Feature Sensitive Surface Simplification

Georgios Stylianou
Department of Computer Science
Cyprus College
gstylianou@cycollege.ac.cy

Abstract

We propose a simplification algorithm that preserves the surface's shape features. In a pre-processing step, crest lines are extracted and used together with a quadric error metric to drive the simplification. The error metric is utilized locally to ensure that every mesh's region is simplified in a similar rate. This algorithm attempts to minimize the creation of tiny or very large faces. A comparison with Garland and Heckbert's simplification algorithm is presented. The experimental results show a substantial improvement of the simplified surface's quality both visually and numerically.

Keywords: simplification, crest lines, quadric error metric.

1 Introduction

Surface simplification is very important for a number of applications including scientific visualization, game programming, fast network transmission and visualization. Surface simplification techniques can be categorized in two major classes, vertex removal and edge contraction.

Vertex removal deletes a vertex of the mesh and retriangulates the hole. This technique was first devised and implemented by Schröder [Schröeder et al. 1992]. Also, Choo [Choo et al. 1999], Bajaj [Bajaj and Schikore 1996], Soucy [Soucy and Laurendeau 1996] use this technique. On the other hand, Edge contraction (fig. 1) replaces an edge $(\mathbf{v}_i, \mathbf{v}_j)$ with a vertex $\overline{\mathbf{v}}$. After each contraction two faces and a vertex are removed from the triangulation. This technique is quite simple and more popular than vertex removal because of the absence of a re-triangulation. There are two vital considerations. The sequence of edges contraction and the position of the new vertex $\overline{\mathbf{v}}$.

The first consideration is hard and it is the one most of the work is focused on. Usually a cost metric is developed. Cost (or error) metrics measure somehow the distance of the simplified mesh from the original mesh. They are used to decide in which sequence the edges are to be contracted and which edges are eligible for contraction. The metrics developed are usually computed locally but used globally.

Garland and Heckbert [Garland and Heckbert 1997] use a quadric error metric, Gueziec [Guéziec 1999] uses a volume preservation metric, Kim et al. [Kim et al. 2002] combines a quadric error metric with tangential and curvature errors, Gao et al. [Gao et al. 2000] use a discrete curvature metric using average planes, Jeong et al. [Jeong et al. 2002] use a modified quadric error metric for simplifying Loop subdivision surfaces, Hoppe [Hoppe et al. 1993; Hoppe 1996] minimizes an energy function. Ronfard and Rossignac [Ronfard and Rossignac 1996] use the maximum distance from the planes that $\mathbf{v}_i$ or $\mathbf{v}_j$ belong.

The second consideration usually is correlated to the cost metric [Guéziec 1999; Kim et al. 2002; Garland and Heckbert 1997; Hoppe 1996]. Since every contraction increases the cost, the cost metric is used to minimize it. In effect, it is usually required to solve a linear system.

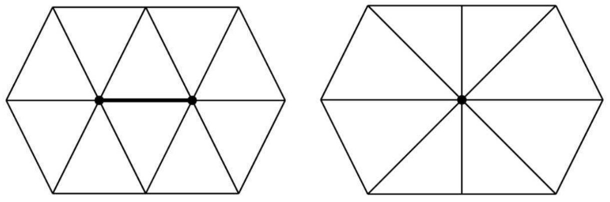
Figure 1: Edge contraction.

Cohen et al. [Cohen et al. 1998] use edge contraction with goal to maintain the surface's appearance. Furthermore, Cohen et al. [Cohen et al. 1996] focus on appearance preserving simplification with a global error guarantee and Kalvin and Taylor [Kalvin and Taylor 1996] simplify while maintaining a bounded total cost. Turk [Turk 1992] re-tiles polygonal surfaces by triangulating a new set of vertices that replaces the original one and Pojar [Pojar and Schmalstieg 2003] uses the quadric error metric for user-controlled multiresolution meshes.

Garland and Heckbert's (G&H) algorithm is the most popular among the rest because it gives as good or better results when compared with the rest and is also fast and the implementation is relatively easy. Watanabe and Belyaev [Watanabe and Belyaev 2001] attempted without much success to improve G&H algorithm by using higher weights on feature triangles and edges. Hoppe [Hoppe 1996] uses constraints to maintain feature edges identified using a threshold on the dihedral angle, G&H do not use any feature preservation constraints. In addition, G&H do not maintain a uniform simplification and tend to oversimplify in certain neighborhoods.

The contribution of this work is:

- **Quality**. Substantial improvement of the simplified surface's quality via explicit shape feature preservation.
- **Fairness and Uniformity**. The simplification is fair and uniform because we tend not to create large and tiny faces and oversimplified and under-simplified regions.
- **Boundary treatment**. We suggest to simplify surface boundaries separately from the rest of the surface.

This work is compared with G&H's simplification algorithm to show that better results are obtained. Next section, gives the background of this work, section 3 describes the various parameters of the proposed algorithm and section 4 shows the results and discusses the improvements over G&H's algorithm.

