A 3D model Compression Method for Large Scenes

Ying Zhou[a], Lingling Wang[a], Lieyun Ding[a], Cheng Zhou[a]

[a]School of Civil Engineering and Mechanics, Huazhong Univ. of Science and Technology, Wuhan, Hubei 430074, China.

E-mail: ying_zhou@hust.edu.cn, 1090196533@qq.com, dly@ hust.edu.cn, chengzhou@hust.edu.cn

Abstract –

Three-dimensional (3D) models are increasingly becoming an important tool in project management. At the same time, the 3D data of construction industry is more and more large, whether it is caused by large model scene or the requirement of model accuracy. In order to meet the requirements of the model application, the 3D model needs to be simplified. Mesh models are one of the most common ways to display 3D models, which provide well-defined object boundaries. The simplified algorithm of the existing mesh model cannot keep the important detail features of the original model once the data volume of the model is further reduced. Combined with Quadric Error Metrics (QEM) algorithm, this paper improves the edge collapse algorithm to solve the problem of undetected model structure and over-simplified model details. The proposed algorithm can preserve the integrity of the model while maintaining a high compression efficiency. Experiments show that the proposed method works well for the compression of large-scale scene models, which includes the compression ratio and the structural preservation in the model. This compression method is well suited for the model with huge amounts of data generated by large scene objects in the construction field.

Keywords –

Mesh model; Simplification; Structure-preserving; Edge collapse

1 Introduction

Three-dimensional (3D) models are increasingly becoming an important tool in project management. 3D model can reproduce building scenes and display building structures, which plays a significant role in construction quality, construction progress, construction site safety [1] [2], such as dimensional measurements of structures [3], remote management of the construction site [4].

At present, in the field of architecture, engineering and construction (AEC), polygon mesh models are one of the most common ways to display 3D models. They provide well-defined object boundaries that are suitable for clearly presenting the structure of 3D objects, which greatly improve the visualization of the construction site. These models have been able to achieve centimeter-level display of the details of the object [5]. In general, a polygon mesh model usually consists of complex triangular mesh, which may contain tens or even millions of vertices and polygons [6]. In particular, there are more vertices and polygons for some models of large scenes. There are higher requirements for grid model applications, including adequate storage capacity, high computational power, and to access over bandwidth-limited links [7]. This is often difficult to achieve. Therefore, to compress 3D model effectively is urgent.

The research on 3D model compression started with Deering, who proposed to reduce the amount of data by quantizing all the components of the input mesh into a certain number of bits [8]. Later, the compression of the mesh model evolved. For the current research, the simplified algorithm still deserves further study in the relatively large 3D model, no matter in the compression effect or in the compression time [9].

Combined with Quadric Error Metrics (QEM) algorithm, this paper improves the edge collapse algorithm to solve the problem of undetected model structure and over-simplified model details [10]. The contents of the article are as follows: the second part is the existing model compression method. The third part presents the proposed algorithm. The fourth part and the fifth part are the verification of the algorithm and the conclusion respectively.

2 Related Work

Generally, two different 3D models are available after collecting construction data. One is point cloud model consisting of a large number of points containing the characteristics of the target surface, the other is mesh model in which objects and the entire environment are drawn by polygons. 3D models have a huge amount of

