.

Algorithm 2 The contour operation algorithm

Input: *points*
Output: Pattern *M*

1. Computer Covariance Matrix(*points*, *xyz_centroid*, *covariance_matrix*);//PCA
2. for each $covariance\_matrix[i]$ //*i*=1,2,3
3. {
4. Project(*points*, *covariance_matrix*[*i*]);
5. SIFT(σ, Project(*points*, *covariance_ matrix*[*i*]));
6. do
7. {
8. find(neighbor);
9. curve(neighbor);
10. outline(Pattern M);
11. while(Pattern $M \neq \phi$)
12. {
13. if($\Delta Cofficients = \text{true}$);
14. Update(Pattern *M* (scales){});
15. ΔPattern *M*;
16. }

17. } Until(ΔPattern $M < \varepsilon$)
18. }
19. Return (Pattern *M*,Scale,Cofficients);
20. **End**

5. Experiment Results and Experiment Analyses

We conduct experiments using the compressing point cloud algorithm provided in this paper, as well as other algorithms on the data sets "Stanford Bunny" and "Fandisk", which come from the Stanford 3D scanning repository.

The experiments aim at illustrating the performance of the compressing point cloud algorithm provided in this paper from the following aspects: (a) the quality of the surface with the compressing algorithm, (b) the running time compared to other algorithms. All experiments are performed on Intel (R) Core (TM) i7-3520M CPU @2.90GHZ, 8G memory, and Windows 7. All the codes are written in VC++.

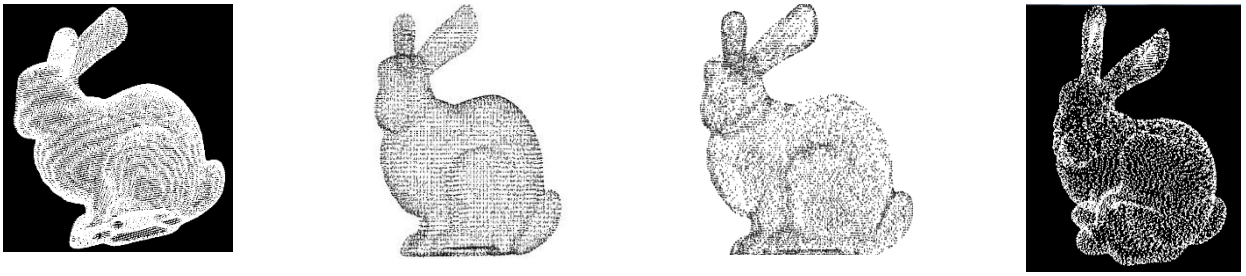
Firstly, the compressing algorithm provided in this paper is carried out on simply point data, and the result of the reduced point is shown in Figure 3. Figure 3(a) is the raw point cloud, and Figure 3(b) is the compressing data by the compressing algorithm provided in this paper. Figure 3(b) shows that the compressing ratio is different according to the different shapes matched. For the area with abundant features, the compressing ratio is lower, and the feature can be saved in the compressing data.

(a) The raw point data generated randomly (b) The result of the reduced point data

Figure 3. The adaptive compressing ratio result for uneven objects

5.1. The Comparison of Compressing Algorithm Results by the Smooth Data Set

Then, the voxel compressing algorithm, the angle of normal compressing algorithm, and the compressing algorithm provided in this paper are carried out with the same data set "Stanford Bunny". The raw data sets are visualized in Figure 4 (a). Compared to the raw data set, the three compressing data sets can be seen with the similar compression ratio. For the voxel compressing algorithm, the compressed data size is 7190; for the vector angle compressing algorithm, the compressed data size is 9490; and for the compressing algorithm provided in this paper, the compressed data size is 7477. However, the effect of surface using the three compressing data sets is different.



