# Efficient and Feature-Preserving Triangular Mesh Decimation

Muhammad Hussain[1,2] Yoshihiro Okada[1,2] Koichi Niijima[1]
[1]Graduate School of Information Science and Electrical Engineering,
Kyushu University, 6-1, Kasuga Koen, Kasuga, Fukuoka, 816-8580, Japan.
[2]Intelligent Cooperation and Control, PRESTO, JST.
{mhussain, okada, niijima}@i.kyushu-u.ac.jp

## ABSTRACT

Most of the existing algorithms for decimation of triangular meshes perform poorly at very low levels of detail. We propose a new automatic method for the decimation of triangular meshes, which performs better as compared to the notable existing algorithms at low levels of detail, preserves visually important parts of the mesh and thus keeps the semantic or high level meaning of the model. The proposed algorithm is based on greedy approach and exploites a new method of measuring geometric error employing a form of vertex visual importance that helps to keep visually impotant vertices even at low levels of detail and causes to remove other kinds of vertices, which do not profoundly influence the overall shape of the model. In addition, the proposed method has less momory overhead as compared to most of the published algorithms and is faster in terms of running times. The results of the algorithm have been compared numerically, visually, in terms of execution times and memory consumption with the sate-of-the-art decimation methods to strengthen the efficiency and quality of the algorithm.

## Keywords

Mesh decimation, Multiresolution modeling, Level of detail, Edge collapse, Vertex visual importance.

## 1. INTRODUCTION

Various applications of interactive computer graphics, like animation, scientific visualization, and virtual reality, involve the manipulation of geometric models that are commonly represented by triangular meshes because of wide acceptance of triangle as a basic primitive on rendering systems. The pursuit of realism and high visual fidelity on one hand and the latest advances on scanning devices and CAD systems on the other hand has given rise to huge triangular meshes whose complexity and size often exceed the capacity of available graphics rendering systems. The only way to deal with this problem and to make such models available for real time applications is to induce different levels of resolution on a mesh so that an application can exploit an appropriate level of detail based on the compromise between visual fidelity, the limitations of rendering system and the requirements of the application. A number of solutions have been proposed for geometric simplification of polygonal models during the last decade addressing the different aspects of the problem and keeping in view different objectives. Mainly, there are three factors that determine the quality of a simplification algorithm: computational efficiency, memory overhead, and the quality of generated models. Different published algorithms have different strengths and weaknesses in terms of the quality of approximations, running times and memory overhead; some methods e.g. [Bro00, Ros93, Sch92] are faster in running times but produce poor approximations; some methods e.g. [Cia96, Hop96, Pet98] generate good quality approximations but perform poorly in running times; the algorithms e.g. [Gar97] are fast and generate good quality approximations but suffer from large memory overhead. According to our information there does not exist any algorithm that produces extremely low level approximations while preserving the semantic meaning of the model and, still is computationally efficient and has low memory overhead.

We propose a new decimation algorithm that is not only memory efficient and involves short running times but also produces approximations at extremely



***Journal of WSCG, Vol.12, No.1-3., ISSN 1213-6972***
*WSCG'2004, February 2-6, 2004, Plzen, Czech Republic.*

low levels of resolution while keeping the semantic meaning of a model. Our algorithm is driven by half-edge collapse operation and a memory efficient global measure of geometric deviation that employs vertex visual importance value to prevent the decimation of visually important parts and high frequency details of the model.

The rest of the paper has been organized as follows. Section 2 gives an overview of some of the state-of-the-art directly related decimation algorithms. The essential ingredients of the algorithm have been presented in detail in Section 3, and Section 4 outlines the algorithm. Section 5 discusses the performance and quality of the algorithm by performing comparison with some published methods. Section 6 concludes the paper.

## 2. RELATED WORK

A lot of algorithms have been proposed addressing the problem of decimation of polygonal models, an interested reader is referred to consult [Hec97, Cig98a, Lue97]. Here we would present an overview of sate-of-the-art edge-collapsed based algorithms to establish a ground for comparison. Progressive Mesh of Hoppe [Hop96] is the first algorithm that employed edge collapse operator; although this algorithm produces good quality approximations but it has large memory overhead and its performance in terms of execution time is very poor. After this, many edge collapse based algorithms have been proposed.