2 Background

The proposed algorithm uses a crest line extraction algorithm [Stylianou and Farin 2004] to constrain and enhance the simplification. For more information about feature extraction refer to [Stylianou and Farin 2004]. The quadric error metric is used for the simplification and the placement of new vertices. These are described in the next sections.

2.1 Crest Lines

A parametric surface is a mapping from $\Re^2$ to $\Re^3$, $\mathbf{x}(\mathbf{u}) = (x(u,v), y(u,v), z(u,v))^T \subset \Re^3$, $\mathbf{u} = (u,v) \subset \Re^2$ is the domain. Normal curvature measures the bending of the surface locally on every point $\mathbf{x}(\mathbf{u})$. Because there are infinitely many directions to compute normal curvature, the standard approach is to compute only the largest and smallest (principal) curvatures denoted by k_1, k_2, respectively. These have associated directions $\mathbf{t}_1$, $\mathbf{t}_2$ which are orthogonal. Principal curvatures can be positive or negative, with the sign denoting whether the surface bends outwards or inwards. Largest curvature k_1 is larger than curvature k_2 in absolute value ($|k_1| > |k_2|$). A crest point is a point of the surface where its largest curvature k_1 is maximum in its corresponding direction. The definition of a crest point is

$$D_{\mathbf{t}_1} k_1(u,v) = 0 \tag{1}$$

where $D_{\mathbf{t}_1}$ is the directional derivative in direction $\mathbf{t}_1 \in \Re^2$, $k_1 \in \Re$ is the largest principal curvature and $\mathbf{t}_1$ is its domain direction on a point (of the surface) with domain coordinates (u,v).

Crest points implicitly trace lines on the surface denoted as crest lines. Crest lines are shape features with the main characteristic of using local information to yield a global description of the surface. Even though crest lines are local shape features, when they are viewed together they partially describe the surface.

Crest points trace ridges (when k_1 is positive) and valleys (when k_1 is negative) on the surface. Even though we can trace ridges and valleys concurrently, it is as useful to trace only ridges or valleys, especially because ridges have different characteristics than valleys, like their k_1 curvature ranges, even though they are features of the same surface.

Figure 2 shows an example of a crest line. Because a crest point has maximum largest curvature in its corresponding direction, a crest line naturally follows the direction of the smallest curvature of its composing crest points.

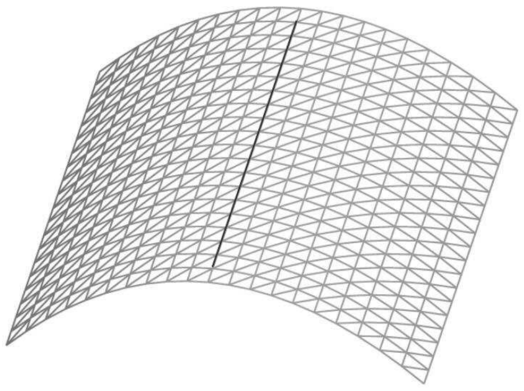

Figure 2: A crest line example.

2.2 Error Quadrics

The error quadric metric [Garland and Heckbert 1997], uses the observation that each vertex is the intersection of a set of planes, the planes of the faces that meet at that vertex. A set of planes is associated with each vertex $\mathbf{v}$ and the error of the vertex $\Delta(\mathbf{v})$ is defined with respect to this set as the sum of squared distances to its planes:

$$\Delta(\mathbf{v}) = \Delta(\begin{bmatrix} \upsilon_x & \upsilon_y & \upsilon_z & 1 \end{bmatrix}^T) = \sum_{\mathbf{p} \in planes(\mathbf{v})} (\mathbf{p}^T\mathbf{v})^2 \tag{2}$$

where $\mathbf{p} = \begin{bmatrix} a & b & c & d \end{bmatrix}^T$ represents the plane defined by the equation $ax + by + cz + d = 0$ where $a^2 + b^2 + c^2 = 1$. The set of planes at a vertex is initialized to be the planes of the triangles that meet at that vertex.

The error metric given in (2) can be rewritten as:

$$\begin{aligned} \Delta(\mathbf{v}) &= \sum_{\mathbf{p} \in planes(\mathbf{v})} (\mathbf{v}^{\mathrm{T}}\mathbf{p})(\mathbf{p}^{\mathrm{T}}\mathbf{v}) \\ &= \sum_{\mathbf{p} \in planes(\mathbf{v})} \mathbf{v}^{\mathrm{T}}(\mathbf{p}\mathbf{p}^{\mathrm{T}})\mathbf{v} \\ &= \mathbf{v}^{\mathrm{T}} \left(\textstyle\sum_{\mathbf{p} \in planes(\mathbf{v})} \mathbf{K_p} \right) \mathbf{v} \end{aligned}$$

where $\mathbf{K_p}$ is the matrix:

$$\mathbf{K_p} = \mathbf{p}\mathbf{p}^{\mathrm{T}} = \begin{bmatrix} a^2 & ab & ac & ad \\ ab & b^2 & bc & bd \\ ac & bc & c^2 & cd \\ ad & bd & cd & d^2 \end{bmatrix}$$

This fundamental matrix $\mathbf{K_p}$ can be used to find the squared distance of any point in space to the plane $\mathbf{p}$. These fundamental matrices can be summed together to represent an entire set of planes by a single matrix $\mathbf{Q}$.

