Computational Intelligence and Data Fusion in Wireless Multimedia Sensor Networks - Research Article

International Journal of
Distributed
Sensor Networks

International Journal of Distributed Sensor Networks
2019, Vol. 15(1)
© The Author(s) 2019
DOI: 10.1177/1550147718822458
journals.sagepub.com/home/dsn
SAGE

Animated mesh simplification based on motion features in visual sensor networks

Hongjie Yang, Fan Zhou, Ge Lin, Mouguang Lin and Shujin Lin

Abstract
The three-dimensional animated model is widely used in scientific visualization, entertainment, and virtual applications, especially in visual sensor networks. The main purpose of simplification is to capture the shape sequence of an object with very few elements while preserving the overall shape. As three-dimensional animated mesh is time-varying in all frames, the trade-off between the temporal coherence and geometric distortion must be considered to develop simplification algorithm. In this article, a novel three-dimensional animated mesh simplification algorithm based on motion features is presented. Here, motion features are the connection areas of the relative movement consisted of vertices and edges. Motion feature extraction is to find a subgraph that has movement property. Dihedral angle of the edge through all frames is used to determine whether an edge is connected or not to the movement parts. Then, a rotation connected graph is defined to extract motion features. Traveling this graph, all motion features can be extracted. Based on the motion features, animated quadric error metric is created and quadric error matrix is built through all frames. Compared with the other methods, the important advantages of this method are high-efficiency simplification process and smoother simplification effects. It is suitable to be used in real-time applications. Experiment results show that the 3D animated mesh simplification effects by our method are satisfactory.

Keywords
Animated mesh, simplification, motion characteristic, adaptive quantum-enhanced metrology

Date received: 8 June 2018; accepted: 3 December 2018

Handling Editor: Sergio Toral

Introduction

With the development of computer technologies, the mobile platform based on network and communication has been applied rapidly in different areas. Wireless multimedia sensor networks (WMSNs), that is, networks composed of a large number of wirelessly connected multimedia sensor nodes, have a wide range of applications such as video surveillance, smart home, traffic monitoring, game, and entertainment.[1] Considering the efficiency of application on the mobile platform, how to obtain and manage the data used in the network is a challenge in research. Nowadays, high-quality animated three-dimensional (3D) surfaces, usually in the form of a dense 3D triangle mesh, can be

School of Data and Computer Science, Sun Yat-sen University, Guangzhou, China

Corresponding authors:
Fan Zhou, School of Data and Computer Science, Sun Yat-sen University, Higher Education Mega Center, Guangzhou 510006, China.
Email: isszf@mail.sysu.edu.cn

Shujin Lin, School of Data and Computer Science, Sun Yat-sen University, Higher Education Mega Center, Guangzhou 510006, China.
Email: linshjin@mail.sysu.edu.cn

acquired by performance capture systems from real-world objects and characters.[2] And more and more animated meshes are frequently used from scientific applications (simulation or physical processes) to animation (movie, games, morphing); these time-varying surfaces are frequently used to represent and visualize models or objects in motion.[3] In many cases, some details of them might be unnecessary.[2] For example, interactive systems often gain performance by using lower resolution models when possible, and offline systems can benefit from the memory savings which lower resolution models provide when the details are unnecessary.[4]

The 3D mesh simplification is an attractive proposition in view of its wide range of applications among levels of detail,[5] shape processing,[6] and analysis.[7] The aims of 3D animated meshes simplification are to capture the shape sequence of an object with very few elements, while preserving the overall shape, volume, and boundaries as much as possible. The general problem is that any simplification of a deforming mesh must preserve the necessary geometry to adequately represent the deformed mesh in all poses, fast visualization processing, simple data structure, and storage efficiency. Some research results around the mesh simplification are reported.[8–13] There is a large range of mesh simplification algorithms, each targeted at different specific applications. However, since the 3D animated mesh data are temporal coherence, traditional simplification methods cannot be directly applied to animated models.[2] Otherwise, it can involve the unpleasant visual artifact, causing the surface to vibrate and twitch. The trade-off between the temporal coherence and geometric distortion must be considered to develop simplification algorithm.

Compared with a single static mesh, for time-varying meshes, the problem of simplification is still a significant problem.[2,14] The 3D animated meshes are generally represented as over-sampled triangular meshes with a static connectivity, involving a large number of unnecessary details for some frames. Deforming surfaces are most of the times represented by dense triangular meshes sharing the same connectivity. For these models, we not only want to simplify the surface to create a faithful approximation to its rest pose but we also would like a simplified surface that approximates the shape well in all possible poses associated with its deformation model.[15]

In this article, we present a novel 3D animated mesh simplification algorithm based on motion features. Initially, we improve quadric error metric (QEM) as an animated quadric error metric (AQEM) to simplify the 3D animated mesh. Considering the relationship between edge collapse cost and order, the edge collapse costs in every frame are integrated to generate the measure of error method. This method can decrease the unexpected jump during animation. Second, we define a new concept about motion features. The main purpose is to protect the valuable part on the animated model. Analyzing the 3D animated model, motions can be decomposed into translation and rotation. In translation operation, the parts on the model are no relative motion; in the rotation operation, some parts on the object are moving relative to each other. The connection areas of the relative movement similar to the joints in the model are defined as motion features. Figure 2(a) shows the motion joints of the dancing men. Figure 2(b) shows the motion features in every single frame, and the complete motion features should be obtained according to all the frames shown in Figure 2(c). Dihedral angle of the edge through all frames is used to determine whether an edge is connected or not to the movement parts. Then, we define a rotation connected graph to extract the motion features. Here, motion feature is a subgraph. Based on motion features, a new weighted QEM for animated meshes is built. Experimental results show that the 3D animated mesh simplification effects by our method are satisfactory, and similarly, the advantages of highly efficient simplification are better for the real-time 3D animated mesh application.

Overall, our full frame is visualized in Figure 1, which consists of two major phases, including motion feature extracting through all frames and AQEM simplification.

The contributions in this article are as follows:

1. A new concept about motion features to protect the valuable part on the animated model during simplification process;
2. A method to calculate the motion feature and new weighted QEM;
3. A fast and efficient simplification algorithm AQEM for 3D animated mesh.

The rest of this article is organized as follows. We begin with a review of related work in mesh simplification. Next, we present our new simplification algorithm and motion feature calculation method. Finally, we show the experimental results by our method and give a results analysis and discussion. The last section is conclusion.

Related works

Animated mesh simplification

Mesh simplification is an important and well-studied problem in computer graphics. The mesh simplification methods can be roughly divided into five categories: vertex decimation, vertex clustering, region merging, subdivision meshes, and iterative edge contraction.[16] Many researchers have tackled the problem of mesh