QEM based algorithm of Garland and Heckbert [Gar97] employs a more general form of edge collapse and stores the geometric deviation as the square of the distance of incident planes on a vertex as 4×4 symmetric matrix, which is based on the idea proposed in [ Ron96] and uses this matrix to compute the cost of an edge and the optimal position of the constituent vertex. This algorithm is one of the most efficient methods in terms of running times and produces good quality approximations but it suffers from large memory overhead, for each vertex it consumes 40 bytes of memory just to store error metric in the form of a 4×4 symmetric matrix. Also it is at a loss to preserve essential features of a model at a very low level of resolution. The idea of measuring geometric fidelity proposed in [Pet98] is a memory efficient form of QEM, but is not as efficient as original QEM in terms of running times; it takes about 5 times more execution time. The decimation algorithm proposed by Broadsky and Watson [Bro00] is based on refinement; it is even faster than QEM algorithm but produces poor approximations. The algorithm proposed by Kim et al [Kim02] employs edge collapse operator and

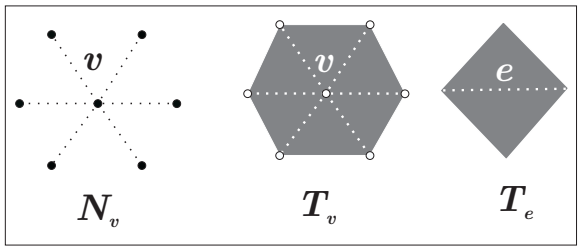


**Figure 1. $N_v$: one ring neighbors of $v$, $T_v$: triangles incident on $v$, and $T_e$: triangles incident on edge $e$.**

exploits discrete curvature norm to measure geometric deviation; this method seems to be good at preserving visually important detail of a model at a low level of detail. Although the authors did not report the running times, but it seems because of the computation of their proposed measure of discrete curvature norm, this algorithm involves very long running times. Algorithms proposed by Hussain et al [Hus03a, Hus03b] are almost as fast as QEM algorithm in execution time and produce comparable results, and have low memory overhead, but they are also unable to preserve semantic meaning of a model at low levels of detail.

## 3. MEASURE OF VISUAL FIDELITY

### Some definitions

Without lose of generality, we assume a triangular mesh $M$ because any polygon can be represented by a set of triangles.

For the sake of clarity and compactness of expression, we introduce the following definitions.

$v$ : a vertex of $M$ with its geometric counterpart as a 3D vertex $v$. $v$ is a flat vertex if $T_v$ is a co-planner set.

$e_{ij}$ : an edge of $M$ connecting the set of vertices $\{v_i, v_j\}$.

$\vec{e}_{ij}$ : a half edge of $M$ represented by the ordered pair $(v_i, v_j)$. $v_i$ and $v_j$ are termed as origin and head of $\vec{e}_{ij}$. Each edge $e_{ij} = \{v_i, v_j\}$ is equivalent to two half edges $\vec{e}_{ij}$, $\vec{e}_{ji}$.

$t$ : a triangular face of $M$ is a set of oriented edges ( $\vec{e}_{ij}$ , $\vec{e}_{jk}$ , $\vec{e}_{ki}$ ) or equivalently an oriented set of vertices $(v_i, v_j, v_k)$.

$T_v$ : the set of triangles incident on vertex $v$ (see Figure 1). It is termed as star of $v$.

$T_e$ : the set of triangles incident on edge $e$ (see Figure 1).

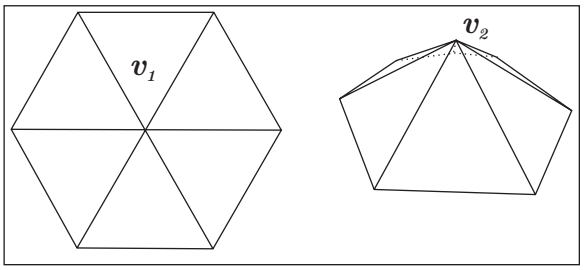


**Figure 2. $v_1$ is a flat vertex i.e. $w_{v_1} = 0$, and $v_2$ has visual importance greater than zero.**

$N_v$ : the set of vertices in the 1-ring neighborhood of vertex $v$.

## Topological Operator

We employ half-edge collapse to simplify the topology of $M$ because it is easy to implement and does not create new geometry, so it makes the progressive transmission more efficient and induces nested hierarchies on unstructured meshes that can facilitate further applications [Kob98].

## Visual Importance of a Vertex

We associate with each vertex a value that represents its visual importance and helps to determine the sequence of edge collapses in such a way that even extremely low level versions of a model keep the semantic meaning of the model. We define the visual importance of vertex $v$ by :

$$w_v = 1 - \|\boldsymbol{k}_v\|,$$

where

$$\boldsymbol{k}_v = \frac{\sum_i \Delta_i \vec{n}_i}{\sum_i \Delta_i},$$

here $\vec{n}_i$ is unit normal to the triangle $t_i \in T_v$, $\Delta_i$ is its area and summation is over all triangles in $T_v$. $\|\boldsymbol{k}_v\|$ is Euclidean norm of $\boldsymbol{k}_v$. It is quite obvious that

$\|\boldsymbol{k}_v\| = 1$ or $w_v = 0 \Leftrightarrow v$ is a flat vertex,

and

$\|\boldsymbol{k}_v\| < 1$ or $0 < w_v < 1 \Leftrightarrow v$ is not a flat vertex.

