THE JOURNAL OF VISUALIZATION AND COMPUTER ANIMATION
J. Visual. Comput. Animat. 2003; **14:** 93–109 (**DOI**: 10.1002/vis.308)

Shape-adaptive 3-D mesh simplification based on local optimality measurement

By In Kyu Park, Sang Wook Lee and Sang Uk Lee*

Mesh simplification is the process of reducing the number of triangles in a mesh representation of object surface. For a given level of detail or error tolerance, the conventional mesh simplification algorithms maximize the edge length globally, without explicitly considering local object shape. In this paper, we present a shape-adaptive mesh simplification algorithm that locally maximizes edge length, depending on local shape. The proposed algorithm achieves shape-adaptive simplification by iteratively maximizing edges between vertices, based on comparison with the 'optimal' edge lengths derived from local directional curvatures for a given error tolerance. Edge-based processing facilitates the local shape adaptation and preserves sharp features. Experimental results demonstrate the efficacy of the proposed algorithm, by showing good visual quality and extremely small approximation error. Copyright © 2003 John Wiley & Sons, Ltd.

Received: 17 July 2002; Revised: 30 October 2002

KEY WORDS: mesh simplification; level of detail; shape-adaptive; local shape; directional curvature; sharp feature

Introduction

In 3-D computer graphics and animation, polygonal mesh models have been commonly used to represent arbitrarily shaped 3-D objects. In addition to artificially generated objects in computer graphics and computer-aided design, 3-D data obtained from a laser range scanner is often represented by polygonal mesh. In spite of the mesh models' ability to represent complex 3-D shapes, the data size of a mesh structure is usually quite large when 3-D details are preserved.

For efficient storage, transmission and rendering, various algorithms have been proposed that reduce the number of polygons, depending on the object complexity to be maintained.[1–13,21] The complexity, i.e. the level of detail (LOD), is determined by viewing distance and direction, lighting, focus of attention, graphics pipeline performance and transmission bandwidth. Once the LOD is determined in terms of an error tolerance, a mesh simplification algorithm iteratively applies mesh operations while the error tolerance criteria are satisfied. An example of mesh simplification is shown in Figure 1.

*Correspondence to: In Kyu Park, Multimedia Lab, Samsung Advanced Institute of Technology, San 14-1, Nongseo-ri, Kiheung-eup, Yongin 449-712, Korea.
E-mail: saitpik@sait.samsung.co.kr

There have been a number of simplification algorithms developed for different goodness-of-fit measures, mesh operations, methods for determining new node positions and selecting vertex/edge orders. The most remarkable previous work is Garland and Heckbert's method, which is based on quadric error metrics (QEM).[6] It is very simple to evaluate and implement the QEM, yielding fast and stable simplification. Schroeder *et al.*[1] introduced iterative vertex decimation and retriangulation to simplify meshes. The change of mesh shape is limited by a global bound on the maximum allowable change in iteration stages. Hoppe *et al.*[2] proposed a global optimization algorithm in which a data-fitting energy function is minimized to reduce the number of vertices, while maintaining a certain mesh shape. Local mesh transformations, such as *edge collapse*, *edge split* and *edge swap*, are used to simplify meshes in iteration stages. In his subsequent work, Hoppe[4] developed a progressive mesh technique, in which *vertex split* is used to transmit and construct mesh progressively. Cohen *et al.*[5] proposed an approach of *simplification envelopes* to guarantee that the distance between original and simplified mesh is within a user-specifiable distance.

On the other hand, if the original mesh and its local shape are taken into account during the mesh simplification, polygon size can be larger where local shape

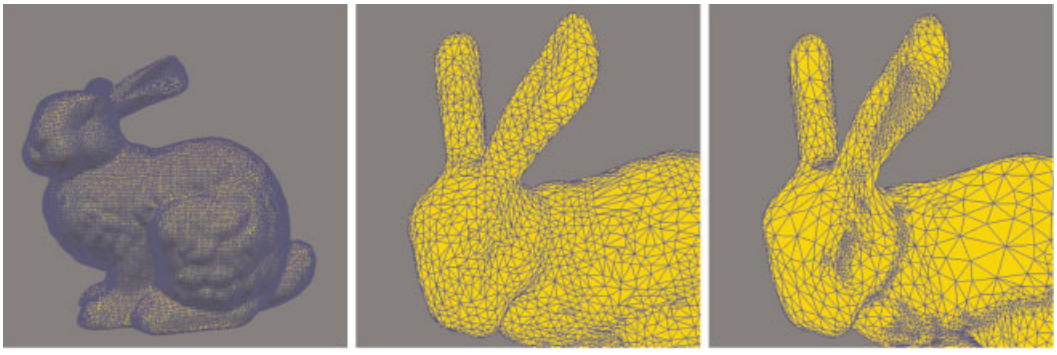


Figure 1. Example of mesh simplification. (a) Original mesh with 69K triangles. (b) Uniformly simplified mesh (20% of the original). (c) Adaptively simplified mesh (proposed algorithm, 20% of the original).

change is smooth, while smaller polygons are required for complex regions. Compared with conventional uniform simplification, shape-adaptive simplification is a very attractive methodology since a much lower number of polygons can represent same original geometry, while keeping similar visual quality. Equivalently, for a given number of polygon, a shape-adaptive mesh yields a smaller approximation error. An example of shape-adaptive mesh simplification is shown in Figure 1(c). Note that previous works are also shape-adaptive to some extent; however, the effect is not outstanding.