(a) The bunny original point cloud (35947) (b) Voxel compressing algorithm (7190) (c) The vector angle compressing algorithm (9490) (d) Compressing algorithm provided in this paper (7477)

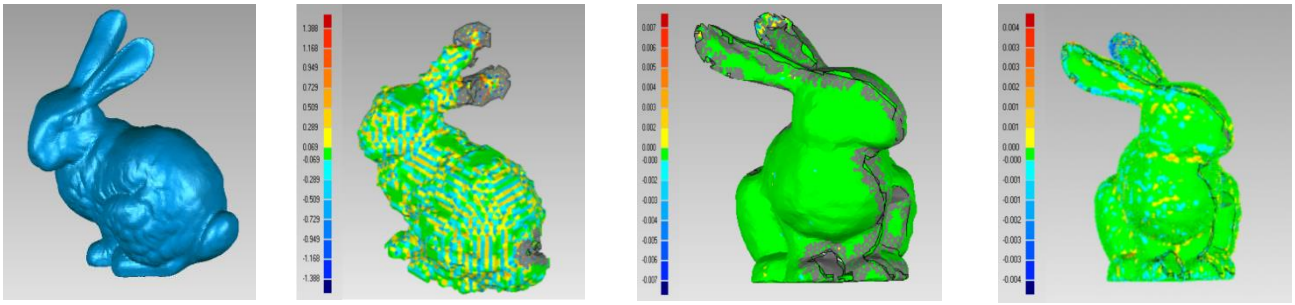
Figure 4. The comparison of the compressing algorithm using the bunny data set

In order to contrast the effect of compressing data, the surfaces will be reconstructed by different compressing algorithms based on the compressing data set and the raw data set. We create projections from the surface of the compressing data set to the surface of the raw data set. Through the projection graph, we compute the distance of the two surfaces by Geomagic Studio, shown in Figure 5(b)-(d). In Figure 5(b)-(d), the different colours represent the deviation value of the surface from the original surface. In contrast to the surface based on the raw data set, the surface based on the compressing data set provided in

this paper has the error range (-0.004, 0.004), the surface using the voxel compressing algorithm produces the error range (-1.388, 1.388), and the surface using the angle between the normal compressing algorithm produces the error range (-0.007, 0.007). Therefore, from the effects of perspective, the experiment shows that the compressing algorithm provided in this paper is better than the others. This means that the features of surface are better reserved.

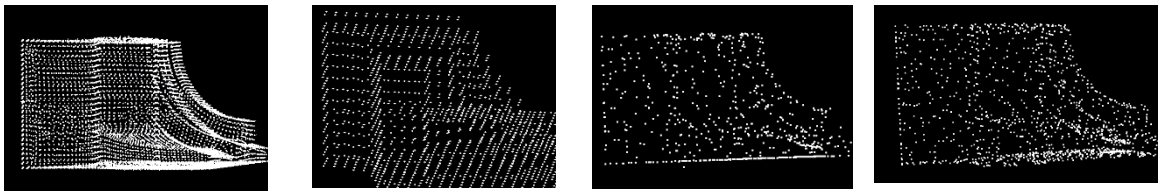
5.2. The Comparison of Compressing Algorithm Results by the Sharp Data Set

In order to compare the algorithm effect on the sharp data set, we implement the compressing data experiment on the Fandisk data set. The Fandisk data size is 6476, and the point cloud view can be seen in Figure 6(a). We use the Marching cube reconstruction algorithm and get the reconstruction surface, which can be seen in Figure 7(a). Compared to the raw data set, the three compressing data sets have similar compression ratios. For the voxel compressing algorithm, the compressed data size is 2212; for the vector angle compressing algorithm, the compressed data size is 2305; and for the compressing algorithm provided in this paper, the compressed data size is 2303.



