LOD modelling of polygonal models

Article · January 2005

CITATIONS
10

READS
19

2 authors, including:

Muhammad Hussain
King Saud University
110 PUBLICATIONS **464** CITATIONS

SEE PROFILE

LOD MODELLING OF POLYGONAL MODELS

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

ABSTRACT
For the decimation of polygonal models and generating their LODs (Levels of detail), an automatic edge-collapse based simplification method has been proposed. The measure of geometric fidelity employed is motivated by the normal space deviation of a polygonal model occurred during its decimation process and forces the algorithm to minimize the normal space deviation. Inspite of being global nature of the evaluation of geometric deviation, the algorithm is memory efficient and involves less execution time as compared to the state-of-the art algorithms on simplification. It prevents automatically the creation of folds and preserves automatically the visually important features of a model even at low levels of detail. LODs generated by our method compare favorably with those produced by the standard QEM based algorithm QSlim [7] in terms of mean and maximum geometric error whereas its performance in preserving normal space of the original model is better than that of QSlim.
KEY WORDS
Triangular meshes, Surface simplification, Level of detail, Edge-collapse, Shape approximation, Multiresolution modeling

1 Introduction

Polygonal models are currently being exploited to render 3D surfaces and serve as the de facto standard for fast interactive visualization applications. Pursuit of realism on one hand and the recent advances in scanning devices and modeling systems on the other hand have given rise to huge and highly complex models of sheer sizes that go over the sustainable graphics performance of the current mid-level graphics systems and make them far less suitable for interactive handling in applications like virtual reality and scientific visualization.

The solution of this problem lies in generating LOD representation of a model, where a model is made available at different levels of accuracy and complexity, and an appropriate level of resolution is utilized thereby trading quality for rendering efficiency. Constant frame rate, for example, can be retained during the navigation through a virtual environment by employing an LOD of a visible object adapted to the complexity of the visible part of the scene and the performance of the graphics hardware being used; a detailed polygonal model is used when the object is close to the viewpoint, and coarser approximations are substituted as the object recedes.

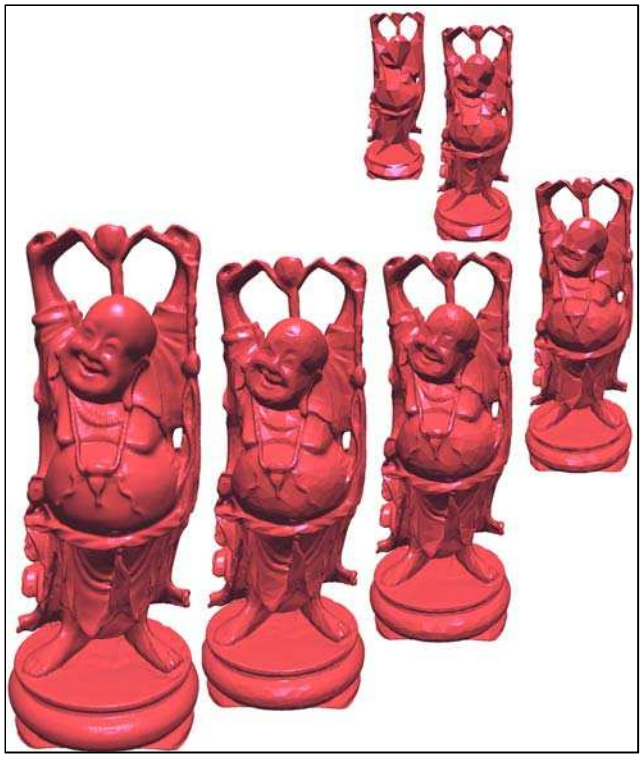

Figure 1. Buddah model at different levels of detail. Original version and the reduced versions consisting of 1085634, 24000, 12000, 5000, 3000, 1000 triangular faces respectively.

Published iterative algorithms for generating LODs are either based on $global\ evaluation$ or $local\ evaluation$ of geometric deviation. Methods [19, 22, 12] driven by local evaluation are usually fast and memory efficient but result in poor approximations; on the other hand methods [11, 6, 3] exploiting global evaluation produce approximations of good visual fidelity but suffer from high memory overhead because of retaining geometric history during decimation process and often involve very long running times. We present a new technique for decimation of polygonal models and generating LODS based on half-edge collapse transformations and a memory-efficient and faster

global evaluation of geometric deviation. The measure of geometric deviation caused by a half-edge collapse transformation is motivated by the requirement that the normal space deviation of a polygonal model is minimized during the decimation process, which in turn adorns it with the potential of keeping the features of high perceptual importance even at extremely low levels of detail and preventing automatically the occurrence of folds. Comparison reveals that resulted algorithm is as fast as QSlim [7] in execution times and, consume less memory like memoryless simplification and performs better to keep high level meaning of a model at extremely low levels of detail. The key features of the presented algorithm are:

- Drastic reduction in memory overhead
- Fast execution time
- Automatic preservation of features of high semantic importance at extremely low levels of resolution
- Automatic prevention of self-intersections
- Ease of implementation

The remainder of this paper is organized as follows. In Section 2, we give an overview of the related work. Section 3 presents the details of our geometric error measure. Outline of the algorithm has been presented in Section 4. In Section 5, we discuss the different aspects of our algorithm and compare it with some state-of-the-art published algorithms. Section 6 concludes the paper.