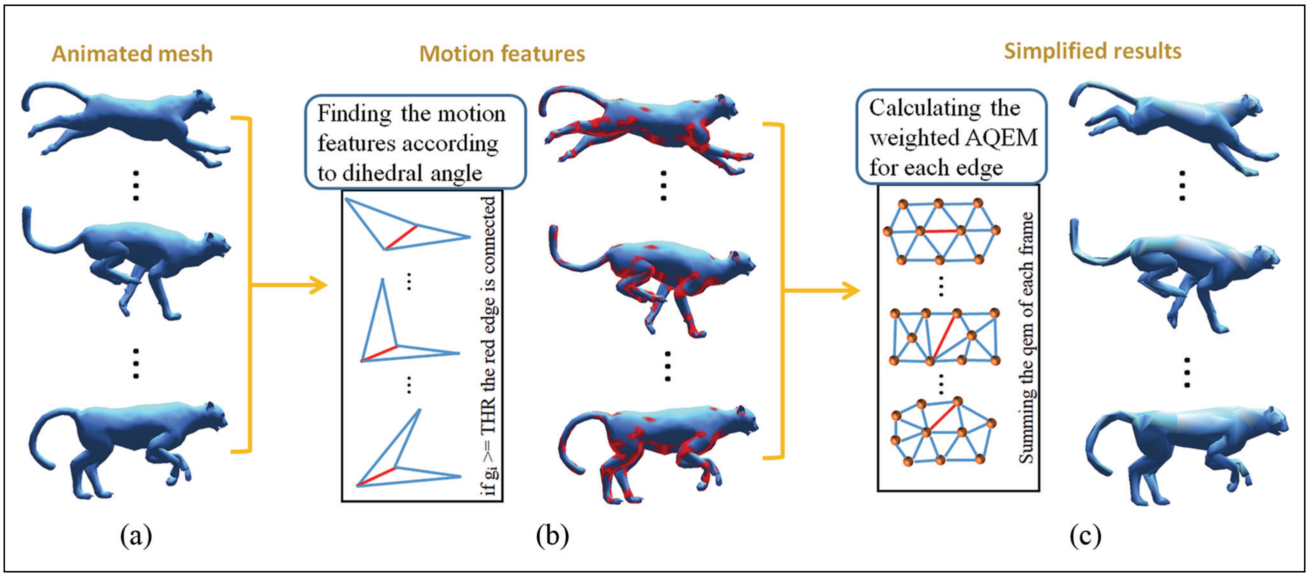
Animated mesh
Motion features
Simplified results
Finding the motion features according to dihedral angle
if g_i >= THR the red edge is connected
Calculating the weighted AQEM for each edge
Summing the qem of each frame
(a)
(b)
(c)

Figure 1. The calculation of the collapse cost using AQEM. (a) The input animation: the animated sequence of a cat; (b) the left column indicates the shorthand version meshes in the pale blue circles in the animated sequence on the right column, and the red line in expresses the same edge in different frames; (c) simplification by AQEM. The final collapse cost of the edge is summed by its collapse cost in each frame. (a) All frames, (b) motion feature extracted, and (c) AQEM simplification.

simplification in recent years, and a variety of approaches have been used to decimate polygonal models.

With the wide applications of 3D animated mesh, the research of 3D animated mesh simplification becomes more and more attractive. Shamir and colleagues[17,18] 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. However, this scheme is very complex and cannot be easily handled.[16] Based on a QEM,[19] some approaches address animated mesh simplification. The researches can create an animated triangle mesh or a set of triangle meshes with minimal frame-to-frame connectivity transformations.[20,3] There are three major advantages of these solutions. First, it is able to simplify dense mesh quite rapidly, which is vital for processing a lot of animated mesh; second, the representations produced by QEM remain faithful to original mesh; and third, an edge of the input consistent mesh for all frame is collapsed, which holds the output connectivity along all frame. Until now, some improvements have been reported.[21–26]

Technical background

The 3D animated mesh is time-varying in all frames; the trade-off between the temporal coherence and geometric distortion must be considered to develop simplification algorithm. At the same time, the efficiency of simplification is also important for some real-time application. We will briefly present the QEM algorithm,[19] which we extend in our method. It is based on the assumption that in a good approximation, points do not move far from their original positions. QEM simplifies a mesh by iteratively selecting an edge (v_i, v_j) with the minimum contraction cost to collapse and replace this edge with a new vertex u which minimizes the contraction cost. The implementation steps are summarized as follows: the collapse cost of each edge was calculated, the edge with the minimum cost was contracted, and the relevant information was updated. All the edges are collapsed iteratively until the desired resolution is reached. It is widely used, relatively straightforward to implement, and it makes a good trade-off between simplification speed and quality.

In this article, initially we define the motion features for all frames, calculate the collapse weight based on the motion features, and then a new weighted QEM for animated meshes is built; in contrast to previous approaches, our method not only approximates the shape well in all frames but also be a highly efficient simplification method for real-time application.

AQEM

In the section of related work, QEM algorithm is reviewed briefly. The example of edge collapse algorithm is shown in Figure 3. It is found that according to the edge collapse iteratively, the 3D mesh can be simplified continually until the desired resolution is reached. The QEM algorithm is widely used due to its

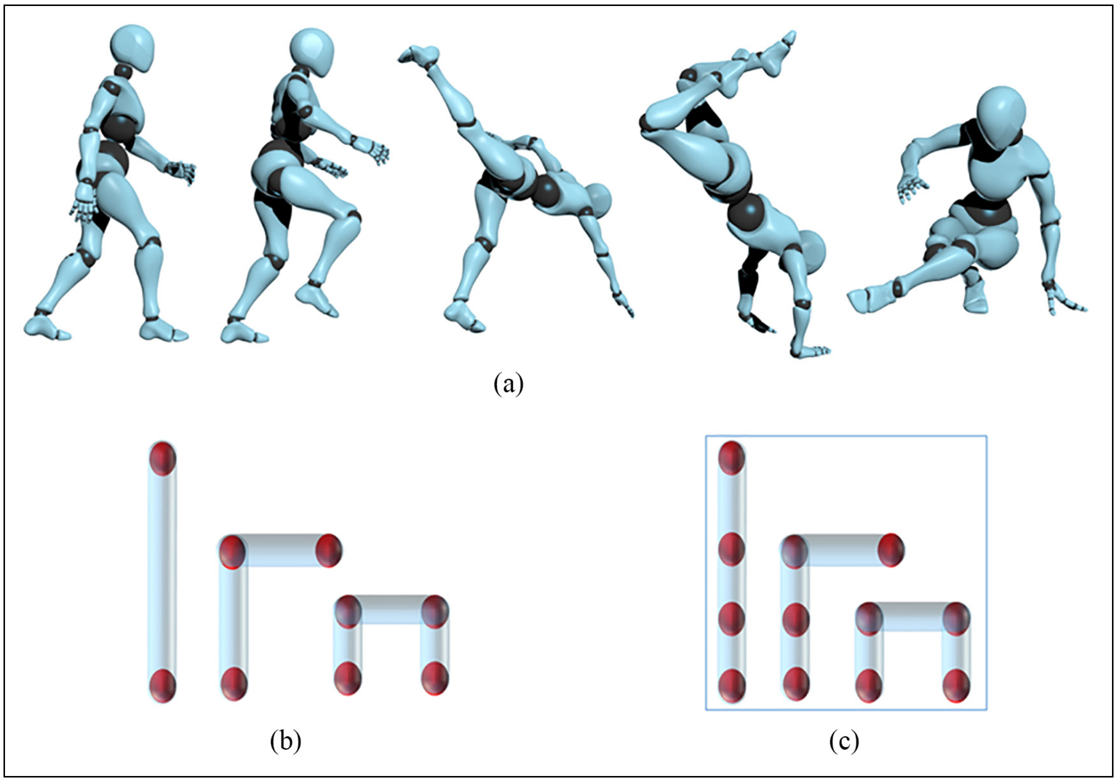

Figure 2. 3D animated model motion feature analysis: (a) dancing men and the motion joints, (b) motion features in a single frame, and (c) complete motion features.

superior comprehensive performance. Existing method to apply QEM to the animated mesh simplification is to use QEM for each frame independently. Though this solution can obtain the minimum error, it does not take into account the temporal coherence, which leads to the unexpected jump through whole animation; as shown in Figure 2(b), the two vertices were preserved in the first frame, while the other three or more vertices were preserved in the next frame, causing the unexpected jump. So, we will report a new method to create a QEM model, namely, AQEM. In the algorithm AQEM, quadric error matrix is built through all frames, and the motion features are added as weights. Because of considering the relevance between all frames and the motion feature on the animated model, the simplification effect is more smooth.

