Deforming Surface Simplification Based on Feature Preservation

Shixue Zhang[1] and Enhua Wu[1,2]

[1] Dept. of Computer and Information Science, University of Macau, Macao, China
[2] State Key Lab. of Computer Science, Institute of Software,
Chinese Academy of Sciences, China
{ya57406,EHWu}@umac.mo

Abstract. In computer graphics, methods for mesh simplification are common. However, most of them focus on static meshes, only few works have been proposed for simplifying deforming surfaces. In this paper, we propose a new method for the multiresolution representation of time-varying meshes based on deformation area and feature preservation. Our method uses the famous QEM (quadric error metric) as our basic metric. The idea is to modify the edge collapse cost by adding the deformation and sharp feature weight to the aggregated quadrics errors when computing the unified edge contraction sequence, then adjust this sequence slightly for each frame to get a minimum geometry distortion. Our approach is fast, easy to implement, and as a result good quality dynamic approximations with well-preserved fine details can be generated at any given frame.

Keywords: Deforming mesh, LOD, Sharp feature, Mesh simplification.

1 Introduction

Nowadays more and more deforming meshes, also called time-varying surfaces or dynamic meshes, are widely used in many applications such as movies, games, simulations and so on. These surfaces are often represented by dense triangular meshes with high resolution, so sometimes we have to decimate the redundant details of the models for more efficient visualization processing, transmission and storage. Many mesh simplification methods have been proposed to simplify original models by repeatedly removing vertices, edges, or faces. However most of them are to deal with static meshes, and very little work has been made to address the problem on how to maintain accurate approximations of time-varying surfaces. In this paper, we propose an efficient method for generating progressive deforming meshes with high quality.

To simplify deforming meshes, one naive way is to simplify the models for each frame independently. This solution has the minimum error from the original model. 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. Moreover, this will waste a great deal of space.

L. Ma, R. Nakatsu, and M. Rauterberg (Eds.): ICEC 2007, LNCS 4740, pp. 139–149, 2007.

Some previous methods focus on preserving the static connectivity, i.e. the connectivity of the deforming surfaces remains unchanged for all frames. Such adaptations are inadequate and would cause arbitrarily bad approximations when deformation is highly non-rigid, since it does not take time-varying deformation into consideration.

We therefore propose a new method for simplifying deforming meshes based on feature preservation. Our method is a better tradeoff between the temporal coherence and mesh distortion, i.e. we try to maximize the temporal coherence while minimizing the visual distortion during the simplification process. Our idea is to first calculate the aggregated QEM error for each edge as DSD [26] method. We use the collapsing cost variation to measure the deformation degree of the deforming mesh. We calculate the sharp features the first and last frame according to the classification of the edges. By adding the deformation and sharp feature weight to the collapsing cost, we finally get the unified edge collapse sequence. For each of the frame we first do a majority of edge collapse operations based on this sequence to maintain the temporal coherence, and do the remainder operations based on the independent QEM sequence to minimize mesh distortion. We demonstrate that this provides an efficient means of multiresolution representation of deforming meshes over all frame of an animation.

The rest of this paper is organized as follows: Section 2 will review the previous works related in deforming mesh simplification. Section 3 will mainly introduce the procedure of our algorithm in detail and discuss its advantage. Section 4 will show the experimental results and compare our method with previous methods. Finally, conclusion will be made and some future work will be given in Section 5.

2 Related Work

Simplification and LOD. There are now extensive papers on the approximation of dense polygonal meshes by coarser meshes that preserve surface detail. These methods can be roughly divided into five categories: vertex decimation [1, 4, 33], vertex clustering [24, 29], region merging [8, 19, 27], subdivision meshes [6, 13, 23, 21], and iterative edge contraction [2, 3, 9, 10, 11, 12, 15, 16, 17, 18, 28, 32]. A complete review of the methods has been given in [7, 25]. Among these methods, the process of iteratively edge contraction is predominantly utilized. Representative algorithms include those from Hoppe [15] and Garland [9]. In such methods, a simple multiresolution structure is generated on the surface that can be used for adaptive refinement of the mesh [34, 16, 22]. Traditional mesh simplification algorithm works fine on a single static model, but it is unable to be directly applied to deforming meshes since no temporal coherence has been considered.