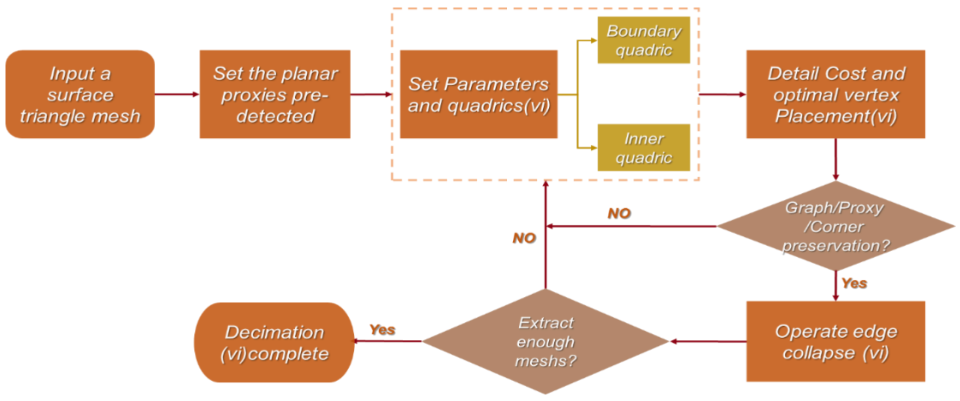

Input a surface triangle mesh
Set the planar proxies pre-detected
Set Parameters and quadrics(vi)
Boundary quadric
Inner quadric
Detail Cost and optimal vertex Placement(vi)
Graph/Proxy /Corner preservation?
NO
Yes
NO
Operate edge collapse (vi)
Extract enough meshs?
Yes
Decimation (vi)complete

Figure 1 Triangular mesh model simplification process

data. For example, the Faro Focus 3D delivers more than 976,000 3D points per second [11]. The state-of-the-art, light-based 3D camera produces 30 frames per second, roughly 300 MB of raw data [12]. Therefore, to store and manage three-dimensional data while preserving useful information as much as possible, it is important to improve the compression technique of the 3D model.

According to the decompression method after model compression, the 3D model compression algorithm is divided into single-rate compression [13] and progressive compression [14]. Single-rate compression technology compresses an entire model and then sequentially decodes the recovery model, focusing on saving bandwidth between the CPU and the graphics card [15-17]. With the increasing volume of 3D geometric data, this technology takes too long to receive the complete model, which cannot meet the real-time rendering requirements [18]. The advent of progressive compression algorithm has provided the possibility of real-time rendering of model compression. This algorithm first compresses the general framework of the model and then gradually supplements the remaining geometric details of the model. Therefore, the 3D model can be continuously reconstructed by the decoder from coarse to fine Levels of Detail (LOD) when receiving the bit stream [19].

The study of 3D point cloud compression algorithm is mainly divided into tree structure method and height map method. Height map-based method is to compress the 3D point cloud information into a two-dimensional matrix. This method was first proposed by Pauly [20]. Later, there have been some improved algorithms, such as adding geometry [21] [22], or preprocessing the data for compression [23]. The core of the tree-based method is to partition the point cloud space. The common tree structure methods include kd-tree [24] [25] and octree [26] [27]. The 3D point cloud can represent a geometric model of arbitrary shape with a simple structure and its compression technology has been relatively mature [28].

Compared with the point cloud model, the surface structure of the 3D mesh model is slightly complicated. However, the polygon structure can improve the visual effect of the model, which is especially suitable for visual inspection of large scene objects in the field of architecture. The study of model simplification includes vertex clustering [29], vertex deletion [30], wavelet transform algorithm [31] [32], edge collapse [33] [34]. These method is under development. How to compress mesh models well with increasing data volume is still a problem worth exploring.

3 Methodology

The proposed algorithm takes a surface triangle mesh of the model as input, since arbitrary polygons can be easily triangulated. Then preprocess the model to detect a set of the planar proxies. By performing a limited number of edge collapse operations on the above model, a simplified model that retains the structural features of the original model is output. The specific process shown in Figure 1. Edge collapse is one of the most common algorithms in model simplification. However, previous edge-collapse algorithms often caused over-simplification of the model.

In this paper, on the basis of the quadric error metric method (Abbreviated as GH) [10] and the volume-preserving approach (Abbreviated as LT) [35], the concept of planar proxies is introduced to ensure that important features of the model after input can be detected. To prevent the model from being oversimplified, inner quadric and boundary quadric is introduced to measure the error and the rule of model structure preservation is formulated.

3.1 Planar proxies

Before the implementation of the algorithm, the 3D model should be initialized to realize the storage format of the triangular mesh data structure and the vertex data structure. Set the planar proxies pre-detected

Suppose a triangle is selected on a plane, so the plane equation and normal vector of this triangle can be calculated. Center around this triangle and spread to the surrounding area of the triangle. Calculate whether the normal vector of the surrounding area is close to the normal vector of this triangle, and if so, it means that the triangle has high planarity. By analogy, calculate the planarity of different triangles, the triangle from which the best planarity is chosen represents the local area. As a result, the region has been expanding. This process is called planar proxies.