In this paper, we present a novel algorithm of shape-adaptive mesh simplification, which uses a new measure of local-shape fitness. Instead of taking the geometric error metrics as the measure for the degree of simplification, we utilize the length of the edges between vertices. We show that the desirable size of triangle elements for a given error tolerance can be effectively represented by this local edge length and it can be efficiently computed from a given mesh. In other words, polygon-size maximization for a given error tolerance is achieved locally by estimating 'optimal' edge length from a given error tolerance and local directional curvature, and mesh simplification is performed in a way to make the actual edge length converge to the optimal edge length. For the simplification procedure to be more stable and rapid, we propose an efficient method to obtain the optimal length without computing the curvature explicitly. Note that we avoid the direct computation of curvature on a polygonal mesh which is computationally complex and unstable.[14–18] Once the optimal edge lengths are computed at an iteration stage, they can be used for guiding mesh operations without the need for error calculation. Sharp features are preserved efficiently considering the geometric relation of triangles connected to an edge. We achieve shape-adaptive simplification while avoiding computationally complex shape analysis. Therefore, the proposed algorithm is quite easy to implement and numerically stable. In addition, this edge-based approach preserves geometric features well.

This paper is organized into several sections. In the next section, we present the physical meaning and the mathematical definition of the proposed 'optimal' edge length. An efficient implementation of computing the optimal length is described in the third section. The iterative mesh simplification is discussed in the fourth section. The experimental results are shown in the fifth section. We conclude our work in the final section.

Optimal Edge Length: Physical Meaning and Mathematical Definition

Edge Optimization

The goal of mesh optimization is to achieve maximal mesh simplification for a given error tolerance level, i.e. to reduce the number of triangles as much as possible, while the approximation error is maintained below a specified threshold, e. Conversely, for a given number of triangles, the goal is to minimize the approximation error as much as possible. In our approach, the mesh optimization is performed by changing the lengths of edges since the 'optimal' edge length can be easily defined and estimated from local directional geometric shape in two dimensions, which is shown later in this paper. Edge optimization means that, for each edge in the triangular

mesh, the edge length is maximized, while keeping the error below the tolerance level. The edge optimization is actually another form of mesh optimization and its advantage is that edge lengths can be determined to represent the local and directional surface structure. We define the 'optimal' edge length so that the geometric curvature can be effectively represented by the edge.

Given an error threshold and initial mesh, the proposed algorithm computes the optimal length of each edge at each stage of iteration, and adjusts the actual length of an edge iteratively towards the optimal edge length. Since the optimal length is the target of actual edge length change at each iteration stage, global or local error calculation is not necessary in our approach.

Since the optimal edge length is used to guide the iterative mesh adaptation, accurate modeling of the optimality and its measurement are of great importance in the proposed algorithm. In what follows, we begin with the modeling of optimal edge length in polygonal approximation of a continuous 2-D curve. Later in this section, we extend this to extract the optimal edge length in polyhedral approximation of a continuous surface.

2-D Case: Polygonal Approximation of Continuous Curve

First, let us consider a simple case of polygonal approximation of a circle. For a given number of line segments, the best approximation that minimizes the approximation error would be the equilateral polygon, of which the circle is a circumcircle, as shown in Figure 2. In this case, the optimal edge length is the length of edge of the equilateral polygon. Conversely, if the error threshold ε is given, the number of line segments can be determined, such that the length of the line segment is the maximum possible length of the chord where the maximum arc–edge distance is less than ε.

We extend this notion to the general 2-D curve case. Let P be a point on a curve $C(t) = (x(t), y(t))$, which is abstracted by an edge L and parameterized linearly along L with parameter interval $[0, 1]$, as shown in Figure 3(a). If $C(t)$ is second-order continuous, then the radius of curvature is defined at every position of P. The radius of curvature at P is by definition the radius of the circle, which is a tangent to C at P and actually the inverse of the curvature κ. The mathematical definition of κ at P (at $t = t^*$) is given by

$$\kappa(t^*) = \frac{\sqrt{|C'(t^*)|^2 |C''(t^*)|^2 - (C'(t^*) \cdot C''(t^*))^2}}{|C'(t^*)|^3} \quad (1)$$

and the radius of the tangential circle at P is given by

$$r_\kappa(t^*) = \frac{1}{\kappa(t^*)} \quad (2)$$

The optimal tangential length at P is then defined as the maximum possible length of the chord in the tangential circle when the maximum arc–edge distance is below e as illustrated in Figure 3(b). This is the same as the case of polygonal approximation of a circle described earlier. Based on this observation, from the simple geometric relation, the optimal length is obtained as

$$L^P_{opt}(t^*) = 2\sqrt{2\, e\, r_\kappa - e^2} \quad (3)$$

Note that the above optimal length is defined only on P. The length of the whole arc can be represented by the average radii of the tangential circles at all the points in the arc. Some examples of the tangential circles are shown in Figure 3(c). The average of the radii of the tangential circles is given by

$$\bar{r}_\kappa = \int_0^1 r_\kappa(t)dt \cong \frac{1}{N}\sum_{k=0}^{N-1} r_\kappa\left(\frac{k}{N}\right) \quad (4)$$

