Detail-Preserving High-Quality Mesh Simplification Algorithm for Textured Models Postprint

Authors: Li Shijun, Xiaotong Jiang, Tang Hui

Date: 2018-11-29T00:00:00+00:00

Abstract

Currently, 3D scene applications are becoming increasingly widespread, and complex models place considerable pressure on real-time rendering, making model simplification a necessary step. To address the problem that most algorithms tend to lose model detail features at high simplification ratios, we introduce the concept of vertex sharpness and propose an improved collapse cost based on the QEM (quadric error metric) collapse cost, which can better preserve model detail features. Simultaneously, to tackle the issue that most simplification algorithms do not incorporate texture processing, we introduce a texture variation factor to better retain texture details. Building upon the simplification, we also propose a mesh local optimization algorithm to solve the problem of low mesh quality after simplification. Experimental results demonstrate that our algorithm can not only maintain model detail features and texture integrity, but also produce high-quality mesh models.

Full Text

Preamble

High-Quality Simplification Algorithm for Textured Models with Detailed Feature Preservation

Li Shijun[1], Jiang Xiaotong[1], Tang Hui[2]†

[1]School of Instrument Science & Engineering, [2]School of Computer Science & Technology, Southeast University, Nanjing 210096, China

Abstract: With the increasing prevalence of 3D scene applications, complex models place significant pressure on real-time rendering, making model simplification a necessary step. To address the problem that most algorithms tend to lose detailed features at high simplification rates, this paper introduces the concept of vertex sharpness and proposes an improved folding cost based on

the QEM (Quadric Error Metric) cost, which better preserves model details. Additionally, to tackle the issue that most simplification algorithms do not incorporate texture processing, a texture variation factor is introduced to retain more texture details. Building upon the simplification process, a mesh local optimization algorithm is also proposed to address the problem of low mesh quality after simplification. Experimental results demonstrate that the proposed algorithm not only preserves model details and texture integrity but also yields high-quality mesh models.

Keywords: 3D model; mesh simplification; vertex sharpness; quadric error metric; local optimization

0 Introduction

With the rise of Internet-based home furnishing, 3D scene applications are becoming increasingly common. In the interior design industry, real objects are often scanned for 3D reconstruction, producing highly detailed models. However, these models typically contain excessive triangular faces, which hinder real-time rendering on mobile devices, making model simplification essential.

Mesh is the fundamental component of 3D models, and model simplification essentially involves simplifying the mesh, with triangular meshes being the most common representation. Mesh simplification algorithms can be broadly categorized into two types: vertex resampling and element deletion. Vertex resampling partitions the model into regions, removes internal vertices within each region, inserts new vertices, and performs retriangulation, controlling simplification rate by adjusting the number of resampling points. Element deletion progressively removes elements from the triangular mesh, which can be further divided into vertex deletion methods [1], triangle contraction methods [2,3], and edge collapse methods [4,5]. Among these, edge collapse has become the most widely used simplification algorithm due to its speed, efficiency, and ability to preserve the original model' s topological structure.

Hoppe [6] first proposed using an energy function to guide model simplification, demonstrating effective results on data models from 3D scanners, though with low efficiency. Garland et al. [7] introduced the use of quadric error to measure edge collapse cost, achieving fast simplification but producing overly uniform results that easily lose detail features. Ozaki et al. [8] extended the QEM algorithm using a machine learning-based linear classifier to partition meshes for simplifying massive models, yet detail features remained prone to loss. To address this issue, Ho et al. [9] proposed a controllable mesh simplification algorithm where users could select regions to preserve details, but this method is cumbersome and demands high user expertise. Huang et al. [10] presented a progressive mesh simplification algorithm that uses normal vector variations of one-ring and two-ring triangles as the collapse cost, preserving detail features well. However, by focusing solely on detail preservation, it fails to adequately

constrain the simplification order in flat regions. Wang et al. [11] proposed an edge curvature-based simplification algorithm that incorporates edge curvature into the quadric error to increase collapse cost in sharp regions, but their edge curvature estimation is overly simplistic. Qian et al. [12] approached simplification from the perspective of local triangle roughness, using triangle folding to simplify models. However, due to the inherent limitations of triangle folding in preserving features, this method is less effective than edge collapse for retaining model details. Liu et al. [13] introduced vertex curvature to enhance feature region preservation, achieving good results. However, their method only considers the maximum normal vector angle and total area of a vertex' s one-ring neighborhood, neglecting the area of the triangle causing the maximum angle, which can lead to misjudgment. Moreover, none of the aforementioned algorithms consider texture variation, yet in practical applications, texture preservation is essential for visual experience.