It means that the visual importance of a vertex is zero if it is a flat vertex, otherwise it is a real value in the range of 0 and 1. A vertex would be removed if its visual importance is 0 otherwise it would be kept according to its importance value.

## Geometric Error Measure

A half–edge collapse transformation, say $\bar{e}_{oi}$ : $(v_0, v_i) \rightarrow v_i$ causes the triangles $T_{e_{oi}}$ to degenerate and the remaining triangles $T_{v_o} - T_{e_{oi}}$ to undergo a transformation. Degenerate triangles $T_{e_{oi}}$ are removed and the transformation of each of $t_i \in T_{v_o} - T_{e_{oi}}$ can be interpreted to be a rotation about its edge opposite to $v_0$ followed by scaling and shearing, see Figure 3; the rotation accounts for geometric error. Consider a typical triangle $t = (v_o, v_1, v_2)$ shown in Figure 3, its rotation about edge e = $\{v_1, v_2\}$ causes the vertex $v_0$ to traverse an arc. Analogous to the arc length of a circle, we define the following quantity to account for the geometric error caused by triangular face $t$:

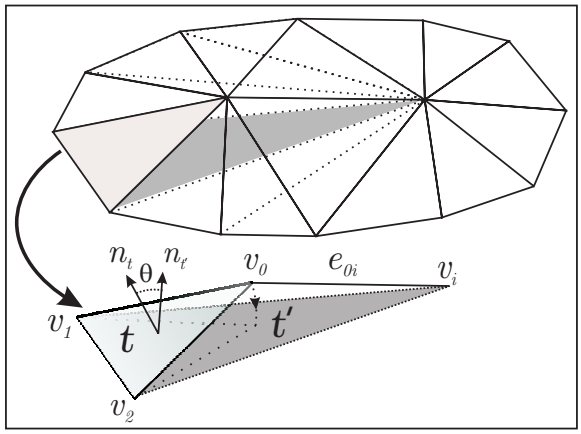


**Figure 3. Half –edge collapse operation.**

$$Q_t = l_t . \theta_t,$$

where $l_t = 0.5(\Delta_t + \Delta_{t'})$ with $\Delta_t$ and $\Delta_{t'}$ to be the areas of triangles $t$ and $t'$, and $\theta_t$ is the angle described by the unit normal to triangle $t$ when it performs rotation about edge $e_{12} = \{v_1, v_2\}$. The computation of $\theta_t$ is involved and would slow down the simplification process, so in our implementation, we replace $\theta_t$ by $1 - \vec{n}_t . \vec{n}_{t'}$ where $\vec{n}_t$ and $\vec{n}_{t'}$ are unit normals to the triangles $t$ and $t'$; for our purpose this approximation works quite well, and is computed efficiently.

The cost of collapse of edge $e_r$ is the sum of geometric errors introduced by each of $t_i \in T_{v_o} - T_{e_{oi}}$. So

$$Cost(\bar{e}_{oi}) = \sum_{t \in T_{v_o} - T_{e_{oi}}} Q_t$$

This measure of geometric error would associate normally less cost with edges on the boundary, and once the algorithm enters a local minima along the boundary, it would be trapped over there and would start to collapse boundary indiscriminately. To tackle this problem, special heuristics are employed. Boundary half-edges are categorized into two main types: (1) the half-edges having either origin or

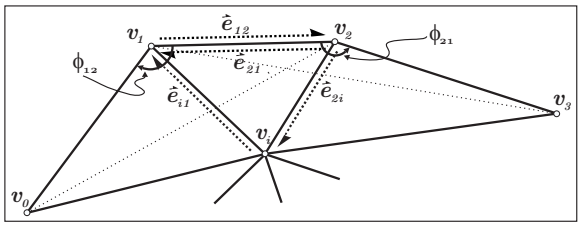


**Figure 4. Vertex $v_i$ is an interior vertex and $v_0$, $v_1$, $v_2$, and $v_3$ are boundary vertices.**

| Model | Model Size (# faces) | MELOD | QSlim |
|---|---|---|---|
| Fandisk | 12,946 | 0.98 | 0.97 |
| Bunny | 69,451 | 4.65 | 3.67 |
| Horse | 96,966 | 6.39 | 5.28 |
| Male | 605,902 | 44.05 | 35.58 |

**Table 1. Execution times (in seconds) of MELOD and QSlim to decimate each model to one face.**

head on boundary e.g. $\vec{e}_{i1} = (v_i, v_1)$ in Figure 4 and (2) the half-edges having both origin and head on boundary e.g. $\vec{e}_{12} = (v_1, v_2)$ in Figure 4. Each is dealt with separately.