2 Related Work

The problem of automatic polygonal decimation or LOD modelling has been intensively studied during the last decade and many iterative algorithms have been proposed with different weaknesses and strengths, and can be broadly categorized into three classes according to topological operator they employ: $vertex\ removal$, $triangle\ collapse$, and $edge\ collapse$. The algorithms proposed in [1, 3, 6] are based on vertex removal and the one presented in [8] employs triangle collapse. The algorithms described in [7, 11, 12, 16, 18] are driven by edge collapse. Edge collapse based algorithms have gained popularity because of their ease in implementation, computational efficiency and better approximation results, so in our ongoing discussion we would mainly focus on this kind of algorithms. For thorough survey of decimation methods, an interested reader is referred to consult [5, 9, 17].

Many edge collapse algorithms have been published, we would mention only a few state-of-the-art algorithms to establish a ground for comparison. Hoppe [11] is the first researcher who employed edge collapse transformation for constructing progressive meshes; his algorithm is based on global optimization framework, while this algorithm produces high quality approximations, it involves very high

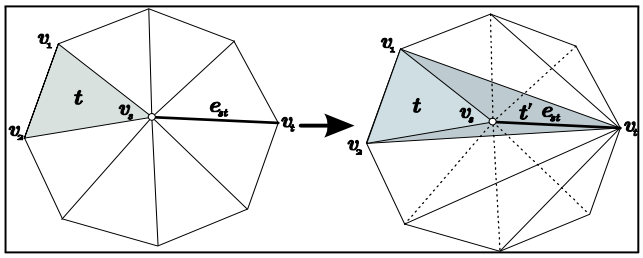

Figure 2. Collapse of edge e_{st} will map the triangle $t = \{v_s, v_1, v_2\}$ onto triangle $t' = \{v_t, v_1, v_2\}$.

memory overhead and is very slow in running times because it needs to store geometric history and has to perform several distance-to-surface measurements. QSlim [7] is currently one of the most popular algorithms; it computes the square of the distance of a vertex from each of the incident planes and stores it as a 4×4 symmetric matrix following the idea presented in [20]; it exploits this quadric error metric (QEM) to put the edges in priority ordering for collapse and to determine the optimal position of the substituent vertex. This algorithm produces fairly good LODs in a short time, but it is not memory efficient and often can not preserve features of high visual importance at extremely low levels of detail. The idea of measuring geometric fidelity proposed in [18] is a memory efficient version of QEM, which is based on the conservation of volume; it produces good quality simplifications and is fairly efficient, particularly in memory consumption, but not in execution times, it is about five times slower than QSlim.

The decimation algorithm proposed by Brodsky and Watson [2] is based on refinement; it is even faster than QEM algorithm but produces poor approximations. The algorithm proposed by Kim et al [14] employs edge collapse operator and 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, it seems this algorithm involves very long running times because of the computation of their proposed measure of discrete curvature norm. Algorithm proposed by Hussain et al [12, 13] is also fast and have low memory overhead, but the idea of accumulation of geometric error leads to error overestimation.

3 Evaluation of Geometric Deviation

A polygonal model can be converted conveniently into a triangular mesh as a preprocessing, so we concentrate on triangular models only in our onward discussion.

First of all, to fix the ideas a brief description of terminology and notation is in place. We represent a vertex of a model M by v with its geometric counterpart as a $3D$ vector $\mathbf{v}$; an edge e_{st} is the set $\{v_s, v_t\}$and is equivalent

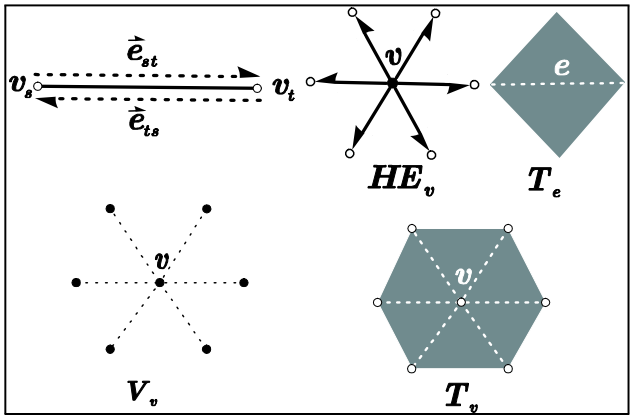

Figure 3. An edge $e_{st} = \{v_s, v_t\}$ and the corresponding half-edges $\vec{e}_{st}$ and $\vec{e}_{ts}$(top left). The remaining diagrams are self-explanatory.

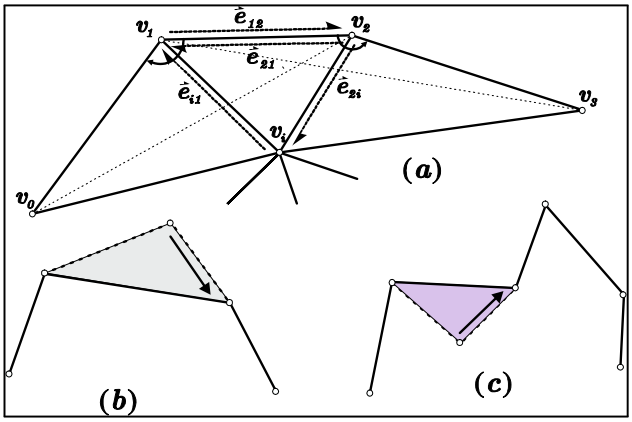

Figure 4. (a) v_i is an interior vertex and v_0, v_1, v_2 and v_3 are boundary vertices. (b) Collapse of the boundary edge (indicated by arrow) will reduce the surface along the boundary by the amount of the area of the shaded triangle. (c) Collapse of the boundary edge (indicated by arrow) will introduce the surface along the boundary by the amount of the area of the shaded triangle. We use this shaded area to penalize the cost of the edges along the boundary.