It is assumed that the input mesh presents a near-flat portion that can be detected by common shape detection methods. These near-plane parts are represented by planar proxies. More specifically, a plane proxy consists of a set of vertices and a plane ax+by+cz+d=0, denoted as [a b c d], where n = [a b c] is a flat unit normal vector. As shown in Figure 2, a processed 3D model, in which different colors represent different plane proxies.

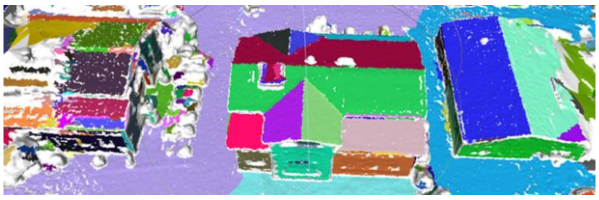


Figure 2 the model of the plane proxies detected

3.2 Edge collapse operator

All text must use the Times New Roman font. Edge Collapse $e=v_0v_1 \rightarrow v$ is a mesh operation that merges two vertices v_0 and v_1 into a unique vertex v. The grid extraction algorithm based on the edge collapse is usually as follows: (1) For each edge fold $v_0v_1 \rightarrow v$, the cost associated with the error metric and its corresponding placement strategy are defined to design the local optimal point location for finding v. (2) Calculate the initial priority heap of the edge folding operator at a cost increase. (3) Iteratively extract the lowest cost operator from the heap and calculate its best location. Collapses the relevant edges and updates the priority heap on the edge of the local neighborhood. Based on the quadric error metric for each defined operator, details of the cost and optimal location are determined.

3.2.1 Title Page Error Quadrics

In the early years, GH associated each vertex with a quadratic matrix, which represented an approximation of the error between the current and initial grids. The quadratic plane is designed as a 4*4 matrix, represented by (1), and then the square sum of the distances to these quadratic planes can be further calculated.

$$Q_P=PP^T=\begin{pmatrix} a^2 & ab & ac & ad \\ ab & b^2 & bc & bd \\ ac & bc & c^2 & cd \\ ad & bd & cd & d^2 \end{pmatrix} \tag{1}$$

The sum of the square of the distance from point v to the plane list (Pi) is represented in (2). The advantage of this quadratic form is that it can be encoded with only 10 coefficients, and this formula can calculate the distance from one point to any plane.

$$\sum d(v,P_i)^2=\sum v^T P_i P_i^T v=v^T\left(\sum Q_{P_i}\right)v \tag{2}$$

Based on the previous matrix method by GH, this article optimizes the square sum of the distance from point to point from the following three aspects: (a) to reduce the square sum of the distance from the vertex to the plane of the triangle mesh adjacent to it; (b) to reduce the sum of the squares of the distances from the vertex to the plane where it is adjacent to the plane proxy set; (c) to reduce the boundaries of agents and grids.

Inner quadric The proposed method adjusts the inner matrix of the structure and optimizes Q_p proposed by GH. For a triangle $t \in T_K$, plane P_t where t is located, we denote as Q_{Pt} the quadric of P_t. For the plane proxy φ, we denote as Q_φ the quadric where proxy is located. The set of proxies that contain t is represented by Proxies (t). The quadric Q_t for each triangle t of T(e) is given by (3).

$$Q_t = \begin{cases} Q_{P_t} & \text{if Proxies(t)}=\emptyset \\ (1\text{-}\lambda)Q_{P_t}+\lambda\sum_{\varphi\in \text{Proxies(t)}} Q_\varphi & \text{otherwise} \end{cases} \tag{3}$$

Then, the quadric form of the edge e is defined in (4)

$$Q_{inner}(e)=\sum_{t\in T(e)}|t|Q_t \tag{4}$$

Here, λ is called an abstraction parameter and provides a way to fidelity conversion between mesh and proxy. For example, when parameter λ = 1, that is, both proxies go through edge e, vertices are selected at the intersection of plane proxies. When the parameter λ=0 or no plane proxies passes edge e, the quadric only approximates to the local error measure of the mesh. A larger value for λ means faster removal of proxy details during decimation and faster abstraction. The higher value for λ results in a lower proxy error, but the approximation error is larger compared to the original mesh.