Simplification of time-varying surfaces. Shamir et al. [30, 31] are the pioneers to address the problem of simplifying efficiently 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 can not be easily handled.

Mohr and Gleicher [26] proposed a deformation sensitive decimation (DSD) method, which directly adapt the Qslim algorithm [9] by summing the quadrics errors over each frame of the animation. The result is a single mesh that attempts to provide a good average approximation over all frames. Consequently this technique provides a pleasant result only when the original surfaces do not present strong deformation.

DeCoro and Rusinkiewicz [5] introduced a method of weighing possible configuration of poses with probabilities. With articulated meshes, skeleton transformation is incorporated into standard QEM algorithm, and users must specify the probability distribution for each joint. This method works quite well, but is limited to a very specific class of deformations.

Kircher and Garland [20] 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 [7]. This method seems to be particularly efficient because of its connectivity transformation.

Huang et al. [14] proposed a method based on the deformation oriented decimation error metric and dynamic connectivity update. They use 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.

3 Our Algorithm

Our algorithm consists of three parts: (1) Add the deformation information to the collapsing cost to preserve areas with large deformation. (2) Add the sharp feature weight to the cost to preserve the fine sharp features of the model. (3) Adjust the edge collapse sequence for each frame to reduce the approximation distortion. Next we will introduce the algorithm in detail.

3.1 Deformation Area Preservation

Our algorithm is based on the QSlim [9] which is now considered as one of the most efficient methods for static mesh simplification, so we should first have a quick review of it. Qslim iteratively selects an edge $(\mathbf{v}_i, \mathbf{v}_j)$ with the minimum contraction cost to collapse and replace this edge with a new vertex $\mathbf{u}$ which minimizes the contraction cost. For measuring the contraction cost of an edge, it utilizes the quadratic error metric (QEM) to measure the total squared distance of a vertex to the two sets of planes $\mathbf{P}(\mathbf{v}_i)$ and $\mathbf{P}(\mathbf{v}_j)$ adjacent to $\boldsymbol{v}_i$ and $\boldsymbol{v}_j$ respectively. A plane can be represented with a 4D vector $\mathbf{p}$, consisting of the plane normal and the distance to the origin. Hence, the squared distance of a vertex $\mathbf{v}$ to a plane $\mathbf{p}$ equals $\mathbf{v}^T(\mathbf{p}\mathbf{p}^T)\mathbf{v}$. The QEM error function $\boldsymbol{ec}_{ij}$ for a vertex $\mathbf{v}$ to replace the edge $(\mathbf{v}_i, \mathbf{v}_j)$ is

$$ec_{ij}(v) = \sum_{p \in P(v_i)} v^T (pp^T) v + \sum_{p \in P(v_j)} v^T (pp^T) v$$
$$= v^T Q_i v + v^T Q_j v$$

Garland also suggests using an area-weighted quadric error metric for better results [7] and defines the QEM error function as:

$$ec_{ij}(v) = v^T(w_iQ_i + w_jQ_j)v = v^TQ_{ij}v$$

where w_i is the total area of triangles adjacent to $\mathbf{v}_i$ and w_j is defined similarly. Hence, the QEM cost QEM_{ij} for contracting an edge $(\mathbf{v}_i, \mathbf{v}_j)$ is defined as $\boldsymbol{ec_{ij}}(\mathbf{u_{ij}})$, in which $\mathbf{u}_{ij}$ is the vertex minimizing $\boldsymbol{ec_{ij}}(\mathbf{v})$. QSlim simplifies a mesh by iteratively finding the edge $(\mathbf{v}_i, \mathbf{v}_j)$ with the minimum QEM_{ij}, performing an edge-collapse operation to replace $(\mathbf{v}_i, \mathbf{v}_j)$ with a new vertex $\mathbf{u}_{ij}$ and updating the edge contraction costs related to $\mathbf{u}_{ij}$ until the desired vertex count $\boldsymbol{m}$ is reached.

