SCIENCE CHINA
Information Sciences

• RESEARCH PAPERS •
Special Focus

July 2010 Vol. 53 No. 7: 1312–1321
doi: 10.1007/s11432-010-4007-8

An improved method for progressive animation models generation

ZHANG ShiXue[1,2]*, ZHAO JinYu[1] & WU EnHua[2,3]

[1]*Changchun Institute of Optics, Fine Mechanics and Physics, Chinese Academy of Sciences, Changchun* 130033*, China;*
[2]*Department of Computer and Information Science, University of Macau, Macao, China;*
[3]*State Key Laboratory of Computer Science, Institute of Software, Chinese Academy of Sciences, Beijing* 100190*, China*

Received November 6, 2009; accepted December 29, 2009

Abstract In computer graphics, animated models are widely used to represent time-varying data. And the progressive representation of such models can accelerate the speed of processing, transmission and storage. In this paper, we propose an efficient method to generate progressive animation models. Our method uses an improved curvature sensitive quadric error metric (QEM) criterion as basic measurement, which can preserve more local features on the surface. We append a deformation weight to the aggregated edge contraction cost during the whole animation to preserve more motion features. At last, we introduce a mesh optimization method for the animation sequence, which can efficiently improve the temporal coherence and reduce visual artifacts between adjacent frames. The results show our approach is efficient, easy to implement, and good quality progressive animation models can be generated at any level of detail.

Keywords animation models, LOD, progressive mesh, mesh optimization

Citation Zhang S X, Zhao J Y, Wu E H. An improved method for progressive animation models generation. Sci China Inf Sci, 2010, 53: 1312–1321, doi: 10.1007/s11432-010-4007-8

1 Introduction

In computer graphics and virtual reality, more and more animation models, which are also called deforming surfaces, are frequently used from scientific applications (simulation or physical process) to animation (movie, games, morphing). In many cases, high-resolution geometric models are required to represent details and fine structures. However, some details might be unnecessary especially when viewing from a distance. Mesh simplification is a process of eliminating such unnecessary or redundant details from high-resolution 3D models by repeatedly removing vertices, edges, or faces. Most of the existing simplification algorithms are to deal with a single static mesh, while very little work has been done to address the problem of approximation for deforming models. In this paper, we propose an efficient method to generate progressive animation models.

To get different level of detail (LOD) animated meshes, one naive way is to simplify the models for each frame independently. This solution can generate the approximations with the minimum geometric distortion. However, since it does not exploit the temporal coherence of the data, it can involve the unpleasant visual artifact, causing the surface to vibrate and twitch. So the existing methods for simplifying static models cannot be directly applied to dynamic meshes.

*Corresponding author (email: shixuezhang@tom.com)

We therefore propose an improved method for generating progressive multiresolution animation models. This method is a better tradeoff between the temporal coherence and geometric distortion, i.e. we try to maximize the temporal coherence while minimizing the visual distortion during the simplification process. We introduce a curvature sensitive edge collapse cost measurement, which improves the famous QEM by considering both the vertex curvature and edge length. We append an additional deformation degree weight to the aggregated edge contraction cost to preserve areas with large deformation. Finally, a mesh optimization algorithm is performed to the simplified models, which can greatly improve the temporal coherence for the whole animation sequence. We demonstrate that this provides an efficient means of multiresolution representation for deforming surfaces over all frames of an animation.

The rest of this paper is organized as follows: Section 2 reviews the previous related works. Section 3 mainly introduces the procedure of our algorithm in detail and discusses its advantage. Section 4 shows our experimental results and makes comparison with previous methods. Finally, conclusion is made and future work is given in section 5.

2 Related work

Simplification and LOD. There are now extensive papers addressing the problem of mesh simplification, which can be treated as the basic operation to generate different LOD models. These methods can be roughly divided into five categories: vertex decimation [1, 2], vertex clustering [3], region merging [4], subdivision meshes [5], and iterative edge contraction [6–10]. A complete review of the methods has been given in [11–13]. Among these methods, the process of iteratively edge contraction is predominantly utilized. Representative algorithms include those from Hoppe [8] and Garland [6]. In such methods, a simple multiresolution structure is generated on the surface that can be used for adaptive refinement of the mesh [14]. Traditional mesh simplification algorithm works fine on a single static model, but it can not be directly applied to deforming meshes since no temporal coherence has been considered.

Mesh optimization. Most of the existing mesh optimization methods are based on the idea of local optimization and require an improvement of such mesh quality parameters as aspect ratio, area, etc. Hoppe et al. [15] described an energy minimization approach to solving the mesh optimization problem. The energy function consists of three terms: a distance energy that measures the closeness of fit, a representation energy that penalizes meshes with a large number of vertices, and a regularizing term that conceptually places springs of rest length zero on the edges of the mesh. In [16], a procedure was presented to improve the quality of complex polygonal surface meshes without an underlying smooth surface using numerical optimization. de Goes et al. [17] treated deformable surfaces using extended intrinsic Laplacian mesh smoothing to get well controlled triangulation that follows surface features. Recently, Liu et al. [18] proposed a non-iterative approach using global Laplacian operator to keep the fairness, and also use the constraint condition to reserve the fidelity to the mesh.

