Computers & Geosciences 83 (2015) 37–45

Contents lists available at ScienceDirect

Computers & Geosciences

journal homepage: www.elsevier.com/locate/cageo

Case study

An improved texture-related vertex clustering algorithm for model simplification

Jing Chen *, Mo Li, Jiawei Li

State Key Laboratory of Information Engineering in Surveying, Mapping and Remote Sensing, Wuhan University, 129 Luoyu Road, Wuhan, Hubei 430079, China

ARTICLE INFO

Article history:
Received 30 June 2014
Received in revised form
19 March 2015
Accepted 9 July 2015
Available online 11 July 2015

Keywords:
3D GIS
Model simplification
Adaptive partitioning
Vertex clustering
Texture-related

ABSTRACT

As an important data source in 3D GIS, 3D landmark models are built to simulate the real-world scenario. However, due to the enormous volume and complexity of 3D models, the data transmission under limited bandwidth and the real-time rendering have always been an open problem. In order to improve the visualization and the efficiency, this paper proposes a novel model simplification algorithm in consideration of texture after the analysis of the existing model simplification approaches. Differing from the previous research, our approach defines a new error metric related to the model texture, which extends the vertex clustering scheme in 3D geometry space and 2D texture space independently. Since the uneven distribution of vertices is taken into account, the clustering unit is divided adaptively in consideration of both geometry and texture information. In view of reducing the memory overhead and improving the algorithm efficiency, we don't create new vertices by iterative calculations, but use the inherent vertices in the initial meshes as the characteristic vertices. To demonstrate the feasibility and effectiveness of our strategy, a series of simplification experiments have been carried out on the platform of DirectX 3D, a widely used 3D application programming interface. The results show that the simplified models in consideration of texture preserve more texture details than those traditional ones. It apparently makes a good balance between the reduction rate and visual effects.

1. Introduction

With the rapid development of computer, network and virtual reality technology, the current Geographic Information Systems and Services demand a unified visual platform for rendering the remote sensing image, regular grid DEM and 3D models together, as the integrated expression of the multi-source spatial geographic information. Especially, 3D building models have the advantage to show the realistic city landscape vividly, owing to their abundant information, intuitive expression, excellent visual effects and suitability for human spatial cognition. Nevertheless, the intricate topology and additional attributes such as material, texture, skeletal animation and so on bring about enormous data redundancy, which creates impediments to providing online services about global three-dimensional spatial information visualization. Therefore, the fast data interpretation and real-time rendering within limited CPU resources have been a critical challenge in 3D GIS, and model simplification is no doubt an effective solution to this problem.

More significantly, however, it is important to balance the contradictions between high simplification rate and satisfactory visual quality. Without appropriate error metrics, oversimplified models will lose material particulars. Motivated by the above, our main contributions are as follows: A novel vertex clustering simplification algorithm is put forward, which takes the texture details into account as well as the geometry information. It introduces appropriate thresholds in texture error metrics, which aims to preserve essential texture pattern details in different levels. Moreover, in order to achieve the partitioned organization of the vertices, we use adaptive subdivision based on Oct-tree to control the scope of vertex clustering dynamically according to the local vertex distribution and error metrics. The optimized algorithm shows stricter error control, higher operation efficiency, lower memory usage and better visual effects, which is conducive to simplifying the massive amount of complicated models.

In this paper, the remainder is organized as follows. Section 2 reviews the previous research on simplification of urban building models, which provides new insights into our approach. Section 3 describes the detailed algorithm process of vertex clustering simplification in consideration of texture, which makes specific explanation of the feasibility of this algorithm. Section 4 demonstrates our approach by rendering a series of complicated building

* Corresponding author. Fax: +86 27 68778229.
E-mail address: jchen@whu.edu.cn (J. Chen).