Because the half-edge collapse transformation $\vec{e}_{oh} : (v_o, v_h) \rightarrow v_h$ eliminates edge $e_{oh} = \{v_o, v_h\}$ by merging $v_o$ to $v_h$, so the collapse of a half-edge having head on boundary needs not special treatment; however if its origin is on boundary, then the collapse would deform boundary severely, so the collapse of such half-edges is restricted. The half-edges of type 2 must be handled tactfully. One possible treatment is to panelize the cost of such edges with the length of the edge, but it would cause to consume more and sliver triangles to preserve boundary. Consider Figure 4, edge $e_{12} = \{v_1, v_2\}$ would be collapsed if either of the half-edges $e_{12}$, $e_{21}$ collapses, but to achieve better results this edge must be collapsed to $v_1$. It is obvious that this objective can be achieved if the costs of collapse of the half edges $\vec{e}_{12}$ and $\vec{e}_{21}$ are panelized with the length of the edge weighted by $\phi_{12}$ and $\phi_{21}$ respectively. So the cost of collapse of the half edge $\vec{e}_{12}$ would be

$$Cost(\vec{e}_{12}) = \lambda \phi_{12} \| v_1 - v_2 \| + \sum_{t \in T_{v_1} - T_{e_{12}}} Q_t$$

where $\phi_{12} = 1 - u_{21} . u_{23}$ with $u_{21}$ and $u_{23}$ being the unit vectors along the edges $\vec{e}_{21} = (v_2, v_1)$ and $\vec{e}_{23} = (v_2, v_3)$ respectively, as shown in Figure 4, and $\lambda$ is a user specified parameter used to control the quality of boundary preservation; during our experiments we found that feasible results can be found using the value of $\lambda$ in the range of 1 to 50, $\lambda$=10 is the default value in our implementation. Nearer the value of $\lambda$ is to 50, tighter the boundary is preserved.

## 4. ALGORITHM

The algorithm is driven by half-edge collapse transformation, visual importance associated with each vertex and the geometric error measure presented in Section 3; it is implemented in greedy framework to obtain sub-optimal approximation of a given geometric model at a certain level of detail. The algorithm takes a triangular mesh as input and yields the original model along with ordered list of edge collapses and their associated cost values. This progressive mesh (PM) [Hop96] representation constitutes an entire continuum of LODs of the model and an LOD approximation of desired complexity can be extracted from this PM.

The algorithm performs the following steps to decimate a triangular mesh and to yield an entire continuum of LODs.

- Compute the visual importance $w_v$ of each vertex $v$ of $M$.
- For each vertex $v_i$ of $M$, compute the cost $c_{ij}$ of each half-edge transformation $\vec{e}_{ij} : (v_i, v_j) \rightarrow v_j, v_j \in N_{v_i}$ exploiting the measure of geometric error proposed in Section 3, select the one (optimal half-edge associated with $v_i$) with minimum cost i.e. $c_i = min\{c_{ij} \; v_j \in N_{v_i}\}$, scale its cost with visual importance of $v_i$ i.e. $w_{v_i}$ and put it in the priority queue.
- Take out of the priority queue the least cost half-edge $\vec{e}_{ij}$ `= $(v_i, v_j)$ and collapse the edge $e_{ij}$ by substituting all occurrences of $v_i$ with $v_j$ and removing the triangles $T_{e_{ij}}$. Neighborhood of each vertex in $N_{v_i}$ has changed, so re-evaluate the visual importance and update the optimal half-edge associated with each $v_j \in N_{v_i}$ by computing its cost $c_j$, accumulate the cost $c_i$ by taking $c_j = max\{c_j, c_i\}$ and scale $c_j$ with $w_{v_j}$. Update the priority queue according to new costs of these half-edges.
- Repeat the previous two steps until there is no half-edge in the priority queue.

## 5. DISSCUSSION

We tested MELOD (Memory-efficient LOD modelling), the implementation of our algorithm on a wide range of public domain triangular meshes and achieved good results. To evaluate our method, we make comparison with the notable published algorithms QSlim [Gar97] and MS (Memoryless Simplification) [Pet98].

### Execution times

Table 1 lists the running times of QSlim and MELOD to decimate various models on 800MHz Intel PentiumIII machine with 384 MB of main memory. It is obvious that execution times of MELOD are quite close to those of QSlim, but it is faster than MS because according to the results reported in [Pet98] (see table 1), MS is about 5 times slower than QSlim.