AQEM algorithm

The main steps included in AQEM algorithm can be described in detail. First, we need to find out all candidate collapse edges on the mesh; second, we need to calculate the collapse cost and the weight generated by motion features of each candidate edge in all frames; third, for each edge on the mesh, collapse cost of each frame is summed, and the summation multiplied by weight is the final collapse cost of this edge, which is different from the calculation in a static model; fourth, all candidate collapse edges were put on the heap structure according to their final collapse cost, and the edge with minimal cost is placed at the top of the heap; and finally, the following steps were repeated until the simplification target is reached:

1. The edge with minimal collapse cost is removed from the heap, assuming (v_i, v_j).
2. The edge was collapsed and the corresponding vertices, faces, and edges were deleted.
3. The collapse cost of the edges was updated, which contains vertex v_i or v_j, and the position of these edges was adjusted in the heap to ensure that the edge with minimal collapse cost remains at the top of the heap.

Most of the algorithms based on edge contraction employ the aforementioned approach. The difference between them is the way of figuring out the collapse cost of the candidate edge. The collapse cost represents the loss degree of model accuracy after an edge is removed. The less cost illustrates more similarity between original mesh and simplified model. Obviously, the calculation about collapse cost is related to the collapse order of the edges and further determines the remaining surface of the original model in the process of simplification.

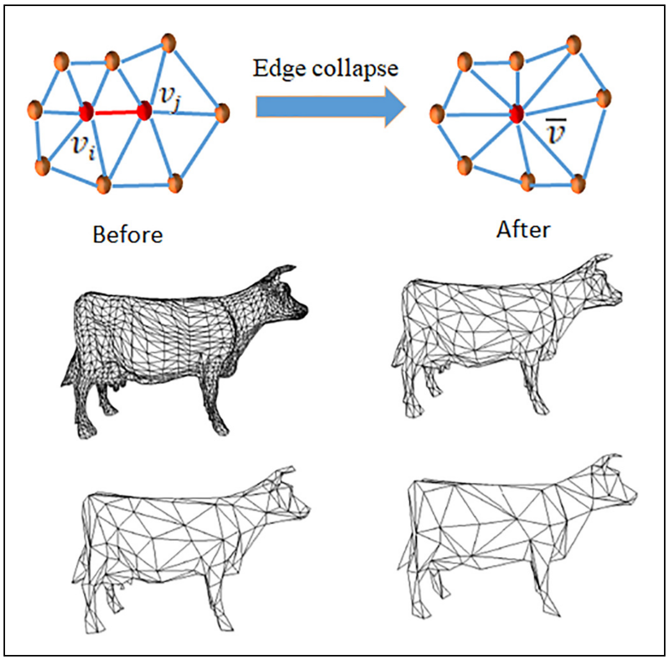

Figure 3. Example of edge collapse.

Notations. In the following, S^n is the set of real symmetric matrices of size $n \times n$, R^n is the set of the real matrices of size $n \times 1$, and R is the set of the real number.

Calculation of the AQEM

The sum of the square of the distance from the new vertex produced by the edge collapse operation to the plane associated by two vertices of collapsed edge will be defined as the measure of error, which is formulated as follows

$$Q(\overline{v}) = \sum_{p \in T(v)} (p^T \overline{v})^2 \tag{1}$$

where $p = [a, b, c, d]^T$ denotes the plane defined by the equation $ax + by + cz + d = 0$, where $a^2 + b^2 + c^2 = 1$ in the 3D space; $T(v)$ is the set of triangles associated with vertex v; and $\overline{v} = [x, y, z, 1]^T$ is the new vertex after collapse in the 3D space. Equation (1) can be described as follows

$$\begin{aligned} Q(\overline{v}) &= \sum_{p \in T(v)} \left(n_p^T + d_p\right)^2 \\ &= \sum_{p \in T(v)} \left[\overline{v}^T \left(n_p n_p^T\right)\overline{v} + 2d_p n_p^T \overline{v} + d_p^2\right] \end{aligned} \tag{2}$$

where $n_p = [a_p, b_p, c_p]^T$ is the normal vector of plane p, $\overline{v} = [x, y, z]^T$ is the new vertex (notice we no longer adopt homogeneous coordinates), and d_p is the scalar d of plane p mentioned in equation (1). The plane can be defined by another equation $a(x - x_o) + b(y - y_o) + c(z - z_o) = 0$, we know that $d_p = -n_p^T v_p^o$, where $v_p^o = [x_o, y_o, z_o]^T$ denotes any point on the plane p. So, equation (2) can be translated into the following

$$Q(\overline{v}) = \sum_{p \in T(v)} \left[\overline{v}^T \left(n_p n_p^T\right)\overline{v} - 2n_p^T v_p^o n_p^T \overline{v} + \left(n_p^T v_p^o\right)^2\right]$$

Furthermore

$$\begin{aligned} Q(\overline{v}) &= \sum_{p \in T(v)} \left[\overline{v}^T A_p \overline{v} - 2b_p^T \overline{v} + c_p\right] \\ &= \overline{v}^T \left(\sum_{p \in T(v)} A_p\right)\overline{v} - 2\left(\sum_{p \in T(v)} b_p^T\right)\overline{v} + \sum_{p \in T(v)} c_p \end{aligned} \tag{3}$$

with

$$\begin{aligned} A_p &= n_p n_p^T \in S^3 \\ b_p &= n_p^T v_p^o n_p \in R^3 \\ c_p &= (n_p^T v_p^o)^2 \in R \end{aligned}$$

The discussions above are based on static mesh. However, for animated mesh sequence, we should build a unified QEM model through all frames. So, we extend the vertex dimension from three dimensions to $3F$ dimensions, where F is the number of frames contained in the animation. And quadric error matrix is built through all frames

Let $v = [x^1, y^1, z^1, x^2, y^2, z^2, \ldots, x^f, y^f, z^f, \ldots, x^F, y^F, z^F]^T \in R^{3F}$ to be a vertex (f indicating the frame f of an animated mesh), similarly

$$\begin{aligned} A_{Fp} &= \begin{bmatrix} n_p^1 n_p^{1^T} & & & & \\ & \ddots & & & \\ & & n_p^f n_p^{f^T} & & \\ & & & \ddots & \\ & & & & n_p^F n_p^{F^T} \end{bmatrix} \in S^{3F} \\ b_{Fp} &= \left[n_p^{1^T} (v_p^1)^0 n_p^1, \ldots, n_p^{F^T} (v_p^F)^0 n_p^F \right] \in R^{3F} \\ c_{Fp} &= \sum_f \left(n_p^{f^T} (v_p^0)^f \right)^2 \in R \end{aligned}$$

Therefore, we can obtain a new measure of error in equation (4):

$$\begin{aligned} Q_F(\overline{v}) &= \sum_{p \in T(v)} \left[\overline{v}^T A_{Fp} \overline{v} - 2 b_{Fp}^T \overline{v} + c_{Fp} \right] \\ &= \overline{v}^T \left(\sum_{p \in T(v)} A_{Fp} \right) \overline{v} - 2 \left(\sum_{p \in T(v)} b_{Fp}^T \right) \overline{v} + \sum_{p \in T(v)} c_{Fp} \end{aligned} \quad (4)$$

During the simplification process, all the edges together with the resulting approximation error are put into a heap H, specifically, an edge linked by two vertices v_i and v_j with error quadrics Q_i and Q_j, respectively, and the edge (i,j) is placed in heap H with collapse cost for quadric $Q_i + Q_j$, and the edge with minimum cost is placed on the top of the heap H.

The top edge (i,j) from H is collapsed, and then, we get a new vertex v_k with $Q_k : = Q_i + Q_j$. Those edges involving v_i or v_j are removed from H and new edges generated due to new vertex are added to H. The algorithm is performed iteratively until the simplification requirement is achieved.