to two half-edges represented by ordered pairs (v_s, v_t) & (v_t, v_s) denoted by $\vec{e_{st}}$ & $\vec{e_{ts}}$ respectively; v_s & v_t are said to be $origin$ and $head$ of the half-edge $\vec{e}_{st}$ respectively. A triangle t is a set of half-edges *i.e.* $t = \{\vec{e_{01}}, \vec{e_{12}}, \vec{e_{20}}\}$ or $t = (v_0, v_1, v_2)$. The set of half-edges having v as their origin is represented by HE_v; similarly T_e represents the triangles incident on the edge e, T_v stands for the set of triangles that meet at the vertex v,and V_v is the set of 1-ring neighbor vertices of v, see Figure 3

The proposed decimation method is driven by half-edge collapse transformation (see Figure 2) and a global evaluation of geometric fidelity. We prefer half-edge collapse transformation because of its simplicity in implementation, local control, and its suitability for efficient progressive transmission, for inducing nested hierarchies [15] on unstructured meshes that can facilitate further applications and for generating smooth transitions between levels of detail. It is not only a special case of $edge\ collapse$ but $vertex\ removal$ as well.

Now we introduce the measure of geometric fidelity that forces to minimize the normal space deviation of a triangular model during decimation process. Consider a typical half-edge collapse transformation $\vec{e}_{st}\ (v_s, v_t) \mapsto v_t$, it eliminates the edge e_{st}, and the triangles $T_{e_{st}}$ incident on the edge e_{st} become degenerate, so they are discarded. Each of the remaining triangles $T_{v_s} - T_{e_{st}}$ incident on v_s undergoes a transformation, note Figure 2, and introduce local geometric error in the model. We measure this geometric error so that the decimation of the model have minimum effect on the normal space of the model. To accomplish this, we measure the deviation of the normal of the each triangle $t \in T_{v_s} - T_{e_{st}}$ with respect to its original normal and scale it with the current area of t; note that this measure is global in the sense that the deviation of the current normal of t is calculated relative to its original normal. So, if θ_t is the deviation of the normal of the triangle t relative to its original normal and Δ_t is its current area, then geometric error introduced by this triangle can be expressed as

$$gdev(t) = \Delta_t \theta_t \tag{1}$$

The normal deviation θ_t can be computed as the difference of unit normals or as the angle between the unit normals, but we approximate θ_t by $1 - \hat{n}_t \cdot \hat{n}_{ot}$ where $\hat{n}_t$ and $\hat{n}_{ot}$ are the current and the original unit normals of the triangles t. Since our purpose is to order the half-edge collapses, so this approximation exactly serves our purpose and it yields the same results as if we use any of the other two measures for θ_t, but it leads to the computationally efficient expression for $gdev(t)$ as will be clear in the sequel.

$$gdev(t) = \Delta_t (1 - \hat{n}_t \cdot \hat{n}_{ot}) \tag{2}$$

Note that a common factor does not matter, so we assume $\Delta_t = \|\vec{n}_t\|$, where $\|.\|$ stands for Euclidean norm and $\vec{n}_t$ is the current normal (not unit normal) to the triangle t. In view of this assumption the above expression for geometric error takes this from

$$gdev(t) = \Delta_t - \vec{n}_t \cdot \hat{n}_{ot} \tag{3}$$

This expression involves just three operations.

The cost of the half-edge $\vec{e}_{st}$ is the sum of geometric errors contributed by the each triangle $t \in T_{v_s} - T_{e_{st}}$ i.e.

$$Cost(\vec{e}_{st}) = \sum_{t \in T_{v_s} - T_{e_{st}}} gdev(t). \tag{4}$$

Note that the quantity given by (1) can also be interpreted as the change in curvature . From (2), it is obvious that this quantity is zero for flat regions and grows gradually for higher curvature regions. It implies that it helps

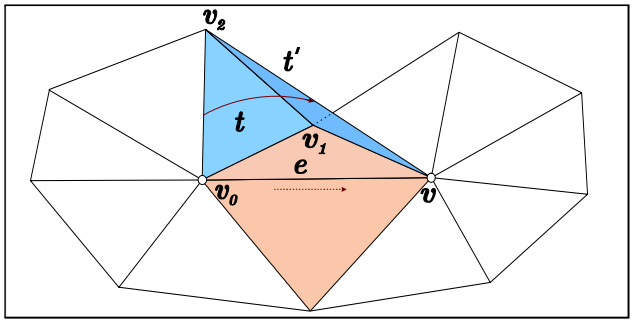

Figure 5. In case of the collapse of edge e, the triangle t will turn through 180° to t' and thus will create a fold.

remove excessive detail on flat regions and keep the necessary detail on high curvature regions of a mesh. Thus it causes to keep the semantic meaning of a mesh. From (3) it is clear that it just converts into an area preserving metric for flat regions, and it can be combined with any triangle aspect ratio measure for creating well-shaped triangles on flat regions; one such triangle aspect ratio preserving measure is

$$ar(t) = \frac{4\sqrt{3}\Delta}{l_1^2 + l_2^2 + l_3^2},$$

where Δ is the area of the triangle t, and l_1, l_2, l_3 are lengths of its sides. It follows from above discussion that the prosed measure also preserves area and curvature.

When the collapse of an edge creates a fold, θ_t in (1) attains its maximum value and causes to assign maximum value to the cost of collapse and thus prevents this edge collapse. Also, in case of a fold, the area of 1-ring neighborhood of the eliminated vertex will not be preserved, so the area preserving nature of the proposed cost of collapse will discourage this collapse, see Figure 5.