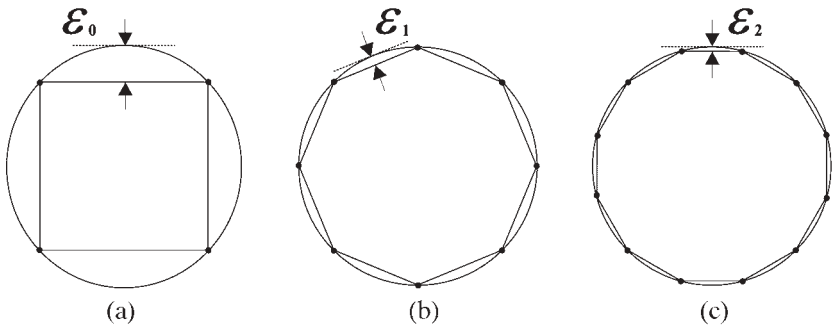


Figure 2. Polygonal approximation of a circle. (a) Approximation by 4 line segments. $\varepsilon_1 = 0.2071$. (b) Approximation by 8 line segments. $\varepsilon_2 = 0.0538$. (c) Approximation by 12 line segments. $\varepsilon_3 = 0.0240$.

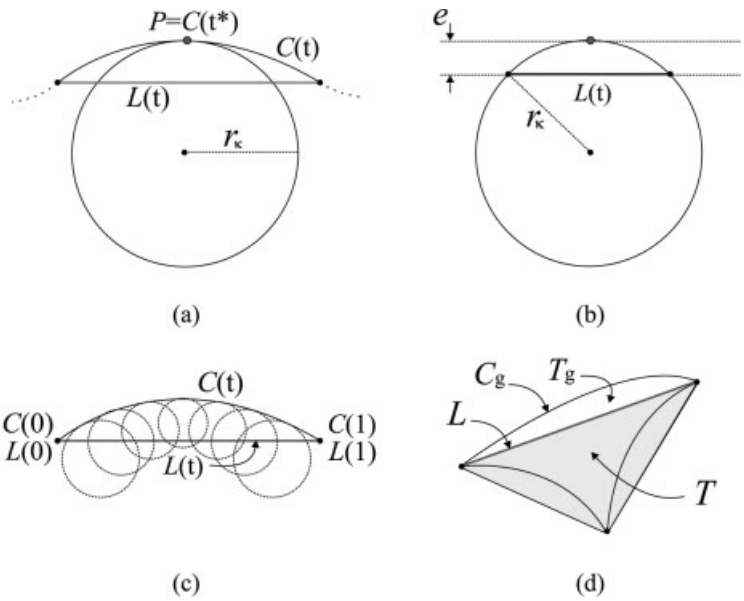


Figure 3. Computing optimal edge length. (a) Radius of curvature at a point. (b) Optimal length computation at P using simple geometry. (c) Optimal edge length for the surface curve $C(t)$. (d) Extension to 3-D mesh.

in which N is the number of samples when approximated using a finite number of points. As in equation (3), the optimal length of the edge is given by

$$L_{opt} = 2\sqrt{2\ e\ \bar{r}_\kappa - e^2} \tag{5}$$

Note that L_{opt} is the desirable value of the edge length in the current configuration of the piecewise approximation. By comparison of L_{opt} and the actual edge length, it can be determined whether the curve is locally oversampled or undersampled, and it guides the edge optimization.

3-D Case: Polyhedral Approximation of Continuous Surface

A 3-D surface is represented by a polygonal mesh, which normally consists of triangular facets, edges, and nodes. While a polyline segment abstracts the corresponding curve in polygonal approximation, a triangular facet abstracts some portion of the surface, which is called the geodesic triangle, as illustrated in Figure 3(d).

Let $S(t,s)$ be the local surface and the geodesic triangle T_g be abstracted by the triangle T as depicted in Figure 3(d). In this case, each side of the geodesic triangle T_g is abstracted by the triangle T. In this geometry, the geodesic curve C_g is the boundary between T_g and its neighbouring geodesic triangle. C_g can be easily obtained as the cross-section of $S(t,s)$ and the plane that is defined by the edge L and its normal vector. The normal vector of L is defined as the area-weighted average of the normal vectors of the triangles which share L. In this geometry, the directional variation along an edge direction is represented on C_g. Note that C_g is represented as a polyline in a real situation, since $S(t,s)$ is in fact the original mesh. Based on the observation that C_g and L lie on the same plane, this is analogous to the 2-D geometry discussed before, i.e. the 2-D curve C_g is abstracted by the edge L. Therefore, the optimal edge length can be computed from the 2-D curve C_g and the same method of 2-D edge optimization can be applied to a 3-D surface.