Quadric minimization

The best position of the new vertex obtained by equation (4) can be shown in another form

$$\begin{aligned} Q_F(\overline{v}) &= (\overline{v})^T X \overline{v} - 2 Y^T \overline{v} + Z \\ &= \begin{bmatrix} (\overline{v})^T & 1 \end{bmatrix} \begin{bmatrix} X & -Y \\ -Y^T & Z \end{bmatrix} \begin{bmatrix} \overline{v} \\ 1 \end{bmatrix} \\ &= P^T Q P \end{aligned} \quad (5)$$

where

$$\begin{aligned} X &= \sum_{p \in T(v)} A_{Fp} \\ Y &= \sum_{p \in T(v)} b_{Fp} \\ Z &= \sum_{p \in T(v)} c_{Fp} \end{aligned}$$

The Q is also a real symmetric matrix since the X is the real symmetric matrix. Furthermore, the Q is a positive definite matrix since $Q(\overline{v})$ is greater than 0. According to the least principle of the linear algebra, we know that $Q(\overline{v})$ gets the minimum when $\overline{v} = X^{-1} Y$.

The final cost is $wQ(\overline{v})$. As mentioned above, the collapse order of the edges determines the surface remaining of the original model in the process of simplification. The role of the weight w generated by motion features is to change the collapse order. So, it is very important to find the motion features of the animated model.

Motion feature

Up to now, we have built a new AQEM algorithm for the animated mesh simplification. We should find a way to find out our motion features. The effect may be unsatisfactory as shown in "Results" section if we simplify the animated model owing to only the collapse cost of edge considered, which makes against the protection of the valuable part on the animated model. So, we define a new concept about motion features for the 3D animated model to avoid half-baked parts.

As we know, the purpose of simplifying a mesh is to drop low-attention places while protecting the high-attention places. In the animated model, movement parts of the object are more sensitive to the static part. Nevertheless, not all animated models in the dynamic processing hold moving parts, such as when we translate a model, there is no relative motion parts on the model. The moving parts emerge only when some parts on the object are moving relative to each other. Then, the moving parts on the model are the motion features we defined.

Motion features are the connection areas of the relative movement similar to the joints that connect bones, as shown in Figure 2(b) or Figure 2(c) in red color. We need to find all motion features of the model. Then, the vertex in these special areas will be processed seperately. The greater weight will be assigned to the edges within those regions. In another situation, the vertex that is not in the special areas will have the same weight. In the following, an algorithm to find out those motion features is described.

Rotation connected graph

In order to find the motion features, the regions are needed to be checked whether they are bent or rotated. Dihedral angle is used to determine whether a region is rotated or bent. We notice that the meshes within motion features will rotate at a certain angle because of the relative movement of the parts it connects. So,

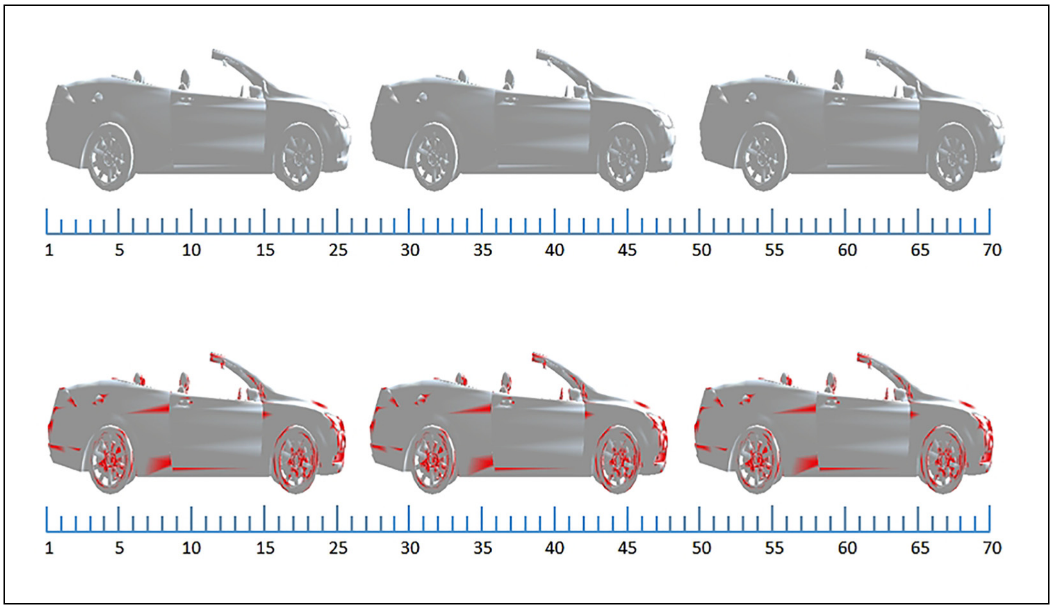

Figure 4. Some errors are generated due to the calculation accuracy. A car traveling straight on the road is equivalent to a translational movement and should have no movement feature as shown in the top, but it is not the case on the bottom row (the red blocks represent motion features), which can be caused by the calculation accuracy. So, we set a limit that the *FCVDA* must be greater than 1°, otherwise unconnected even if it is greater than the *THR*.

dihedral angle of adjacent triangles can be the index of the motion features. We define that if the change of dihedral angle corresponding to the edge is greater than the threshold, then this edge is connected, otherwise disconnected. Specifically, we monitor the change value of dihedral angle among all frames and simultaneously redefine the connectivity of edges. The corresponding edge will be regarded as connectivity if the change value of dihedral angle is greater than a certain threshold.

The 3D animated model is considered as a graph at first, and then, all the connected subgraphs are found on the model by traversing algorithm. Each connected subgraph is called rotation connected graph. The rotation connected graph is motion feature. For a given edge, we compute the change in its dihedral angle through all frames and record the maximum change to be the factor of edge that is connected if the factor is greater than the threshold, seeing the entire model as a graph and traversing it to find all connected subgraphs.

It is easy to see that the choice of threshold will be the key to finding a rotation connected graph. An animated model with slight motion should be allocated a smaller threshold such as a beating heart; on the contrary, the model with strenuous activity will be of a greater threshold, that is, the threshold variety according to the input animated model. Based on the experimental results, we choose the average value of the factor of all edges on the model as the threshold to find the motion features.

Letting g_i to be the factor of edge i, $CVDA_i^{(f-1)-f}$ is the change value of the dihedral angle of the edge i between frame $f-1$ and frame f. Then

$$g_i = max\left\{CVDA_i^{1-2}, CVDA_i^{2-3}, \ldots, CVDA_i^{(f-1)-f}\right\}$$

We use the mean of the factor of all edges as the threshold: $THR = \left(\sum_{i=1}^{n} g_i\right)/n$, where *THR* is the threshold of dihedral angle and n is the number of edges on the model. When g_i is greater than the *THR*, the two vertices corresponding to the edge are set to be connected, otherwise be unconnected. As a special case, some edges may only be associated with a triangle, such as the boundary, having no dihedral angle. In the static mesh simplification, based on the boundary preserving principle, the larger weight is given to the boundary; similarly, in our method, the vertices on the boundary were also set to be connected. So, a boundary corresponds to a motion feature.

Some errors may be generated when determining whether an edge is connected using our method due to the calculation precision, such as motion features are found in a translational car as shown in Figure 4. In order to avoid this situation, we set a limit that the g_i must be greater than 1°, otherwise unconnected even if it is greater than the *THR*.

Weight allocation

After finding the motion feature, all vertices on the model can be divided into two classes: vertex within motion feature or outside motion feature. For each motion feature, we record the total number w of vertices contained in it and assign the weight w to each vertex within it. We set the weight of the vertex outside the motion feature to 1.