### Numerical Comparison

For thorough numerical comparison, we employ maximum geometric error measure and compute it using version 2.5 of well-known I.E.I-CNR Metro tool [Cig98a] developed to compare triangular meshes. Graphs shown in Figures 5 illustrate the maximum geometric error between the original and the simplified models created by QSlim, MS and MELOD; MS is not available in public domain, metro results are the courtesy of Peter Linstrom. It is apparent that our algorithm performs better than QSlim and MS in terms of maximum geometric error.

### Memory Consumption

MELOD is memory efficient just like MS; it needs not to store any kind of geometric history. Global evaluation of geometric error is accomplished by accumulating the cost of collapse associated with half-edges, as has been explained in Step 3 of the algorithm; it does not consume extra memory unlike the accumulation of error as proposed in [Baj96, Cia96]. In addition to the storage for the mesh itself and priority queue, QSlim needs memory for storing 10 floats per vertex, so it suffers from an additional memory overhead of $40n$ bytes, where $n$ is the number of vertices in $M$.

### Visual Comparison

MELOD can decimate a model to an extremely low level of resolution while keeping its semantic meaning. Here we perform visual comparison only with QSlim (because it is available in public domain) making use of different models of different complexities. Figure 7 depicts the cow model, original and decimated versions generated by MELOD (middle) and QSlim (bottom); MELOD keeps the overall appearance of the original model by preserving visually highly important parts like eye, snort, hoofs and nipples, whereas QSlim blurs these features of high level perceptual importance.

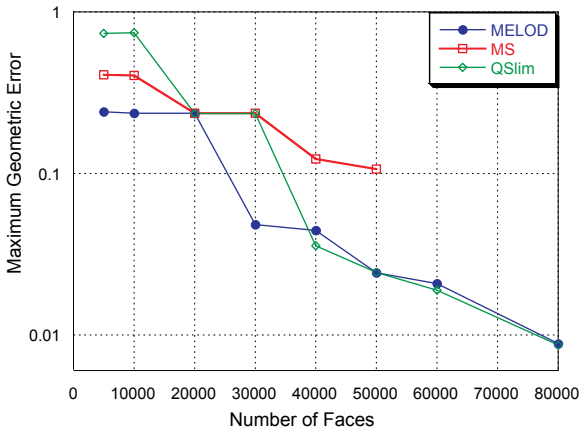


**Figure 5. Maximum geometric error for horse model**

Consider spider model shown in Figure 6, it is quite clear that the parts of the model having high level visual importance have been preserved by MELOD, especially one can see palps (leg-like structures attached to the front) and joints of legs are quite visible; QSlim is devoid of the potential to keep these parts. Although, MELOD scales off abdomen a little bit but in spite of this overall appearance is very close to the original model.

Close-up of head and front legs of horse model has been shown in Figure 9; it is apparent that MELOD keeps parts of high semantic meaning like ears, nostrils and hoofs after drastic decimation of 98.35%. Also, note Figure 8, it can be seen that high semantic importance features like eyes, nose, lips and ears remain on male model decimated by MELOD even after 99.71% reduction , whereas some of these have been completely removed or blurred by QSlim. Close examination of male model, and head and front legs of horse model reveals that MELOD spends small and more triangles to preserve high level perceptually important portions of a model, which usually are high curvature regions, and play crucial rule in the semantic meaning of a model.

Buddha model shown in Figure 10 consists of more than one million triangular faces; models in the middle and to the left of the figure are decimated versions simplified by MELOD and QSlim respectively and each consists of 3266 triangular faces. One can see the version generated by MELOD gives the feel of a happy Buddha in spite of 99.69% reduction in size.