To solve these problems, this paper introduces the concept of vertex sharpness to improve the Q value of vertices in the QEM algorithm, enabling accurate identification of sharp vertices and increasing their collapse cost while effectively constraining flat regions. Additionally, a texture variation factor is introduced to maximize texture integrity preservation without increasing algorithmic complexity. Furthermore, a mesh local optimization algorithm is proposed to address the issue of low mesh quality caused by non-uniform simplification. Experimental results show that the proposed algorithm effectively preserves model details and texture integrity while producing uniformly distributed meshes with good visual effects.

---

2 Improved Simplification Cost

2.1 Basic Concepts

Edge Collapse Process: Figure 1 [Figure 1: see original paper] illustrates the edge collapse operation. The algorithm sorts all edges in the triangular mesh by their deletion cost in ascending order. Each operation selects the edge at the top of the queue (i.e., with the smallest collapse cost), such as edge (v_1, v_2) shown in Figure 1. The two vertices v_1 and v_2 are merged into a new vertex v that minimizes the error, deleting vertices v_1 and v_2, adding new point v, removing the two triangles sharing both vertices, deleting the corresponding edges, and updating the topological information of surrounding vertices and edges. This process repeats until the number of remaining triangles meets the requirement.

One-Ring Neighborhood: The one-ring neighborhood of a vertex V refers to the set of all triangles containing that vertex. As shown in Figure 2 [Figure 2: see original paper], all triangles constitute the one-ring neighborhood of vertex V_0.

Boundary and Complex Vertices: These are defined as follows: Count

the occurrence frequency of each adjacent vertex in all triangles adjacent to the target vertex. If all adjacent vertices appear exactly twice in the adjacent triangles, the vertex is a regular vertex (e.g., vertex A in Figure 3 [Figure 3: see original paper]). If an adjacent vertex appears only once in all adjacent triangles, both the adjacent vertex and the target vertex are boundary vertices (e.g., vertices B and C in Figure 3). If an adjacent vertex appears more than twice (three times), both the adjacent vertex and the target vertex are complex vertices (e.g., vertices D and E in Figure 3). If a vertex has no adjacent triangles, it is an isolated vertex (e.g., vertex F in Figure 3).

Notation: The following symbols are used throughout this paper: - v_t denotes the t-th vertex of the model mesh - S_{tvi} denotes the area of the i-th triangle in the one-ring neighborhood of vertex v_t - N_{tvi} denotes the normal vector of the i-th triangle in the one-ring neighborhood of vertex v_t - $S_{vtex,tvi}$ denotes the texture area corresponding to the i-th triangle in the one-ring neighborhood of vertex v_t - S_{tv} denotes the sum of areas of all triangles in the one-ring neighborhood of vertex v_t - $\langle a, b\rangle$ denotes the angle between vectors a and b

2.2 Vertex Sharpness Calculation

In models with numerous detail features, using only quadric error metric causes sharp features to be easily lost. Therefore, this paper introduces the concept of vertex sharpness and integrates it into the vertex matrix to effectively preserve these details at high simplification rates.

As shown in Figure 4 [Figure 4: see original paper], in regions with sharp vertices, the normal vector angles between adjacent triangles are large, whereas in flat regions these angles are relatively small, becoming zero when two faces are perfectly flat. Thus, vertex sharpness σ_{tv} can be reflected by the weighted average of normal vector angles between all triangle pairs in its one-ring neighborhood:

$$\sigma_{tv} = \frac{1}{n_{tv}} \sum_{i=1}^{n_{tv}} \sum_{j=i+1}^{n_{tv}} \langle N_{tvi}, N_{tvj} \rangle \cdot \frac{S_{tvi} + S_{tvj}}{2S_{tv}}$$

where n_{tv} is the number of triangles in the one-ring neighborhood of vertex v_t. In this formula, the normal vector angles between triangles in the vertex' s one-ring neighborhood are computed sequentially and averaged with area weighting. The area weighting is included because although some triangle pairs may have large normal vector angles, if the triangles themselves are small, their loss during simplification is minimal. Therefore, the contribution of such angles should be appropriately reduced to more accurately reflect the region' s sharpness.

2.3 Texture Factor

Most existing mesh simplification algorithms do not consider texture variation, yet texture is an essential component of visual experience in practical appli-

cations. After studying texture importance, this paper proposes the following texture factor formula:

$$T_{tv} = \frac{\sum_{i=1}^{n_{tv}} S_{vtex,tvi}}{\sum_{i=1}^{n_{tv}} S_{tvi}}$$

This formula considers that if a triangle' s texture area occupies a large proportion of its geometric area, simplification in that region will cause significant texture loss. Thus, the formula expresses the importance of texture in that region. During simplification, texture coordinates undergo the same update operations as vertex coordinates.