Boundary quadric For any edge e'∈R, a plane e'∈R, define $Q_{e'\varphi}$ as the quadric of the plane that passes through edge e' and is orthogonal to plane R. $E_{\partial K}$ represents a set of edges in K|φ that is only contained by one triangle. For model boundaries $e' \in E_{\partial K}$, t_e' is the only triangle in K|φ that goes through e'. The boundary quadric of an edge e is then defined in (5).

$$Q_{bdry}(e)=\sum_{e'\in E_{\partial K}}\left|t_e'\right|Q_{e',t_e'}+\sum_{E_{\partial K|\varphi}}\left|t_e'\right|Q_{e',\varphi} \tag{5}$$

In (5), the first part avoids overly simplifying the mesh boundaries to prevent the flat mesh from collapsing to a single point at any cost. The second part preserves the structural characteristics of the proxies. The proxy φ accommodates more information by restrictive triangulation than just an infinite plane. This process increases the sensitivity of the mesh to a simplified structure.

3.2.2 Cost and Placement

The GH algorithm consists of calculating the initial quadric Qv for each vertex v and then summing the distances from each edge collapse operator to the plane. More specifically, for the collapse operator $e=v_0v_1 \rightarrow v$, the quadric of e (Qe) is calculated by adding quadrics of its two vertices v_0, v_1 so that the optimal vertex position v is obtained via the minimized cost V^TQ_VV. Compared with the GH, we recalculate Qe from the current mesh before each collapse operator in terms of improving the efficiency of approximation error.

The process of our method is as follows.

The decomposition quadric matrix Qe is given by (6).

$$Q_e=\begin{pmatrix} A & -f \\ -f^T & g \end{pmatrix} \tag{6}$$

Among (6), A is a 3*3 matrix, f is a vector. Minimizing V^TQ_VV involves solving linear problems Ax = f. When all vertices adjacent to e are distributed in only one or two planes, the determinant of A is 0.

Compute the singular value decomposition *A* in (7).

$$A=U\Sigma V^T \tag{7}$$

where U and V are 3*3 matrixes, whose column vectors are mutually orthogonal. Σ is a 3*3 diagonal matrix whose diagonal values, called singular values, decrease from the upper left corner to the lower right corner.

The diagonal value in Σ is defined in (8), λ_i is the eigenvalue of A.

$$\Sigma_{ii}=\sqrt{\lambda_i} \tag{8}$$

The diagonal value Σ_{ii} decreases in descending order, then truncate the small eigenvalues of Σ and store the result in Σ^+, as shown in (9).

$$\Sigma_{ii}^{+}=\begin{cases} 1/\Sigma_{ii}, & \text{if } \Sigma_{ii}/\Sigma_{11}>\varepsilon \\ 0, & \text{otherwise} \end{cases} \tag{9}$$

where Σ_{11} denotes the largest singular value, and the parameter ε is set to 10^{-3}.

$\hat{x}$ denotes the center of gravity of the vertices of v_0 and v_1, the singular value decomposition of Qe is shown in (10).

$$x=\hat{x}+V\Sigma+U^T(f-A\hat{x}) \tag{10}$$

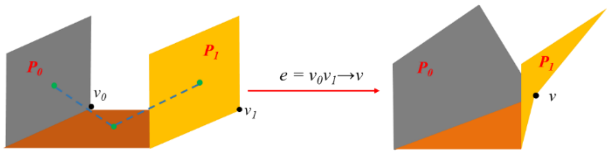
Calculate the optimal position of the vertex v after the collapse operation $e=v_0v_1 \rightarrow v$. Proxies(v) set is determined by Proxies(v_0) ∪ Proxies(v_1) when collapsing the edge e, and the quadric of e is defined in (11).

$$Q_e=(1-\mu)Q_{inner}(e)+\mu Q_{bdry}(e) \tag{11}$$

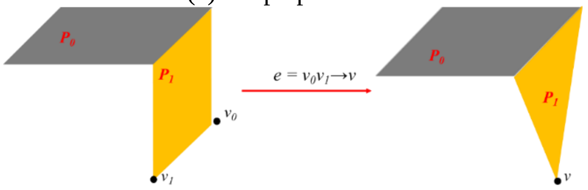
Where $\mu\in(0,1)$ is a parameter used for internally simplifying transaction boundaries. The optimal position for *v* is obtained by the solution of Qe and the cost V^TQ_VV of folding edge *e* is decided.