(a) The reconstructed surface of original point cloud (b) The compressing result of voxel compressing algorithm (c) The compressing result of the vector angle compressing algorithm (d) The compressing result of the algorithm provided in this paper

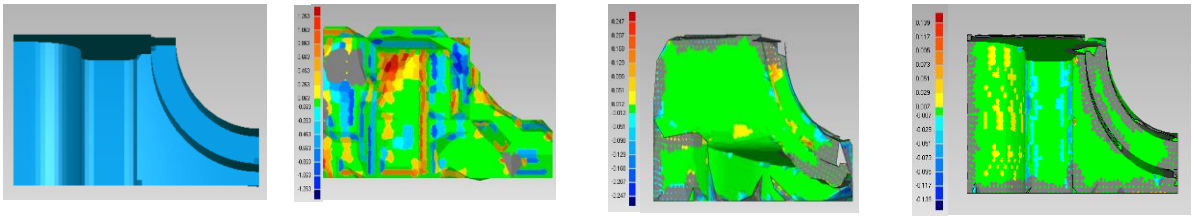
Figure 5. The comparison of the compressing result after reconstruction using the bunny data set



(a) The original point cloud (b) Voxel compressing algorithm (c) The vector angle compressing algorithm (d) Compressing algorithm provided in this paper

Figure 6. The comparison of the compressing algorithm using the fandisk data set

Through the projection graph of Fandisk, which shown as Figure 7, the surface based on the compressing data set provided in this paper has the error range (-0.139, 0.139); the surface using the voxel compressing algorithm produces the error range (-1.263, 1.263); the surface using the angle between normal compressing algorithm produces the error range (-0.247, 0.247). So, from the effects of perspective, the experiment shows the compressing algorithm which provided in this paper is better than the others. As Figure 7 shows that the algorithm which provided in this paper has better effect on the surface with sharp feature.



(a) The reconstructed surface of original point cloud (b) The compressing result of voxel compressing algorithm (c) The compressing result of the vector angle compressing algorithm (d) The compressing result of the algorithm provided in this paper

Figure 7. The comparison of the compressing result after reconstruction using the fandisk data set

We have compared the running time of the compressing algorithm and the other compressing algorithms, the vector angle compressing method, and the compressing method based on the voxel compressing algorithm. The running time is shown in Figure 8. According to the graph, we have observed that the running time of the compressing algorithm provided

in this paper is longer than that of the bounding box compressing algorithm. The reason is that the compressing algorithm provided in this paper is based on the matching model; the execution time of the algorithm in the process of model matching is longer, while the voxel compressing algorithm is based on grid and its execution efficiency is higher.

In the vector angle method experiment, data is selected with the neighbourhood radius of 0.025. The vector angle method is compact in calculating the vector dot product of the neighbourhood data normal vector and the average normal vector. Because both the normal vector estimation process and the dot product operation are time-consuming, the vector angle method performs longer than the algorithm in this paper and the voxel compressing algorithm. However, according to Figure 8, we have found that the running time of the compressing algorithm provided in this paper is shorter than that of the vector angle compressing experiment, especially when the reduced ratio is higher.

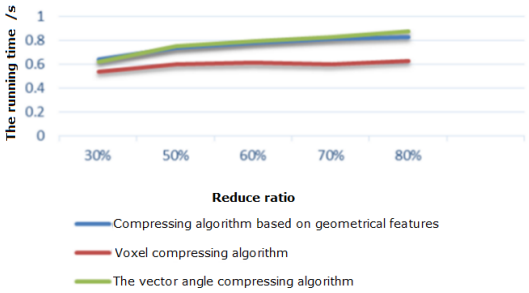
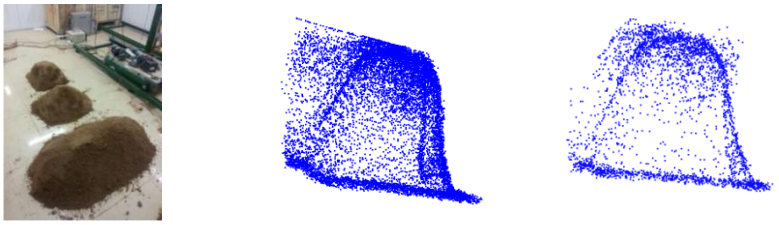


Figure 8. The running time of the algorithm

5.3. The Algorithm Effect based on the Noise Data Set

In order to test the robustness of the compressing algorithm, we measure the point data of piles using the SICK LMS100. The real view of the measured object is shown in Figure 9(a). We extract one pile's point cloud, as shown in Figure 9(b). Through the compressing algorithm, the point cloud can be reduced at the compression ratio 68.84%, and the result can be seen in Figure 9(c).