Given a collapse $(u, v) \xrightarrow{cl} \bar{v}$, its weight can be calculated as $W_{\bar{v}} = (W_u + W_v)/2$. The $W_{\bar{v}}$, W_u, and W_v denote the weight of the vertex $\bar{v}, u, v$, respectively. Combining the weight to equation (5), we get the final collapse cost

$$Q_F(\bar{v}) = P^T W_{\bar{v}} Q P \qquad (6)$$

Moreover, for the vertex u and v, if one of them is within the motion feature, the other is outside, $W_{\bar{v}} = (1 + W_v)/2$ or $W_{\bar{v}} = (W_u + 1)/2$, which is greater than both u and v outside the motion feature. Such that, the edges within the motion feature are not only preserved, but also the edges connected with the motion features can be retained.

The change in the collapse cost of an edge is not obvious if the weight assigned to the edge is too small. In our work, the weight of a motion feature should depend on the size of the motion feature (i.e. the number of edges within the motion feature); the more the number of edges contained in the motion feature, the greater the weight assigned to the motion feature.

Forbidding triangle inversion

Similar to the research work reported by Garland,[19] when considering a possible collapse, we need to compare the normal vector of the neighboring triangles before and after the collapse. We prevent collapsing two vertices which lead to flip the orientation of the triangle. For animated mesh, the method of detecting mesh inversion is somewhat different. Equation (7) is used to detect an inversion

$$\sum_f n(t^f)^T \cdot n(t'^f) \cdot \sqrt{A(t^f) \cdot A(t'^f) \cdot w_t^f \cdot w_t' f} < 0 \qquad (7)$$

where t' denotes the triangle of t after the collapse, $A(t)$ is the area of the triangle t, and w_t is the weight of t calculated by $w_t = w_{v_i} + w_{v_j} + w_{v_k}$ (v_i, v_j, and v_k are three vertices of the triangle t and w_{v_i}, w_{v_j}, and w_{v_k} are the weight of the vertices v_i, v_j, and v_k, respectively). In other words, our scheme forbids large triangles, and the triangles with greater weight (i.e. the triangles are in the motion feature areas) will be inverted for a significant amount of frames.

Simplification algorithm outline

Up to now, the animated mesh simplification based on motion features can be described in detail. The data flowchart can be shown in Figure 1. The 3D animated mesh are inputed. Motion features are extracted through all frames. Then, the weight of each edge is determined. AQEM is created according to the quadric error matrix built through all frames. Apply AQEM to simplify 3D animated mesh until the desired resolution is reached.

The main steps of simplification algorithm are described as follows:

Step 1. Collect all edges (v_i, v_j) of each mesh in the animated sequence;
Step 2. Calculating Q matrices for all vertices;
Step 3. Search all motion features;
Step 4. Allocate weight w_i for each vertex v_i;
Step 5. For each edge,

1. Compute the quadric Q: $Q = (Q_i + Q_j)/2$;
2. Compute the weight of edge: $w = (w_i + w_j)/2$;
3. Compute the position of target vertex $\bar{v}$;

Step 6. Compute the collapse cost of each edge by equation (6);
Step 7. Build a min-heap according to the collapse cost of each edge;
Step 8. Remove the edge with the minimum cost from the heap;
Step 9. Fold edge (v_i, v_j) to $\bar{v}$ and arrange the weight $(w_i + w_j)/2$ to $\bar{v}$;
Step 10. Update the collapse cost of the affected edge;
Step 11. Rebuild the min-heap;
Step 12. Repeat Steps 8–11 until reaching the desired approximation.

Results

We implemented our simplification method in C + + and used QT to build our interface for the animated model approximation. All motion features on the model are marked in red. Some experimental results of our method will be presented and compared with other methods. First of all, the detection results of the motion features will be displayed. By observing the movement sequence, it can be learned that the motion features detected by our method are reasonable and almost all the regions connecting two moving parts on the model are found. Second, the simplification results of the animated sequences are shown at different levels. Third, some comparisons are made between our method and other method.

The detection results of motion feature

A lot of animated models are used in the experiment. Some detection results of motion features are shown in Figure 5. There are five animated sequences in Figure 5 containing a breathing spider, a walking spider, a running leopard, a running horse, and a training knight,

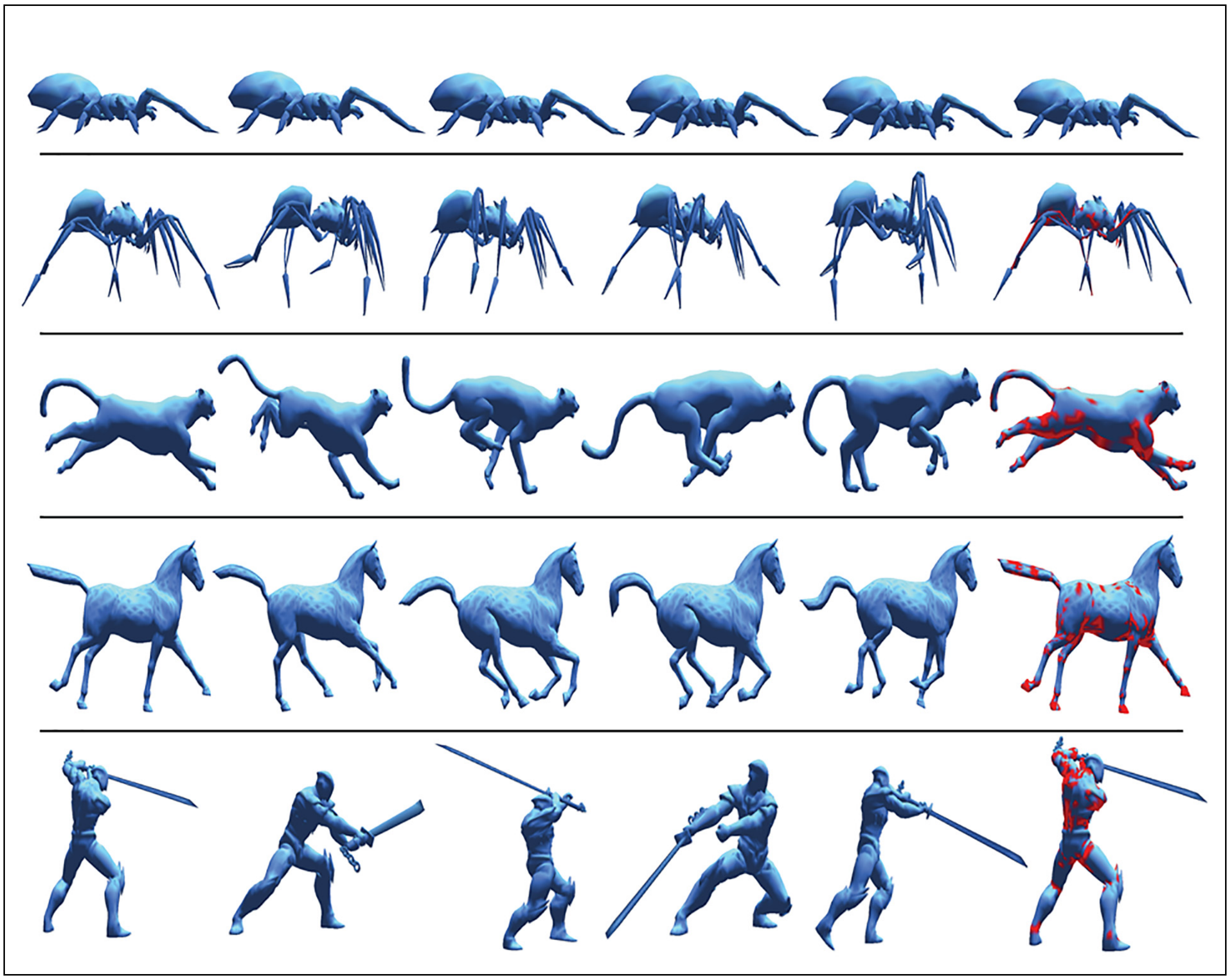

Figure 5. The detection results of motion feature. It contains five animated models and each of them is represented by five frames (the first five columns). The last column is the results of the motion features detected using our method where the regions with red color represent the motion features.

with 30 frames, 33 frames, 15 frames, 48 frames, and 20 frames, respectively. We extract five frames for each animation to display. The last column is the results of the motion features detected using our method. The movement features are well found. As we can see in the first row of Figure 5, the breathing spider has no motion feature. That is because breathing is a very mild exercise, like the heart beating, which is not the motion feature we want to find. The leopard and the horse's tail swings, their movements are fast and large in amplitude, the joints of the limbs when they are running are well found, and the motion features are detected and marked in red color. On the last row, a swordsman is practicing sword. We can observe that his upper body moves frequently, and most parts on the body are moved during practice. So, the motion features on the upper body are all distributed. The experimental results show that the motion features can be detected by our method efficiently.