http://dx.doi.org/10.1016/j.cageo.2015.07.005
0098-3004/

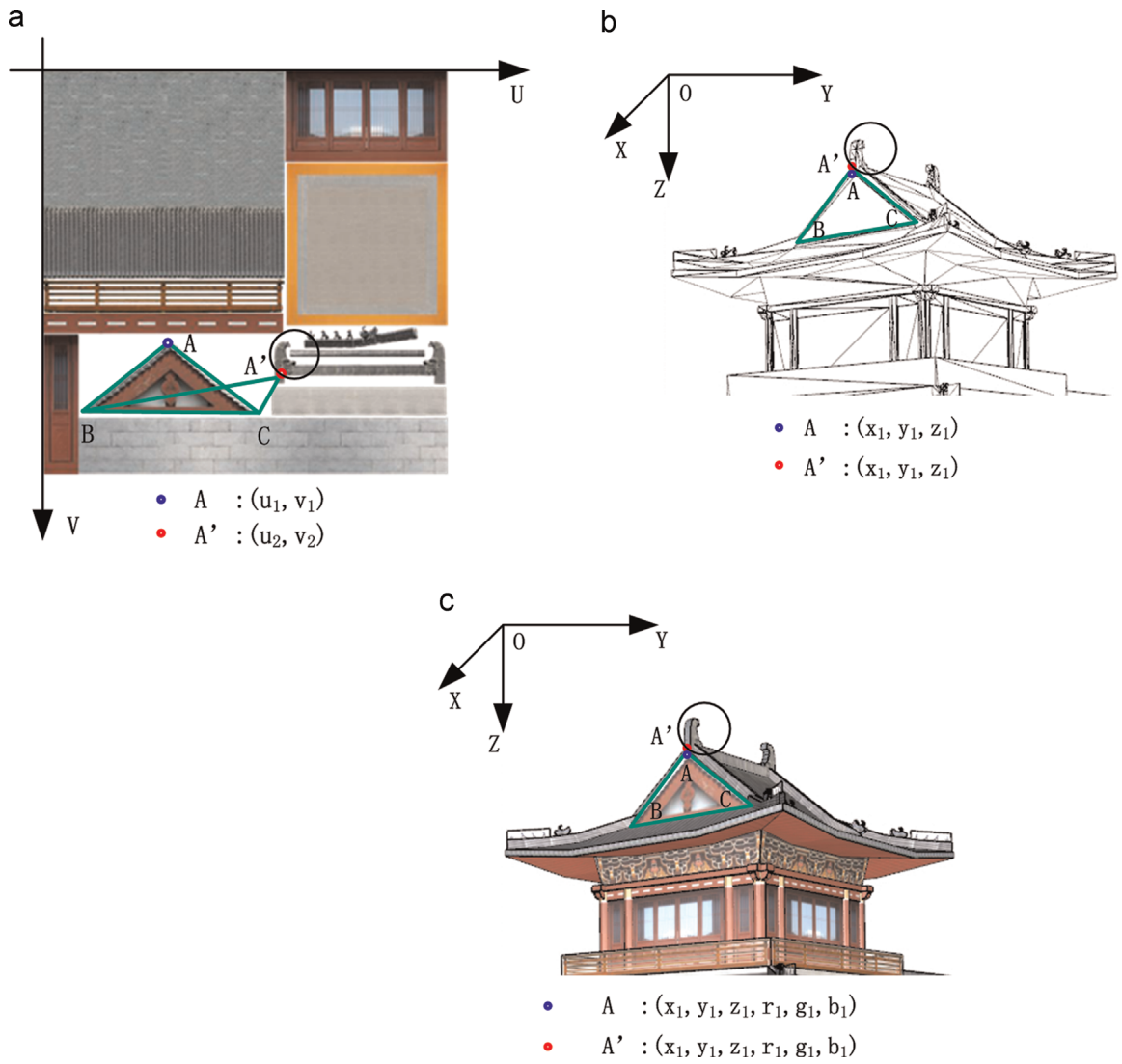

Fig. 1. Vertex mapping relationships in different rendering spaces. (a) Texture space. (b) Geometry space. (c) Geometry-color space.

models in DirectX 3D. The performances are compared with those using the traditional strategies. Section 5 concludes and discusses the future work.

2. Related work

The most notable available simplification algorithms include vertex clustering algorithm (Chan and Kok, 2001; Li et al., 2008; Tsuchie et al., 2014), wavelet decomposition algorithm (Esteve et al., 2000; Wang and Zhang, 2010), vertex decimation algorithm (Kobbelt et al., 1998; Franc and Skala, 2002), iterative edge contraction algorithm (Heckbert and Garland, 1999; Ungvichian and Kanongchaiyos, 2011) and so on. These ordinary model simplifications are more likely to aim at streamlined models, that consist of smooth and regular surfaces without sharp edges and corners. However, unlike the 3D game environments, as for 3D GIS, we are more interested in geographical landmark models that are built in accordance with the reality and keep more complex geometric structure. In order to achieve the hierarchical visualization in 3D GIS, various methods have been proposed for model simplification, generalization, aggregation and merger in 3D virtual scenes. Thiemann and Sester (2006) proposed a generic set of adaptive templates to replace the original urban landmarks. Characteristics can be highlighted or enhanced easily since the semantic information is known, but the templates adaptation is limited. Royan et al. (2007) developed a merging algorithm based on Delaunay triangularization and PB-Tree structure used for hierarchical rendering, but too complex hierarchy would affect the rendering efficiency. Another scale-space approach worked for generalization by shifting parallel facets to erase cracks and adaptive squaring non-orthogonal structures (Forberg, 2007). Mao et al. (2011) put forward a multiple representation data structure named CityTree for dynamic visualization of generalized 3D city models. It works well for simple building blocks, but not for those with complex roofs. Xie et al. (2012) presented a view-dependent scheme consisting of simplification of individual buildings and aggregation of building groups. The former includes footprint correction, special structure removal, roof simplification, oblique rectification and facade shifting, while the latter includes hierarchical clustering, footprints generation and hierarchical tree establishment. Although the above algorithms cover a wide variety of situations, the texture information still has no role in model simplification.

After taking texture constraint into account, Chang et al. (2008) introduced a hierarchical texture rebuilding approach, which combined the textures into texture atlases according to geographical adjacency. However, this approach is designed rather for urban legibility, not for improvement of visual quality in its

strictest sense. A structure-preserving reshape scheme in consideration of texture was presented by Cabral et al. (2009), which kept the structural rigidity and used texture synthesis to recover details in stretched image areas. This algorithm accelerates modeling by chaining the existing pieces together to construct complex models, but doesn't suit for multi-scale model simplification in 3D GIS. Zhang et al. (2012) proposed a component structure named FEdge, with which the cluster instance was mapped into different striped texture segments, and the FEdge indices were organized into a EBT (Evolved Buffer Tree) structure for recombining models.

Comprehensively analyzing the above texture related methods, we can easily find that most large-scale architectural scenes discussed above consist of simple building blocks with repetitive texture patterns, while our approach will focus on more complicated model structure with richer texture information such like models of Chinese classic pavilions. In other words, the former algorithms belong to fragment-level model simplification, and this algorithm focuses more on vertex-level simplification, which is more suitable for the hierarchical visualization of a massive amount of elaborate building models in 3D GIS.

As for vertex-level model simplification in consideration of texture information, a common method is to add a (r, g, b) 3D texture color space (Shen et al., 2007; Feng and Zhou, 2009). The fundamental principle of this method is as follows: Find out the color information (r, g, b) of the given vertex corresponding to its texture coordinates (u, v), and then calculate the distance error in color space to define the texture tolerance, according to the vertex correspondence between the simplified model and the initial model. The main disadvantages lie in the following two aspects: Firstly, it reduces the algorithm efficiency. Getting color information from texture coordinates needs loading the texture data into memory frequently and performing the extraction operation from (u, v) to (r, g, b) for each vertex, which apparently increases the resource consumption and running time. Secondly, it doesn't work well for models with complex structure and composite texture patterns. It is because color coordinates in (r, g, b) format are not gentle gradients in texture space and they only represent a fraction of the texture information. Fig. 1 shows an example of the Chinese classical airy pavilion, which reflects the vertex mapping relationships in different rendering spaces, left for 2D texture space, middle for 3D geometry space, right for composite geometry-color space. It is obvious in the illustration that vertex *A, A′* share the same position and color value, in other words, their distance in geometry-color space is regarded as zero. But if we replace the facet *ABC* with the facet *A′BC*, it will result in the distortion and deformation of the texture mapping, which means that the visual effects are beyond the estimate of the error metric. This paper will pay more attention to avoid this mistake and make improvement on the visual effects of the simplified results.

3. Methodology

3.1. Overall thought

Through the analyses above, it is easy to find that the vertex texture coordinates are the key point to measure the texture distance rather than the vertex color coordinates. Thus, we decide to introduce texture coordinates into metric space. Texture error is defined as the distance value between the characteristic vertex and the initial vertex according to the correspondence in simplification:

$$E(M_1,\ M_2) = \sum_{V_1 \in M_1,\ V_2 \in M_2} \sqrt{(u_1 - u_2)^2 + (v_1 - v_2)^2}$$

Note: $(u_1,\ v_1)$ and $(u_2,\ v_2)$ are texture coordinates of vertices V_1,V_2. M_1, M_2 refer to the initial mesh and simplified mesh.