They implicitly track sets of planes using a single matrix by simply adding two matrices. For a given contraction $(\mathbf{v}_1, \mathbf{v}_2) \rightarrow \bar{\mathbf{v}}$ the new matrix $\bar{\mathbf{Q}}$ that approximates the error at $\bar{\mathbf{v}}$ is $\bar{\mathbf{Q}} = \mathbf{Q}_1 + \mathbf{Q}_2$, where the matrices $\mathbf{Q}_1$, $\mathbf{Q}_2$ are used to approximate the error at vertices $\mathbf{v}_1$, $\mathbf{v}_2$, respectively.

Observe that this metric over-estimates the real error because the error obtained from common faces is added twice. Therefore, it cannot lead to optimum results as it will be shown in section 4.

2.3 Placing the new vertex

The position for the new vertex $\bar{\mathbf{v}}$ is chosen such that it minimizes the error $\Delta(\bar{\mathbf{v}}) = \bar{\mathbf{v}}^T \bar{\mathbf{Q}} \bar{\mathbf{v}}$ [Garland and Heckbert 1997]. Thus, $\bar{\mathbf{v}}$ is found by solving $\partial\Delta/\partial x = \partial\Delta/\partial y = \partial\Delta/\partial z = 0$. This is equivalent to solving:

$$\begin{bmatrix} q_{11} & q_{12} & q_{13} & q_{14} \\ q_{12} & q_{22} & q_{23} & q_{24} \\ q_{13} & q_{23} & q_{33} & q_{34} \\ 0 & 0 & 0 & 1 \end{bmatrix} \bar{\mathbf{v}} = \begin{bmatrix} 0 \\ 0 \\ 0 \\ 1 \end{bmatrix}$$

The linear system does not always have a solution as stated by G&H and as observed experimentally. Sometimes an alternative must be used. A good alternative solution is to place the new vertex $\bar{\mathbf{v}}$ somewhere on the contracted edge. Suppose we contract the edge $(\mathbf{v_i}, \mathbf{v}_j)$. The new vertex will be a linear combination of them $\bar{\mathbf{v}} = (1-t)\mathbf{v_i} + t\mathbf{v}_j$, with $t \in [0,1]$. t is weighted using the incident edges of $\mathbf{v}_i, \mathbf{v}_j$. We set $t = n_j/(n_i + n_j)$, where $n_i = d_i - 1$, $n_j = d_j - 1$ and d_i, d_j are the valence of $\mathbf{v}_i$, $\mathbf{v}_j$, respectively. Because the error increases when the distance of $\bar{\mathbf{v}}$ from the planes that meet on this vertex increases, this solution attempts to place $\bar{\mathbf{v}}$ as close as possible to most of the planes. This is achieved by placing $\bar{\mathbf{v}}$ closer to the initial vertex ($\mathbf{v}_i$ or $\mathbf{v}_j$) that is incident to the most planes. This solution is more stable than the vertex placement by G&H, but yields more error.

3 Proposed Algorithm

The proposed algorithm's components are given in the following sections. These are the feature constraints, boundary handling and how fairness and uniformity can be enforced.

3.1 Feature Constraints

The proposed simplification algorithm uses surface shape features, in this case crest lines, to constrain the simplification. The motivation is that the regions of a surface besides its features do not usually hold any important shape information. Hence, the simplification efforts must be concentrated in those regions. In addition, feature lines should be preserved and simplified, as well. The feature constraints used to decide valid edges for contraction are:

1. *an edge can be contracted if* ***neither of*** *its vertices are crest points,*

2. *an edge can be contracted if* ***both of*** *its vertices are crest points.*

3. *a boundary edge* ***cannot*** *be contracted.*

The first rule says every edge not belonging partly or totally to a crest line (feature) is valid for contraction. This protects semi-crest edges, edges with one crest vertex and one non-crest vertex, from being simplified; this could disconnect the feature lines and eventually distort them. The second rule gives the opportunity to simplify edges belonging totally to a crest line (crest edges). This simplifies and preserves crest lines because their endpoints are not eligible for simplification. The third rule is used because boundaries are not handled at this point of the algorithm. For more detail on boundary handling see section 3.2.

3.2 Boundary Handling

Boundary edges, edges incident only to one face, must be handled differently than the rest of the edges.The reason is the incompleteness of their neighborhood. This problem creates less quadric error when simplifying boundary edges than non-boundary edges, leading to an oversimplification of the boundary edges and bad results.