3.3 Structure-preserving

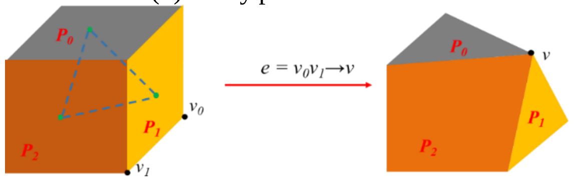
During model simplification, the best graph of the proxy coincides with the corresponding adjacency graph, such that a cube is represented by a double octahedron. In practice, however, it is not possible to have all the proxies correctly detected, and all the vertices are arranged on the same plane as the one to which they belong. Using the sufficient information about the adjacency graph in terms of the arrangement of the structure, three model structure preservation rules are formulated to prevent the structure of the model from being destroyed. These three rules are graph preservation, proxy preservation and corner preservation, Figure 3 is the rule diagram.



(a) Graph preservation



(b) Proxy preservation



(c) Corner preservation

Figure 3 Three types of edge collapsing prohibited

Graph preservation The two parts that are not connected in the graph should not be connected during the simplification. Only when all the proxy pairs in $Pv = Pv_0 \cup Pv_1$ are edges in the original set of G, do we perform edge collapse $e=v_0v_1 \rightarrow v$. Like Figure 3(a), the original two planes P_0P_1, which do not intersect, cannot intersect.

Proxy preservation A planar proxy may degenerate into a vertex or an edge during model simplification. To prevent this situation, as shown in Figure 3(b), the improved QEM algorithm refuses to perform this edge folding operation when the number of vertices forming this proxy is below the user-specified parameters after the edge-folding operation.

Corner preservation To prevent losing corners or migrating vertices away from corners, like Figure 3(c), we need to accurately detect and protect these corners during simplification. The vertices intersected by every three proxies are generally different, so we need to find the best vertex position for the intersecting proxy, taking advantage of the scale-space traversal.

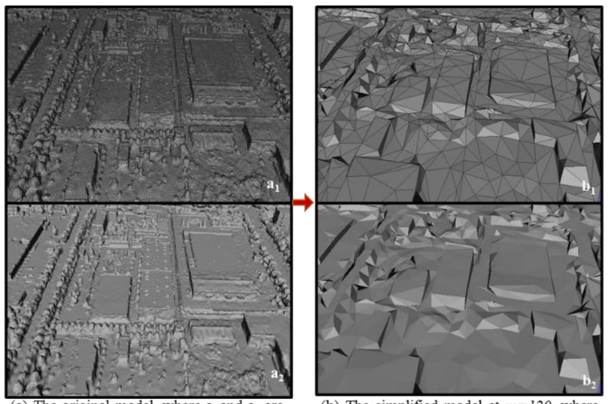
4 Results and Discussions

The focus of this paper is to study mesh model compression techniques to enable the construction industry to conveniently apply 3D models. Therefore, this paper selects two 3D models, an industrial building and a residential building, respectively, for model compression experiments. Table 1 shows the specific information of the two models before and after compression. Figure 4 and Figure 5 show the factory model and the resident building model, respectively.

It can be seen from Table 1, Figure 4 and Figure 5 that the simplified model still retains certain details when the simplified multiples reach high levels. In particular, some of the more complex structures, such as trees, etc., are less simplified than the rest of the model. For large complex models such as factories and residential buildings, the proposed algorithm can maintain the approximate shape of the model and reduce the loss of structural details of the model, while achieving a large simplification multiple.

This algorithm is further compared with GH and LT methods, taking the resident building model as an example. The effect is shown in Figure 6.