2.4 Improved Edge Collapse Cost

Building upon the QEM edge collapse cost, this paper presents an improved edge collapse cost function that integrates vertex sharpness and texture variation factors directly into the vertex' s Q matrix, avoiding additional computation during simplification and reducing algorithmic complexity. The improved edge collapse cost is:

$$\varepsilon_{v_{1v}2} = (v^T(Q(v_1) + \sigma_{v_1}R + T_{v_1}A)v + v^T(Q(v_2) + \sigma_{v_2}R + T_{v_2}A)v)$$

where R and A are adjustment matrices. To simplify, the edge collapse cost can be defined as the sum of the quadric errors of the two vertices:

$$\varepsilon_{v_{1v}2} = Q(v_1) + Q(v_2) + \sigma_{v_1}R + \sigma_{v_2}R + T_{v_1}A + T_{v_2}A$$

Let $M = Q(v_1) + Q(v_2) + \sigma_{v_1}R + \sigma_{v_2}R + T_{v_1}A + T_{v_2}A$ be the error matrix. The collapse cost for each edge is:

$$\varepsilon_{v_{1v}2} = v^T M v$$

For each edge, three candidate points (the left endpoint, right endpoint, and midpoint) are evaluated, and the point with the minimum cost is selected as the collapse target. This cost value is recorded and the edge is inserted into a priority queue.

---

3 Mesh Local Optimization

Due to inherent limitations of the edge collapse algorithm, despite preserving model details well in the improved simplification algorithm, the resulting mesh quality is generally low, often producing narrow triangles that cause visual discrepancies and complicate subsequent processing. Therefore, based on studies

in [14,15], this paper proposes a local mesh optimization algorithm to reduce the number of low-quality meshes.

3.1 Mesh Quality Criterion

Mesh quality is typically determined by triangle regularity [16], which measures how close a triangle is to being equilateral. The regularity of a triangle can be estimated using:

$$r = \frac{4\sqrt{3}A}{l_1^2 + l_2^2 + l_3^2}$$

where l_1, l_2, l_3 are the lengths of the triangle' s three edges and A is its area. The value r represents triangle regularity: $r = 1$ for an equilateral triangle, while $r \approx 0$ for extremely narrow triangles. This formula serves as the criterion for mesh optimization, with triangles having $r < 0.5$ defined as narrow triangles.

3.2 Optimization Algorithm

Global optimization would lose some model details, so this paper performs only local optimization to essentially eliminate narrow triangles.

3.2.1 Planar Mesh Optimization Reference [14] studied applications of Voronoi polygons for planar mesh optimization. As shown in Figure 5 [Figure 5: see original paper], the optimization steps are: 1. Identify all triangles in the one-ring neighborhood of vertex V 2. Construct the Voronoi polygon for vertex V 3. Move vertex V to the centroid of the Voronoi polygon to complete local optimization

Since Voronoi polygon construction is complex, the centroid of the one-ring neighborhood polygon can be used as a simpler optimization point when high precision is not required.

3.2.2 3D Mesh Flattening The algorithm in Section 3.2.1 only works for planar meshes, while this paper deals with 3D meshes. Reference [15] proposed a least-squares conformal mapping algorithm for surface parameterization (LSCM algorithm) that can effectively flatten complex meshes with stable, unique solutions. Therefore, this paper adopts the LSCM algorithm to flatten local regions requiring optimization.

As shown in Figure 6 [Figure 6: see original paper], conformal mapping ϕ maps a parametric domain D to a surface domain S, where tangent vectors at corresponding points are orthogonal and equal in length. Simply put, conformal mapping transforms the unit circle in the parametric domain to the unit circle on the surface' s tangent plane.

3D mesh flattening requires the original surface to be nearly planar to avoid large errors during subsequent recovery. Therefore, when flattening a vertex' s one-ring neighborhood, the region' s sharpness must be considered, and optimization should only be performed on relatively flat regions. This paper uses vertex sharpness as the criterion, optimizing only regions where sharpness is less than 0.3.