Efficient Evaluation and Implementation of the Optimal Edge Length

Fast Algorithm Using Circular Fitting and Table Look-Up

Instead of the expensive computation of equations (1–5), we suggest that the optimal length can be estimated, by directly fitting a circle to the polyline. But, its computation can be easily realized by efficient table look-up. A basic assumption in this procedure is that the local surface variation is smooth enough to be approximated using the cubic Hermite polynomial.[19] Based on this, it is found experimentally that the proposed fast algorithm

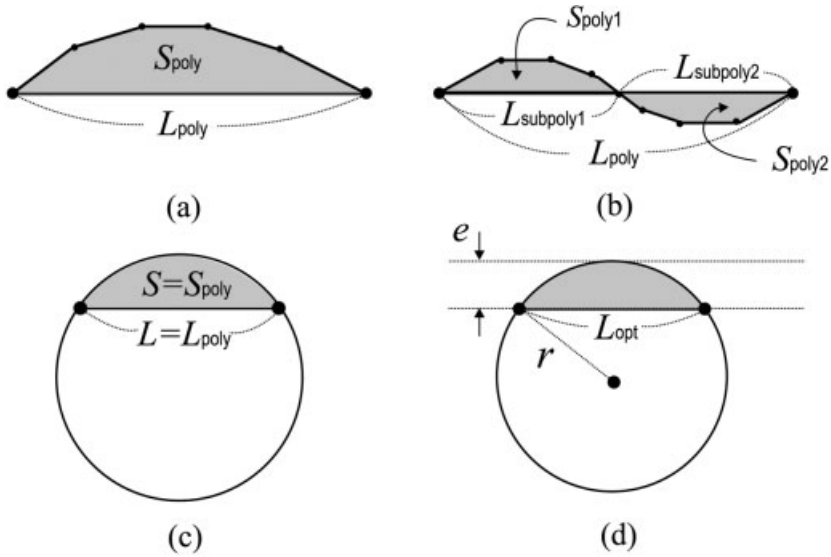


Figure 4. Efficient computation of the optimal length. (a) Edge and convex polyline. (b) Edge and non-convex polyline. (c) Projection of the area and edge length in (a) on a circle. (d) Finding radius of circle and the optimal length.

yields results that are very close to those obtained from evaluation of the mathematical definition.

In Figure 4, the process of circular fitting is shown. First, the area of the region between the polyline and the edge is computed. In Figure 4(a), S_{poly} and L_{poly} denote the area between the polyline and the edge and the distance between the end points of the polyline, respectively. Second, the radius of the fitted circle is computed. When the area S and the chord length L are defined as in Figure 4(c), the criterion for circle fitting is that the partial area S is the same as S_{poly} and the chord length L is the same as L_{poly}. Once the radius r of the fitted circle is determined, L_{opt} is obtained for a given threshold e as shown in Figure 4(d). Now we shall describe how the radius r can be estimated efficiently using a look-up table.

The relations of S_{poly} and L_{poly} to the radius r and the sweep angle θ are given by

$$\begin{aligned} L &= L_{poly} = 2r\sin\left(\frac{\theta}{2}\right) \\ S &= S_{poly} = \frac{1}{2}r^2(\theta - \sin\theta) \end{aligned} \tag{6}$$

Figure 5 shows the relationship between S and L by varying θ and r. The solid curves denote the relation of S

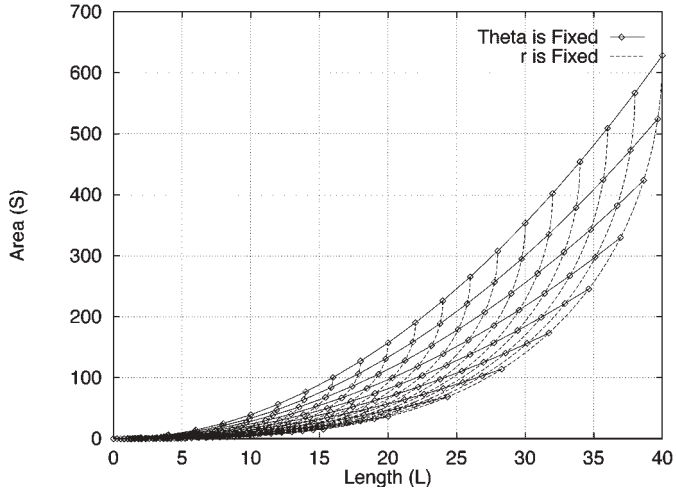


Figure 5. L versus S with r and θ varying.

and L when θ is fixed and r is changing, and the dotted curves denote the relation when r is fixed and θ is changing.

Given S_{poly} and L_{poly}, r could be computed using equation (6). However, since equation (6) is actually a set of non-linear equations, the explicit solution of r, in terms of S_{poly} and L_{poly}, cannot be obtained directly. If θ is fixed to θ_0, after removing the parameter r, the relation of S and L can be represented as

$$f(S, L) = S - kL^2 = 0 \tag{7}$$

where

$$k = \frac{\theta_0 - \sin\theta_0}{8\sin^2(\frac{\theta_0}{2})} \tag{8}$$

Therefore, given S_{poly} and L_{poly}, the θ_{opt} which minimizes $f(S_{poly}, L_{poly})$, can be obtained by a simple numerical search, given by

$$\theta_{opt} = \underset{0<\theta<180}{argmin}\,(S - kL^2)|_{S=S_{poly},\, L=L_{poly}} \tag{9}$$

Finally, r_{opt} is then obtained by using equation (6) and θ_{opt}.