The animated model approximation results at different levels

Figures 6 and 7 show the results of animated model simplication at different levels. A running leopard animated model contains 15 frames and 4876 faces. The animated model is simplified with 3041 faces, 2012 faces, 1200 faces, and 800 faces in different levels. Compared with the original model shown in the first row, we can see that the simplification results are well faithful to the original model, even if 84% of its components are removed on the last row. A more complicated example is shown in Figure 7, a running horse with 48 frames. The original model has 16,843 faces as shown in the first row. The 39%, 58%, 68%, and 86% of its components have been simplified by the simplification processing. The simplification results show that the main features of the animated model are retained. So, our method can be used to simplify 3D animated model and the simplification results are better.

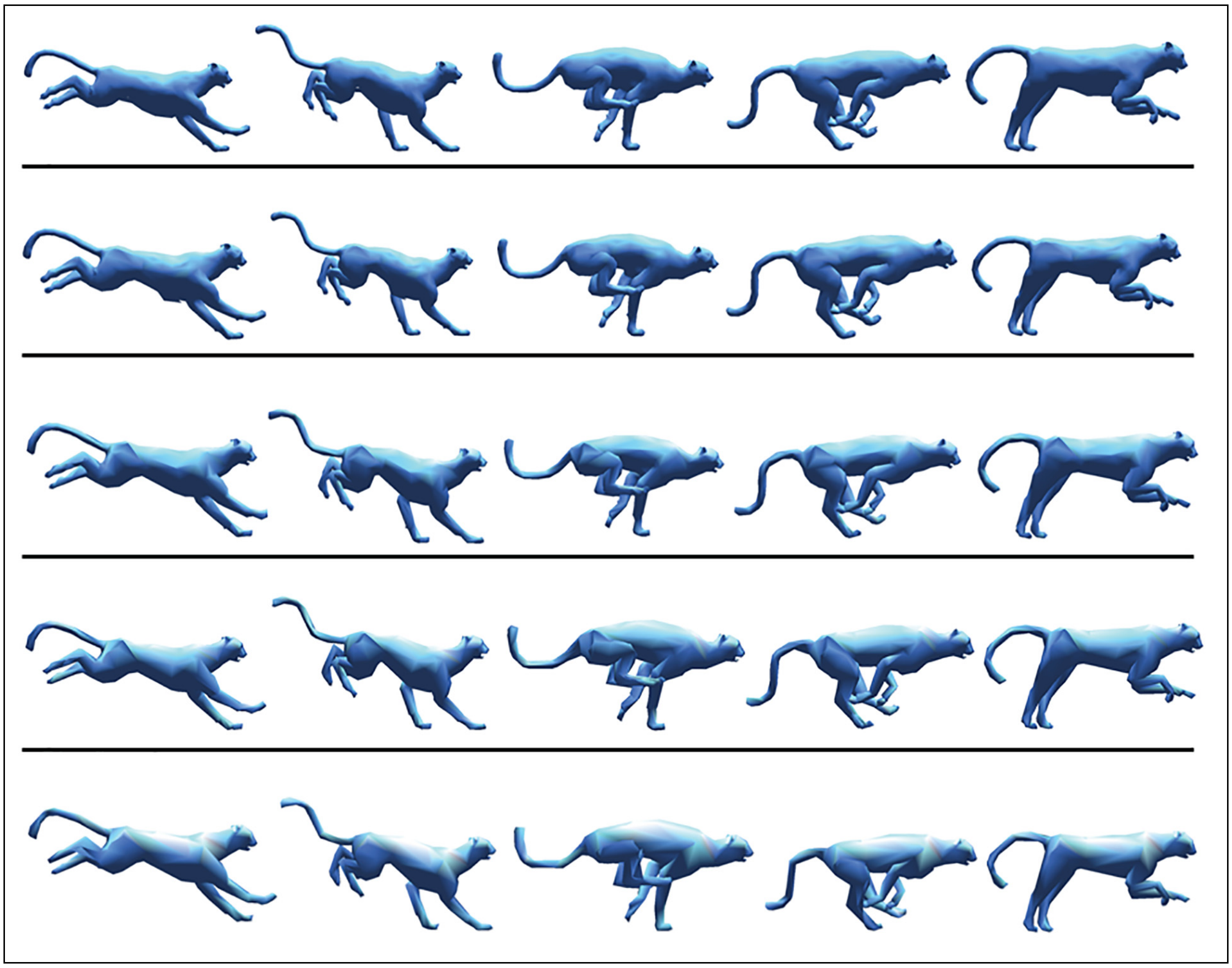

Figure 6. A running leopard animation with 15 frames. Top row: original mesh with 4876 faces; second row: 3041 faces approximation; third row: 2012 faces approximation; fourth row: 1200 faces approximation; and bottom row: 800 faces approximation.

Comparison of the simplified method

In this article, a new concept of motion feature is defined. The effect of motion feature is to avoid half-baking of some of the parts. Figure 8 shows the motion feature effect analysis. In Figure 8(a), leopard is simplified by QEM (without the motion features). So, we can find the simplification result of a running leopard with 500 faces on the second frame, some half-baked on the tail part, and disappeared ears of leopard. Figure 8(b) shows the motion features detected by our method. The motion features are marked in red on leopard's ears and in the middle of the tail part due to their wobble. So, these two parts should be preserved in the simplification process. Then, simplification results of AQEM are shown in Figure 8(c). It is clear that the simplification result is better, and the tail part and ears are protected. Figure 9 shows another example of simplifying a lion that is jumping. There are a lot of motion features (marked in red color) on the body because of its intense exercise, and the region around its eye is one of them in that edges around the eye are the boundary of the model (considering the boundary protection, we formulate that the edges on the boundary are also a motion feature). So, our method can remain as the lion's eye information; otherwise, the lion's eye information will be be half-baked during simplification processing. The experimental results show that the simplification effects are of high quality and satisfactory.

Comparing the computational efficiency, the method reported in this article is highly efficient. Figure 10 shows the main calculation processes about our method and Thiery et al.'s[2] method. In Thiery's algorithm, the radius bound is needed to calculate every iteration, which will require a lot of computational costs. In our method, we do not need this calculation part. The other parts are same. Therefore, the simplification efficiency of our methods is higher. The advantages of highly efficient simplification are better for the real-time 3D animated mesh application.