The proposed local measure of geometric deviation would associate normally less cost with edges on the boundary and there is a strong tendency that the algorithm would get trapped in a local minimum along the boundary and would cause the surface to shrink out of existence. As a remedy to this problem, we categorize boundary half-edges into three main types: (1) half-edges having origin on boundary, e.g. $\vec{e_{2i}} = (v_2, v_i)$ in Figure 4, (2) half-edges having head on boundary,e.g. $\vec{e}_{i1} = (v_i, v_1)$ in Figure 4, and (3) half-edges having both origin and head on boundary, e.g. $\vec{e}_{12} = (v_1, v_2)$ in Figure 4.

Because the half-edge collapse transformation $\vec{e}_{st}$ $(v_s, v_t) \mapsto v_t$ substitutes the half-edge with v_t, so collapse of the half-edge having $head$ on the boundary does not need special treatment. However if $origin$ is on the boundary, then collapse of the half-edge will deform the boundary severely; so the collapse of such half-edge is not allowed.

The half-edge having both origin and head on the boundary must be dealt tactfully. Since the expression of the geometric error proposed above have the dimensions of area, so to keep the expression dimensionally consistent, we penalize the cost of such edges with the area of the triangle swept out by the edge collapse, see Figure 4. Ultimately, the cost of collapse of the half edge $\vec{e_{12}} = (v_1, v_2)$ having $origin$ and $head$ on boundary is

$$Cost(\vec{e}_{12}) = \lambda\Delta + (1-\lambda)\sum_t gdev(t) \qquad (5)$$

Here Δ is the area of the triangle swept out by the edge collapse (the shaded regions in Figures 4(b, c)), and $0 \leq \lambda \leq 1$ is a user specified parameter used to control the quality of boundary preservation, experimentally, we found that $\lambda = .5$ gives pleasing results.

4 Outline of the Algorithm

The measure of geometric deviation elaborated in Section 3 can be implemented using well-known greedy framework as well as the multiple choice optimization framework [24]. We have decided to employ the greedy framework because publicly available standard algorithm QSlim also uses greedy framework and it will help to compare the effects of our error measure on the same grounds without any bias.

The input is the original model M consisting of n vertices and r ($\approx 2n$) triangular faces and the output is the original model along with ordered list of n half-edge collapse transformations and their associated cost values. This progressive mesh (PM) [11] representation constitutes an entire continuum of LODs of the model and an LOD approximation of desired complexity can be extracted from this PM.

Implementation of the algorithm needs two very simple data structures: $Face$ and $Vertex$. The $Face$ contains pointers to three vertices that make up the face and holds original unit normal vector of the face. The $Vertex$ contains a list of pointers to incident faces T_v, a pointer to a vertex that implicitly defines the optimal half-edge and a float value that is the cost of this half-edge. It is to be noted that for each vertex v, we traverse the set HE_v, the half-edge with minimum cost is termed as optimal half-edge.

The algorithm based on greedy approach executes the following steps.

1. For each vertex v_i of M, exploiting spatial coherence traverse each half-edge $\vec{e}_{ij} \in HE_{v_i}$ computing its cost c_{ij} using the criterion described in Section 3, and select the optimal half-edge $\vec{e}_{im}$ with minimum cost $c_{im} = \min\{c_{ij} \mid e_{ij} \in HE_{v_i}\}$ and put it in priority queue and discard the rest of the half-edges $HE_{v_i} - \{e_{im}\}$.

2. Take out of the priority queue the half-edge $\vec{e_{st}} = (v_s, v_t)$ having minimum cost. Collapse edge e_{st} by eliminating triangles $T_{e_{st}}$ and by replacing all occurrences of v_s with v_t in the remaining edges $HE_{v_s} - \{\vec{e}_{st}\}$ and triangles $T_{v_s} - T_{e_{st}}$ incident upon v_s.

Model	Model Size	FMLOD	QSlim
Fandisk	12,946	.33	0.35
Bunny	69,451	1.41	1.45
Horse	96,966	1.93	1.90
Male	605,902	14.11	15.43
Buddha	1085,634	24.85	25.42

Table 1. Time taken in seconds to reduce to one face. Here model size is in the nubmer of triangular faces.

3. For each of the vertices $v_j \in V_{v_s}$, update optimal half-edge $\vec{e}_{jm}$ traversing all half-edges HE_{v_j} and put it in the priority queue.

4. Repeat steps 2 through 3 until the priority queue is empty.

It is interesting to note that we do not define an explicit data structure for storing half-edges, implicit definition and selection of optimal half-edge is accomplished by assigning its cost and the pointer to its $head$ to the corresponding fields in the data structure of its $origin$ that represents it in the priority queue. In this way, the priority queue is in fact vertex based and it needs only $8n$ Bytes (for each vertex , 4 Bytes for priority value and 4 Bytes for vertex pointer) instead of $48n$ Bytes (if explicit data structure for half-edges is used then there are $6n$ half-edges because in a model M consisting of n vertices there are roughly $2n$ faces and each face contains 3 half-edges).

5 Results and Discussion

FMLOD, the implementation of our algorithm, has performed well on a wide range of public domain triangular meshes. We discuss here the performance of FMLOD in terms of memory consumption, execution times and the quality of generated approximations. To validate the efficiency and strength of FMLOD, we make comparison with the simplification algorithm based on quadric error metric [7]; the reason for choosing this method for comparison is that it is considered as a standard method for LOD modeling and simplification of polygonal models and its quality and efficiency is reinforced after theoretical analysis by Heckbert and Garland [10] and its implementation QSlim is available in public domain.