Approximation of animation models. Shamir et al. [19, 20] are the first to address the problem of simplifying deforming surfaces. They designed a global multiresolution structure named time-dependant directed acyclic graph (TDAG) which merges each individual simplified model of each frame into a unified graph. TDAG is a data structure that stores the life time of a vertex, which is queried for the connectivity updating. Unfortunately this scheme is very complex, and cannot be easily handled. Mohr and Gleicher [21] proposed a deformation sensitive decimation (DSD) method, which directly adapts the QEM algorithm [6] by summing the quadrics errors over each frame of the animation. This technique provides a pleasant result only when the original surfaces do not present strong deformation. Chen et al. [22] modified DSD method by considering only the maximum quadric errors for each frame during the simplification, but this method only works well on simple animation models. Kircher and Garland [23] proposed a multiresolution representation with a dynamic connectivity for deforming surfaces. By their method, the simplified model for the next frame is obtained by a sequence of edge-swap operations from the simplified model at the current frame. They treat a sequence of vertex contraction and their resulting hierarchies as a reclustering process [11]. This method seems to be particularly efficient because of its connectivity transformation. A similar approach has been used by Payan et al. [24], which starts from a simplification from the first frame and makes progressive updates over time. Tseng [25] used the temporal discrete shape operator to annlyze the animated surface variations. Huang et al. [26] proposed a method based on the deformation oriented decimation error metric and dynamic connectivity update. They used vertex tree to further reduce geometric distortion by allowing the connectivity to change. Their method can provide a good approximation of deforming surfaces, but requires a complex structure. Zhang and Wu [27] proposed a shape feature based method to deal with deforming meshes. Recently, Landreneau et al. [28] proposed a method for simplifying a polygonal character with an associated skeletal deformation, and Merry et al. [29] proposed an influence simplification method for animated characters. These two methods work well, but can be only applied to articulated meshes.

3 Algorithm

Our algorithm is composed of three parts: 1) Use a curvature sensitive error metric to measure the edge contraction cost. 2) Define a deformation degree weight during the animation to preserve areas with large deformation. 3) Propose a mesh optimization algorithm for the whole animation sequence to improve the temporal coherence. Next we will introduce each of the steps in detail.

3.1 Edge contraction measurement

Up to now, QEM [6] measurement is widely adopted for edge-collapse based simplification, so first we should have a quick review of it.

In QEM-based simplification, the sum of squared distance at each internal mesh vertex is computed. Assume that vertex v is the new vertex resulting from an edge contraction for edge (v_i, v_j). Then, the geometric error $\Delta(v)$ at v is defined as the sum of squared distances $\Delta_i^2(v)$ of the vertex v to certain faces on the original mesh. We represent $\Delta_i^2(v)$ as the squared distance between the vertex v and the plane of the i'th facet, where the plane's equation is $a_i x + b_i y + c_i z + d_i = 0$, and we represent its normal as $n_i = [a_i \quad b_i \quad c_i \quad d_i]^{\mathrm{T}}$. The $\Delta_i^2(v)$ can be written as

$$\Delta_i^2(v) = \left(\begin{bmatrix} a_i & b_i & c_i & d_i \end{bmatrix} \begin{bmatrix} x \\ y \\ z \\ 1 \end{bmatrix} \right)^2 = (n_i^{\mathrm{T}} v)^2 = (v^{\mathrm{T}} n_i)(n_i^{\mathrm{T}} v) = v^{\mathrm{T}}(n_i n_i^{\mathrm{T}}) = v^{\mathrm{T}} K_{p_i} v,$$

where $v^{\mathrm{T}} = \begin{bmatrix} x & y & z & 1 \end{bmatrix}$ and K_{p_i} is

$$K_{p_i} = \begin{bmatrix} a_i a_i & a_i b_i & a_i c_i & a_i d_i \\ b_i a_i & b_i b_i & b_i c_i & b_i d_i \\ c_i a_i & c_i b_i & c_i c_i & c_i d_i \\ d_i a_i & d_i b_i & d_i c_i & d_i d_i \end{bmatrix}.$$

Garland and Heckbert [6] call matrix K_{p_i} the fundamental error quadric. Thus, the error $\Delta(v)$ can be written as

$$\Delta(v) = \sum_{t=1}^{m} v^{\mathrm{T}} K_{p_i} v = v^{\mathrm{T}} Q_v v,$$

where $Q_v = \sum_{t=1}^{m} K_{p_i}$, and m is the number of facets to be considered. In applying QEM in edge contraction with v_f, the replacement for (v_i, v_j), $\Delta(v_f)$ is taken as

$$\Delta(v_f) = v_f^{\mathrm{T}} [Q_{v_i} + Q_{v_j}] v_f = v_f^{\mathrm{T}} Q_{ij} v_f. \tag{1}$$

For a vertex v of the original mesh, the K_{p_i}'s are for faces meeting at v. In simplification, as edges are contracted, sums of Q's are aggregated for each replacement vertex.