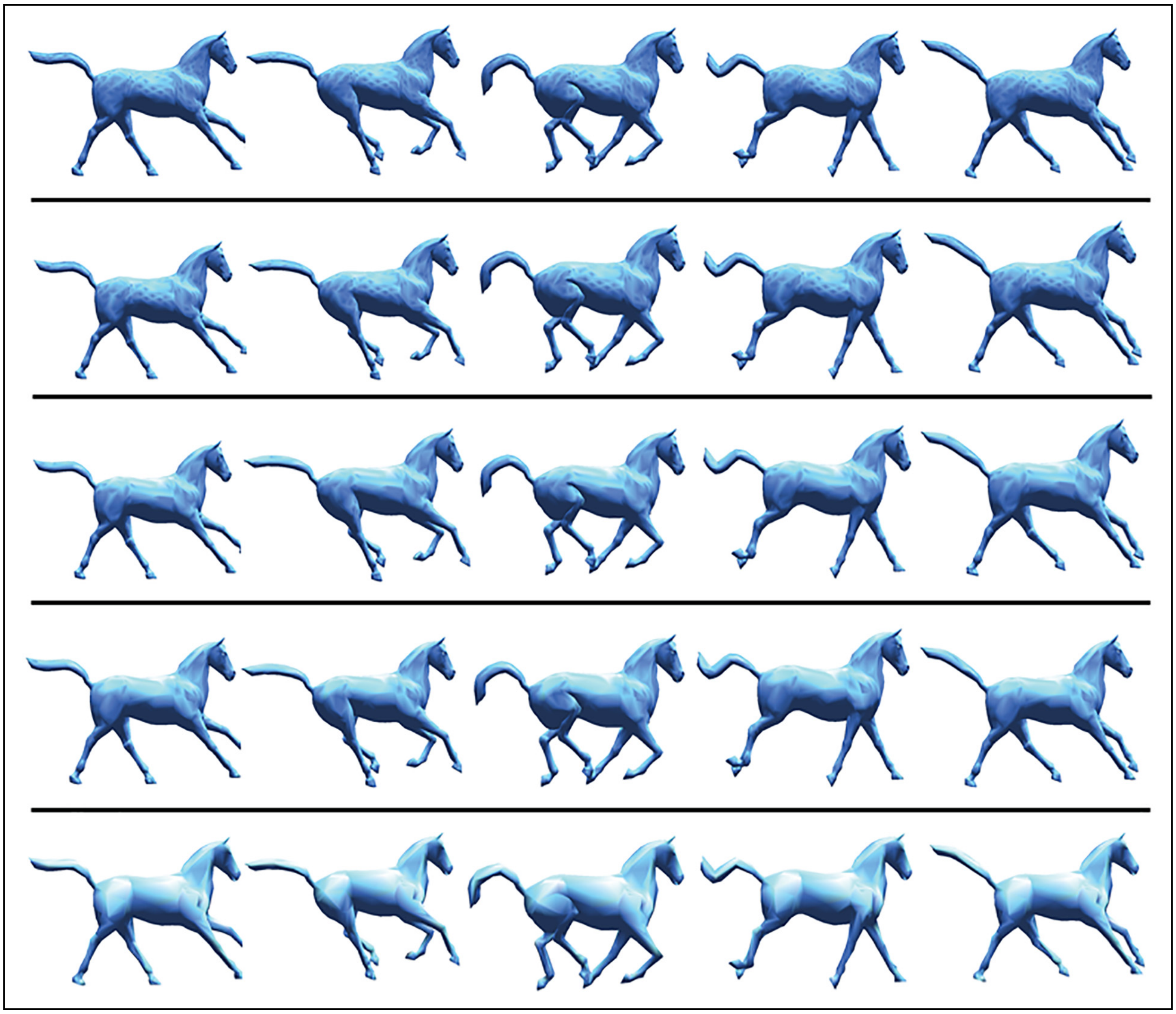

Figure 7. A running horse animation with 48 frames. Top row: original mesh with 16,843 faces; second row: 10,327 faces approximation; third row: 7059 faces approximation; fourth row: 5397 faces approximation; and bottom row: 2278 faces approximation.

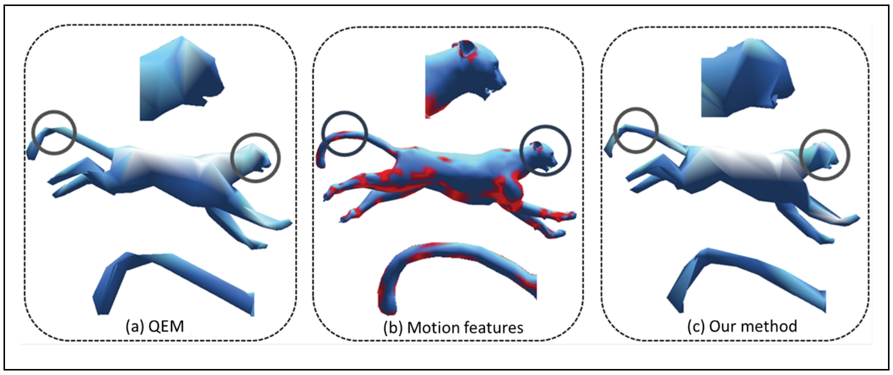

Figure 8. Comparisons of (a) QEM (number of the motion features) and (c) our method (right) applied on the second frame in a running leopard, and the simplified versions contain 500 faces. (b) The motion features found using our method and the areas surrounded by a black circle are the regions we want to compare.

Conclusion

Compared with a single static mesh, for time-varying meshes, the problem of simplification is still a significant problem. High efficiency and satisfying quality of simplification must be considered, especialy in the real-time applications. To achieve this purpose, in this article, we propose a novel 3D animated mesh simplification algorithm based on motion features. We define a new concept about motion feature to explain the motion characters during motion. Thse features can be used not only to preserve the valuable parts but also to avoid some parts be half-baked. In order to find the motion features, dihedral angle of the edge through all frames is used to determine whether an edge is connected or not. Considering the effects of temporal coherence during the motion, AQEM model is designed based on QEM, quadric error matrix is built through all frames, and the simplification results will decrease unexpected jump in the animation. The weight is used to improve the QEM of each vertex, which can produce satisfactory simplification results for animated model. The complete algorithm is reported in detail in this article. This method is better to be used in real-time application because of the high simplification efficiency. Next, we will improve our method further. The fine motion features may be extracted, which can be used to visualize the tiny deformation during simplification. The weight determination is needed to be more accurate.

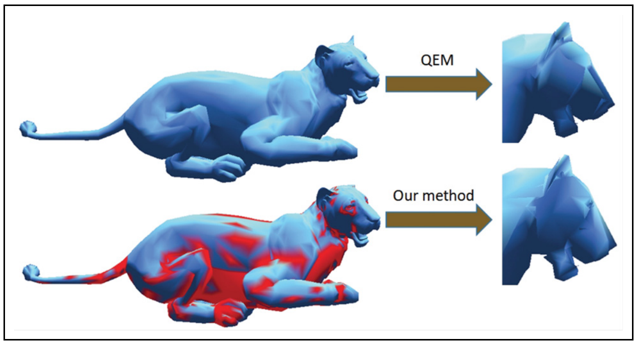

Figure 9. A lion animation with intense exercise, 90% of its components are reduced using QEM (no the motion features) on the top row, and our method is shown in the bottom row. We choose its head to display.