Table 1 lists the execution times spent by QSlim and FMLOD to simplify models of varying complexities to one face on 2.9GHz Intel Pentium(R) 4 machine with 512 MB of main memory; it is obvious that in terms of running times, FMLOD competes well with QSlim.

The only geometric history that is retained by FMLOD is the original unit normal of each triangle; unless a triangle is not eliminated its current normal continue to change so we do not store it, compute it online and instead we store only its original normal i.e. FMLOD consumes

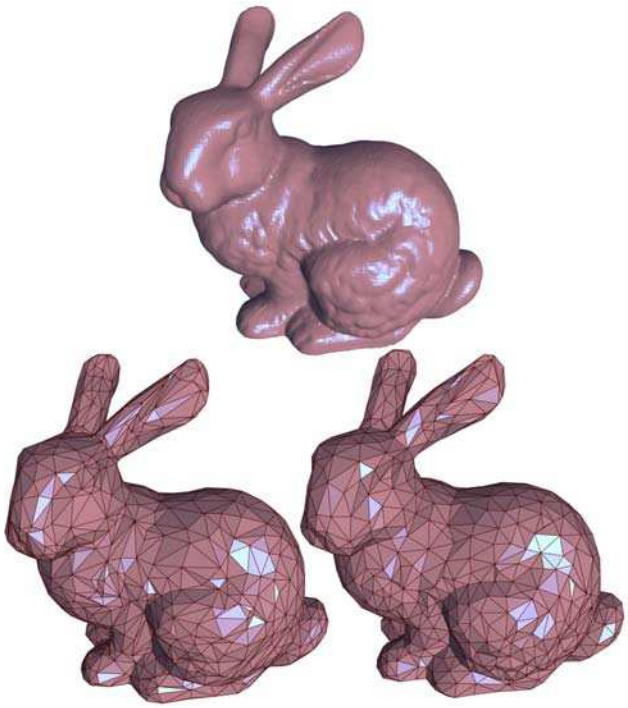

Figure 6. Original bunny model (top) consists of 69,451 faces. Simplified version (#F 2000) generated by FMLOD (bottom left) and QSlim (bottom right).

12bytes per triangle to store triangle normal, which is normally consumed by almost every simplification algorithm to store the normal of triangles. But QSlim needs memory for storing 10 entries of the QEM symmetric matrix in addition to the storage for the mesh itself and priority queue, and so it suffers from an additional memory overhead of at least $40n$ Bytes for a mesh M consisting of n vertices.

So, our algorithm is memory efficient like memoryless simplification [18] and is computationally efficient like QSlim; note that memoryless simplification is at least five times slower than QSlim [18].

For thorough numerical comparison, we employ mean and maximum geometric error measures, and mean and maximum normal deviation measures. We compute mean and maximum geometric errors using version 2.5 of well-known I.E.I-CNR Metro Tool [4] and measure mean and maximum normal deviations using MeshDev v0.4 tool [21]; both are available in the public domain. We present here the results for bunny and male models because of their complex structures. Graphs shown in Figures 9 through 16 depict the mean and maximum geometric errors and normal deviations for different levels of detail of bunny and male models generated by FMLOD and QSlim. It is apparent that FMLOD compares well with QSlim in terms of mean geometric error and normal space deviation whereas it outperforms in terms of maximum geometric error and normal space deviation..

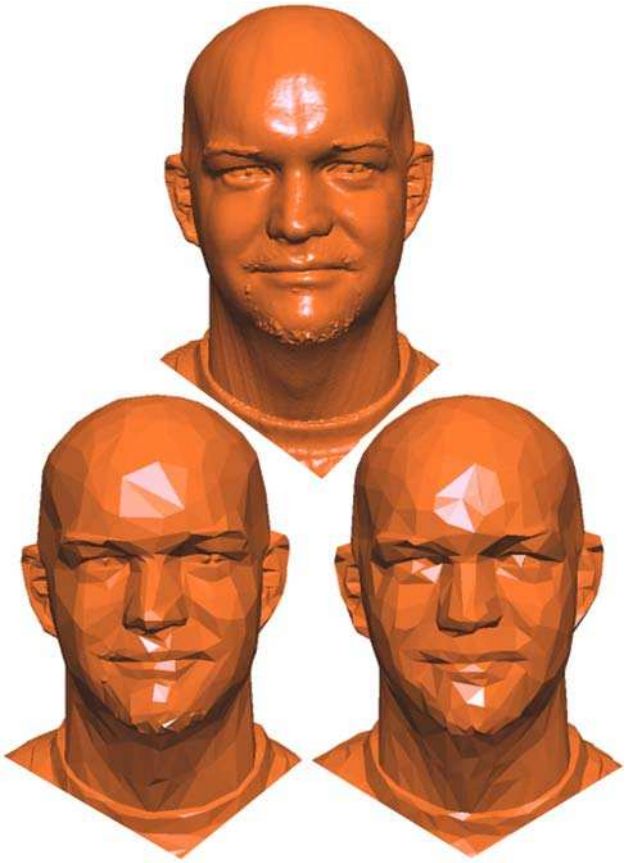

Figure 7. Original male model (top) consists of 605,902 faces. An LOD (#F 3000) produced by FMLOD (bottom left) and QSlim (bottom right).