G&H suggest [Garland and Heckbert 1997] that for each face surrounding a particular boundary edge, to generate a perpendicular plane running through the edge, rewrite those planes as fundamental matrices $\mathbf{K_p}$, weight them by a large penalty factor and add them into the initial matrices for the endpoints of the edge. After this transformation, these edges will be contracted as they were non-boundary edges. Even though, it is stated this works well, the penalty factor is quite arbitrary.

Instead, we suggest to do something simpler and more intuitive. When the algorithm terminates, we re-execute it but now only boundary edges are considered for contraction. Since all the edges to be contracted are boundary edges, then the error produced, while contracting, will be fair for all edges.

3.3 Enforcing Fairness and Uniformity

In other methods, an edge with minimum cost among all the edges is contracted. Here we contract an edge with minimum cost locally. When an edge $(\mathbf{v}_i, \mathbf{v}_j)$ is simplified and replaced by vertex $\overline{\mathbf{v}}_i$, we do not allow an edge incident to vertex $\overline{\mathbf{v}}_i$ to be simplified in the same iteration. The effect of this constraint is that every neighborhood is simplified in the same rate. This tends to keep the triangulation as uniform as possible. Figure 3 shows the standard deviation of the triangle area for brain, bunny and hygea after 75% and 90% contraction for the proposed and G&H algorithms. Standard deviation shows the compactness of triangle mean area. If standard deviation is large, then there are very large and very small triangles, in the mesh, with respect to their area. Observe that there is substantial difference on standard deviation for the two algorithms. The area of the triangles generated by the proposed algorithm is much closer to the mean area than the G&H algorithm. This means that the G&H algorithm does not lead to as uniform triangulation, for reasons such as oversimplification, as the proposed algorithm because it creates very small and very large triangles.

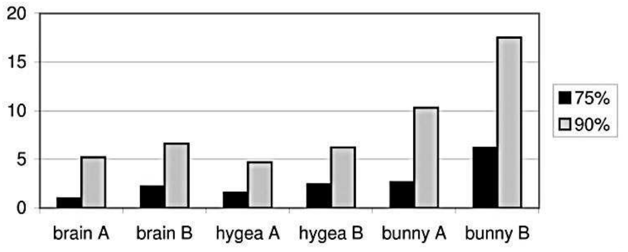

Figure 3: The graph for the standard deviation of triangle area for the simplified objects. Objects A use the proposed algorithm, objects B use the G&H algorithm.

3.4 Summary

The proposed algorithm is iterative as it has to go through the heap of eligible edges for contraction more than one time, until it simplifies up to the user-defined level or until it cannot simplify no more.

A property of the proposed method is that the simplification has a limit. G&H's algorithm can simplify a surface indefinitely and eventually delete most of the features that make the object identifiable. The proposed algorithm cannot simplify indefinitely (unless we relax the feature constraint) and it always preserves all the important features. The simplification's limit is dependent on the number of features and the density of the triangulation.

The proposed algorithm keeps the pre-processing procedure of G&H's algorithm essentially the same. We give it here for completeness. The pre-processing procedure is:

- Compute the $\mathbf{Q}$ matrices for all initial vertices.

- Select all valid point pairs.

- Compute the optimal contraction target $\overline{\mathbf{v}}$ for each valid pair $(\mathbf{v}_i, \mathbf{v}_j)$. The error $\overline{\mathbf{v}}^T(\mathbf{Q}_i + \mathbf{Q}_j)\overline{\mathbf{v}}$ of this target vertex becomes the cost of contracting that pair.

- Place all the pairs in a heap keyed on cost with the minimum cost pair at the top.

The only change is the criteria for selecting valid edges for contraction. The contraction procedure changes as follows:

```
Set $cur\_iter = 1$. For every vertex $\mathbf{v}_i$ set $\mathbf{v}_i.iter = 0$.
Repeat
    Remove the pair $(\mathbf{v}_i, \mathbf{v}_j)$ of least cost from the heap such that $\mathbf{v}_1.iter < cur\_iter$ and $\mathbf{v}_j.iter < cur\_iter$.
    Increase the value of the property iter for all vertices incident to $\mathbf{v}_i$ and $\mathbf{v}_j$ including $\mathbf{v}_i$ by one and contract this pair.
    Update the costs of all valid pairs involving $\mathbf{v}_i$.
    Set $cur\_iter = cur\_iter + 1$.
Until the termination criterion is not reached.
```

The variable *cur_iter* is used to count the algorithm's iterations. Variable *iter* is used to decide if an edge is eligible for contraction during the current algorithm's iteration. The contraction procedure terminates when a user-defined number of vertices remain. The algorithm terminates by doing at most one pass on the heap.