Based on this error metric criterion, we take precedence to remove the vertex with smaller texture error, so as to make the simplified results preserve more texture details. In terms of the data organization of the simplification, one available structure is the progressive mesh proposed by Hoppe (1996). Then view-dependent refinement criteria (Hoppe, 1997) and two reordering techniques (Hoppe, 1999) were presented to supplement the structure. The method which applies the progressive mesh construction into the vertex clustering scheme was proposed and improved for super size meshes (Lindstrom, 2000; Li and Chen, 2008). We extend the vertex clustering scheme and use existing vertices instead of new vertices so that the simplified results can be updated smoothly without considering the data dependencies.

The general idea of our approach is as follows: based on adaptive space partition into octree structure, we can get the initial clustering cells in various sizes roughly according to the distribution of vertices. Then we need to calculate the weight value for each vertex in view of the aforementioned factors, and then choose an appropriate characteristic vertex for each clustering cell. When we check the geometric error metric and the texture error metric, if either of them is greater than the given threshold, the current node will need further splitting. With diverse error metrics, we can construct multiple level of details (LOD) models after the entire simplification process (Funkhouser and Sequin, 1993; Cao et al., 2000; Ribelles et al., 2010).

3.2. Detailed algorithm steps

3.2.1. Calculation of vertex weight

The main goal of model simplification is to maximize the removed vertices on the premise that the visual effects of the simplified result are not badly changed compared with the initial model. Therefore, in the process of vertex clustering, the selected characteristic vertex should reflect the model characteristics and preserve the contour structure. In order to save memory resources and improve the algorithm efficiency, we don't create new vertices but use existing vertices in initial meshes as the characteristic vertices instead. Normally, vertices located in areas with obvious characteristic changes such as boundary, corner and embossing should be assigned larger weight. In contrast, vertices in smooth areas should be given smaller weight. It means the more impacts the vertex has on visualization, the larger weight it has to be assigned. When setting clustering characteristic vertices, we always choose those in larger weight. With this principle, the following gives a method of calculating the vertex weight.

Set the weight range $w_{ij} \in (0, 1)$, and the weight value reflects its importance. For each single vertex V in 3D meshes, it is surrounded by a set of triangles $\boldsymbol{T_S} = \{t_0, t_1, \ldots t_m\}$, which means all the triangles contain the vertex V. Suppose the two triangles t_i and t_j are adjacent, then we give the following definition:

$$w_{ij} = \left(1 - dot\left(n_i, n_j\right)\right)/2 = (1 - \cos\theta)/2$$

Among them, n_i and n_j are normalized normal vectors of triangle t_i and t_j, θ means the angles between n_i and n_j, *dot* means dot product operation, and w_{ij} ranges from 0 to 1. As we all know, the cosine function is a monotony decrease function when the angle value varies from 0 to π. Whether the angles are acute or obtuse, the greater the θ value is, the smaller the cosine value is, and so the greater the weight value is. It indicates more obvious geometrical features. Then the weight of vertex V can be expressed as follows:

$$\begin{cases} w = max\{w_{ij}\} \\ \quad i \neq j \\ i,\ j \in (0,\ m) \end{cases}$$

Besides, since boundary is an important guarantee for maintaining model outlines, for boundary vertices, the weight should be set to 1 as maximum. A distinguishing method for boundary vertices is given: Suppose there is a vertex V, and the related edge set and triangle set are $\boldsymbol{L_S}=\{l_0, l_1,\ldots l_n\}$ *and* $\boldsymbol{T_S}=\{t_0, t_1,\ldots t_m\}$. If there is an existing edge l_i which is only contained in a single triangle t_j, then this vertex must be a boundary vertex. Otherwise, it cannot be a boundary vertex. Based on the above discussions, the weights of all vertices can be calculated.

3.2.2. Initial mesh partitioning

Based on the predetermined geometry unit size and texture unit size, the 3D meshes will be divided into five dimensions (X, Y, Z, U, V), where (X, Y, Z) refer to geometry space coordinates and (U, V) refer to texture space coordinates.

As mentioned above, the corresponding geometry space of a 3D mesh is its external cubic bounding box, and the corresponding texture space is a 2D texture plane, where the coordinates in both directions range from 0 to 1. Before recursive subdivision, either we can estimate the initial cell length of the parent node in the octree according to the preset average vertex number in a cluster, or we can fix the cell length according to specific reduction rates, such as 1 m. Next, we have to take the local vertex distribution into account. It means adaptive recursive node combination and node split will be continuously performed until satisfying the demand of the minimum vertex number in a cluster, resulting in multi-scale leaf nodes of spatial octree. The schematic diagram is shown in Fig. 2. At the same time, in view of the vertex subdivision in texture space, each texture image can be divided into 10×10 chessboard style texture units, waiting for further subdivision of the octree leaf nodes. After adaptive mesh partitioning, we can get the initial clustering leaf node. As various error metrics will be introduced in the subsequent steps, further node split operations may generate more leaf nodes.

3.2.3. Calculation of error metrics

The texture coordinates (u, v) adequately reflect the texture mapping relationship of the 3D geometry vertex. For triangles that contain vertices with significantly different texture coordinates, they are more likely to provide abundant texture information, which means deleting these triangles will cause greater texture error. By contrast, deleting triangles that share similar texture coordinates will lose less texture details.

Error control used in this paper consists of geometry error control and texture error control. As for geometry error control, curvature is selected to be a metric, as is described in details below:

Assume that there is a vertex collection in a leaf node named $\boldsymbol{V_S}=\{V_w, V_0, V_1\ldots V_n\}$, where V_w stands for the clustering characteristic vertex and V_0, $V_1\ldots V_n$ refer to the other $n+1$ vertices in the same leaf node. Then to any single vertex V_i in the collection, there is a related triangle set named $\boldsymbol{T_S}=\{t_0, t_1,\ldots t_m\}$. After replacing all the vertices with the vertex V_w, we can get a new triangle set named $\boldsymbol{T_S}'=\{t_0', t_1',\ldots t_m'\}$. If n_j represents the normal vector of triangle t_j in $\boldsymbol{T_S}$, and n_j' represents the normal vector of triangle t_j' in $\boldsymbol{T_S}'$, then the geometry error of vertex V_i can be estimated with a metric ε_i:

$$\varepsilon_i = \frac{1}{m+1}\sum_{j=0}^{m}\frac{1 - dot\left(n_j,\ n`_j\right)}{2}$$

Where *dot* is the vector dot product of n_j and n_j', and j ranges from 0 to m.