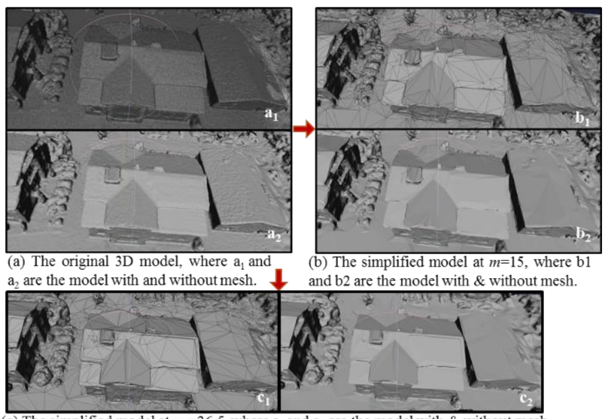
Both methods show the approximate shape of the simplified model. However, the proposed method is highly simplified in the planar area, and the meshes in the highly complex details are basically preserved, and the reduction effect is better. The flat part of the model, such as the roof part, etc., because of its simple structure, can be relatively more streamlined to save more space. This method flattens the roof plane into several large triangles, saving space for characterizing the undulating models. Non-planar parts, such as trees, cars, etc. in the model, are less simplified to better illustrate these details, which reduces the loss of structural details. In contrast, the proposed algorithm simplifies the various parts of the model in different degrees according to the complexity of the structure. The proposed algorithm can preserve the integrity of the model while maintaining a high compression efficiency, whose compression method is feasible for models with a large amount of data.



(a) The original model, where a$_1$ and a$_2$ are the model with & without mesh.

(b) The simplified model at *m=120*, where b$_1$ and b$_2$ are the model with & without mesh.

Figure 4 The factory model



(a) The original 3D model, where a$_1$ and a$_2$ are the model with and without mesh.

(b) The simplified model at *m*=15, where b1 and b2 are the model with & without mesh.

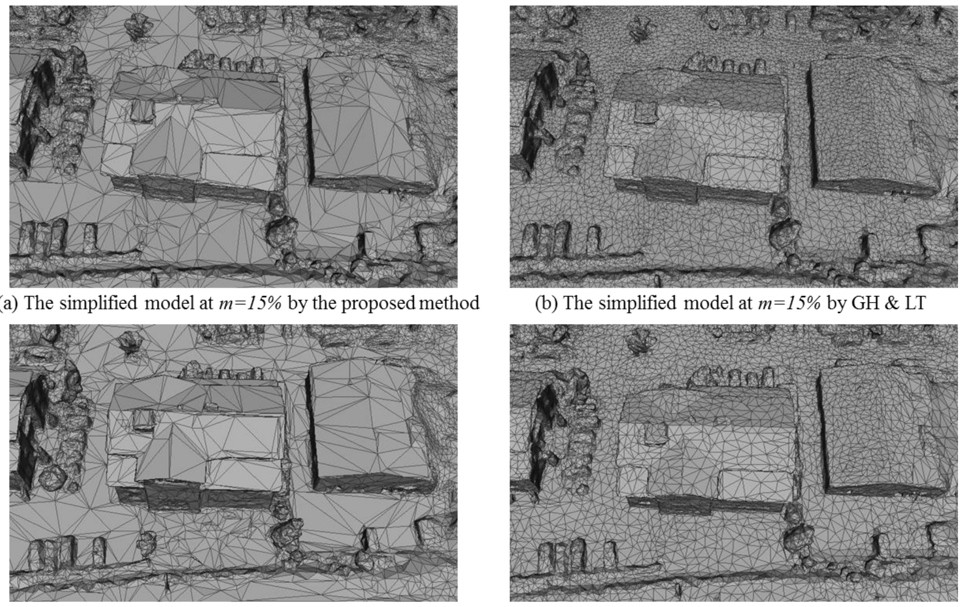
(c) The simplified model at *m=26.5*, where c$_1$ and c$_2$ are the model with & without mesh.