3.2.3 3D Mesh Recovery Reference [17] proposed a mesh reconstruction algorithm whose planar vertex recovery method can be adapted for 3D mesh recovery in this paper. The method is described as follows:

For original boundary vertices, their original 3D coordinates can be used. For interior new vertices v, the 3D coordinates are solved by: 1. Find three original boundary vertices with 3D coordinates (x_1, y_1, z_1), (x_2, y_2, z_2), (x_3, y_3, z_3) and corresponding planar coordinates (u_1, v_1), (u_2, v_2), (u_3, v_3), where point v lies inside the triangle formed by these three vertices 2. Let $A(u_1, u_2, u_3)$ denote the area of the triangle formed by points u_1, u_2, u_3 3. The corresponding 3D coordinate is:

$$v = k_1 v_1 + k_2 v_2 + k_3 v_3$$

where $k_i = \frac{A(u, u_j, u_k)}{A(u_1, u_2, u_3)}$ for cyclic indices i, j, k.

To ensure accurate 3D coordinate recovery, this paper uses all triangles in the vertex' s one-ring neighborhood polygon that contain point v, computes the 3D coordinates using the above formula for each, and averages them for more precise reconstruction.

3.3 Algorithm Flow

The complete algorithm is described as follows:

Initialization: Mark the model' s topological relationships and identify boundary or complex vertices. Set the collapse cost to maximum for any edge containing a boundary or complex vertex. For all collapsible edges, compute the collapse cost to both endpoints and the midpoint, select the minimum, record the collapse point, and insert this cost into a priority queue.

Simplification Process: Execute the operation at the top of queue Q, update collapse costs for all affected edges, recompute coordinates and texture coordinates of updated points, update topological relationships, and reinsert modified edges into the priority queue based on their new costs. Repeat until the triangle count reaches the target.

Optimization: After simplification, traverse the mesh vertices. For vertices with sharpness less than 0.3, check for narrow triangles in their one-ring neighborhood. If found, flatten the region using the LSCM algorithm, optimize using

the method in Section 3.2.1, then recover 3D coordinates using the method in Section 3.2.3. Continue until all vertices are optimized.

---

4 Experimental Results and Analysis

To validate the proposed algorithm, experiments were conducted using Microsoft Visual Studio 2013 and OpenGL on a computer with a 2.5 GHz Intel Core i5-3230M CPU and 5 GB RAM.

Figure 7 [Figure 7: see original paper] shows the effect of vertex sharpness and texture factors on a character model simplified to 60% of its original size. The original model contains 4,019 triangles. Figure 7(b) shows the result without vertex sharpness, where the sword model in the marked region is severely degraded. Figure 7(c) shows the result without texture factor constraints, where eye textures are noticeably distorted. Figure 7(d) presents the result using the proposed folding cost function, where both detail features and textures are well preserved, providing visual quality closest to the original model.

Figure 8 [Figure 8: see original paper] demonstrates the impact on a zombie model simplified to 65% (original: 12,686 triangles). Figures 8(b) and 8(c) show texture distortion in the necktie region when vertex sharpness or texture factors are omitted. The proposed algorithm (Figure 8(d)) maintains high consistency with the original model' s tie texture. These experiments verify the effectiveness of the proposed folding cost in preserving detail features and important textures.

For more comprehensive evaluation, three models with large triangle counts were compared against the QEM algorithm and Liu Jun' s algorithm [13] in terms of geometric error, average mesh quality, and computation time. Geometric error was measured using RMS error from metro [18], and mesh quality was estimated using equation (8).

Figure 9 [Figure 9: see original paper] shows an 80% simplification of a pigeon model (16,960 triangles). Figure 10 [Figure 10: see original paper] shows an 85% simplification of a dog model (21,293 triangles). Figure 11 [Figure 11: see original paper] shows an 85% simplification of a fish model (29,188 triangles). Visual comparison demonstrates that the proposed algorithm and Liu Jun' s algorithm produce higher quality results than QEM.

TABLE:1 Comparison of Error, Mesh Quality, and Simplification Time

Model	Method	Mesh Quality (Eq. 8)	Geometric Error (RMS)/10^{-2}	Time (ms)
Pigeon	QEM	0.72	0.18	45
	Liu Jun	0.81	0.21	52

Model	Method	Mesh Quality (Eq. 8)	Geometric Error (RMS)/10^{-2}	Time (ms)
	This paper	**0.85**	**0.22**	**58**
Dog	QEM	0.68	0.25	78
	Liu Jun	0.79	0.28	89
	This paper	**0.83**	**0.29**	**95**
Fish	QEM	0.65	0.31	112
	Liu Jun	0.76	0.35	128
	This paper	**0.81**	**0.36**	**135**

Table 1 reveals that the proposed method achieves the best mesh quality, followed by Liu Jun' s method, with QEM producing the lowest quality (generating more narrow triangles). Geometric errors are minimal for all three methods, remaining close to the original model. QEM has the smallest error, while the proposed method and Liu Jun's method are comparable but slightly larger, as local vertex movement during optimization introduces additional error. However, this small error trade-off for significantly higher mesh quality is worthwhile.