The geometry error of the entire vertex set $\boldsymbol{V_S}$ after clustering is calculated according to the following formula:

$$\varepsilon_s = max\{\varepsilon_i\}$$

where *max{}* represents the maximization operation, and i ranges from 0 to n.

The basic rules of texture error calculation are as follows:

Assume that there is a vertex collection in a leaf node named $\boldsymbol{Vs}=\{V_w, V_0, V_1\ldots V_n\}$, where V_w stands for the clustering characteristic vertex, whose texture coordinates are (u_w, v_w). Meanwhile to any other vertex V_i in the vertex collection, the texture coordinates are (u_i, v_i), then the texture error generated by vertex substitution can be estimated with a metric ε'_i:

$$\varepsilon'_i = \sqrt{(u_i - u_w)^2 + (v_i - v_w)^2}$$

The texture error of the entire vertex set$\boldsymbol{V_S}$ after clustering is calculated as follows:

$$\varepsilon'_s = max\{\varepsilon'_i\}$$

Where *max{}* represents the maximization operation, and i

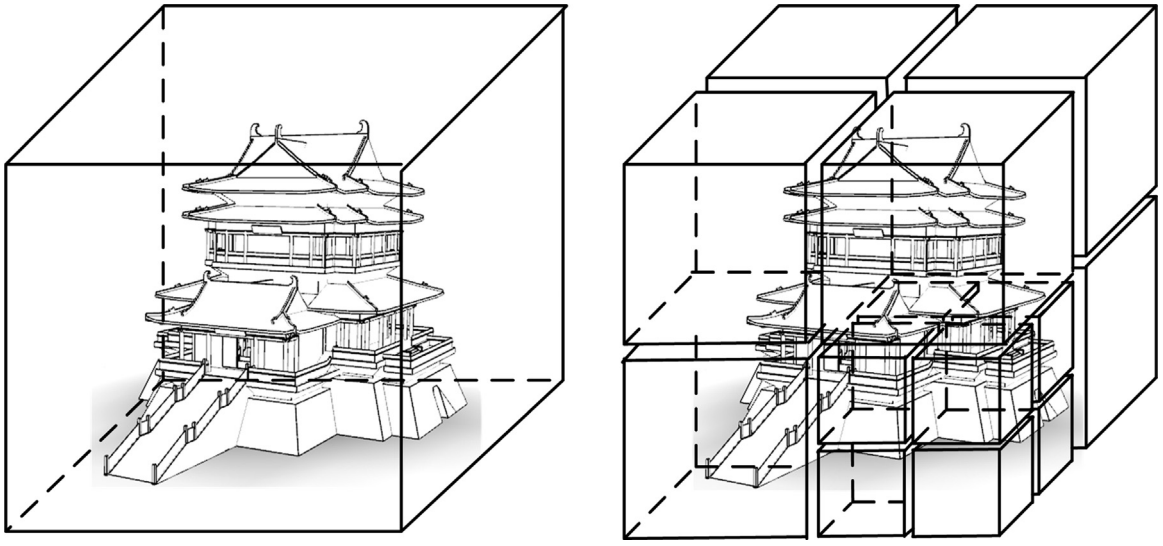

Fig. 2. Adaptive space partition into octree structure.

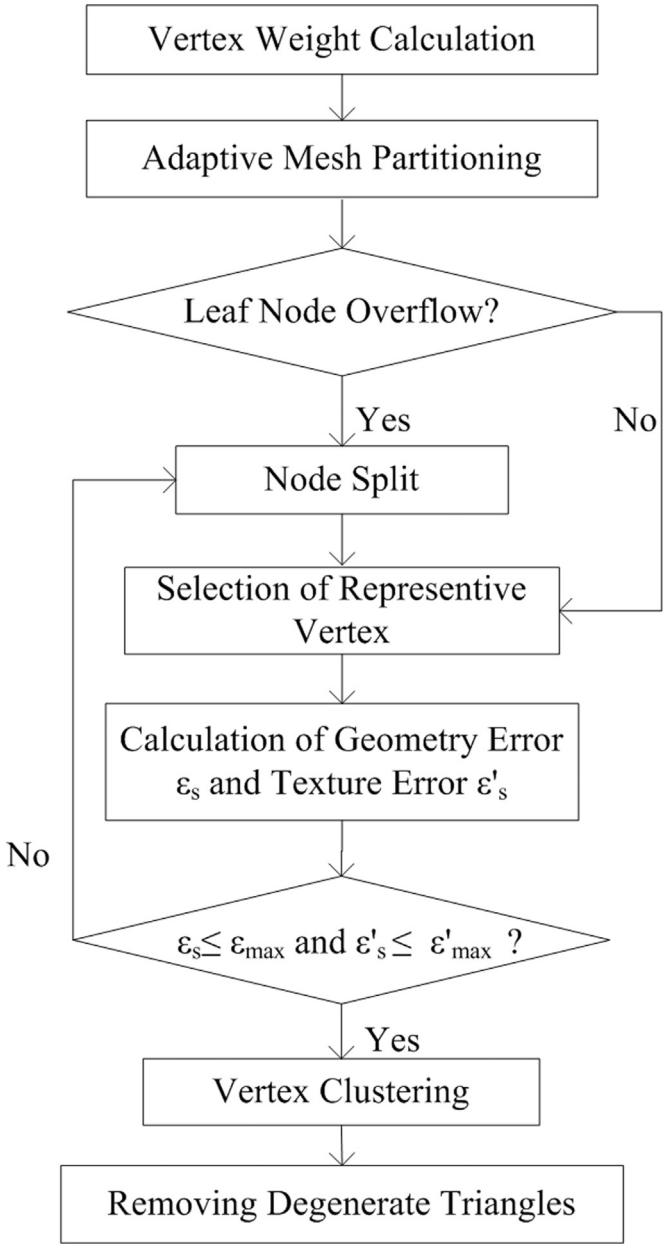

Fig. 3. Algorithm flow diagram.

Table 1
Vertex and triangle simplification details for 3D models with multiple meshes.

Mesh ID	Original vertex number	Original triangle number	Simplified vertex number	Simplified triangle number
0	5053	3616	584	462
1	5505	3660	1971	1543
2	5601	3839	1856	1642
3	1780	1001	236	167
4	231	137	159	99
5	143	122	92	74
6	1098	1007	508	497
7	201	213	184	198
8	25	23	17	15
9	94	60	64	47
10	8	4	8	4
11	324	287	242	216
12	1093	743	199	169
Sum	21156	14712	6120	5133

Table 2
Simplification performance test results.