To extend QEM to handle deforming meshes, a naive way is to use QEM to obtain an edge-collapse sequence for the first frame, and then apply this sequence to all frames. Since all frames use the same edge-collapse sequence, they have the same connectivity. The disadvantage of this approach is obvious. Features of other frames might be removed if they are not features in the first frame. The deformation sensitive decimation (DSD) algorithm addresses this problem by summing QEM costs across all frames [26]. The DSD contraction cost for an edge $(\mathbf{v}_i, \mathbf{v}_j)$ is defined as

$$DSD_{ij} = \sum_{t=1}^{f} QEM_{ij}^{t} = \sum_{t=1}^{f} u_{ij}^{t\,T} Q_{ij}^{t} u_{ij}^{t}$$

where u_{ij}^{t} minimizes the QEM cost QEM_{ij}^{t} for the edge $(\mathbf{v}_i, \mathbf{v}_j)$ at frame t. Hence, DSD tends to preserve edges that are geometric features more often in the animation.

We use the change of edge-collapse cost to measure the surface deformation degree. For areas with large deformation, the change of the collapsing cost must be prominent. On the other hand, collapsing cost may change slightly in areas with small deformation. We append additional deformation weight to the DSD cost, which can postpone the edge contraction in areas with large deformation. We define the deformation weight to be:

$$\sum_{t=1}^{f} \left| ec_{ij}^{t} - \overline{ec}_{ij} \right|$$

where $\overline{ec}_{ij}$ is the average collapse cost of edge $(\mathbf{v}_i, \mathbf{v}_j)$ over all of the frames. We add this cost to the DSD contraction cost:

$$cost_{ij}^{'} = DSD_{ij} + k_1 * \sum_{t=1}^{f} \left| ec_{ij}^{t} - \overline{ec}_{ij} \right| = \sum_{t=1}^{f} ec_{ij}^{t} + k_1 * \sum_{t=1}^{f} \left| ec_{ij}^{t} - \overline{ec}_{ij} \right|$$

where $\boldsymbol{k_1}$ is a user-specified coefficient to adjust the influence the deformation degree. In our experiment, we simply set $\boldsymbol{k_1}$ to 1.

3.2 Sharp Feature Preservation

In order to preserve the sharp features of the original model, we should first define the sharp features. And this involves the classification of edges as Fig. 1 shows. The calculation of sharp features in each frame is obviously a time-consuming process,

also wastes a lot of space. Our idea is to only compute the sharp features in the first and last frame, which can represent most of the characteristics of the animation. If the model is not much deformed during the animation, that is the first and last frame may have the identical sharp features, and these features are enhanced. On the other hand, if the deformation is totally non-rigid, that is the first and last frame may have totally different features, our method may possibly preserve most of the respective features during the animation. We implement the above by adding the feature weight into the collapse cost, as the follow equation shows:

$$cost_{ij} = (1 + k_2 * Feature\_Weight) \sum_{t=1}^{f} ec_{ij}^{t} + k_1 * \sum_{t=1}^{f} \left| ec_{ij}^{t} - \overline{ec}_{ij} \right|$$

where k_2 is between 0 and 1 to adjust the influence of the feature weight. If the user knows whether the deformation is rigid in advance, we can set more proper k_2 to get better result. Next we will discuss how to assign the value of *Feature_Weight*, and this involves the classification of edges [2]. We classify the edges into the following five types as Fig. 1 shows:

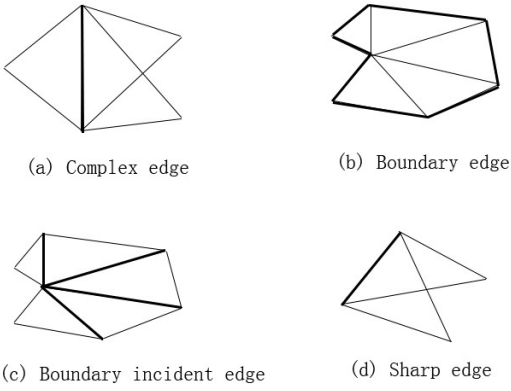

Fig. 1. Classification of edges

①Complex Edge

Since shape changes of the 3D model due to contraction of complex edges could be severe, we give a large penalty on contracting the complex edges so that the contraction operation of complex edges can be conducted later relative to other types of edges. We can easily expect that the more incident triangles from the edge exist, the severer shape variation occurs. Hence, we define the *Complex_Weight* to be proportional to the number of incident triangles from the edge in the 3D model.

Complex_Weight = (# of incident triangles – 3)

②Boundary Edge

The shape variation of the 3D model becomes severe after contracting the boundary edges as well. Therefore, we must place a large penalty on contracting the boundary

edges. In addition, it would be more effective if we weigh differently according to the features in the 3D model.

We define θ as the angle between the incident boundary edge and itself, so a boundary edge may have two values of θ (θ_1 and θ_2) as Fig. 2 shows.

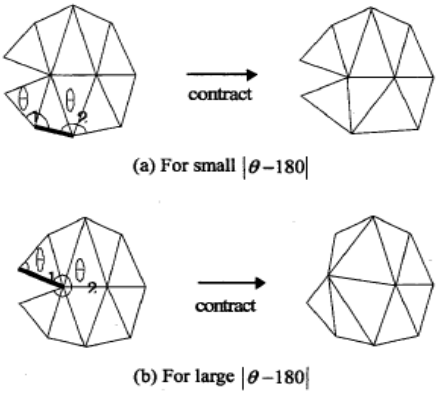

Fig. 2. Contraction of boundary edge

We have the following conclusion by investigating the variation after collapsing.

(1) If the value of |θ-180| is small, the shape variation of the model is limited.

(2) If the value of |θ-180| is large, the shape variation of the model is considerable.

Thus, we define the *Boundary_Weight* to be proportional to the maximum value of cosθ.

$$Boundary\_Weight = max\{cos\theta_1, cos\theta_2\}$$

③Boundary Incident Edge

Contraction of boundary incident edges can yield substantial changes on the shape of the 3D model. Since the shape change depends on the number of incident boundary edges, we define the *Boundary_Incident_Weight* to be proportional to the number of incident boundary edges.

$$Boundary\_Incident\_Weight = (\#\ of\ incident\ boundary\ edges) / 4$$

④Sharp Edge

Sharp edges connect two faces in the original mesh, and we define it as the angle (α) between the two neighboring faces should be smaller than a user-specified threshold (we set it to be 60°). Collapsing this kind of edges may cause large distortion of the model, so we define the *Shape_Weight* of it as sin α.

$$Sharp\_Weight = \sin\alpha$$

⑤Interior Edge

The rest of the edges are considered as interior edges. Since shape changes of the 3D model after contracting the interior edges is milder relative to any other types of edges, no penalty on contracting the interior edges would be reasonable.

Finally we define the *Feature_Weight* as follows:

$$\begin{aligned} Feature\_Weight = {} & IsComplex * Complex\_Weight \\ & + IsBoundary * Boundary\_Weight \\ & + IsBoundaryIncident * Boundary\_Incident\_Weight \\ & + IsSharp * Sharp\_Weight \end{aligned}$$

where *IsComplex, IsBoundary, IsBoundarylncident* and *IsSharp* are Boolean variables. In other words, they can take 1 for TRUE or 0 for FALSE.