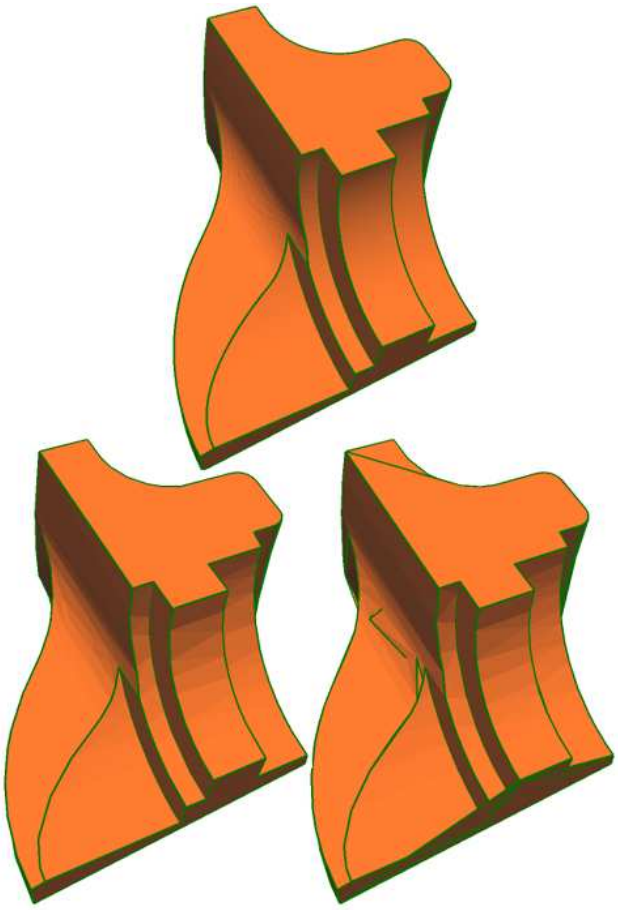

Figure 8. Original fandisk model (top) consists of 12,946 faces. An LOD (#F400) generated by FMLOD (bottom left) and QSlim (bottom right).

For visual comparison, observe models shown in Figures 6, 7, 8, 17 and 18 approximated by FMLOD and QSlim. It can be seen that major features of the original models still remain in models simplified by FMLOD in spite of being highly simplified and these models not only compare with those produced by QSlim but in some cases show better results. One important characteristic of any simplification algorithm is whether it preserves sharp edges or discontinuity curves present in a polygonal model. FMLOD automatically preserves sharp edges and lines of discontinuity as can be seen in Figure 8; simplified version of fandisk model consists of 400 triangles, 3% of the original; feature lines are preserved albeit highly reduced. Folds may appear when an edge to be collapsed is surrounded by a very concave polygon; FMLOD have the potential to prevent the occurrence of self-intersections automatically whereas QSlim needs special heuristics to deal with the problem of folds; again note Figure 8 to this effect, folds are visible on the fandisk model simplified by QSlim.In some cases certain features of a model are very important and form the basis of human perception of the definition of a model; these features are usually high curvature regions and it is desirable that these must not be removed or blurred severely. Note horse model shown in Figure 18, ears, nostrils and the cavity between eye and nostril are apparent in the model simplified by FMLOD, whereas these features have been severely blurred by QSlim. Also, observe male model shown in Figure 7, it is obvious that FMLOD keeps parts of high semantic importance like eyes, eye brows, lips even after 99.5% decimation whereas QSlim completely blurs the eyes. Figure 17 shows the effects of boundary preservation.

FMLOD can efficiently simplify very large models. Figure 1 shows Buddha model at different levels of detail; the lowest level of detail consists of 1000 faces only, 0.09% of the original version; although most of the detail has gone, but overall structure is preserved.

6 Summary and Future Work

A new method for automatically generating levels of detail of a polygonal model have been proposed; this algorithm is driven by half-edge collapse transformations and a mea-

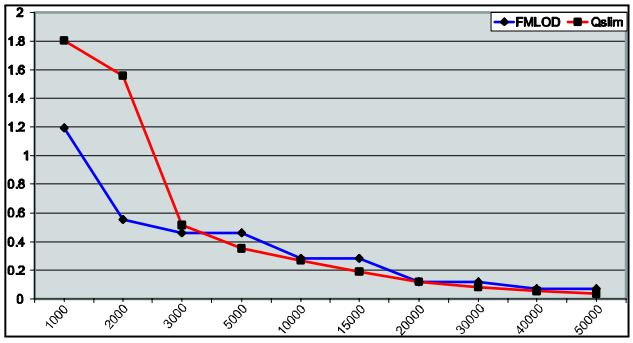

Figure 9. Maximum geometric error for bunny model.

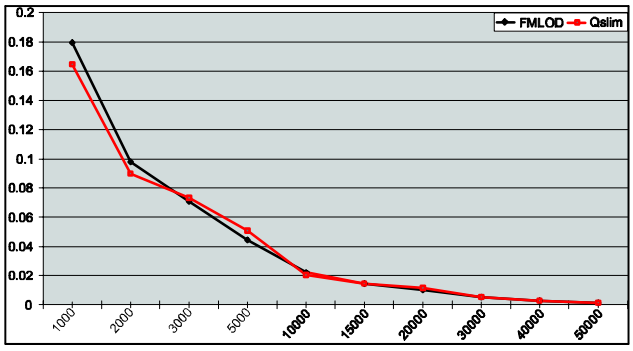

Figure 10. Mean geometric error for bunny model.