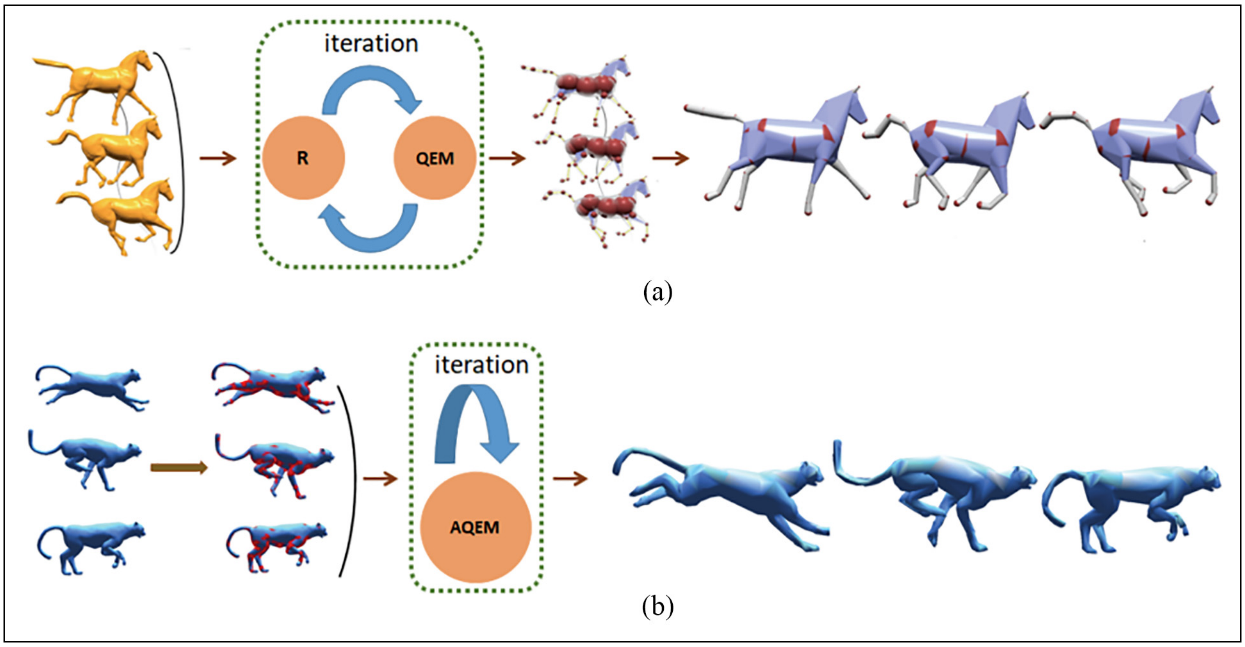

Figure 10. Calculation flowchart of (a) Thiery's and (b) our algorithm, and the most time-consuming steps have been circled with the green dotted line and the time-consuming other steps can be ignored comparing with it. In the process of simplifying iteration, Thiery's method calculates the radius every time, which takes a lot of time. Our method does not need this step. So, the simplification efficiency of our methods is higher. (a) Thiery et al.'s calculation process. (b) Our calculation process.

Declaration of conflicting interests

The author(s) declared no potential conflicts of interest with respect to the research, authorship, and/or publication of this article.

Funding

The author(s) disclosed receipt of the following financial support for the research, authorship, and/or publication of this article: The work is supported by the National Natural Science Foundation of China (Nos. 61432003, 61672547, and 61502546), Science and Technology Planning Project of Guangdong Province (No. 2015B010129008), and Innovation and development major Project of Guangxi (No. 2018AA12003).

References

1. Liu Y, Zeng QA, Wang YH, et al. Data processing techniques in wireless multimedia sensor networks. *Int J Distrib Sens N* 2015; 2015: 260384.
2. Thiery JM, Boubekeur T and Eisemann E. Animated mesh approximation with sphere-meshes. *ACM T Graphic* 2016; 35(3): 1–13.
3. Payan F, Hahmann S and Bonneau GP. Deforming surface simplification based on dynamic geometry sampling. In: *IEEE international conference on shape modeling and applications (SMI'07)*, Lyon, 13–15 June 2007, pp.71–80. New York: IEEE.
4. Mohr A and Gleicher M. Deformation sensitive decimation. Technical report, University of Wisconsin, Madison, WI, 4 July 2003.
5. Soonjo K, Hyungki K, Duhwan M, et al. Determination of appropriate level of detail of a three-dimensional computer-aided design model from a permissible dissimilarity for fully automated simplification. *Adv Mech Eng* 2017; 9(7): 1–12.
6. Huska M and Morigi S. A meshless strategy for shape diameter analysis. *Visual Comput* 2017; 33(3): 303–315.
7. Dassi F, Ettinger B, Perotto S, et al. A mesh simplification strategy for a spatial regression analysis over the cortical surface of the brain. *Appl Numer Math* 2015; 90: 111–131.
8. Wei N, Xu T, Gao K, et al. Mesh simplification weighted by Voronoi poles feature computed saliency. *J Graph* 2017; 38(3): 314–319.
9. Zhu Y, Tian L and Wan T. Automatic polygon mesh repair and simplification for three-dimensional human modeling. *Mod Phys Lett B* 2017; 31(19–21): 1740077.
10. Chen HK and Li MW. A novel mesh saliency approximation for polygonal mesh segmentation. *Multimed Tools Appl* 2018; 77: 17223–17246.
11. Dehne F, Langis C and Roth G. Mesh simplification in parallel. In: Goscinski A, Zhou W, Ip HHS, et al. (eds) *Algorithms and architectures for parallel processing: ICA^{3}PP 2000*. Ottawa, ON, Canada: National Research Council of Canada, 2000, pp.281–290.
12. Papageorgiou A and Platis N. Triangular mesh simplification on the GPU. *Visual Comput* 2015; 31(2): 235–244.
13. Hua Z, Huang Z and Li J. Mesh simplification using vertex clustering based on principal curvature. *Int J Multimed Ubiquitous Eng* 2015; 10(9): 99–110.
14. Kavan L, Sloan P and O'Sullivan C. Fast and efficient skinning of animated meshes. *Comput Graph Forum* 2010; 29(2): 327–336.
15. Landreneau E and Schaefer S. Simplification of articulated meshes. *Eurographics* 2009; 28(2): 347–353.
16. Zhang S, Zhao J and Wang B. A local feature based simplification method for animated mesh sequence. In: *International conference on computer engineering and technology (ICCET)*, Chengdu, China, 16–18 April 2010, vol. 1, pp.681–685. New York: IEEE.
17. Shamir A, Bajaj C and Pascucci V. Multi-resolution dynamic meshes with arbitrary deformations. In: *Proceedings of the conference on visualization*, Salt Lake City, UT, 8–13 October 2000, pp.423–430. New York: IEEE.
18. Shamir A and Pascucci V. Temporal and spatial level of details for dynamic meshes. In: *Proceedings of the ACM symposium on virtual reality software and technology*, Banff, AB, Canada, 15–17 November 2001, pp.77–84. New York: ACM.
19. Garland M. Surface simplification using quadric error metrics. *ACM SIGGRAPH Comput Graph* 1997; 1997: 209–216.
20. Houle J and Poulin P. Simplification and real-time smooth transitions of articulated meshes. In: *Proceedings of graphics interface*, Ottawa, ON, Canada, 7–9 June 2001, vol. 2001, pp.55–60. Toronto, ON, Canada: Canadian Human-Computer Communications Society.
21. Zhang Y, Ma L, Zhou Y, et al. Automatic superpixel generation algorithm based on a quadric error metric in 3D space. *Signal Image Video P* 2017; 11(3): 471–478.
22. Wang X and Yuan Z. A novel weak form three-dimensional quadrature element solution for vibrations of elastic solids with different boundary conditions. *Finite Elem Anal Des* 2018; 141: 70–83.
23. Lastra R and Paktunc D. An estimation of the variability in automated quantitative mineralogy measurements through inter-laboratory testing. *Miner Eng* 2016; 95: 138–145.
24. Bors AG and Luo M. Optimized 3D watermarking for minimal surface distortion. *IEEE T Image Process* 2013; 22(5): 1822–1835.
25. Xing L, Zhang X, Wang CCL, et al. Highly parallel algorithms for visual-perception-guided surface remeshing. *IEEE Comput Graph* 2014; 34(1): 52–64.
26. Zolotarev PN, Arshad MN, Asiri AM, et al. A possible route toward expert systems in supramolecular chemistry: 2-periodic H-bond patterns in molecular crystals. *Cryst Growth Des* 2014; 14(14): 1938–1949.