So far, a general solution of equation (6) is shown and it is equivalent to finding the nearest solid and dotted curves in Figure 5 for given L and S. Since it is computationally inefficient to evaluate equation (9) for every polyline, equation (9) can be pre-computed for a range of sampled S_{poly} and L_{poly} values and a look-up table can be constructed. Obviously, look-up table referencing is much faster than the evaluation of equation (9).

In this procedure, L_∞ is used for the error metric. Thus, the maximum distance between the fitting circle and the edge should be maintained less than the given threshold e. By allowing the maximum error below the threshold, the optimal length L_{opt} is obtained from the simple geometric relations shown in Figure 4(d). Thus, we have

$$L_{opt} = 2\sqrt{2er_{opt} - e^2} \tag{10}$$

and the optimality value Ω is given by

$$\Omega = L_{poly} - L_{opt} \tag{11}$$

The computation of L_{opt} is the same for a concave polyline. If the polyline is neither convex nor concave as illustrated in Figure 4(b), the polyline is subdivided into convex subpolylines and the optimal length is computed for each subpolyline. In this case, the optimal length of the edge is the weighted average of the optimal lengths of the polylines.

Stability of Computing Optimal Length

The optimal edge length is defined in terms of surface curvature, but direct computation of the curvature is known to be unstable since the computation of first- and second-order derivatives is highly sensitive to input noise. In our approach, however, the estimation of the optimal edge length does not rely on equation (1). Instead, an efficient method for obtaining curvature information, based on circle fitting, is proposed. It is found from experimental study that this curve-fitting approach provides highly reliable curvature information.

Mesh Simplification

In this section, we describe the iterative simplification algorithm using the measured optimal edge length, discussed in the previous section. The order of simplification and the selection of mesh operation are controlled by the optimality value, Ω in equation (11); the least optimal edge is processed first.

Mesh Operation

In our approach, we use a few basic mesh operations, such as edge collapse, edge swapping, removal of simple vertex star, and removal of vertex diamond, which are illustrated in Figure 6(a–d). In addition, simplification operations for two special configurations are contrived, which are shown in Figure 6(e, f). They are indeed combinations of Figure 6(a–d). By employing them, a few important improvements are achieved. For example, overall simplification is made faster, undesirable mesh-folding is reduced significantly, and manifold mesh structure (i.e., each edge is shared by at most two triangles) is easier to preserve. Note that, although it is possible to check and prevent mesh-folding and non-manifold meshing at run time, it is too expensive to be performed on-the-fly.

In Figure 7, examples of generating undesirable mesh-folding and non-manifold meshing are shown, when simple edge collapse is applied instead of special operations. Mesh-folding invokes several problems. The resultant mesh would be very irregular both in 3-D shape

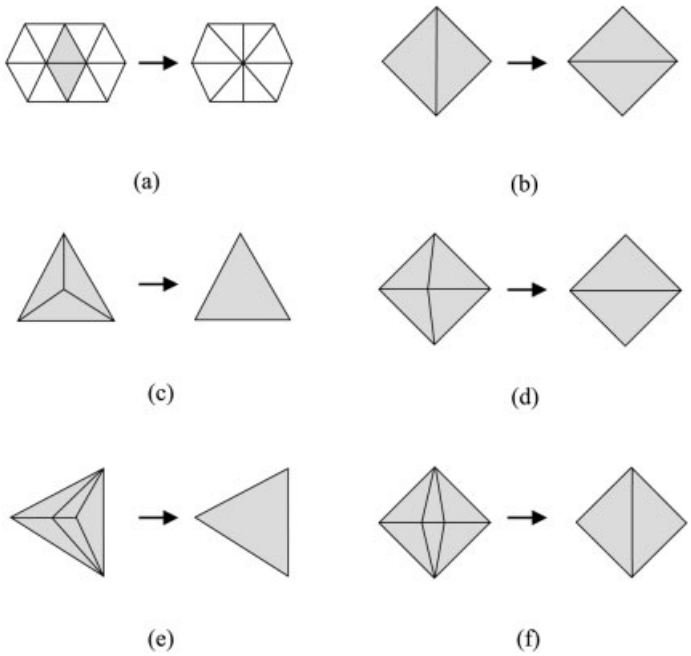


Figure 6. Employed mesh operations. (a) Edge collapse. (b) Edge swapping. (c) Removal of simple vertex star. (d) Removal of vertex diamond. (e, f) Simplification of special configuration.