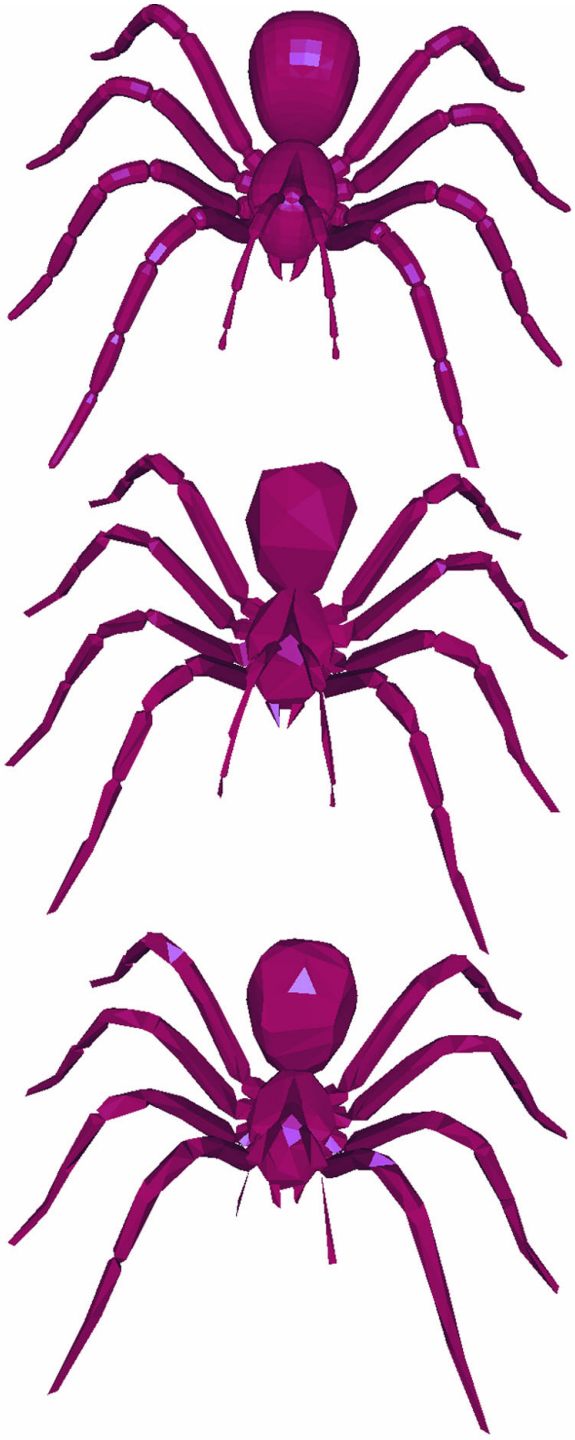

**Figure 6. Spider model: original (top) #faces: 9286, decimated by MELOD( middle) and QSlim(bottom): #faces: 1144 (each).**

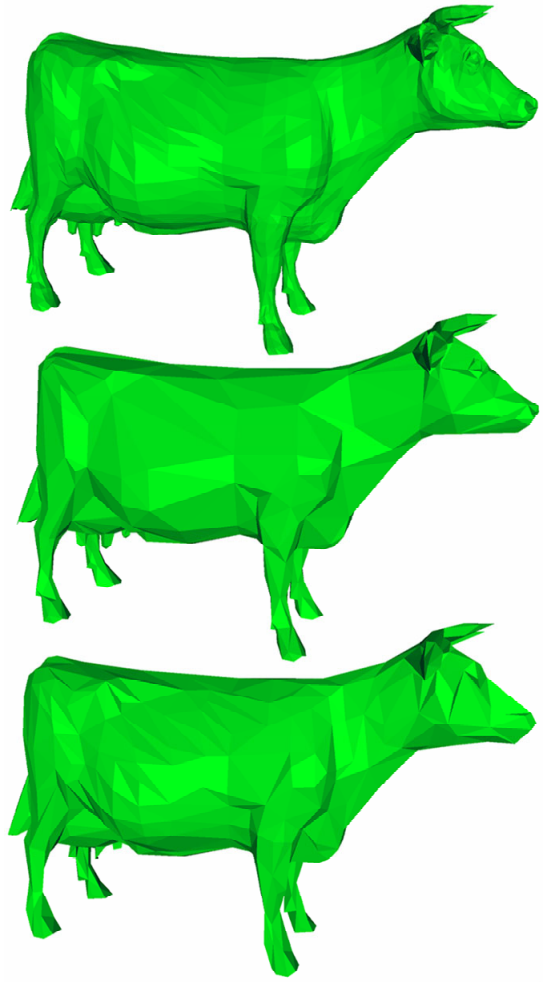

**Figure 7. Cow model: original (top) #faces: 5804, decimated by MELOD( middle) and QSlim(bottom): #faces: 1198 (each).**

## 6. CONCLUSIONS

We have presented a new method for generating LODs of triangular meshes. Our algorithm is not only faster than many existing algorithms in terms of execution times and has low memory overhead as compared to most of the notable decimation algorithms but also preserves automatically the essential parts of a mesh and its visually important features, and keeps its semantic meaning. Numerically the simplification results of MELOD are comparable with those of QSlim and MS. This can be employed for applications which require visual fidelity and the semantic meaning of the model to be preserved at very low levels of resolution, not tight error bound, and the set of vertices of the simplified version to be a proper subset of original vertices. The relation of the proposed measure of geometric error with distance metric is not clear; it is not obvious that how this can be extended to include surface attributes. These are future directions for investigation.