4 Results

We have experimented on different surfaces. The first two surfaces are bivariate test functions $f(x,y)$ defined over $[-1,1]\times[-1,1]$ and evaluated on a 50×50 grid with equidistant spacing. These are x^3-3xy^2 (monkey saddle) and $\cos(\pi x)+\cos(\pi y)$ (trig. function). The other surfaces are the Stanford bunny, Hygea (a digitized Greek statue) and a part of a brain (cortical) surface. Tables 1, 2, 3, 4 show the mean vertex quadric error (cost), the max valence of a vertex and the mean valence of the vertices for the G&H algorithm and the proposed algorithm, respectively. The mean vertex quadric error is the average quadric error produced by each of the simplified edges. The number does not reflect the object's complexity or size because the quadric error's magnitude depends on the object's size.

Table 1: G&H's algorithm. Mean vertex quadric error, max and mean valence are recorded for 75% simplification.

surface	cost	max	mean
monkey saddle	.287	16	6.28
trig. function	.565	14	6.23
bunny	1.11	28	5.24
hygea	0.12	23	5.95
brain part	0.01	27	6.11

Table 2: Proposed algorithm. Mean vertex quadric error, max and mean valence are recorded for 75% simplification.

dataset	cost	max	mean
monkey saddle	0.212	15	6.27
trig. function	0.176	13	6.29
bunny	0.175	18	5.25
hygea	0.05	15	5.95
brain part	0.002	19	6.12

Table 3: G&H's algorithm. Mean vertex quadric error, max and mean valence are recorded for 90% simplification.

dataset	cost	max	mean
bunny	5.15	35	4.05
hygea	0.67	22	5.88
brain part	0.064	32	6.52

The proposed algorithm shows much better results on the cost for all the surfaces. The proposed algorithm has strictly smaller max valence and equal or slightly greater mean valence. Smaller max valence means that the surface is simplified uniformly instead of concentrating a lot of effort in certain neighborhoods.

While numbers say a lot about the two algorithms, figures can say even more. Figures 4, 5 show the simplified triangulation and the alteration of the crest lines after 75% reduction on the number of vertices for both algorithms. G&H clearly distorts a lot the features of the monkey saddle but less the features of the trigonometric function. The reason for the difference in feature distortion is that the features of the trigonometric function are more pronounced in contrast to monkey saddle's features. In addition, it leaves many neighborhoods untouched while other neighborhoods are heavily simplified. The proposed algorithm simplifies more uniformly and preserves all the features while simplifying them, as well.

Furthermore, we have experimented with the Stanford bunny (fig. 6a), hygea and a brain part (fig. 6b). Initially the bunny, hygea and the brain part had 35905, 33587 and 19902 vertices, respectively. Figures 7, 9 and 11 show the modification of the features by G&H and their preservation by the proposed algorithm, after 75% reduction on the number of vertices. After 75% reduction, the bunny, hygea and the brain part have 8976, 8396 and 4975 vertices, respectively. Again, figures 8, 10 and 12 show the modification of the features by G&H and their preservation by the proposed algorithm, after 90% reduction on the number of vertices. After 90% reduction, the bunny, hygea and the brain part have 3594, 3358 and 1995 vertices, respectively. The surface qualitative difference can be easily identified. The proposed algorithm simplifies them but their smoothness and their features remain significantly unchanged, even after 90% reduction. In contrast, G&H's algorithm creates much rougher surfaces and also distorts or deletes some features.

Table 4: Proposed algorithm. Mean vertex quadric error, max and mean valence are recorded for 90% simplification.

dataset	cost	max	mean
bunny	1.39	18	4.06
hygea	0.36	17	5.88
brain part	0.038	30	6.54

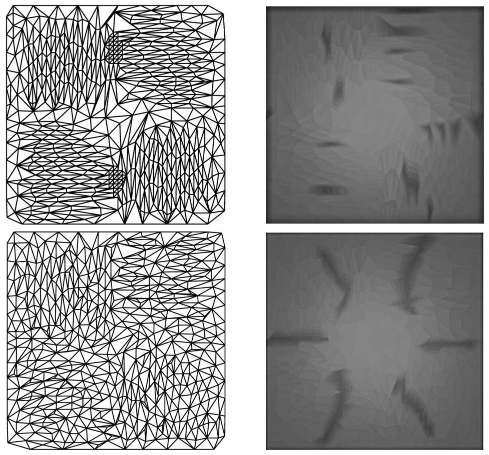

Figure 4: Monkey saddle after 75% simplification. Left column shows the triangulation, right column show the domain with crest lines. Top is G&H algorithm. Bottom is the proposed algorithm.

G&H's algorithm is very good but it does not give the best results. It usually contracts feature edges unless they are very sharp. Also, it tends to create long thin faces which is not necessarily a desirable effect. In addition, it does not preserve the surface's smoothness because some vertices have high valence while other vertices have low valence. By introducing crest lines as a constraint and modifying the contraction criterion, this algorithm is enhanced significantly as the results show.

5 Conclusion and Future Work

A simplification algorithm has been presented which improves substantially a simplified surface's quality. This is achieved because surface shape features are used to constrain the algorithm and fairness and uniformity were enforced by applying the cost metric locally instead of globally. In addition, we proposed a simple technique for simplifying boundaries. A complete comparison was performed with Garland and Heckbert's simplification algorithm.