Such QEM method can achieve good result with a fast simplification speed. However, it has some deficiencies such as ignoring some important features and excessive simplification in some areas. We improve it by considering the vertex curvature and edge length together to be integrated into the quadric error matrix. The curvature represents the shape feature of a vertex and the length shows the influence region of the edge.

To get the curvature, we should calculate the normal n_v of the vertex. The simplest measure of the vertex's normal is the mean normal of its adjacent triangles. The triangle normal can be calculated by the equation as below:

$$n = \begin{pmatrix} v_2.x - v_1.x \\ v_2.y - v_1.y \\ v_2.z - v_1.z \end{pmatrix} \times \begin{pmatrix} v_3.x - v_2.x \\ v_3.y - v_2.y \\ v_3.z - v_2.z \end{pmatrix}.$$

Here v_1, v_2 and v_3 are three vertices of the triangle.

The vertex normal n_v can be calculated by the following equation:

$$n_v = \sum_{p \in planes(v)} n_p.$$

Here, $planes(v)$ is the adjacent triangles of the vertex v. $n_p = n/|n|$ is the unit normal of a triangle and the vertex normal n_v commonly is united immediately.

With the vertex normal n_v, we can calculate the vertex curvature with the following equation:

$$c_v = \max_{p \in planes(v)} \theta(n_v, n_p),$$

where $\theta(n_v, n_p)$ denotes the angle between n_v and n_p. c_v is called relative curvature, which represents the geometric importance of the vertex v. The vertex with a high value of c_v is more likely to hold salient feature of the model.

Based on the above equations, the feature value $F(i,j)$ of the edge (v_i, v_j) can be calculated as follows:

$$F(i,j) = l_{ij}^2 \times \left[1 + \frac{1}{2}(c_i + c_j)\right],$$

where l_{ij} is the length of edge (v_i, v_j). With this feature value, we can modify the edge contraction cost of eq. (1) into

$$\Delta'(v_f) = v_f^{\mathrm{T}} Q_{ij} v_f + \lambda \cdot F(i,j), \tag{2}$$

where λ is a coefficient to adjust the influence of feature value. Also we can define $\lambda \cdot F(i,j)$ as a 4×4 matrix as follows:

$$K_f = \begin{bmatrix} 0 & 0 & 0 & 0 \\ 0 & 0 & 0 & 0 \\ 0 & 0 & 0 & 0 \\ 0 & 0 & 0 & \sqrt{\lambda \cdot F(i,j)} \end{bmatrix}.$$

Thus we can rewrite eq. (2) as

$$\Delta'(v_f) = v_f^{\mathrm{T}} [Q_{ij} + K_f] v_f.$$

In this case, the collapse cost of edges in high-curvature region and with long length can be increased significantly. With the integrated feature matrix K_f, the edges with high collapse cost will be collapsed in posterior sequence.

3.2 Deformation measurement

The deformation sensitive decimation (DSD) algorithm addresses the deforming surface approximation by summing QEM costs across all frames [21]. The DSD contraction cost for an edge (v_i, v_j) is defined as

$$\mathrm{DSD}_{ij} = \sum_{t=1}^{f} (v_f^t)^{\mathrm{T}} Q_{ij}^t v_f^t,$$

where v_f^t minimizes the edge contraction cost for the edge (v_i, v_j) at frame t, and $Q_{ij}^t = Q_{v_i}^t + Q_{v_j}^t$.

In our algorithm, we add an additional deformation degree weight to the DSD cost, which can preserve large deformation areas. We use the change of frame-to-frame edge collapse cost to measure this deformation degree. In areas with large deformation, the change of the collapsing cost must be prominent, while collapsing cost may change slightly in areas with small deformation. By appending this additional deformation weight, we can postpone the edge contraction in areas with large deformation. The deformation weight is defined to be

$$\sum_{t=1}^{f} |\Delta_{ij}^t - \overline{\Delta}_{ij}|,$$

where Δ_{ij}^t is the collapse cost of edge of (v_i, v_j) in frame t, and $\overline{\Delta}_{ij}$ is the average collapse cost of edge (v_i, v_j) over all of the frames. We add this cost to the DSD contraction cost:

$$\cos t_{ij} = \mathrm{DSD}_{ij} + k_d * \sum_{t=1}^{f} |\Delta_{ij}^t - \overline{\Delta}_{ij}|,$$

where k_d is a user-specified coefficient to adjust the influence of deformation in the overall animation.

3.3 Mesh optimization

Next we propose an animated mesh optimization method. By readjusting triangle shapes, it can improve the temporal coherence between adjacent frames.