## ACKNOWLWGEMENTS

We are thankful to Peter Lindstrom for providing the metro results for different LODs of horse model generated by his Memoryless Simplification

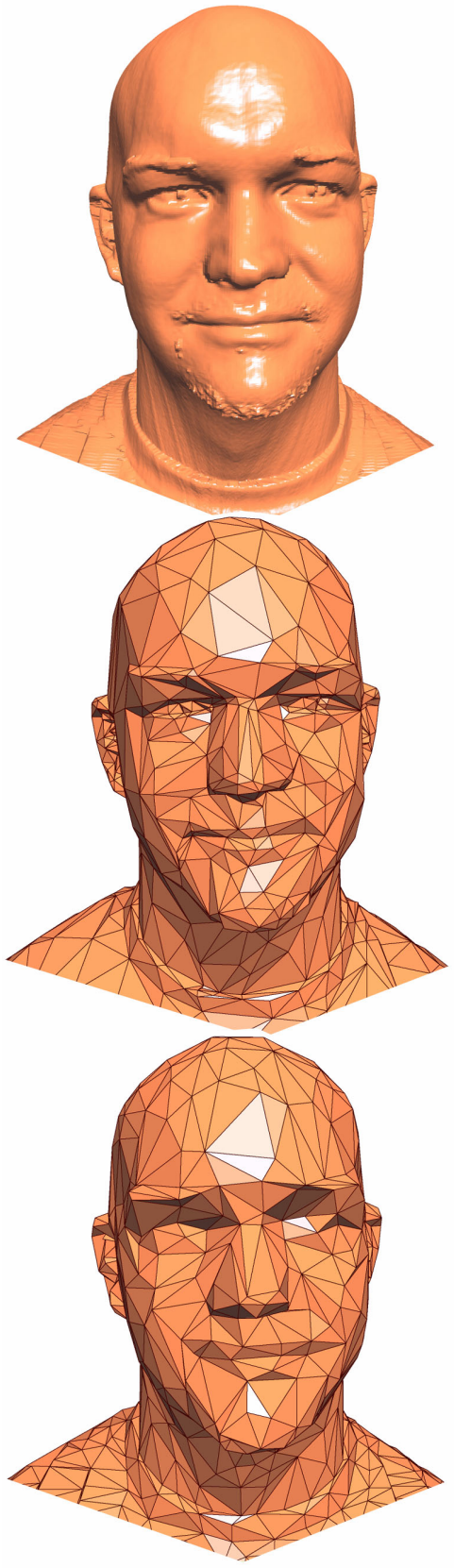

**Figure 8. Male model original (top) # faces: 605,902; versions decimated by MELOD (middle) and QSlim (bottom) #faces: 1709 (each).**

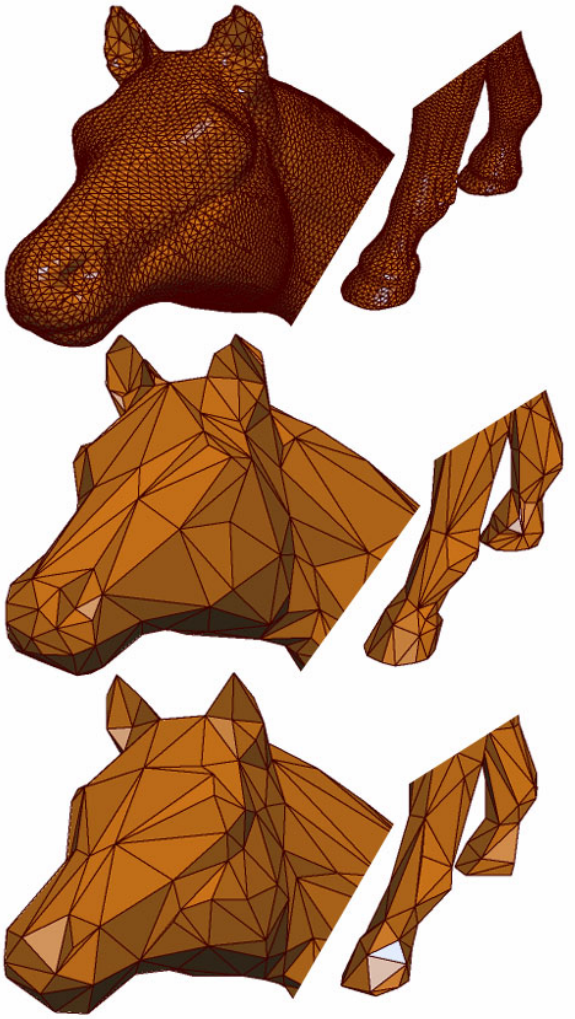

**Figure 9. Cose-up of head and front feet of horse model original (top) # faces: 96,966; versions decimated by MELOD (middle) and QSlim (bottom) #faces: 1596 (each).**

algorithm. We also acknowledge the valuable comments of the anonymous reviewers for the improvement of the paper.

## 7. REFERENCES

[Baj96] Bajaj, C. L and Schikore, D. R. Error bounded reduction of triangle meshes with multivariate data. SPIE, 2656:34-45, 1996.

[Bro00] Brodsky, D., and Watson, B., Model simplification through refinement. In Proc. Graphics Interface'00, pages 221-228, 2000.