Figure 5 The resident building model

Table 1. the specific information of the models before and after compression

Model	The number of		Data size (KB)	Simplify multiple (m)*
	vertices	faces		
Original factory	234118	467672	8682	——
Simplified factory	1920	3999	74	120
Original resident	533995	1067818	19815	——
Simplified resident 1	35079	69999	1300	15
Simplified resident 2	20191	39999	745	26.5

* Simplify multiple (m) is approximately equal to the ratio of the number of vertices or faces, or data size in the original model to the number of vertices or faces, or data size in the simplified model



(a) The simplified model at *m=15%* by the proposed method (b) The simplified model at *m=15%* by GH & LT

(c) The simplified model at *m=26.5%* by the proposed method (d) The simplified model at *m=26.5%* by GH & LT

Figure 6 Compression of the two methods for residential building models

5 Conclusion

In this paper, an improved method for simplifying complex models of large scenes in architecture is proposed, which can better preserve the details of the model compared with the classical algorithms. Experiments show that the proposed algorithm simplifies the various parts of the model in different degrees according to the complexity of the structure. This preserves the details of the model as much as possible. With the same simplify multiple, the proposed method has a greater advantage in displaying the details of the model. The proposed algorithm greatly improves the accuracy of the simplified model under the premise of ensuring the compression ratio. However, this paper uses the region growing algorithm to detect plane proxies. This process is complicated and takes some time, which increases the compression time of the model. Future directions will be to refine algorithms to reduce the time required for model compression.

Acknowledgement

The presented work has been supported by the National Science Foundation of China (NSFC) through grant 71471072.

References

[1] Son, H. and Kim, C. 3D structural component recognition and modeling method using color and 3D data for construction progress monitoring. *Automation in Construction*, 19(7):844-854, 2010.

[2] Lattanzi, D. and Miller, G.R. 3D Scene Reconstruction for Robotic Bridge Inspection. *Journal of Infrastructure Systems*, 21(2):04014041, 2015.

[3] Liu, Y. F., Cho, S., Spencer, B. F., & Fan, J. S. Concrete Crack Assessment Using Digital Image Processing and 3D Scene Reconstruction. *Journal of Computing in Civil Engineering*, 30(1): 04014124, 2016.

[4] Golparvar-Fard, M., Bohn, J., Teizer, J., Savarese, S., & Peña-Mora, F. Evaluation of image-based modeling and laser scanning accuracy for emerging automated performance monitoring techniques. *Automation in Construction*, 20(8): 1143-1155, 2011.

[5] Khaloo, A., Lattanzi, D. Hierarchical Dense Structure-from-Motion Reconstructions for Infrastructure Condition Assessment. *Journal of Computing in Civil Engineering*, 31(1):04016047, 2016.

[6] Cheng, S. C., Kuo, C. T., & Wu, D. C. A novel 3D mesh compression using mesh segmentation with multiple principal plane analysis. *Pattern Recognition*, 43(1):267-279, 2010.

[7] Peng, J., Kim, C. S., & Jay Kuo, C. C. Technologies for 3D mesh compression: A survey. Journal of *Visual Communication & Image Representation*, 16(6), 688-733, 2005.

[8] Deering, M. Geometry compression. *Conference on Computer Graphics and Interactive Techniques DBLP*:13-20, 1995.

[9] Lyuu, Y. D., Ma, T. M., & Ti, Y. W. Linear-time compression of 2-manifold polygon meshes into information-theoretically optimal number of bits. *Applied Mathematics & Computation*, 217(21):8432-8437, 2011.

[10] Garland, M. Surface simplification using quadric error metrics. *Acm Siggraph Computer Graphics*, 1997:209-216, 1997.

[11] Elseberg, J., Borrmann, D., & Nüchter, A. One billion points in the cloud – an octree for efficient processing of 3D laser scans. *Isprs Journal of Photogrammetry & Remote Sensing*, 76(1):76-88, 2013.

[12] Hou, J., Chau, L. P., Zhang, M., & Magnenat-Thalmann, N. A Highly Efficient Compression Framework for Time-Varying 3-D Facial Expressions. IEEE Transactions on Circuits & Systems for Video Technology, 24(9):1541-1553, 2014.

[13] Gumhold, S., Kami, Z., Isenburg, M., & Seidel, H. P. Predictive point-cloud compression:137, 2005.

[14] Schnabel, R., Möser, S., & Klein, R. Fast vector quantization for efficient rendering of compressed point-clouds. *Computers & Graphics*, 32(2):246-259, 2008.

[15] Chow, M. M. Optimized geometry compression for real-time rendering. *Visualization '97. Proceedings IEEE*:347-ff, 1997.