Both the proposed and Liu Jun' s methods include local optimization steps, resulting in slightly longer computation times than QEM, but all remain in the millisecond range with negligible differences. Notably, Figure 11(b) vs. 11(c) shows that without local optimization, the proposed algorithm' s non-uniform simplification produces more narrow triangles than QEM, underscoring the importance of mesh local optimization.

5 Conclusion

This paper presents a novel mesh simplification algorithm based on QEM that integrates vertex sharpness and texture factors into the edge collapse cost function. Experiments demonstrate that this improved cost function preserves both model details and important texture regions. Additionally, a local mesh optimization algorithm combining surface parameterization and Voronoi polygon applications is proposed to improve mesh quality after simplification. Experimental results show the algorithm effectively enhances mesh quality and visual perception. However, optimization is currently limited to regions with low vertex sharpness. Future work will investigate optimization methods for regions with more pronounced features.

References

[1] Luo Kun, Huang Kuidong, Lian Mingming. New method of triangular mesh simplification based on vertex culling [J]. *Microelectronics & Computer*, 2009, 26(5): 142-143.

[2] Duan Liming, Shao Hui, Li Zhongming, et al. Simplification method for feature preserving of efficient triangular mesh model [J]. *Optics and Precision Engineering*, 2017, 25(2): 460-468.

[3] Yan Tao. Triangular mesh simplification based on Gauss curvature [J]. *Computer Engineering & Science*, 2012, 34(12): 126-129.

[4] Wang Xiaozhe, Liu Yongji, Zhao Longbo, et al. Semi-automated and dynamic mesh simplification algorithm based on feature preserving [J]. *Application Research of Computers*, 2015(9): 2839-2843.

[5] Tang H, Shu H Z, Dillenseger J L, et al. Moment-based metrics for mesh simplification [J]. *Computers & Graphics*, 2007, 31(5): 710-718.

[6] Hoppe H. Progressive meshes [C]// *Proc of SIGGRAPH*. New York: ACM Press, 1996: 99-108.

[7] Garland M, Heckbert P. Surface Simplification Using Quadric Error Metric [C]// *Proc of the 24th Annual Conference on Computer Graphics and Interactive Techniques*. New York: ACM Press, 1997: 209-216.

[8] Ozaki H, Kyota F, Kanai T. Out-of-core framework for QEM-based mesh simplification [C]// *Proc of Eurographics Symposium on Parallel Graphics and Visualization*. Eurographics Association, 2015: 87-96.

[9] Ho T C, Chuang J H. A Simple Yet effective user controllable mesh simplification [J]. *Journal of Information Science & Engineering*, 2013, 29(3).

[10] Huang Jia, Wen Peizhi, Li Lifang, et al. Local error progressive mesh simplification algorithm for keeping detailed features [J]. *Journal of Computer Applications*, 2016, 36(6): 1704-1708.

[11] Wang Jun, Wang Xiuhui. Mesh simplification algorithms based on edge curvature metrics [J]. *Journal of China Jiliang University*, 2016, 27(1): 73-79.

[12] Qian Xunbo, Luo Lihong. Method of mesh simplification based on feature preserving [J]. *Mechanical & Electrical Engineering Magazine*, 2017, 34(10): 1224-1228.

[13] Liu Jun, Fan Hao, Sun Yu, et al. Mesh simplification algorithm combined with edge collapse and local optimization [J]. *Journal of Computer Applications*, 2016, 36(2): 535-540.

[14] Du Q, Faber V, Gunzburger M. Centroidal Voronoi tessellations: applications and algorithms [J]. *SIAM Review*, 1999, 41(4): 637–676.

[15] Lévy B, Petitjean S, Ray N, et al. Least squares conformal maps for automatic texture atlas generation [J]. *ACM Transactions on Graphics*, 2006, 21(3): 362-371.

[16] Liu Siyan, Liao Wenhe, Liu Hao. Triangle regularity measurement based on cosine sum of inner angles and mesh optimization [J]. *Mechanical Science and Technology for Aerospace Engineering*, 2007, 26(4): 420-423.

[17] Surazhsky V, Gotsman C. Explicit surface remeshing [C]// *Proc of Eurographics//ACM SIGGRAPH symposium on Geometry processing.* Aire-la-Ville, Switzerland: Eurographics Association, 2003.

[18] Cignoni P, Rocchini C, Scopigno R. Metro: Measuring error on simplified surfaces [J]. *Computer Graphics Forum*, 1998, 17(2): 167-174.

Note: Figure translations are in progress. See original paper for figures.