The basic idea for controlling the shapes of the mesh triangles is to iteratively apply averaging operations to its vertices. Assume that we have for every vertex v_i and each of its neighbours v_j a weight λ_{ij}. Then the optimized position v_i' of the vertex v_i is given by

$$v_i' = \sum_{v_j \in N(v_i)} \lambda_{ij} v_j,$$

where $N(v_i)$ is v_i's neighbour set. More precisely, we require that all weights λ_{ij} are positive and sum to unity

$$\sum_{v_j \in N(v_i)} \lambda_{ij} v_j.$$

So the above equation describes a convex combination and hence v_i' always lies inside the convex hull of its neighbours v_j, $v_j \in N(v_i)$.

For example, if all weights λ_{ij} are identical, then v_i' will be located at the barycentre of its neighbours and iteratively applying above equation to all vertices of M will lead to a uniform distribution of points and tends to create equilateral triangles. Instead, if we let λ_{ij} be the mean value coordinates [30] of v_i with respect to its neighbours v_j, then the shapes of the triangles are nicely preserved (see Figure 1).

In an animation mesh sequence, we want the triangles to have similar shapes between adjacent frames, and thus the output animation can maintain optimal temporal coherence. Inspired by the deformation transfer method of Sumner and Popovic [31], we realized that instead of computing the mean value weights from M, we can also compute the weights λ_{ij} from a second mesh N and thus "copy-and-paste" the shapes of the triangles in N to the triangles in M. Of course, this requires N to have the same connectivity as M. The simplified models generated using our method obviously satisfy this condition.

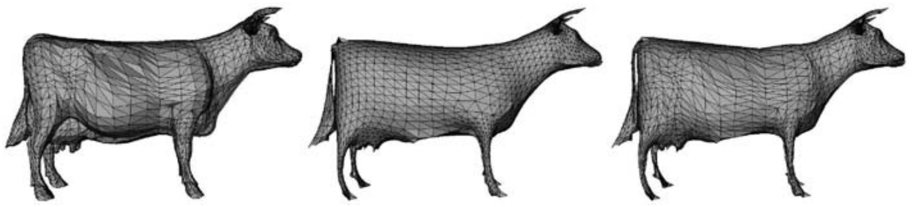

Figure 1 Applying averaging operators to a mesh (left) with uniform (middle) and mean value weights (right).

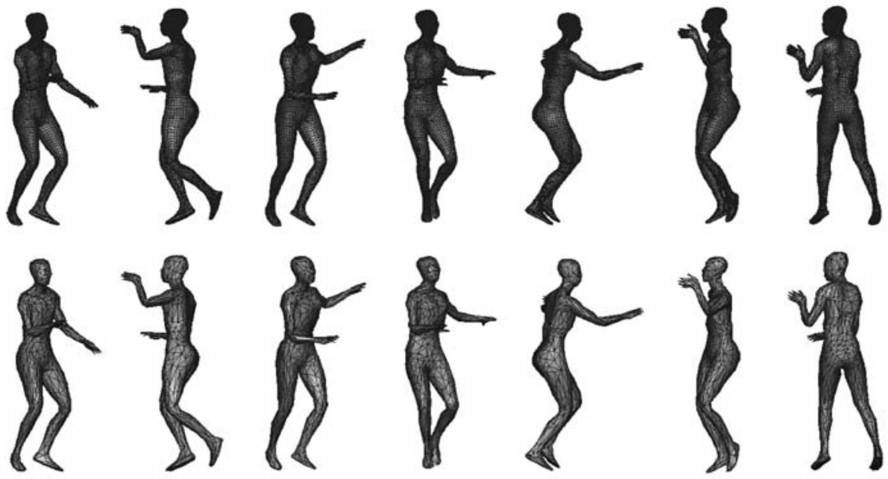

Figure 2 Dancer animation models with 48 frames. Upper row: original sequence with 7061 vertices. Bottom row: simplified models with 700 vertices (remove 90%).

In our method, we first compute the mean value coordinate weight of the first frame model, and then use it to move the vertices on the second frame model. Similarly, we use the weights calculated on the second frame model to move vertices in the third frame model, and so on. So we can transfer the triangle shapes in the current frame to the next frame. Since such reshape process is performed on each frame only once, it will not influence the outline feature. Finally, in the whole animation sequence, two adjacent frames have the same connectivity and similar triangle shapes, so the output visual artifact can be greatly reduced.

3.4 Algorithm outline

Our algorithm can be summarized into the following steps:

1. Calculate edge contraction cost on the original models using the error metric described in subsection 3.1.
2. For each edge over the whole animation sequence, compute the aggregated contraction cost as the DSD method.
3. Appending the deformation degree weight to the aggregated cost, and get the unified edge collapsing order.
4. Perform the edge collapse operation for each frame to get the desired resolution.
5. Use the mesh optimization algorithm described in subsection 3.3 to improve the temporal coherence of the animation sequence.