Future work includes extending this algorithm to handle surfaces with texture and the development of a real-time feature sensitive multiresolution algorithm.

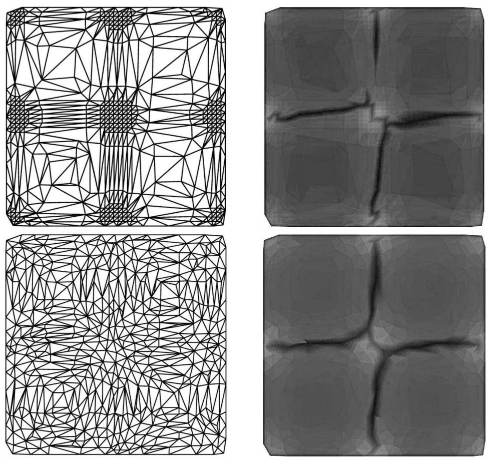

Figure 5: Trigonometric function after 75% simplification. Left column shows the triangulation, right column show the domain with crest lines. Top is G&H algorithm. Bottom is the proposed algorithm.

References

BAJAJ, C., AND SCHIKORE, D. 1996. Error-bounded reduction of triangle meshes with multivariate data. *SPIE 2656*, 34–45.

CHOO, K., YUN, I., AND LEE, S. 1999. Edge-based approach to mesh simplification. In *Proc. of the IEEE Second Int. Conf. on 3-D Digital Imaging and Modeling*, 368–377.

COHEN, J., VARSHNEY, A., MANOCHA, D., TURK, G., WEBER, H., AGARWAL, P., BROOKS, F., AND WRIGHT, W. 1996. Simplification envelopes. *Computer Graphics*, 119–128.

COHEN, J., OLANO, M., AND MANOCHA, D. 1998. Appearance-preserving simplification. *ACM Siggraph*, 115–122.

GAO, J., ZHOU, M., AND WANG, H. 2000. Mesh simplification with average planes for 3-D image. In *IEEE Int. Conf. on Systems, Man and Cybernetics*, vol. 2, 1412–1417.

GARLAND, M., AND HECKBERT, P. 1997. Surface simplification using quadric error metrics. *ACM Siggraph*, 209–216.

GUÉZIEC, A. 1999. Locally toleranced surface simplification. *IEEE Trans. on Visualization and Computer Graphics 5(2)*, 168–189.

HOPPE, H., DEROSE, T., DUCHAMP, T., MCDONALD, J., AND STUETZLE, W. 1993. Mesh optimization. *ACM Siggraph*, 19–26.

HOPPE, H. 1996. Progressive meshes. *ACM Siggraph*, 99–108.

JEONG, W.-K., KAHLER, K., AND SEIDEL, H.-P. 2002. Subdivision surface simplification. In *Proc. of the 10th Pacific Conf. on Computer Graphics and Applications*.

KALVIN, A. D., AND TAYLOR, R. H. 1996. Superfaces:polygonal mesh simplification with bounded error. *IEEE Computer Graphics and Applications 16(3)*, 64–77.

KIM, S.-J., KIM, S.-K., AND KIM, C.-H. 2002. Discrete differential error metric for surface simplification. In *Proc. of the 10th Pacific Conf. on Computer Graphics and Applications*.

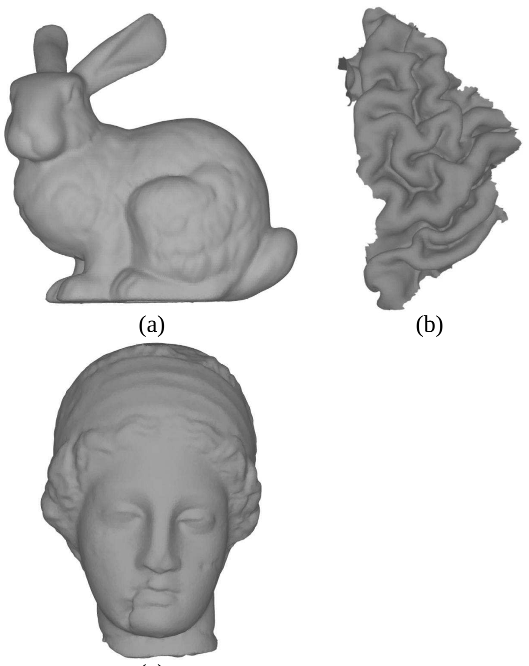

Figure 6: Original models. (a) Stanford bunny, (b) Brain part and (c) Hygea.

POJAR, E., AND SCHMALSTIEG, D. 2003. User-controlled creation of multiresolution meshes. In *Proc. of the Symposium on Interactive Computer Graphics*.