3.3 Distortion Adjustment

Based on the final collapse cost described above, we can finally get the overall edge-collapse sequence. This unified collapse sequence can maintain the connectivity of the whole mesh sequence unchanged. And also it has the optimal temporal coherence. However, since the unified order is applied, approximation can't be very satisfying on every frame. So our idea is to adjust this sequence slightly to get better approximation and preserve more fine details.

Considering the model in a certain frame, the independent QEM method can generate the optimal edge-collapse sequence. We can first do most of the collapse operation (we assume the proportion to be $\boldsymbol{P}$, $\boldsymbol{P}$ is very near to 1) based on the previous calculated cost, then do the rest collapse based on the QEM cost. Since most of the operation is based on static connectivity, the temporal coherence is still maintained. We have tested different $\boldsymbol{P}$ during the animation, and we found that set $\boldsymbol{P}$ to around 96% can generate elegant result.

3.4 Algorithm Outline

Our algorithm can be summarized into the following steps:

1. Calculate the *QEM* collapsing cost of edges for each of the frame, to obtain the aggregate errors as the *DSD* method.
2. Based on the deforming sequence, measure the deformation degree to obtain the metric $cost_{ij}'$
3. Calculate the *Feature_Weight* of the first and last frame. Combined with previous cost, we get the destination cost $cost_{ij}$. Thus the final unified collapsing sequence is obtained.
4. For each of the frame, first do a majority of the edge collapse operations according to this sequence, and then do the rest of the operation following the QEM collapsing sequence to obtain the desired resolution.

4 Experimental Result

We test the result of our algorithm on a computer with Pentium4 3.2G CPU and 2G memory. We use OpenGL to render the models. The simplification of the horse gallop animation is shown in the Fig. 3. Compared to the upper full resolution models, the bottom shows our simplified results when 90% of its components are reduced. We could see that most of its features are preserved.

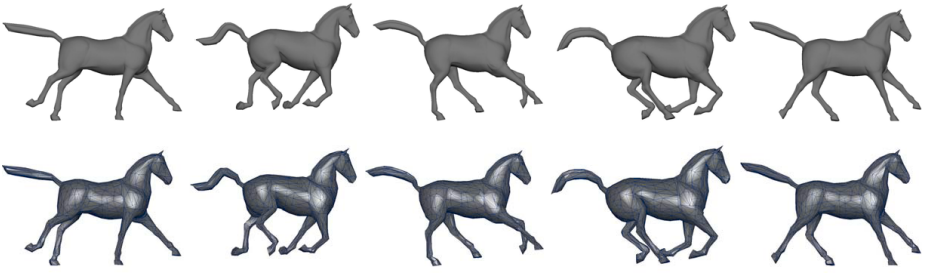

Fig. 3. Horse-gallops animation simplification with 48 frames. Original sequence: 8431v (upper), simplified models: 800v (bottom).

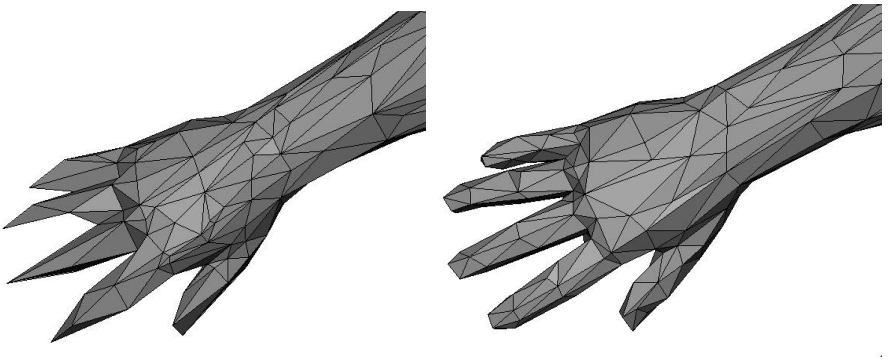

Fig. 4. Comparison of [20]'s method(left) and our method(right) of human's hand in horse-to-human morphing, the approximated versions contains 3200 vertices and 6396 triangles