4 Experimental result

We test the result of our algorithm on a computer with Pentium4 3.2 G CPU and 4 GB memory, using VC2005 and OpenGL as programming environment. The dancer animation sequence is shown in Figure 2. Compared to the original animation sequence on the top row, the bottom row shows our simplified

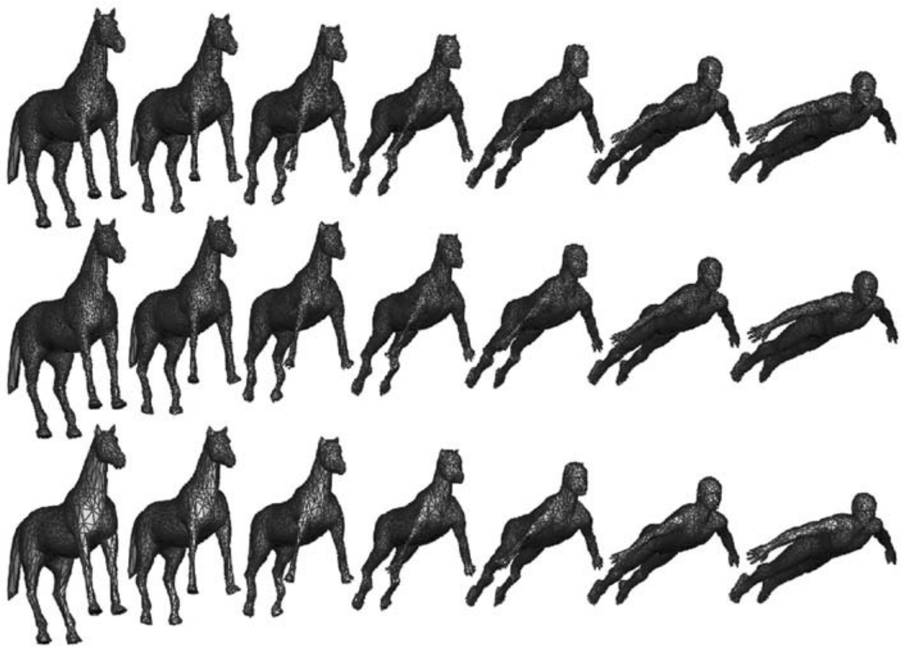

Figure 3 A horse-to-human morphing animation with 200 frames. Top: The original sequence (17489 vertices). Middle: 3200-vertices approximation. Bottom: 800-vertices approximation.

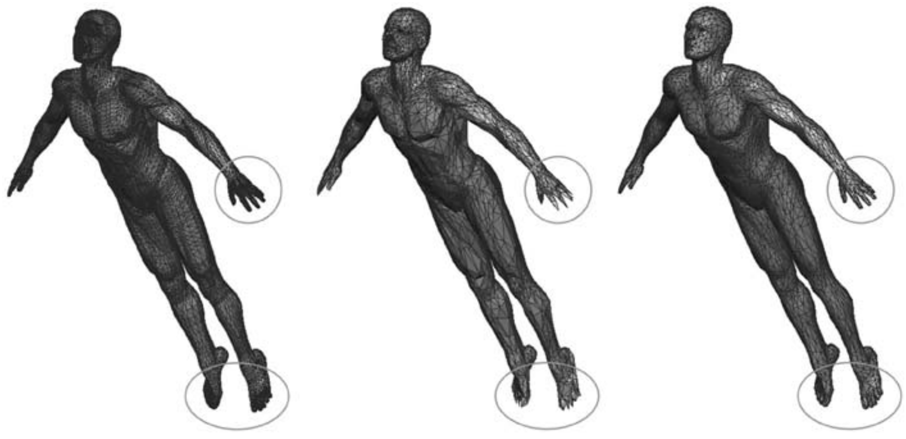

Figure 4 Comparison of the last frame of horse-to-man morphing sequence. Left: original model with 17489 vertices. Middle: the simplified versions (3200 vertices) generated using the method in [23]. Right: result of our method.

results with 90% of its components reduced. And most of the shape features in the original models are preserved well. Another more complicated horse-to-man animation is shown in Figure 3. It is composed of 200 frames, and its deformation is totally non-rigid. Compared to the original morphing models in the top row, we generate approximations with 80% and 95% of its components reduced respectively. Both the simplified versions can be rendered faithfully to the original ones, and also the geometric and motion features are preserved very well.

Next we compare our method with previous ones. In Figure 4, the results of method of [23] (middle) and our method (right) are compared. And our method obviously preserves the features of the hands and feet better. Because the hands and feet are morphing from the hoof of the horse, the deformation degree of this area is relatively strong, our method pays more attentions to such areas and can preservemore local and motion features. Figure 5 shows the simplification results of elephant-to-horse morphing animation. The horse model generated by our method preserves more features on the ears and tail; moreover the triangulation in such areas is more reasonable than the method of [23].