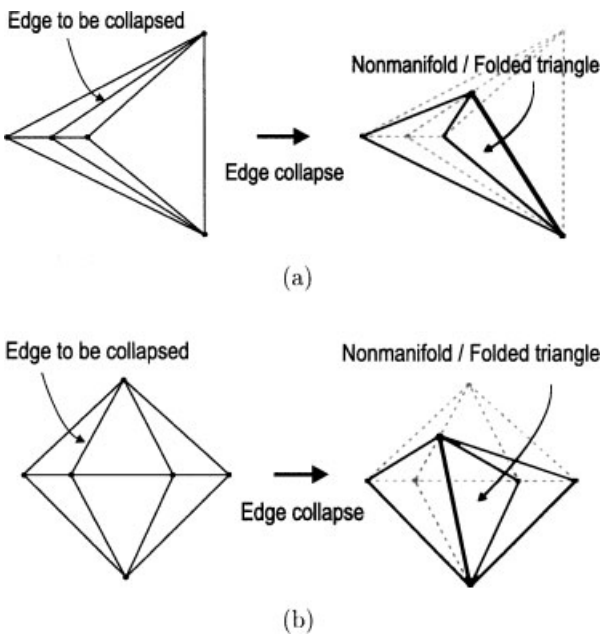


Figure 7. Examples of mesh folding and non-manifold generation.

and rendered image. In addition, it often breaks manifold structure. Preserving the manifold is an important issue for a wide range of applications in geometrical modeling in CAD and computer graphics.

Although the proposed approach utilizes local directional characteristics on the surface, the results presented below show that the simplified mesh yields a regular structure, with a small number of elongated triangles. This is due to the employment of the mesh operations shown in Figure 6(c–f). In our implementation, the number of edges connected to a vertex is limited (explicitly enforced), in order to avoid the extremely elongated triangles and to make the mesh more regular.

The position of the new node induced by the operation shown in Figure 6(a) should be carefully determined, since improper position occasionally causes mesh-folding. In our approach, each edge in the current mesh has its corresponding polyline on the reference mesh. Therefore, the new node is forced to be placed on the polyline, making it possible to locate the new node on the reference mesh. This constraint is very useful for preserving the shape feature during simplification.

In our implementation, the actual position is chosen as the point on the circular approximation of the

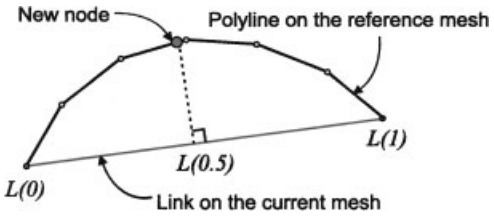


Figure 8. Determining the position of new node.

polyline which has the parameter value 0.5 (centre of the edge), as shown in Figure 8. Indeed, the position is obtained during the calculation of optimality length of the link. Computing optimal length is numerically stable. As an intermediate result of this procedure, calculation of the new node is numerically stable too.

Iterative Edge Simplification

Before iteration, all the edges are initially sorted, depending on the optimality value, and put into a priority queue such that the least optimal edge is located at the first position of the queue. When the length and the direction of an edge are changed during the iteration or a new edge is created, the optimality value is computed and the queue is also updated (locally sorted), according to the optimality value. The mesh operation is applied to the first element of the queue, and the type of mesh operation is determined in such a way as to maximize the optimality gain. The gain is defined as the difference between the average optimality values of the involved edges before and after a specific operation. It may be noted that at each iteration stage the optimal value and the polyline list of edges are always updated once it is involved in the mesh operation. However, it is not necessary to compute the approximation error either before or after an operation or the global approximation error during iteration. Computation of edge optimality and insertion into a priority queue sort are only required during the iteration. Note that sorting in a priority queue can be implemented efficiently using a heap-sorting algorithm.[20] Assuming the total number of edges in the model is N, then the complexity of heap sorting would be $C \cdot O(\log N)$, in which C is the number of edges involved in a certain mesh operation. It does not require additional memory more than the queue size.

In our implementation, we limit the maximum allowable number of edges connected to a node to 12 (which means 30° of average angle between neighbouring edges). With this setting, the maximum value of C is observed during an edge collapse which produces a maximum number of edges. Assuming that the nodes of an edge which is to be collapsed have M and N incident edges, then the resultant number of edges would be $M + N - 4$ by edge collapse as shown in Figure 6(a). Since it should be less than 12, the allowed maximum of $M + N$ is 16 and all of them would be involved in heap sorting. Therefore, the maximum value of C is 16.

Termination Condition

Since iteration is guided by the optimality gain, the mesh operation increases optimality. If the mesh operation tends to decrease optimality, it is rejected. Therefore, it is expected that the optimality converges to its lower bound. Note that an ideal upper bound is zero, implying that the current mesh is perfectly optimal. In our approach, the iteration is terminated if enough edge values fall into a predetermined interval centred at zero (perfect optimality). As will be shown in the experimental results, the convergence of the simplification yields narrow optimality histogram distributions centred at zero.

Experimental Results

In order to evaluate the performance of the proposed algorithm, intensive computer simulations have been carried out on both synthetic and real mesh data. We have tested three synthetic and three scanned models. The experiments for the synthetic models (simple CSG models) are intended to show the sharp-edge preserving properties. As the main experiments, we have tested properly sampled 3-D mesh models obtained by 3-D laser scanning and available in the public domain (http://www.cyberware.com/samples/index.html). In Table 1, the statistical and geometric information of the mesh data is listed. As observed, there exist various types of surfaces with different local topology and complexity, which is desirable for evaluating the proposed algorithm.