Model ID	Original triangle number	Simplified triangle number	Reduction rate (%)	Running time (s)
0	3774	1432	62.06	1.26
1	4701	1851	60.63	1.48
2	14,712	5278	64.12	4.86
3	82,646	34,495	58.26	26.86
4	352,902	102,231	71.03	133.71

ranges from 0 to n.

3.2.4. Node split principle

For all the initial leaf nodes generated from mesh partitioning, traversal operation has to be performed to judge that if the nodes need to be further split or not. After choosing the highest weight vertex as the characteristic vertex in the leaf node as mentioned above, it is easy to estimate the geometry clustering error $\mathcal{E}_s$ and texture clustering error $\mathcal{E}'_s$. If $\mathcal{E}_s$ exceeds the setting error threshold $\mathcal{E}_{max}$ or $\mathcal{E}'_s$ exceeds the setting error threshold $\mathcal{E}'_{max}$, node split operations will be performed in a loop until the leaf node is empty, or only one vertex is valid, or the clustering error is within the scope of admissible error. The default thresholds used for controlling the leaf node capacity and error allowance are related to the elaborate degree of 3D modeling, which can be adjusted based on the actual situation. The so-called node split means splitting each vertex cluster into two sections according to the (X, Y, Z) and (U, V) coordinates.

3.3. 5. Vertex clustering

Through error calculation in advance, invalid leaf nodes are filtered out and the remaining leaf nodes get involved in vertex clustering. The essence of clustering operation is to replace the other vertices in a cluster with the selected representative vertex and update the coordinates and adjacency topology relationship accordingly. For the existing 3D modeling technology, a triangle list is used to express the contour shape of 3D models. It consists of a collection of triangles which contain triangle index, triangle normal vector and the corresponding vertex indices. After clustering, a set of degenerate triangles formed by three collinear vertices may be generated. The degenerate triangles should be removed from the initial triangle list. Then the sequence numbers of the triangle list need to be readjusted and the normal vectors need to be recalculated.

To sum up, the operating process of this algorithm is shown below in Fig. 3:

4. Experiments and discussions

In order to verify the feasibility of the proposed algorithm, a series of simplification experiments were carried out, mainly focusing on architectural models with complex texture. This section will evaluate and analyze the experiment results mainly from two aspects: model simplification performance and model simplification quality. It's remarkable that the test models are realistically generated by close-range photogrammetric technology, in accordance with the actual scale of the pavilion platform buildings. These models are assembled with multiple meshes, each of which consists of complicated geometric structure and is decorated with several groups of finely combined texture patterns.

Table 3
Impact of different UV error threshold settings on running time & reduction rate.

UV error threshold	Model1 (14712 faces)		Model2 (3455 faces)		Model3 (7940 faces)	
	Reduction rate (%)	Running time (ms)	Reduction rate(%)	Running time (ms)	Reduction rate(%)	Running time (ms)
0.05	49.986	969.2	70.304	251.4	59.244	402.9
0.10	52.073	995.7	71.983	260.8	61.587	414.1
0.15	52.726	1003.2	72.214	268.3	62.481	425.6
0.20	52.875	1006.1	72.504	269.7	62.620	430.9

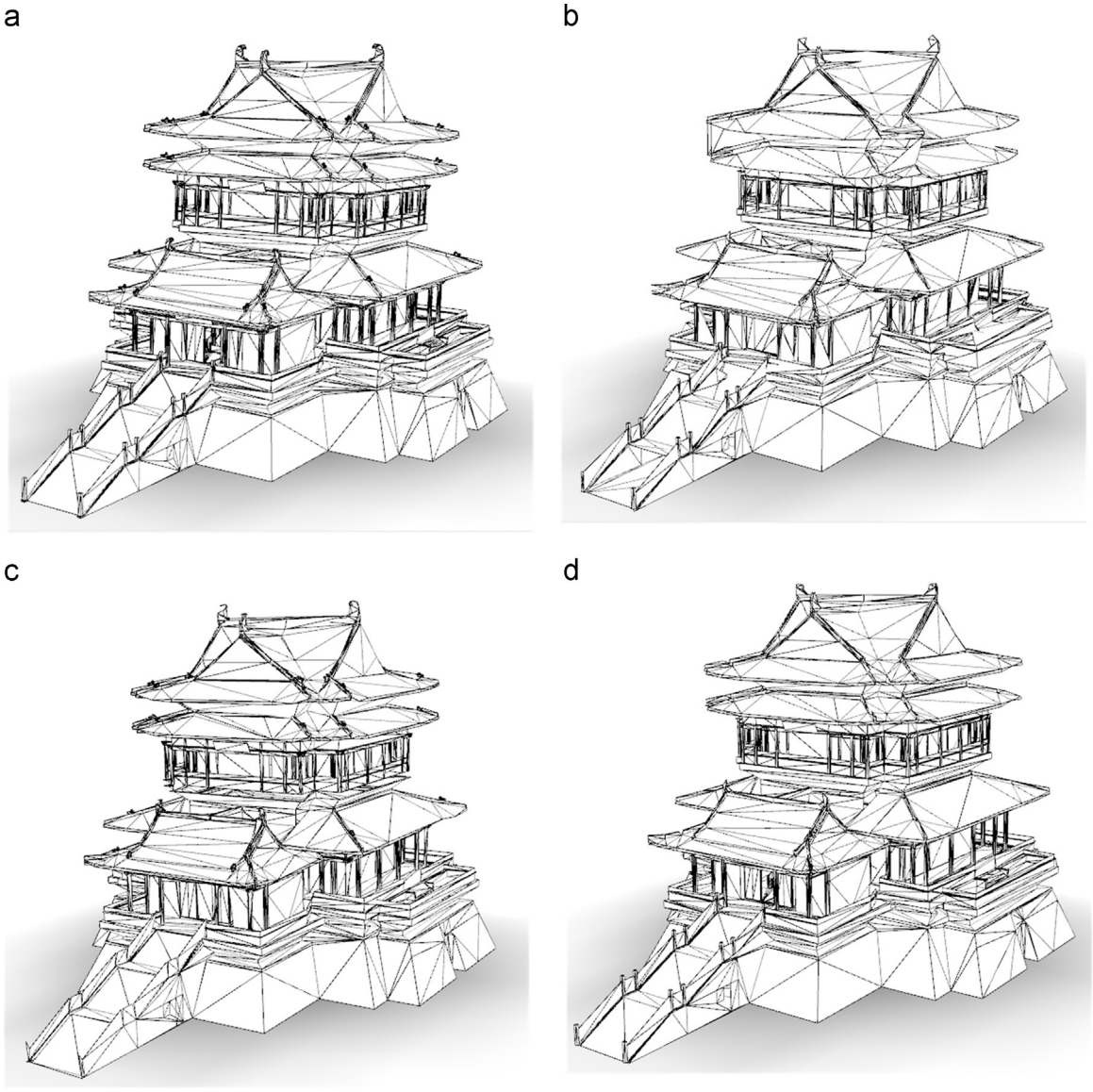

Fig. 4. Wireframe comparison diagrams of simplified results based on diverse algorithms. (a) Initial mesh model before simplification. (b) Traditional vertex clustering simplification. (c) Simplified result with RGB error metric. (d) Simplified result with UV error metric.