Figure 6 and Figure 7 show the RMS error statistics for the horse-gallop and horse-to-man animation. The RMS error line of our method (red) lies between DSD and independent QEM on the whole, and is

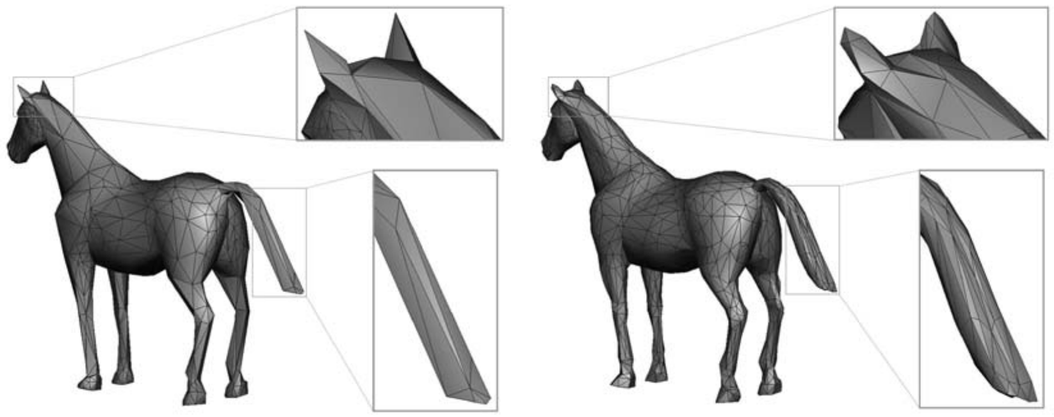

Figure 5 Comparison of the last frame of elephant-to-horse morphing sequence. Left: Result of [23]'s method. Right: result of our method.

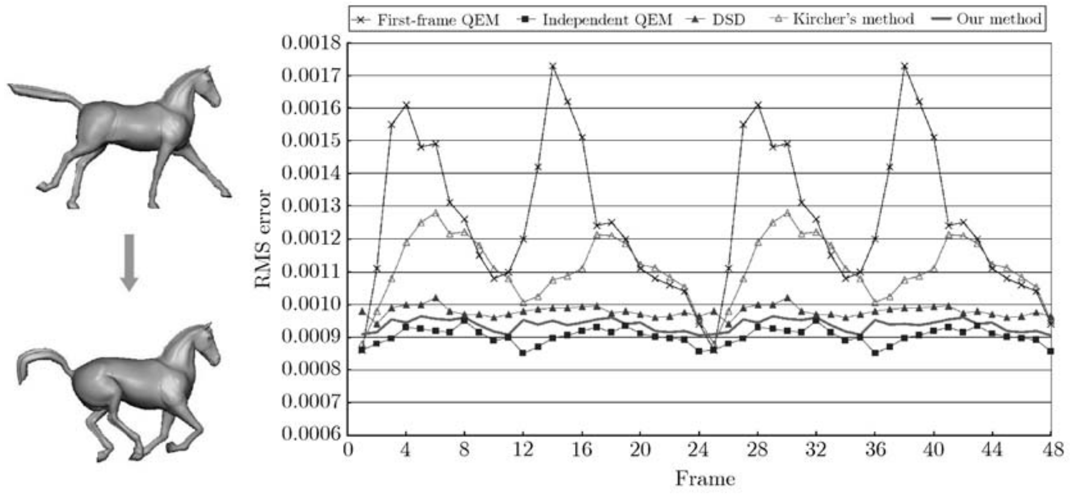

Figure 6 RMS error metric results of the horse-gallop animation (48 frames). Vertical axis indicates the error value, and the horizontal axis indicates the animation frame.

obviously lower than other methods. Especially in the horse-gallop animation, the RMS error of our result is very close to the independent QEM error. So we can demonstrate that our method can generate better result both in visual effect and error statistic.

Figure 6 and Figure 7 show the RMS error statistics for the horse-gallop and horse-to-man animation. The RMS error line of our method (red) lies between DSD and independent QEM on the whole, and is obviously lower than other methods. Especially in the horse-gallop animation, the RMS error of our result is very close to the independent QEM error. So we can demonstrate that our method can generate better result both in visual effect and error statistic.

To demonstrate the various applicability of our algorithm, we finally show an example of facial-expression animation in Figure 8. Our method can still generate faithful models for different levels of details.

The models generated by our method have the same connectivity for different frames; moreover the optimization process further conform the triangle shapes. So the temporal coherence of our result can be much better than the results of other methods.