(a) Figure of measured object (b) The data before performing reduced algorithm (c) The data after performing reduced algorithm (68.84%)

Figure 9. The compressing result using the measured data set

6. Conclusions

With the progress of scanner technology, the increasing volume of data brings many problems in point cloud processing. Therefore, compressing data sets is an important method during pre-processing. To balance effectiveness and accuracy, we provided a point cloud compressing algorithm based on geometrical features. In order to find the feature of the measured object, we provided a compressing model, CM. Through the matching operation, we can measure the features of data sets, and eventually, the reserved data and the reduced data can be identified. We provided experiments to prove the feasibility and effect of the compressing algorithm. In this paper, we carried out the experiment on one computer; we deduce that the algorithm can be completed in parallel environments. We will use this algorithm on data sets by the other 3D scanner in the future, especially using the portable devices Kinect and RealSense.

Acknowledgements

This research is supported by the National Natural Science Foundation of China (No. 51271033), Natural Science Foundation of Hebei Province of China (No. 2018208116), Higher School Science and Technology Research Project of Hebei Province of China (No. Z2017008), and PhD Early Development Program of Hebei University of Science and Technology. The authors also gratefully acknowledge the helpful comments and suggestions of the reviewers, which will improve the presentation.

References

1. S. Song, J. Liu, and C. Yin, "Data Reduction for Point Cloud using Octree Coding," in *Proceedings of International Conference on Intelligent Computing*, pp. 376-383, Springer, 2017
2. R. Schnabel and R. Klein, "Octree-based Point-Cloud Compression," in *Proceedings of Eurographics Symposium on Point-Based Graphics*, pp. 111-120, 2006
3. Q. Xie and X. Xie, "Point Cloud Data Reduction Methods of Octree-based Coding and Neighborhood Search," in *Proceedings of 2011 International Conference on Electronic and Mechanical Engineering and Information Technology* (*EMEIT*), pp. 3800-3803, 2011
4. Y. Wang, J. Tang, Q. Rao, and C. Yuan, "High-Efficient Point Cloud Simplification based on the Improved K-Means Clustering Algorithm," *Journal of Computational Information Systems*, Vol. 11, No. 2, pp. 433-440, 2015
5. H. Benhabiles, O. Aubreton, H. Barki, and H. Tabia, "Fast Simplification with Sharp Feature Preserving for 3D Point Clouds," in *Proceedings of International Symposium on Programming and Systems*, pp. 47-52, 2013
6. B. Q. Shi, J. Liang, and Q. Liu, "Adaptive Simplification of Point Cloud using K-Means Clustering," *Computer-Aided Design*, Vol. 43, No. 8, pp. 910-922, 2011
7. T. Li, Q. Pan, L. Gao, and P. Li, "A Novel Simplification Method of Point Cloud with Directed Hausdorff Distance," in *Proceedings of IEEE International Conference on Computer Supported Cooperative Work in Design*, pp. 469-474, 2017
8. X. C. Yuan, W. U. Lu-Shen, and H. W. Chen, "Feature Preserving Point Cloud Simplification," *Optics & Precision Engineering*, Vol. 23, No. 9, pp. 2666-2676, 2015
9. W. Zhang and J. Kosecka, "Image based Localization in Urban Environments," in *Proceedings of International Symposium on 3d Data Processing, Visualization, and Transmission*, pp. 33-40, 2006
10. K. Ni, A. Kannan, A. Criminisi, and J. Winn, "Epitomic Location Recognition," *IEEE Transactions on Pattern Analysis & Machine Intelligence*, Vol. 31, No. 12, pp. 2158, 2009