4.1. Statistical analysis of algorithm performance

As for model simplification performance, the statistical results mainly focus on the reduction rates of the vertices and triangles in multiple meshes, and the algorithm running time is recorded to evaluate the simplification efficiency. Taking the main building of the Overlooking Hall model in Jiangsu Province of China as an example, Table 1 records the simplification details of vertices and triangles in 13 meshes. The experimental results show that for the same threshold, diverse meshes will adjust the simplification rate according to their respective vertex distribution and triangle size. Under the premise of reducing obvious texture distortion and damage to the framework of the model, the vertex reduction rate is up to 71.07%, while the triangle reduction rate is up to 65.11%. Statistics indicate that via texture-related vertex clustering simplification, the entire model reduces almost two thirds of the vertices and triangles. The data quantity is also reduced from 2 658 kb to 781 kb. The model simplification rate basically satisfies the rendering requirement of the refined building models in complex 3D scene.

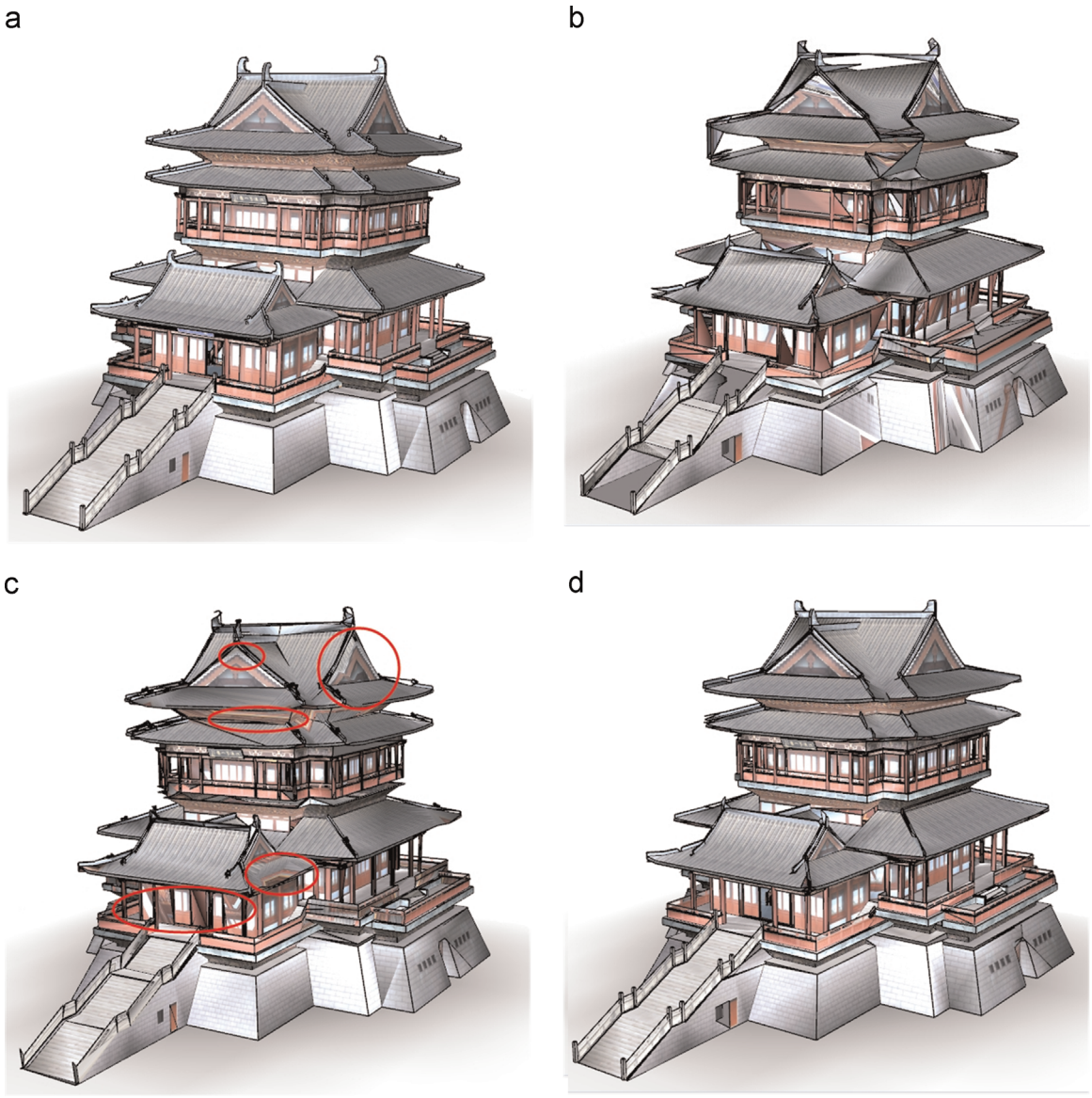

Fig. 5. Texture rendering comparison diagrams of simplified results based on diverse algorithms. (a) Initial mesh model before simplification. (b) Traditional vertex clustering simplification. (c) Simplified result with RGB error metric. (d) Simplified result with UV error metric.

In order to further demonstrate the algorithm adaptability, more pavilion platform building models with larger data volume are involved in this simplification experiment. In view of the realistic visual effects of the test models, the data redundancy is limited. Table 2 shows the simplification performance of this algorithm. From the experimental results we can draw a conclusion that when we aim at a particular reduction rate for the simplification, the running time of the entire clustering operations is directly proportional to the data volume of the initial model. Thus the model simplification efficiency can be estimated and predicted based on the initial model data volume and the preset reduction rate. In other words, the running time is controllable, which meets the demands of applications in practice. Table 3 shows the impact of different UV error threshold settings on running time & reduction rate. The models were simplified within a fixed geometry error metric 0.4. The results show that, with bigger error threshold settings, coarser clustering cells will be generated and more clustering operations need to be performed, resulting in higher reduction rate and more running time. But the impact is disproportionate.