5 Conclusion and future work

In this paper, we propose an improved method for generating progressive animated models. Given a sequence of meshes representing time-varying 3D data, our method produces optimized simplified versions

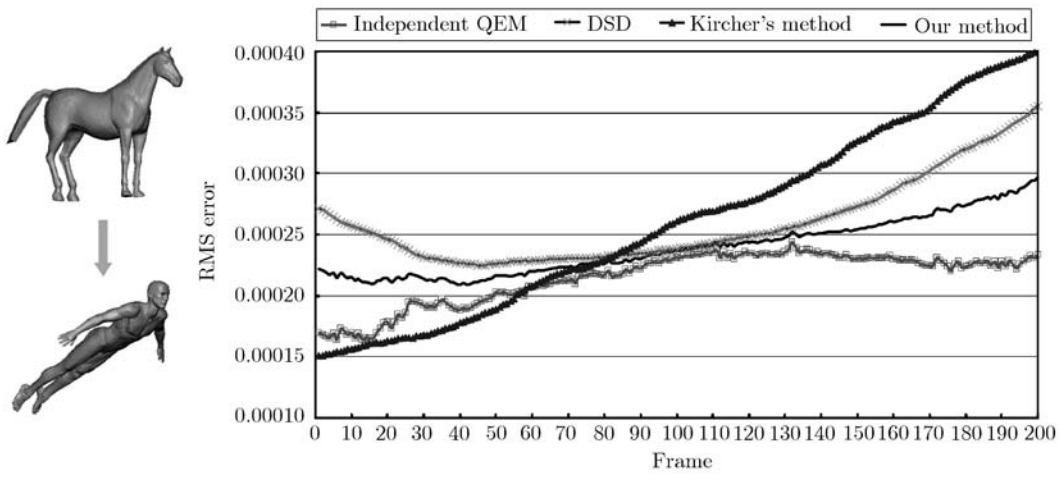

Figure 7 RMS error metric results of the horse-to-man animation (200 frames). Vertical axis indicates the error value, and the horizontal axis indicates the animation frame.

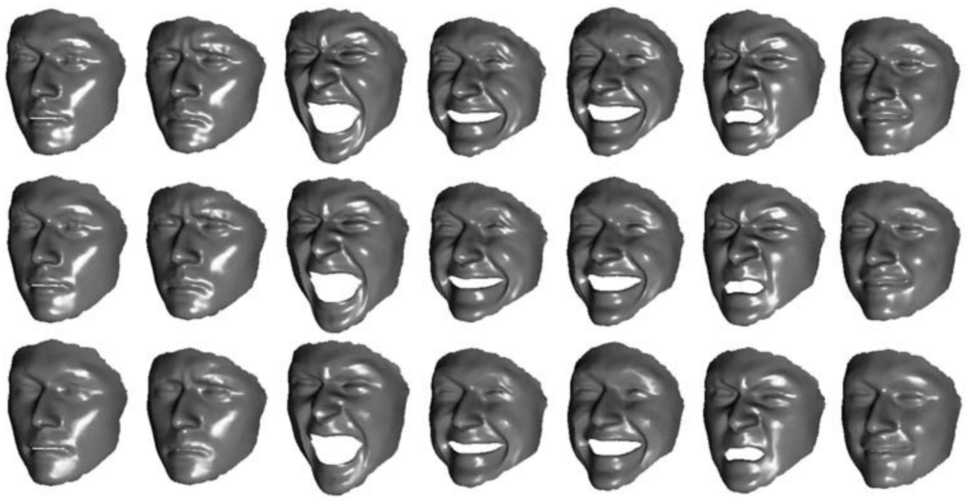

Figure 8 A face-expression animation with 192 frames. Top: The original sequence with 23725 vertices and 46853 faces. Middle: 3201-vertices and 5825-faces approximation (80% reduced). Bottom: 1200-vertices and 1874-faces approximation (95% reduced).

at any level of detail. We use an improved quadric error metric as our basic measurement, which can preserve more local features. By adding an additional deformation weight assessment, we successfully preserve the features in areas with large deformation. At last, a mesh optimization method is proposed to improve the temporal coherence of the animation sequence.

There are certainly further improvements that could be made to our algorithm. For example, we believe that there must be a way to extend our algorithm to be view-dependent.

Acknowledgements

This work was supported by the National High-Tech Research & Development Program of China (Grant No. 2008AA01Z301, 2008AA8080202), the National Natural Science Foundation of China (Grant No. 60833007), and PhD Studentship & Research Grant of University of Macau. We wish to thank Scott Kicher for providing the horse-to-man animation sequence, Robert W. Sumner and Igor Guskov for providing the horse-gallop, face-expression and dancer data.