[Cia96] Ciampalini, A., Cignoni, P., Montani, C., and Scopigno, R. Multiresolution decimation based on global error. The Visual Computer, 13:228-246, 1997.

[Cig98a] Cignoni, P., Rocchini, C., and Scopigno, R. Metro: Measuring error on simplified surfaces. Computer Graphics Forum, 17(2):167-174, June 1998.

[Cig98b] Cignoni, P., Montani, C., and Scopigno, R. A comparison of mesh simplification algorithms. Computer & Graphics, 22(1):37-54, 1998.

[Gar97] Garland, M., and Heckbert, P.S. Surface simplification using quadric error metric. In Proc. SIGGRAPH'97, pages 209-216, August 1997.

[Hec97] Heckbert, P. S., and Garland, M. Survey of surface simplification algorithms. Technical report, Carnegie Mellon University-Dept. of Computer Science, 1997.

[Hop96] Hoppe, H. Progressive meshes. In Proc. SIGGRAPH'96, pages 99-108, August 1996.

[Hus03a] Hussain, M., Okada, Y. and Niijima, K. Fast, simple, feature-preserving and memory efficient simplification of triangle meshes. International Journal of Image and Graphics, 3(4):1-18, 2003.

[Hus03b] Hussain, M., Okada, Y. and Niijima, K. LOD modeling of polygonal models based on multiple choice optimization. In proc. MMM04, to appear.

[Kim02] Kim, S. J., Kim, C. H., and Levin, D. Surface simplification using a discrete curvature norm. Computers and Graphics, 26:657-663, 2002.

[Kob98] Kobbelt, L., Campagna, S., and Seidel, H.P. A general framework for mesh decimation. In Proc. Graphics Interface'98, pages 311-318, October 1998.

[Lue97] Luebke, D. A survey of polygonal simplification algorithms, Technical Report TR97-045, Department of Computer Science, University of North Carolina, 1997.

[Pet98] Lindstrom, P., and Turk, G. Fast and memory efficient polygonal Simplification. In Proc. IEEE Visualization'98, pages 279-286, 544 Oct. 1998.

[Ron96] Ronfard, R., and Rossignac, J. Full range approximation of triangular polyhedra. Computer Graphics Forum (Proc. Eurographics'96), 15(3), 1996.

[Ros93] Rossignac, J., and Borrel, P. Multi-resolution 3D approximation for rendering complex scenes. In Geometric Modeling in Computer Graphics, pages 455-465, Springer Verlag, 1993.

[Sch92] Schroeder, J., Jonathan A. Zarge, and William E. Lorenson. Decimation of triangle meshes. Computer Graphics (Proc. SIGGRAPH'92), 26(2):65-70, July 1992.

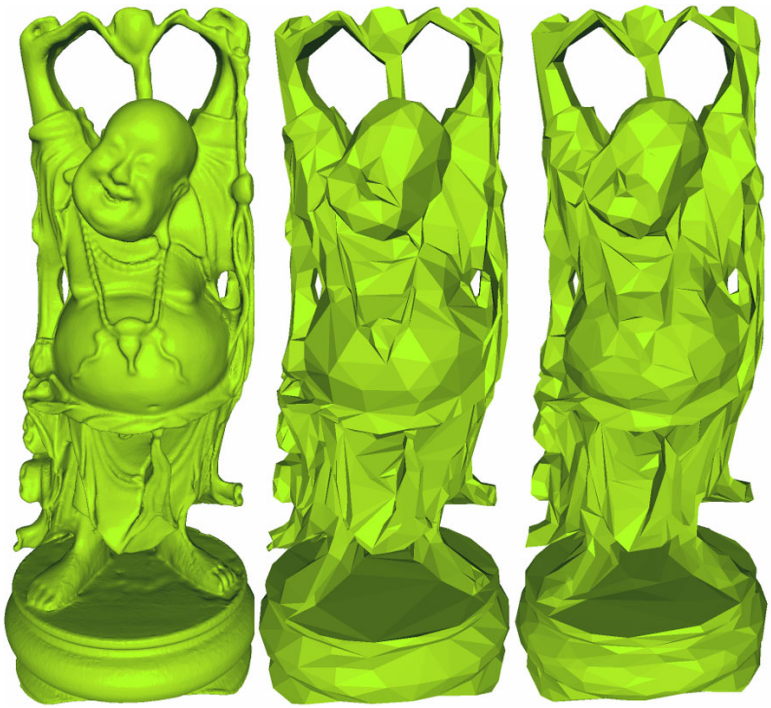

**Figure 10. Buddha model original (left) # faces:1085, 634; versions decimated by MELOD (middle) and QSlim (right) #faces: 3266.**