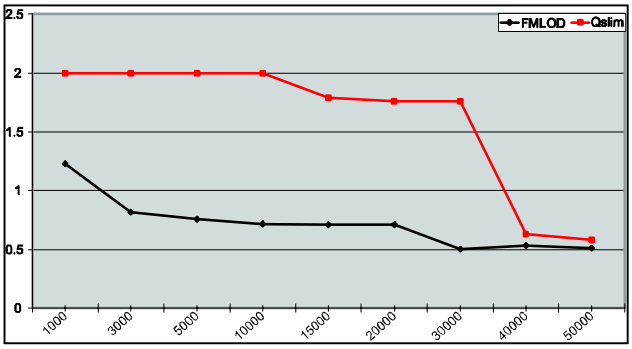

Figure 11. Max normal deviation for bunny model.

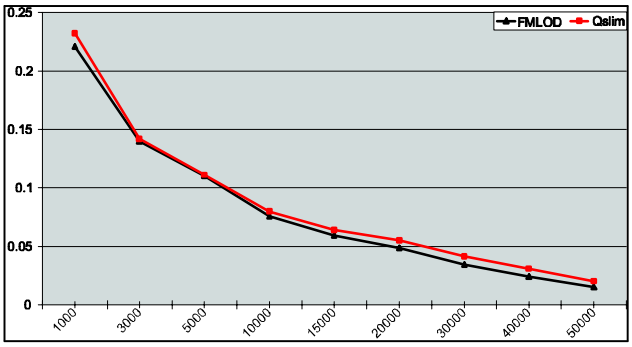

Figure 12. Mean normal deviation for bunny model.

sure of geometric deviation that forces it to minimize the normal space deviation. It competes well with the standard algorithm QSlim in terms of execution time, but performs better in memory consumption, quality of approximations and preserving features of high perceptual importance. It can be adopted for applications where visual fidelity, not tight error bound, is the prime need and the set of vertices of the simplified version to be a proper subset of the original vertices.

References

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

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

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

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

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

[6] Cohen, J., Varshney, A., Manocha, D., Turk, G., Weber, H., Agarwal, P., Brooks, F., and Wright, W. Simplification envelopes. In *Proc. SIGGRAPH'96*, pages 119-128.

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

[8] Gieng, T., S., Hamann, B., Kenneth. I. Joy, Gregory L. Schlussmann, and Isaac J. Trotts. Smooth hierarchical surface triangulations. In *Proc. IEEE Visualization'97*, pages 379-386, 1997.

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

[10] Heckbert, P., and Garland, M. Optimal Triangulation and Quadric-Based Surface Simplification. Journal of Computational Geometry: Theory and Applications, 14(1-3), pp. 49-65, November 1999.

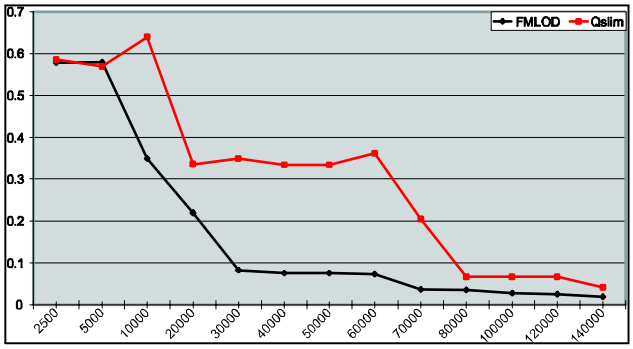

Figure 13. Maximum geometric error for male model.

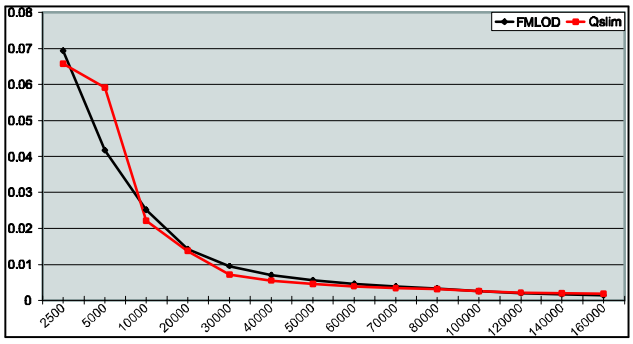

Figure 14. Mean geometric error for male model.

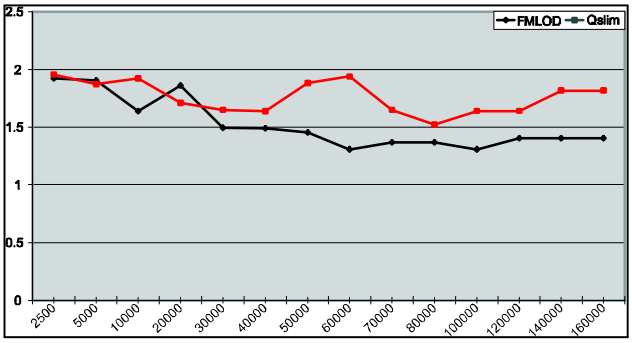

Figure 15. Maximum normal deviation for male model.

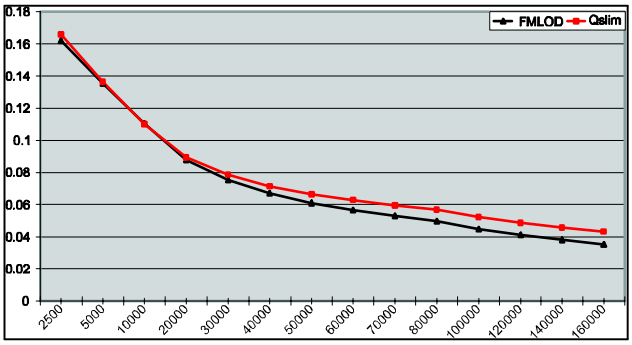

Figure 16. Mean normal error for male model.

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

[12] 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.