RONFARD, R., AND ROSSIGNAC, J. 1996. Full-range approximation of triangulated polyhedra. In *Proc. of Eurographics, Computer Graphics Forum*, Blackwell, J. Rossignac and F. Sillon, Eds., vol. 15(3), Eurographics, C67–C76.

SCHRÖEDER, W., ZARGE, J., AND LORENSEN, W. 1992. Decimation of triangle meshes. *ACM Siggraph*, 65–70.

SOUCY, M., AND LAURENDEAU, D. 1996. Multi-resolution surface modeling based on hierarchical triangulation. *Computer Vision and Image Understanding 63*, 1–14.

STYLIANOU, G., AND FARIN, G. 2004. Crest lines for segmentation and flattening. *IEEE Trans. on Visualization and Computer Graphics*.

TURK, G. 1992. Re-tiling polygonal surfaces. *Computer Graphics 26(2)*, 55–64.

WATANABE, K., AND BELYAEV, A. 2001. Detection of salient curvature features on polygonal surfaces. In *Proc. of Eurographics*, A. Chalmers and I.-M. Rhyne, Eds., vol. 20(3).

About the author

Georgios Stylianou is a lecturer in the Department of Computer Science at Cyprus College, Cyprus. He received his PhD in Computer Science from Arizona State University, USA, in 2003 and his BSc in Computer Science from the University of Cyprus in 1998. His research interests lie in the area of geometric modelling and include parametrization and multi-resolution methods, feature extraction, deformation and registration.

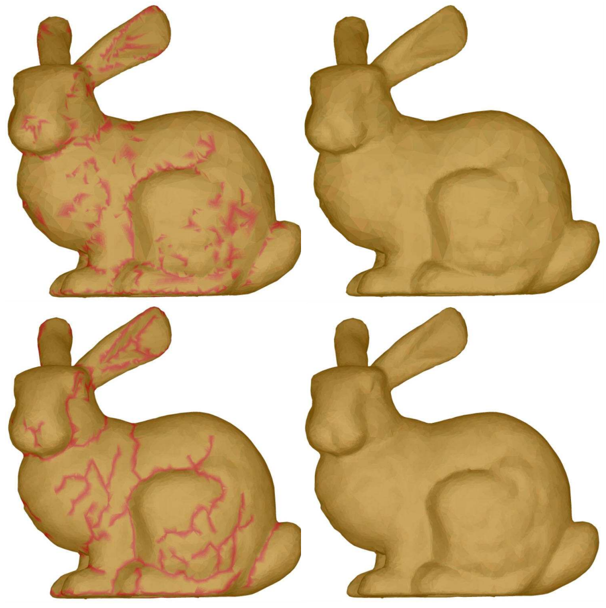

Figure 7: Stanford bunny after 75% simplification. Top: G&H algorithm. Bottom: proposed algorithm.

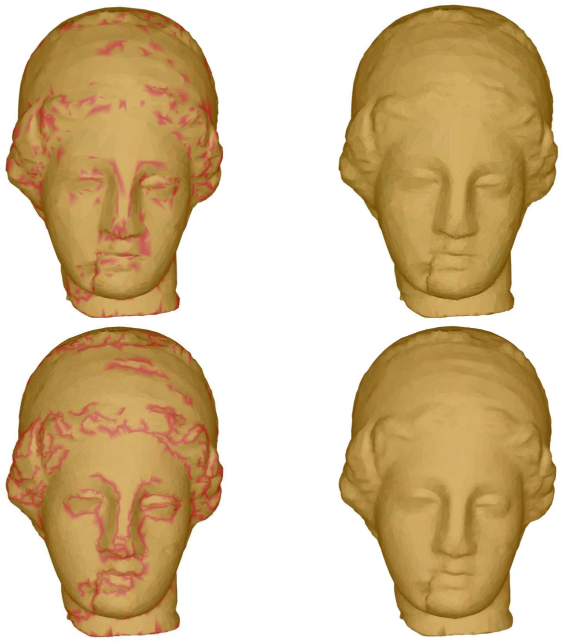

Figure 9: Hygea after 75% simplification. Top: G&H algorithm. Bottom: proposed algorithm.

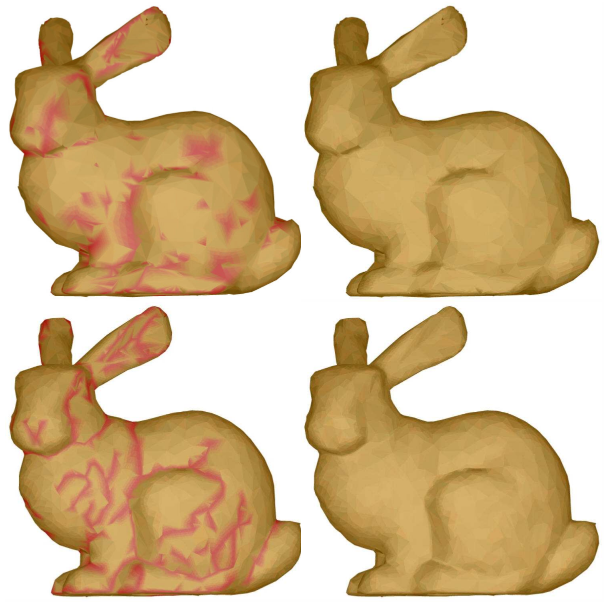

Figure 8: Stanford bunny after 90% simplification. Top: G&H algorithm. Bottom: proposed algorithm.