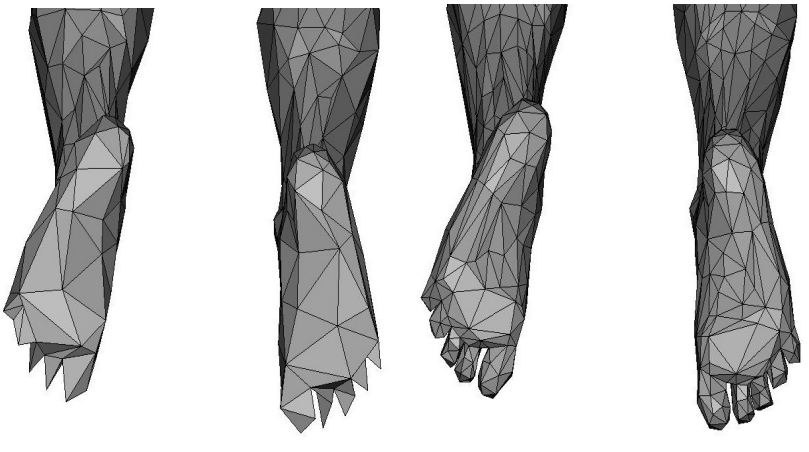

Fig. 5. Comparison of [20]'s method(left) and our method(right) of human's feet in horse-to-man morphing when 80% of its component is reduced

The next figures show the comparison of our method with the method in [20]. From Fig. 4 we could see that our method obviously preserves more details in the human's hand, also the distribution of triangle shape is more reasonable than [20]'s method. Since the features of the human's hands are not the features of the horse in the previous frames, and its deformation is totally non-rigid, our method can preserve such features much better than [20]. Another example shows the human feet approximation in Fig. 5. Our method also shows a much better simplification result than [20] with the toe details much better preserved. Since the deformation degree is so much in this area, the number of triangles in this area is much greater than [20]'s results.

An example of simplifying the whole sequence of horse-to-human morphing by using our method is shown in Fig. 6. Even the original model was reduced by 95%, our method can generate very faithful approximations.

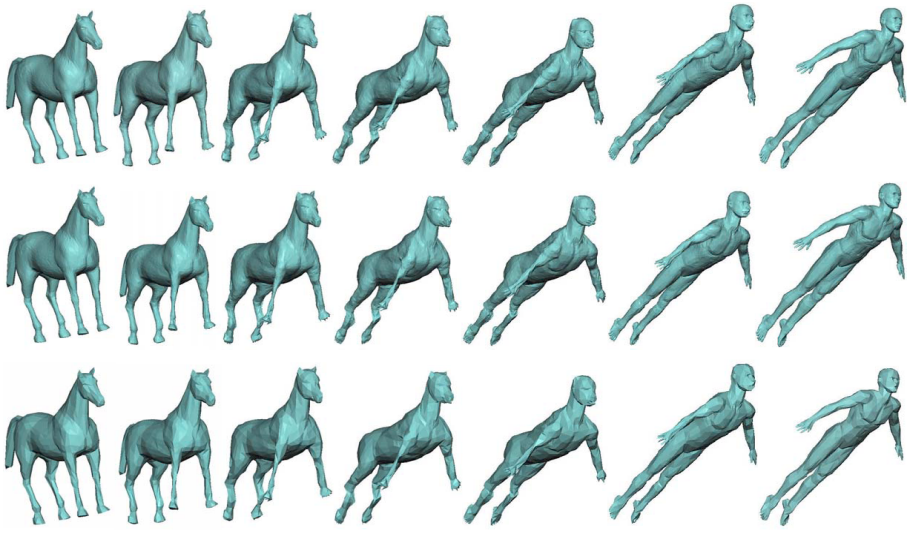

Fig. 6. A horse-to-human morphing animation with 200 frames. Top: The original sequence (17489 vertices). Middle: 3200-vertices approximation. Bottom: 800-vertices approximation.