First, we obtain the optimality map and the optimality histogram distribution for the Bunny model using different error thresholds, which are shown in Figure 9. Perfectly optimal regions are shown in pure purple colour. Red colour indicates that the current length of the edge is shorter than the optimal length, and therefore it can be simplified. On the other hand, blue colour indicates that the current length of edge is longer than

Experimental data		Statistical feature: Triangles	Edges	Vertices	Geometric feature
Synthetic	Box	12,288	18,432	6,146	Sharp edge and corner
	Tetrahedron	16,384	24,576	8,194	More sharp edge and corner
	Cylinder	20,736	31,104	10,370	Rounded sharp edge
Real	Bunny	69,451	104,288	35,947	Thin ear and hole at bottom
	Teeth	116,602	174,903	58,303	Diverse surface complexity
	Santa	75,778	113,667	37,891	Narrow body part

Table I. Statistical and geometric features of the test models

the optimal length. As expected, given an error tolerance, structurally less complex regions yield red colour regions, which will be simplified first. If we increase the error threshold, then the overall optimality map becomes red. This means that the object can be approximated with fewer triangles. In Figure 9(b) and (d), the optimality deviation histograms are shown for the optimality maps in Figure 9(a) and (c), respectively. The

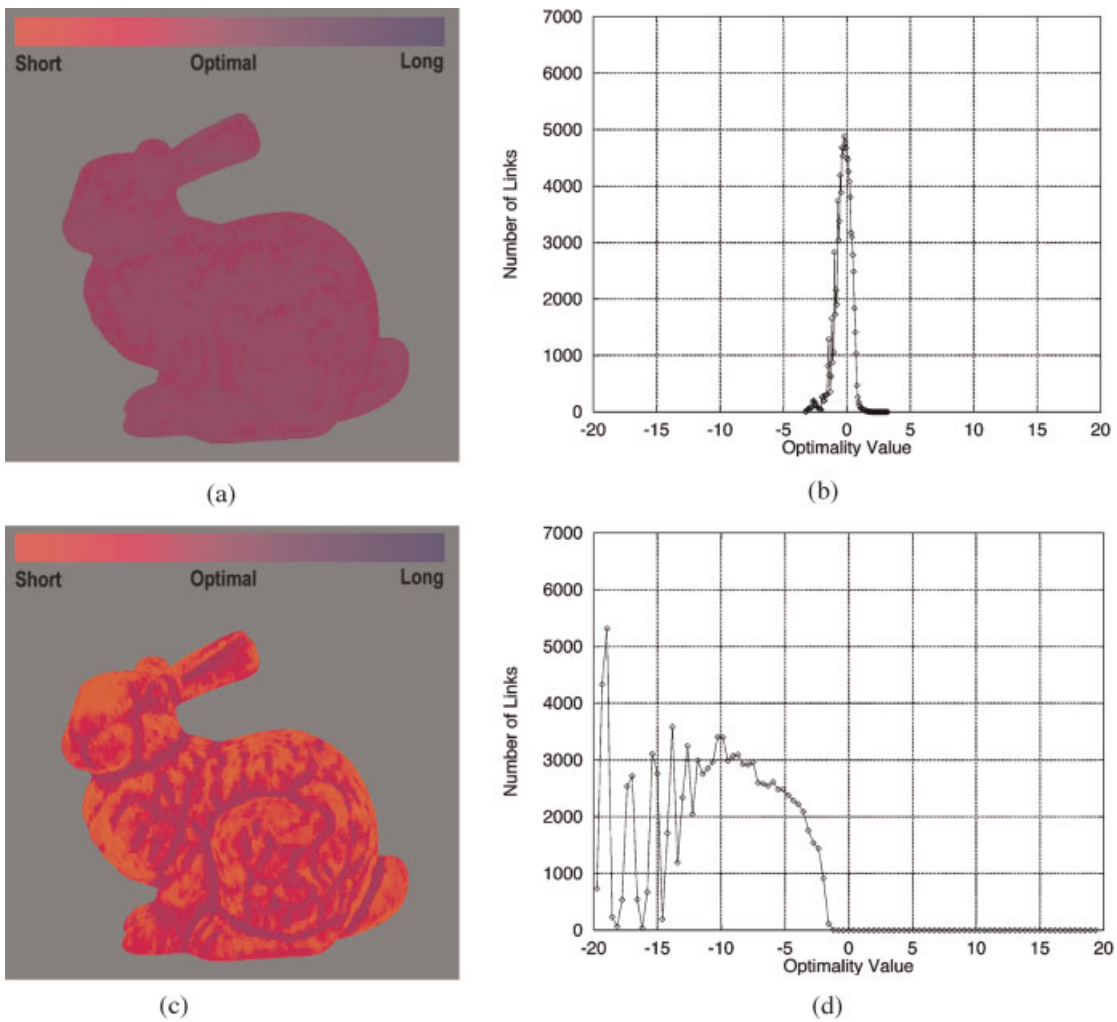


Figure 9. The optimality map and optimality histogram of Bunny model. (a, b) $e = 0.01$. (c, d) $e = 1.0$.

zero value of the x-axis describes the perfectly optimal position. As the error threshold increases, the overall distribution is shifted in the negative non-optimal direction and it spreads more widely.

