An Improved Method for Generating Multiresolution Animation Models

Shixue Zhang\textsuperscript{1,3}  
Enhua Wu\textsuperscript{1,2}

\textsuperscript{1} Dept. of Computer and Information Science, University of Macau, Macao, China  
\textsuperscript{2} State Key Lab. of Computer Science, Institute of Software, Chinese Academy of Sciences, China  
\textsuperscript{3} Changchun Institute of Optics, Fine Mechanics and Physics, Chinese Academy of Sciences, China

\{ya57406, ehwu\}@umac.mo

Abstract

In computer graphics, animated models are widely used to represent time-varying data. In this paper, we propose an improved method to generate multiresolution animation models. We use a curvature sensitive quadric error metric (QEM) criterion as our basic measurement, which can preserve local features on the surface. We append a deformation weight to the aggregated edge contraction cost for the whole animation to preserve areas with large deformation. At last, we introduce a mesh optimization method to deal with the animation sequence, which can efficiently improve the temporal coherence and reduce visual artifacts. The results show our approach is efficient, easy to implement, and good quality progressive animation models can be generated at any level of detail.

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 to animation. In many cases, some details on such models 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. Most of the existing simplification algorithms are to deal with a single static mesh, while very little work has been proposed to describe deforming model simplification.

To get different level-of-detail animated models, 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 can’t be directly applied on dynamic meshes.

We therefore propose an improved method for generating multiresolution animation models. Our method is a better tradeoff between the temporal coherence and geometric distortion. We introduce a curvature sensitive edge collapse cost measurement, which improves the famous QEM. 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 of deforming surfaces over all frames of an animation.

2. Related work

Simplification and LOD. There are now extensive papers addressing the problem of mesh simplification, which can be roughly divided into five categories: vertex decimation, vertex clustering, region merging, subdivision meshes, and iterative edge contraction [3, 5]. A complete review of the methods has been given in [9].

Mesh optimization. Hoppe et al. [4] described an energy minimization approach to solving the mesh optimization problem. In [2], a procedure was presented to improve the quality of complex polygonal surface meshes without an underlying smooth surface using numerical optimization. Recently, Liu et al. [8] propose 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. [12, 13] are the first to address the problem of simplifying deforming surfaces. They designed a global multiresolution structure named Time-dependant Directed Acyclic Graph (TDAG). Mohr and Gleicher [MG03] proposed a deformation sensitive decimation (DSD) method, which directly adapt the QEM algorithm [3] 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. Kircher and Garland [6] 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. This method seems to be particularly efficient because of its connectivity transformation. Similar approach has been used by Payan et al. [11] and Zhang et al. [15, 16]. Recently, Landreneau et al. [7] propose a method for simplifying a polygonal character with an associated skeletal deformation. This method works well but can only be 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 smoothing algorithm for the whole animation sequence to improve the temporal coherence.

3.1. Edge contraction measurement

QEM [3] measurement is widely adopted for edge-collapse based simplification. It computes the edge contraction cost as the sum of squared distance at each internal mesh vertex to its adjacent planes. Assume that vertex \( v_i \) is the new vertex resulting from an edge contraction for edge \((v_i, v_j)\). QEM takes the following equation as the edge contraction cost:

\[
\Delta(v_j) = v_j^T \left[ Q_{v_i} + Q_{v_j} \right] v_j = v_j^T Q_{v_j} v_j
\]

where \( Q_{v_i} \) and \( Q_{v_j} \) are the quadric error matrix for each vertex.

QEM method 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 by the following equation:

\[
n_v = \frac{\sum_{p \in \text{planes}(v)} n_{p}}{n_{v}}
\]

Here, \( \text{planes}(v) \) is the adjacent triangles of the vertex \( v \) and \( n_{p} = n_{v}/n \) is the unit normal of a triangle. Then, we can calculate the vertex curvature as the following equation:

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

where \( \theta(n_v, n_p) \) denote 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 equation (1) as:

\[
\Delta'(v_j) = v_j^T Q_{v_j} v_j + \lambda \cdot F(i,j)
\]

where \( \lambda \) is a coefficient to adjust the influence of feature value. Also we can define \( \lambda F(i,j) \) 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 the equation (2) as:

\[
\Delta'(v_j) = v_j^T \left[ Q_{v_j} + K_f \right] v_j
\]

3.2. Deformation measurement

The deformation sensitive decimation (DSD) algorithm addresses edge contraction cost by summing QEM costs across all frames [10]:

\[
\text{DSDE}_f = \sum_{t=1}^{T} (v'_j)^T Q_{v_j} v'_j
\]

where \( v'_j \) minimizes the edge contraction cost for the edge \((v_i, v_j)\) at frame \( t \), and \( Q_{v'_j} = Q_{v_j} + Q'_{v'_j} \).

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. The deformation weight is defined to be:

\[ \sum_{t=1}^{f} \left| \Delta'_{ij} - \bar{\Delta}_{ij} \right| \]

where \( \Delta'_{ij} \) is the collapse cost of edge \((v_i, v_j)\) in frame \(t\), and \(\bar{\Delta}_{ij}\) 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:

\[ \text{cost}_{ij} = DSD_{ij} + k_d \times \sum_{t=1}^{f} \left| \Delta'_{ij} - \bar{\Delta}_{ij} \right| \]

\[ = \sum_{i=1}^{f} \Delta'_{ij} + k_d \times \sum_{t=1}^{f} \left| \Delta'_{ij} - \bar{\Delta}_{ij} \right| \]

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 \(\lambda_{ij}\), 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 \(\lambda_{ij}\) are positive and sum to unity

\[ \sum_{v_j \in N(v_i)} \lambda_{ij} = 1 \]

If all weights \(\lambda_{ij}\) 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 \(\lambda_{ij}\) be the mean value coordinates [1] of \(v_i\) with respect to its neighbours \(v_j\), then the shapes of the triangles are nicely preserved.

In an animation mesh sequence, we want the triangles have similar shapes between adjacent frames, thus the output animation can maintain optimal temporal coherence. 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. 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.

![Figure 1. Dancer animation models with 48 frames. Upper row: original sequence with 7061 vertices. Bottom row: simplified models with 700 vertices (remove 90%).](image)

![Figure 2. Comparison of the last frame in horse-to-man morphing sequence. Left: original model with 17489 vertices. Middle and right are the simplified versions (3200 vertices) generated using [6]'s method(middle) and our method(right).](image)

### 4. Experimental result

We test the result of our algorithm on a computer with Pentium4 3.2G CPU and 4G memory, using VC2005 and OpenGL as programming environment. The dancer animation sequence is shown in the Figure 1. In simplified sequence, most of the shape features on the original models are preserved well.

In Figure 2, we compare the results in [6] (middle) and our method (right). The models generated by our method obviously preserve the features on the hands and feet better.
Figure 3 shows the RMS error statistics of horse-gallop results. The RMS error line of our result (red) lies between DSD and independent QEM on the whole, and is obviously lower than other methods. Thus our method can be demonstrated to generate better results both in visual effect and error statistic.

Figure 3. 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.

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 a sequence of simplified models 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. A mesh optimization method is finally performed to improve the temporal coherence of the animation.

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 viewpoint-dependent.

Acknowledgement

The work has been supported by the PhD Studentship & Research Grant of University of Macau & China 863 project (2008AA01Z301) & NSFC (60833007).

References