5 Conclusion and Future Work

In this paper, we propose a simplification method for deforming surfaces based on deformation area and feature preservation. Given a sequence of meshes representing time-varying 3D data, our method produces a sequence of simplified meshes that are good approximation of the original deformed surface for a given frame. Our method extends the DSD formulation by adding deformation cost to preserve areas with large deformation. Feature weight is added to the collapse cost to preserve the sharp features of the original model. Finally, edge collapse sequence is adjusted for each frame to reduce the geometry distortion. Our method is easy to implement and produces better approximations than previous methods.

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.

Acknowledgments. We wish to thank Scott Kicher for providing the horse-to-man morphing sequence, Robert W. Sumner and Jovan Popovic for providing the horse-gallop data. The work has been supported by the Studentship & Research Grant of University of Macau.

References

1. Ciampalini, A., Cignoni, P., Montani, C., Scopigno, R.: Multiresolution Decimation Based on Global Error. The Visual Computer 13(5), 228–246 (1997)
2. Chang, E.Y., Ho, Y.S.: Three-dimensional Mesh Simplification by Subdivided Edge Classification. In: IEEE Region 10 International Conference on Electrical and Electronic Technology, pp. 39–42. IEEE Computer Society Press, Los Alamitos (2001)
3. Cohen, J., Olano, M., Manocha, D.: Simplifying Polygonal Models Using Successive Mappings. In: Proc. Visualization '97, pp. 395–402 (October 1997)
4. Cohen, J., Varshney, A., Manocha, D., Turk, G., Weber, H., Agarwal, P., Brooks, F., Wright, W.: Simplification Envelopes. In: ACM SIGGRAPH 1996 Conference Proceedings, pp. 119–128. ACM Press, New York (1996)
5. Decoro, C., Rusinkiewicz, S.: Pose-independent Simplification of Articulated Meshes. In: Proceedings of Symposium on Interactive 3D Graphics and Games, pp. 17–24 (2005)
6. Eck, M., Derose, T., Duchamp, T., Hoppe, H., Lounsbery, M., Stuetzle, W.: Multiresolution Analysis of Arbitrary Meshes. In: ACM SIGGRAPH 1995 Conference Proceedings, pp. 173–182. ACM Press, New York (1995)
7. Garland, M.: Multiresolution Modeling: Survey & future opportunities. In: Proceedings of Eurographic, Milano, pp. 49–65 (1999)
8. Garland, M., Willmott, A., Heckbert, P.S.: Hierarchical Face Clustering on Polygonal Surfaces. In: Proc. ACM Symp. Interactive 3D Graphics, pp. 49–58 (March 2001)
9. Garland, M., Heckbert, P.S.: Surface Simplification using Quadric Error Metrics. In: ACM SIGGRAPH 1997 Conference Proceedings, pp. 209–216. ACM Press, New York (1997)
10. Gieng, T.S., Hamann, B., Joy, K.I., Schussman, G.L., Trotts, I.J.: Constructing Hierarchies for Triangle Meshes. IEEE Trans. Visualization and Computer Graphics 4(2), 145–161 (1998)
11. Guéziec, A.: Surface Simplification with Variable Tolerance. In: Proc. Second Ann. Int'l Symp. Medical Robotics and Computer Assisted Surgery, pp. 132–139 (November 1995)
12. Guéziec, A.: Locally Toleranced Surface Simplification. IEEE Trans. Visualization and Computer Graphics 5(2), 168–189 (1999)
13. [GVS00] Guskov, I., Vidimce, K., Sweldens, W., Schroder, P.: Normal Meshes. In: SIGGRAPH '00 Conf. Proc., pp. 95–102 (2000)
14. 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, pp. 53–62. ACM Press, New York (2006)
15. Hoppe, H.: Progressive meshes. In: ACM SIGGRAPH 1996 Conference Proceedings, pp. 99–108. ACM Press, New York (1996)
16. Hoppe, H.: View-dependent Refinement of Progressive Meshes. In: ACM SIGGRAPH 1997 Conference Proceedings, pp. 189–198. ACM Press, New York (1997)