11. E. Kalogerakis, O. Vesselova, J. Hays, and A. A. Efros, "Image Sequence Geolocation with Human Travel Priors," in *Proceedings of IEEE International Conference on Computer Vision*, pp. 253-260, 2010
12. H. Houshiar, D. Borrmann, J. Elseberg, and A. Nüchter, "Panorama based Point Cloud Reduction and Registration," in *Proceedings of International Conference on Advanced Robotics*, pp. 1-8, 2013
13. P. Gospodarczyk, "Degree Reduction of Bézier Curves with Restricted Control Points Area," in *Proceedings of International Conference on Computer Supported Cooperative Work in Design*, pp. 649-656, 2011
14. V. Morell, S. Orts, M. Cazorla, and J. Garcia-Rodriguez, "Geometric 3D Point Cloud Compression," *Pattern Recognition Letters*, Vol. 50, No. C, pp. 55-62, 2014
15. T. Whelan, L. Ma, E. Bondarev, P. H. N. D. With, and J. Mcdonald, "Incremental and Batch Planar Simplification of Dense Point Cloud Maps," *Robotics & Autonomous Systems*, Vol. 69, No. C, pp. 3-14, 2015
16. G. Shi, X. Dang, and X. Gao, "Research on Adaptive Point Cloud Simplification and Compression Technology based on Curvature estimation of Energy Function," *Revista de la Facultad de Ingenier á UCV*, Vol. 32, pp. 336-343, 2017
17. H. S. Park, Y. Wang, E. Nurvitadhi, J. C. Hoe, Y. Sheikh, and M. Chen, "3D Point Cloud Reduction using Mixed-Integer Quadratic Programming," in *Proceedings of IEEE Conference on Computer Vision and Pattern Recognition Workshops*, pp. 229-236, 2013
18. H. Houshiar and A. Nüchter, "3D Point Cloud Compression using Conventional Image Compression for Efficient Data Transmission," in *Proceedings of 2015 XXV International Conference on Information, Communication and Automation Technologies* (*ICAT*), pp. 1-8, IEEE, 2015
19. H. Kim, B. Liu, C. Y. Goh, S. Lee, and H. Myung, "Robust Vehicle Localization using Entropy-Weighted Particle Filter-based Data Fusion of Vertical and Road Intensity Information for a Large Scale Urban Area," *IEEE Robotics and Automation Letters*, Vol. 2, No. 3, pp. 1518-1524, 2017
20. X. Zhang, W. Wan, and X. An, "Clustering and DCT based Color Point Cloud Compression," *Journal of Signal Processing Systems*, Vol. 86, No. 1, pp. 41-49, 2017

Shiquan Qiao received his Master's degree in test measurement technology and instruments from Hebei University of Science and Technology in 2011. He works as a lecturer in the School of Information Science and Engineering at Hebei University of Science and Technology. His main research interests include the research and implementation on image processing and algorithm analysis.

Kun Zhang received her Ph.D. in circuits and systems and her Master's degree in software theory from Yanshan University in 2016. She works as a lecturer in the School of Information Science and Engineering at Hebei University of Science and Technology. She is also a member of the China Computer Federation (CCF). Her main research interests include computer graphics, machine learning, and applications of intelligent algorithms.

Kai Gao received his Ph.D. from Shanghai Jiaotong University. He is a professor in the School of Information Science and Engineering at Hebei University of Science and Technology. He is a senior member of the China Computer Federation (CCF) and a member of Chinese Information Technology (CCF TCCI) & Computer Applications (CCF TCAPP). He is also an associate editor of the EI Compendex indexed UK journal "International Journal of Computer Applications in Technology". His research interests include artificial intelligence, natural language processing, social network computing, web information retrieval, and big data mining.