[16] Ahn, J. H., Kim, C. S., & Ho, Y. S. Predictive compression of geometry, color and normal data of 3-D mesh models. *IEEE Transactions on Circuits & Systems for Video Technology*, 16(2):291-299, 2006.

[17] Luffel, M., Gurung, T., Lindstrom, P., & Rossignac, J. Grouper: A Compact, Streamable Triangle Mesh Data Structure. *IEEE Transactions on Visualization & Computer Graphics*, 20(1):84-98, 2014.

[18] Vasa, L., Brunnett, G. Exploiting Connectivity to Improve the Tangential Part of Geometry Prediction. *IEEE Transactions on Visualization & Computer Graphics*, 19(9):1467-1475, 2013.

[19] Pajarola, R., & Rossignac, J., Compressed Progressive Meshes. *IEEE Transactions on Visualization & Computer Graphics*, 2002, 6(1):79-93.

[20] Pauly, M., Gross, M. Spectral processing of point-sampled geometry. *Conference on Computer Graphics and Interactive Techniques ACM*:379-386, 2001.

[21] Tilo, O., & Dietmar, S. Image - Based Surface Compression. *Computer Graphics Forum*. *Blackwell Publishing Ltd*:1647-1663, 2008.

[22] Schnabel, R., Möser, S., & Klein, R. A Parallelly Decodeable Compression Scheme for Efficient Point-Cloud Rendering. Symposium on Point Based Graphics, Prague, Czech Republic, 2007. *Proceedings DBLP* :119-128, 2007.

[23] Golla, T., & Klein, R. Real-time point cloud compression. I*eee/rsj International Conference on Intelligent Robots and Systems*. *IEEE*:5087-5092, 2015.

[24] Friedman, Jerome, H., Bentley, Louis, J., Finkel, & Ari, R. An Algorithm for Finding Best Matches in Logarithmic Expected Time. *Acm Transactions on Mathematical Software*, 3(3):209-226, 1977.

[25] Sang, W. B., Sang, W. B., & Choi, S., 3D medial axis point approximation using nearest neighbors and the normal field. *Springer-Verlag New York, Inc*. 2012.

[26] Elseberg, J., Borrmann, D., Nüchter, A. One billion points in the cloud-an octree for efficient processing of 3D laser scans. *ISPRS Journal of Photogrammetry and Remote Sensing*, 76: 76-88, 2013.

[27] Tang, L., Da, F. P., & Huang, Y. Compression algorithm of scattered point cloud based on octree coding. I*EEE International Conference on Computer and Communications*. *IEEE*, 2017.

[28] Navarrete, J., Morell, V., Cazorla, M., Viejo, D., García-Rodríguez, J.,& Orts-Escolano, S. 3DCOMET: 3D compression methods test dataset. *Robotics & Autonomous Systems*, 75:550-557, 2016.

[29] Lindstrom, P. Out-of-core simplification of large polygonal models. *ACM SIGGRAPH*. *CiteSeer*:259-262, 2000.

[30] Schroeder, W. J. Decimation of triangle meshes. *Acm Siggraph Computer Graphics*, 26(2):65-70, 1992.

[31] Valette, S., & Prost, R., Wavelet-based progressive compression scheme for triangle meshes: wavemesh. *Visualization & Computer Graphics IEEE Transactions on*, 10(2):123-129, 2004.

[32] Avilés, M., Morán, F., & García, N., Progressive Lower Trees of Wavelet Coefficients: Efficient Spatial and SNR Scalable Coding of 3D Models. *Pacific-Rim Conference on Multimedia*. *Springer Berlin Heidelberg*:61-72, 2005.

[33] Oya, T., Tanamura, T., Aoyama, H., & Higashi, M. Mesh Simplification Based on Feature Preservation and Distortion Avoidance for High-Quality Subdivision Surfaces. *Computer-Aided Design and Applications*, 10(3): 541-550, 2013.

[34] Wang, H., Qiao, F., & Zhou, B. Multi-Feature Metric-Guided Mesh Simplification. *Advances in Intelligent Systems & Computing*, 250:535-542, 2014.

[35] Lindstrom, P., Turk, G. Fast and memory efficient polygonal simplification. *Visualization '98. Proceedings*. *IEEE*:279-286, 1998.