17. Hoppe, H.: New Quadric Metric for Simplifying Meshes with Appearance Attributes. In: Proc. IEEE Visualization '99, pp. 59–66 (1999)
18. Hoppe, H., DeRose, T., Dunchamp, T., McDonald, J., Stuetzle, W.: Mesh Optimization. In: SIGGRAPH '93 Conf. Proc., pp. 19–25 (1993)
19. Kalvin, A., Taylor, R.: Superfaces: Polygonal Mesh Simplification with Bounded Error. IEEE Computer Graphics and Applications 16, 64–77 (1996)
20. Kircher, S., Garland, M.: Progressive Multiresolution Meshes for Deforming Surfaces. In: Proceedings of ACM SIGGRAPH/Eurographics Symposium on Computer Animation, pp. 191–200. ACM Press, New York (2005)
21. Lounsbery, M., DeRose, T., Warren, J.: Multiresolution Analysis for Surfaces of Arbitrary Topological Type. ACM Trans. on Graphics 16(1), 34–73 (1997)
22. Luebke, D., Erikson, C.: View-dependent Simplification of Arbitrary Polygonal Environments. In: ACM SIGGRAPH 1997 Conference Proceedings, pp. 199–208. ACM Press, New York (1997)
23. Lee, A., Moreton, H., Hoppe, H.: Displaced Subdivision Surfaces. In: SIGGRAPH '00 Conf. Proc., pp. 85–94 (2000)
24. Low, K.L., Tan, T.S.: Model Simplification Using Vertex-Clustering. In: Proc. ACM Symp. Interactive 3D Graphics, pp. 75–82 (1997)
25. Luebke, D., Reddy, M., Cohen, J.: Level of Detail for 3-D Graphics. Morgan Kaufmann, San Francisco (2002)
26. Mohr, A., Gleicher, M.: Deformation Sensitive Decimation. Tech. rep., University of Wisconsin (2003)
27. Mangan, A.P., Whitaker, R.T.: Partitioning 3D Surface Meshes Using Watershed Segmentation. IEEE Trans. Visualization and Computer Graphics 5(4), 221–308 (1999)
28. Popovic, J., Hoppe, H.: Progressive: Simplicial Complexes. In: SIGGRAPH '97 Conf. Proc., pp. 217–224 (August 1997)
29. Rossignac, J., Borrel, P.: Multi-Resolution 3D Approximations for Rendering Complex Scenes. In: Falcidieno, B., Kunii, T. (eds.) Modeling in Computer Graphics, pp. 455–465. Springer, Heidelberg (1993)
30. Shamir, A., Bajaj, C., Pascucci, V.: Multi-resolution Dynamic Meshes with Arbitrary Deformations. In: IEEE Visualization 2000 Conference Proceedings, pp. 423–430. IEEE Computer Society Press, Los Alamitos (2000)
31. Shamir, A., Pascucci, V.: Temporal and Spatial Level of Details for Dynamic Meshes. In: Proceedings of ACM Symposium on Virtual Reality Software and Technology, pp. 77–84. ACM Press, New York (2001)
32. Sander, P.V., Snyder, J., Gortler, S.J., Hoppe, H.: Texture Mapping Progressive Meshes. In: SIGGRAPH '01 Conf. Proc., pp. 409–416 (2001)
33. Schroeder, W.J., Zarge, J.A., Lorensen, W.E.: Decimation of Triangle Meshes. In: ACM Computer Graphics (SIGGRAPH 1992 Conference Proceedings), vol. 26(2), pp. 65–70 (1992)
34. Xia, J.C., Varshney, A.: Dynamic View-dependent Simplification for Polygonal Models. In: IEEE Visualization 1996 Conference Proceedings, pp. 327–334. IEEE Computer Society Press, Los Alamitos (1996)