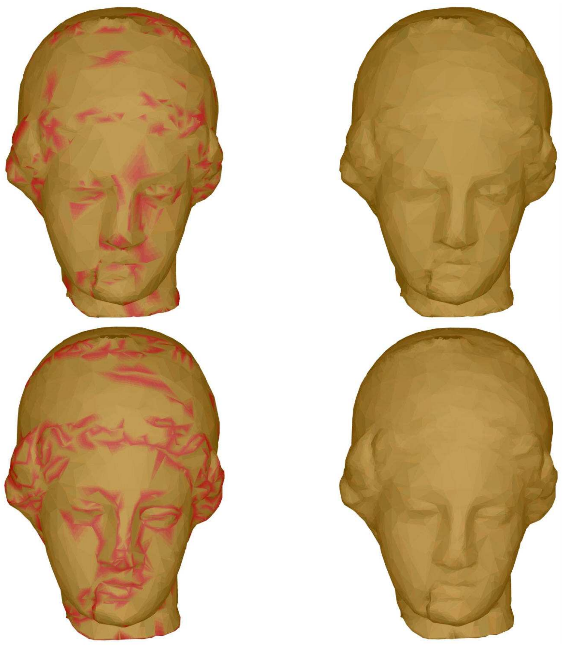

Figure 10: Hygea after 90% simplification. Top: G&H algorithm. Bottom: proposed algorithm.

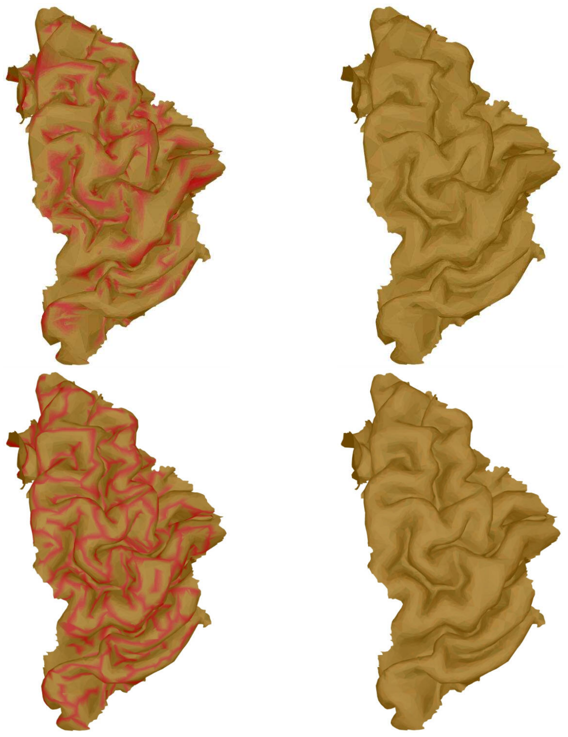

Figure 11: Brain part after 75% simplification. Top: G&H algorithm. Bottom: proposed algorithm.

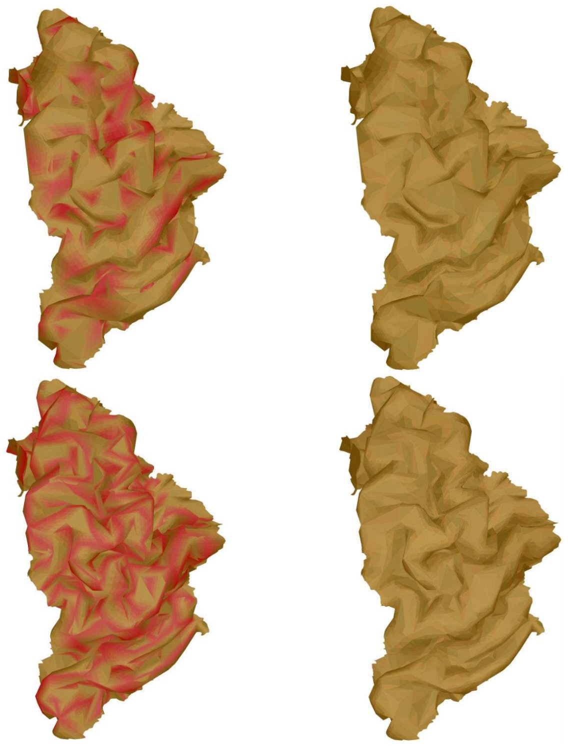

Figure 12: Brain part after 90% simplification. Top: G&H algorithm. Bottom: proposed algorithm.