References

1 Cohen J, Varshney A, Manocha D, et al. Simplification envelopes. In: ACM SIGGRAPH 1996 Conference Proceedings, New Orleans, LA, 1996. 119–128

2 Schroeder W J, Zarge J A, Lorensen W E. Decimation of triangle meshes. In: ACM SIGGRAPH 1992 Conference Proceedings, Chicago, 1992. 65–70

3 Low K L, Tan T S. Model simplification using Vertex-Clustering. In: ACM Symposium on Interactive 3D Graphics, New York, 1997. 75–82

4 Garland M, Willmott A, Heckbert P S. Hierarchical face clustering on polygonal surfaces. In: Proc ACM Symp Interactive 3D Graphics, New York, 2001. 49–58

5 Lee A, Moreton H, Hoppe H. Displaced subdivision surfaces. In: ACM SIGGRAPH 2000 Conference Proceedings, New York, 2000. 85–94

6 Garland M, Heckbert P S. Surface simplification using quadric error metrics. ACM SIGGRAPH 1997 Conference Proceedings, New York, 1997. 209–216

7 Guéziec A. Locally toleranced surface simplification. IEEE Trans Visual Comput Graph, 1999, 5: 168–189

8 Hoppe H. Progressive meshes. In: ACM SIGGRAPH 1996 Conference Proceedings, New Orleans, 1996. 99–108

9 Lee C H, Varshney A, David J. Mesh saliency. In: Proceedings of ACM SIGGRAPH, Los Angeles, 2005. 659–666

10 Yan J, Shi P, Zhang D. Mesh simplification with hierarchical shape analysis and iterative edge contraction. IEEE Trans Visual Comput Graph, 2004, 10: 142–151

11 Garland M. Multiresolution modeling: survey & future opportunities. In: Proceedings of Eurographic, Milano. 1999. 49–65

12 Luebke D, Reddy M, Cohen J. Level of Detail for 3-D Graphics. San Frausisco, CA: Morgan Kaufmann, 2002

13 Oliver M van K, Hélio P. A comparative evaluation of metrics for fast mesh simplification. Comput Graph Forum, 2006, 25: 197–210

14 Xia J C, Varshney A. Dynamic view-dependent simplification for polygonal models. In: IEEE Visualization 1996 Conference Proceedings, Los Alamitos, CA, 1996. 327–334

15 Hoppe H, DeRose T, Dunchamp T, et al. Mesh optimization. In: ACM SIGGRAPH 1993 Conference Proceedings, Anaheim, CA, 1993. 19–25

16 Garimella R, Shashkov M. Polygonal surface mesh optimization. Eng Comput, 2004, 20: 265–272

17 de Goes F, Bergo F P G, Falcao A X, et al. Adapted dynamic meshes for deformable surfaces. In: Brazilian Symposium on Computer Graphics and Image Processing, Manaus, Amazona, Brazil, 2006. 213–220

18 Liu L, Tai C, Ji Z, et al. Non-iterative approach for global mesh optimization. Comput Aided Design, 2007, 39: 772–782

19 Shamir A, Bajaj C, Pascucci V. Multiresolution dynamic meshes with arbitrary deformations. In: IEEE Visualization 2000 Conference Proceedings, Salt Lake City, Utah, 2000. 423–430

20 Shamir A, Pascucci V. Temporal and spatial level of details for dynamic meshes. In: Proceedings of ACM Symposium on Virtual Reality Software and Technology, Banff, Alberta, Canada, 2001. 77–84

21 Mohr A, Gleicher M. Deformation sensitive decimation. Technical Report, University of Wisconsin, 2003

22 Chen B Y, Cho S Y, Johan H, et al. Progressive 3D animated models for mobile & web uses. The J Society Art Sci, 2005, 4: 145–150

23 Kircher S, Garland M. Progressive multiresolution meshes for deforming surfaces. In: Proceedings of ACM SIGGRAPH/Eurographics Symposium on Computer Animation, New York, 2005. 191–200

24 Payan F, Hahmann S, Bonneau G P. Deforming surface simplification based on dynamic geometry sampling. In: Proceedings of IEEE International Conference on Shape Modeling and Applications, Lyon, France, 2007. 71–80

25 Tseng J L. Progressive compression and surface analysis for 3D animation objects using temporal discrete shape operator. In: International Conference on Information, Communications & Signal Processing, Singapore, 2007. 1–5

26 Huang F C, Chen B Y, Chuang Y Y. Progressive deforming meshes based on deformation oriented decimation and dynamic connectivity updating. In: Proceedings of ACM SIGGRAPH/Eurographics Symposium on Computer Animation, Vienna, Austria, 2006. 53–62

27 Zhang S, Wu E. A shape feature based simplification method for deforming meshes. Geomet Model Process, 2008, 4975: 548–555

28 Landreneau E, Schaefer S. Simplification of articulated meshes. In: Proceedings of EUROGRAPHICS, Munich, Germany, 2009

29 Merry B, Marais P, Gain J. Analytic simplification of animated characters. In: International Conference on Computer Graphics, Virtual Reality, Visualisation and Interaction in Africa (Afrigraph 2009), Pretoria, South Africa, 2009. 37–45

30 Floater M S. Mean value coordinates. Comput Aid Geometr Design, 2003, 20: 19–27

31 Sumner R W, Popovic J. Deformation transfer for triangle meshes. In: ACM SIGGRAPH 2004 Conference Proceedings, Los Angeles, 2004. 399–405