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

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

[15] Kobbelt, L., Campagna, S., Vorsatz, J., and Seidel, H., P. Interactive multi-resolution modeling on arbitrary meshes. *Proc. SIGGRAPH'98,* pages 105-114, 1998.

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

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

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

[19] Algori, M., E., and Schmitt, F. Mesh simplification. *Computer Graphics Forum* (*Proc. Eurographics'96*), 15(3), August 1996.

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

[21] Roy, M., Foufou, S., and Truchetet, F., Generic attribute deviation metric for assessing mesh simplification algorithm quality. In Proc. IEEE International Conference on Image Processing, pp. 817-820, September 2002, Rochester, USA. http://meshdev.sourceforge.net/.

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

[23] Veron, P.,and Leon, J., C. Shape preserving polyhedral simplification with bounded error. Computers & Graphics, 22(5):565-585, 1998.

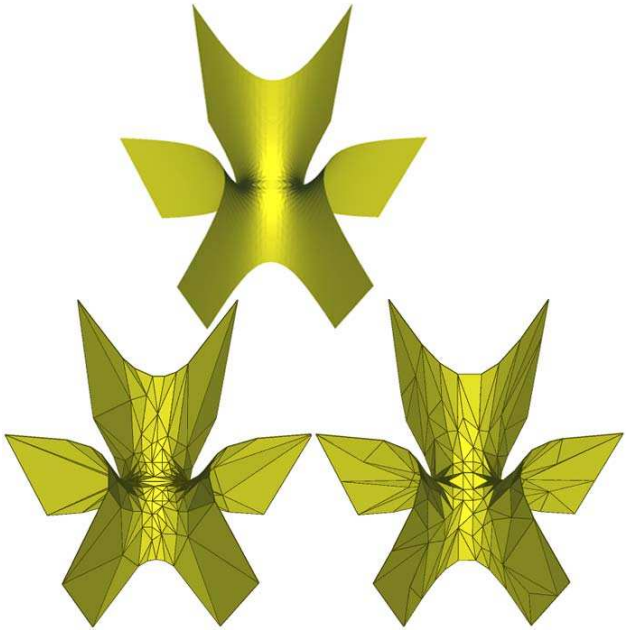

Figure 17. Original monkey model (top) consists of 19,829 faces. An LOD (#F 232) generated by FMLOD (bottom left) and QSlim (bottom right).

[24] Wu, J., Kobbelt, L. Fast mesh decimation by multiple-choice techniques. In *Proc. Vision, Modeling, and Visualization 2002*, Erlangen, Germany, November 2002.

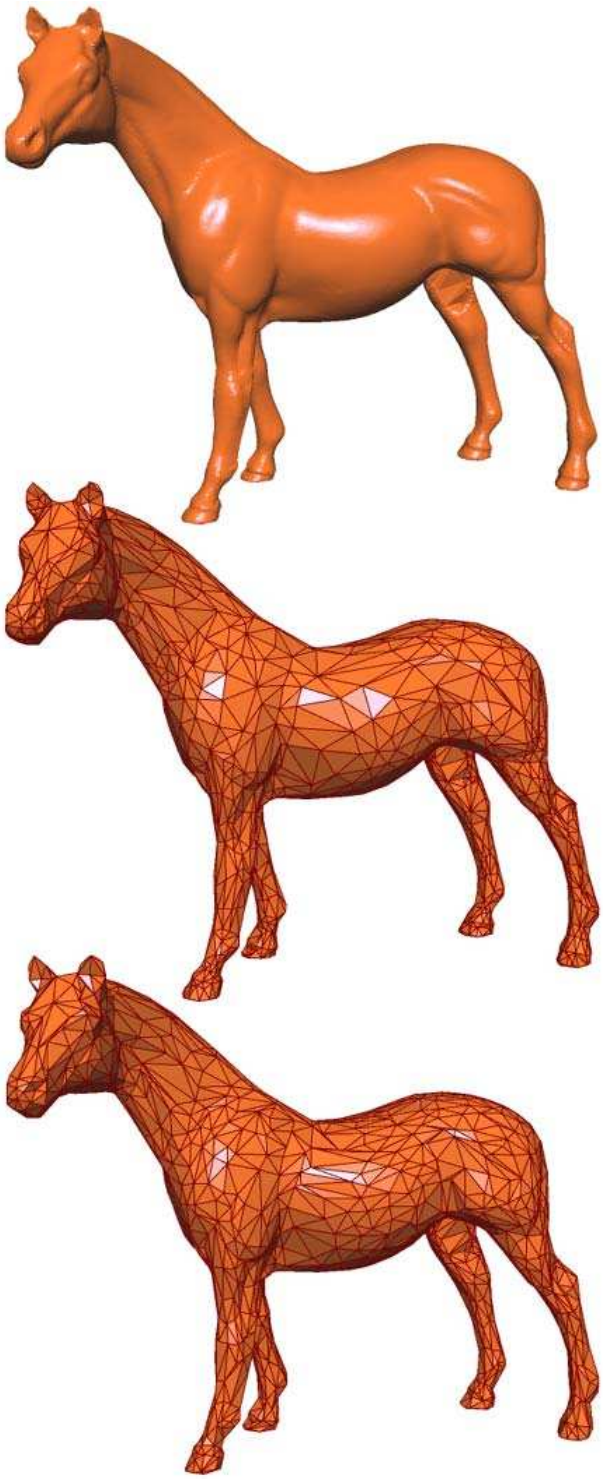

Figure 18. Original horse model (top) consists of 96,966 faces. An LOD (#F 3000) produced by FMLOD (middle) and QSlim (bottom).