4.2. Simplification quality comparison of different algorithms

To check the visual effects of the algorithm in terms of texture feature protection, we conducted an experiment to compare this algorithm with the traditional vertex clustering simplification and RGB-related simplification. The same model is used as the raw data and the same simplification rate is predetermined, while three different algorithms are adopted to control the simplification. Figs. 4 and 5 respectively show the comparison diagrams of different model simplification approaches in both wireframe mode and texture rendering mode.

As is shown in Fig. 4, in wireframe mode, the contour structure and the geometry feature are both better protected in the RGB-related algorithm and the UV-related algorithm, compared with the partially damaged result generated by traditional vertex clustering. But when it comes to the texture rendering mode, the disadvantages of the RGB-related algorithm emerge gradually. With an RGB error metric threshold 0.15, it makes contributions to protecting partial texture details, but there exists obvious texture mapping errors marked by the red circles in Fig. 5c, especially in the ridge of a roof for a pavilion platform building model. In

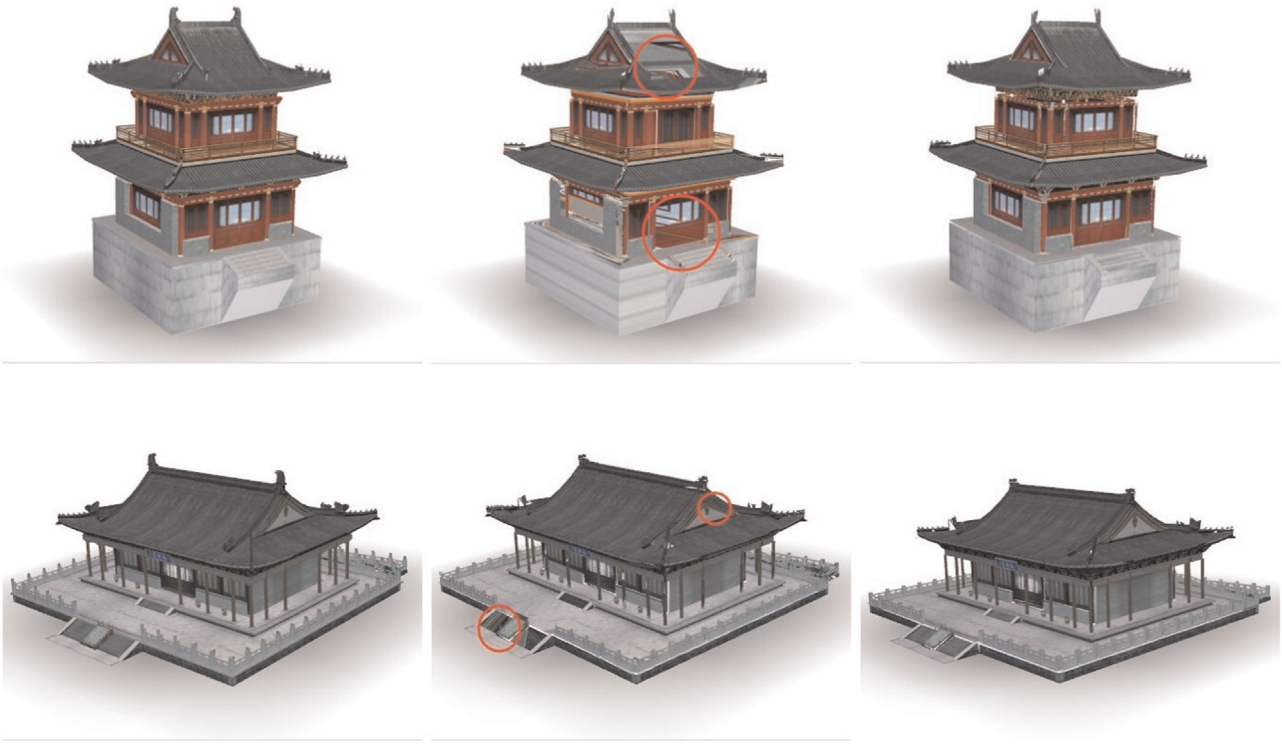

Fig. 6. Simplified results of other models in comparison.

Fig. 5d, texture distortions have been effectively controlled with an UV error metric threshold 0.05. Fig. 6 shows another two simplification comparison experiments. The origin models, the RGB-related simplified results and the UV-related simplified results are shown from left to right in turn. The two algorithms share similar reduction rates (about 50%). It is not difficult to see that texture distortions exist more or less in the simplified results with the RGB-related algorithm, and the obvious distortions have been marked with red circles. In comparison, our algorithm reduces damage to the visual effects of the simplified results under the premise of maintaining considerable reduction rate. That is because the color coordinates only reflect part of the texture information. In a composite texture block that consists of multiple texture patches, even if two vertices share the same geometry position and texture color information, it is possible that they are mapped into diverse texture patches coordinates. Once one of the vertices is replaced by the other one in clustering, it will cause serious texture distortion, which can badly influence the visual effects of the 3D models. By contrast, since we introduce the UV texture coordinates into the simplification, it will protect the texture mapping in 3D models from distortion to a certain extent.

Compared with the RGB color error metric, the UV texture error metric has one less dimension but stricter error control to preserve the texture information. Since it is no longer necessary to pick up the corresponding RGB value in the texture image for every vertex, it reduces the computational complexity effectively, and the algorithm efficiency is improved correspondingly.

5. Summary and future work

This paper proposes a novel model simplification algorithm based on vertex clustering in consideration of texture information. In the process of adaptive vertex subdivision, vertex distributions in both geometry space and texture space are taken into account. As for the error control, geometry error metric and texture error metric are introduced into this simplification process. Corresponding thresholds are used to decide whether to perform node split or vertex clustering. After describing the algorithm theory in details, we compared our approach with the traditional ones, mainly focusing on the differences of simplification performance and simplification quality. The experimental results prove that our approach has a certain inhibiting effect on both geometry deformation and texture distortion.

In fact, the model simplification is a process of losing detail information (including geometry details and texture details) gradually. It compresses the data quantity by omitting less important details, so geometry deformation and texture distortion are inevitable, but the extent can be controlled. The significance of introducing the error metrics lies in that it gives priority to simplify the parts which have less impact on the visualization. It makes a balance between the reduction rate and the model fidelity.

This algorithm is applicable to the simplification of triangular mesh models. The polygon mesh models must be preprocessed by triangulation. In addition, the vertex clustering algorithm does not take the topology into account, so too large clustering cells may cause broken structure. There exists very narrow cracks between the roof and the walls in the above simplified building models, but fortunately it does not affect the overall appearance of the models. So it is important to set reasonable clustering cell size and error thresholds in consideration of the sophistication degree and data quantity of the models. The future research can focus on this, and good data organization scheme for progressive transmission over the network environment is expected to be proposed.

Acknowledgments

This research is supported by the National Natural Science Foundation of China (No. 40801163 and No. 41023001) and the Fundamental Research Funds for the Central Universities (No. 2014619020203). The authors would like to thank the anonymous

reviewers and the editor for their comments that strongly improved this paper.

References

Cabral, M., Lefebvre, S., Dachsbacher, C., Drettakis, G., 2009. Structure-preserving reshape for textured architectural scenes. Comput. Graph. Forum 28, 469–480.

Cao, W.Q., Bao, H.J., Peng, Q.S., 2000. Level of detail model by merging near-coplanar faces based on Gauss sphere. Ruan Jian Xue Bao/J. Softw. 11, 1607–1613.

Chan, K.F., Kok, C.W., 2001. Mesh simplification by vertex cluster contraction. In: Shum, H.Y., Liao, M. Chang, S.F. (Eds.), Advances in Mutlimedia Information Processing-Pcm 2001, Proceedings. Springer-Verlag Berlin, Berlin, pp. 1114–1119.

Chang, R., Butkiewicz, T., Ziemkiewicz, C., Wartell, Z., Ribarsky, W., Pollard, N., 2008. Legible simplification of textured urban models. IEEE Comput. Graph. Appl. 28, 27–36.

Esteve, J., Brunet, P., Vinacua, A., 2000. Generation of Solid Multiresolution Models. Springer-Verlag, Berlin, Berlin.

Feng, X., Zhou, M., 2009. Algorithm for simplification of 3D models with texture. Jisuanji Fuzhu Sheji Yu Tuxingxue Xuebao/J. Comput.-Aided Des. Comput. Graph. 21, 842–846+852.

Forberg, A., 2007. Generalization of 3D building data based on a scale-space approach. ISPRS J. Photogramm. Remote Sens. 62, 104–111.

Franc, M., Skala, V., 2002. Fast algorithm for triangular mesh simplification based on vertex decimation. In: Proceedings of the International Conference on Computational Science. ICCS 2002, 21–24 April 2002. PART 2 ed. Springer Verlag, Amsterdam, Netherlands, pp. 42–51.

Funkhouser, T.A., Sequin, C.H., 1993. Adaptive display algorithm for interactive frame rates during visualization of complex virtual environments. In: Proceedings of the ACM SIGGRAPH '93 Conference on Computer Graphics. 1–6 August 1993. Publ by ACM, Anaheim, CA, USA, pp. 247–254.

Heckbert, P.S., Garland, M., 1999. Optimal triangulation and quadric-based surface simplification. Comput. Geom.: Theory Appl. 14, 49–65.

Hoppe, H., 1996. Progressive meshes. In: Proceedings of the 23rd Annual Conference on Computer Graphics and Interactive Techniques. ACM, pp. 99–108.

Hoppe, H., 1997. View-dependent refinement of progressive meshes. In: Proceedings of the 1997 Conference on Computer Graphics. SIGGRAPH, 3–8 August 1997. ACM, Los Angeles, CA, USA, pp. 189–198.

Hoppe, H., 1999. Optimization of mesh locality for transparent vertex caching. In: Proceedings of the 26th Annual Conference on Computer Graphics and Interactive Techniques. ACM Press/Addison-Wesley Publishing Co., pp. 269–276.

Kobbelt, L., Campagna, S., Seidel, H.-P., 1998. General framework for mesh decimation. Proc.-Graph. Interface, 43–50.

Li, J., Chen, Y., 2008. A fast mesh simplification algorithm based on octree with quadratic approximation. In: Proceedings of the 9th International Conference for Young Computer Scientists. ICYCS, 18–21 November 2008. Inst. of Elec. and Elec. Eng. Computer Society, Zhang Jia Jie, Hunan, China, pp. 775–780.

Li, N., Gao, P.D., Lu, Y.Q., Li, A.M., Yu, W.H., 2008. A New Adaptive Mesh Simplification Method Using Vertex Clustering with Topology-and-Detail Preserving. IEEE Computer Soc, Los Alamitos.

Lindstrom, 2000. Out-of-core simplification of large polygonal models. In: Proceedings of the 27th annual Conference on Computer Graphics and Interactive Techniques. ACM Press/Addison-Wesley Publishing Co., pp. 259–262.

Mao, B., Ban, Y.F., Harrie, L., 2011. A multiple representation data structure for dynamic visualisation of generalised 3D city models. ISPRS J. Photogramm. Remote Sens. 66, 198–208.

Ribelles, J., Lopez, A., Belmonte, O., 2010. An improved discrete level of detail model through an incremental representation. In: Proceedings of the 8th Theory and Practice of Computer Graphics Conference. TPCG 2010, 6–8 September 2010. Eurographics Association, Sheffield, United kingdom, pp. 59–66.

Royan, J., Balter, R., Bouville, C., 2007. Hierarchical representation of virtual cities for progressive transmission over networks.

Shen, X., Zhao, X., Zhao, Q., 2007. Feature-preserved progressive texture-mesh in digital museum. Jisuanji Yanjiu yu Fazhan/Comput. Res. Dev. 44, 1097–1104.

Thiemann, F., Sester, M., 2006. 3d-Symbolization Using Adaptive Templates. ISPRS Technical Commission II Symposium, Vienna, Austria, pp. 109–113.

Tsuchie, S., Hosino, T., Higashi, M., 2014. High-quality vertex clustering for surface mesh segmentation using Student-t mixture model. Comput.-Aided Des. 46, 69–78.

Ungvichian, V., Kanongchaiyos, P., 2011. Mesh Simplification Method Using Principal Curvatures and Directions. CMES-Comput. Model. Eng. Sci. 77, 201–219.

Wang, W.Y., Zhang, Y.J., 2010. Wavelets-based NURBS simplification and fairing. Comput. Methods Appl. Mech. Eng. 199, 290–300.

Xie, J.H., Zhang, L.Q., Li, J., Wang, H., Yang, L., 2012. Automatic simplification and visualization of 3D urban building models. Int. J. Appl. Earth Obs. Geoinf. 18, 222–231.

Zhang, M., Zhang, L.Q., Mathiopoulos, P.T., Xie, W.Q., Ding, Y.S., Wang, H., 2012. A geometry and texture coupled flexible generalization of urban building models. ISPRS J. Photogramm. Remote Sens. 70, 1–14.