Based on the optimality measurement, we perform mesh simplification and present the results in Figures 10–13. From the results on synthetic data, shown in Figure 10, it is observed that the sharp edge and corner of the 3-D model are preserved faithfully, because of the shape-adaptive property of the proposed algorithm. Preserving sharp features is an important property for mesh simplification, since visual quality of the simplified mesh is significantly improved if those features are preserved.

For the Bunny model, the shape adaptiveness is clearly observed on the Bunny's eye, ears and other regions, as shown in Figure 11(c) and (e). Note that the obtained mesh exhibits quite a regular structure. As

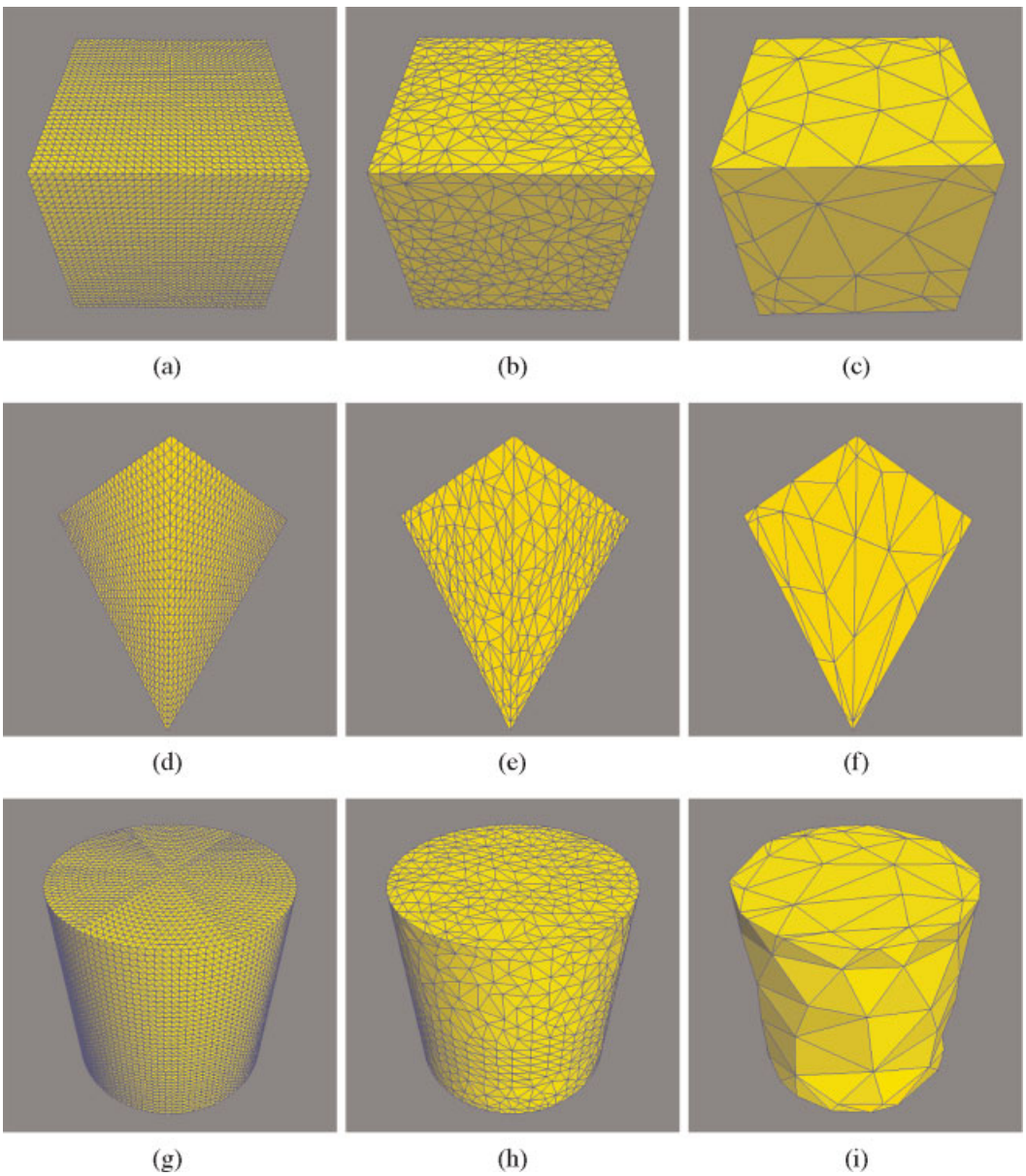


Figure 10. Result of simplification: synthetic mesh. (a, d, g) High-resolution model (original mesh). (b, e, h) Middle-resolution model (20% of the original). (c, f, i) Low-resolution model (1% of the original).

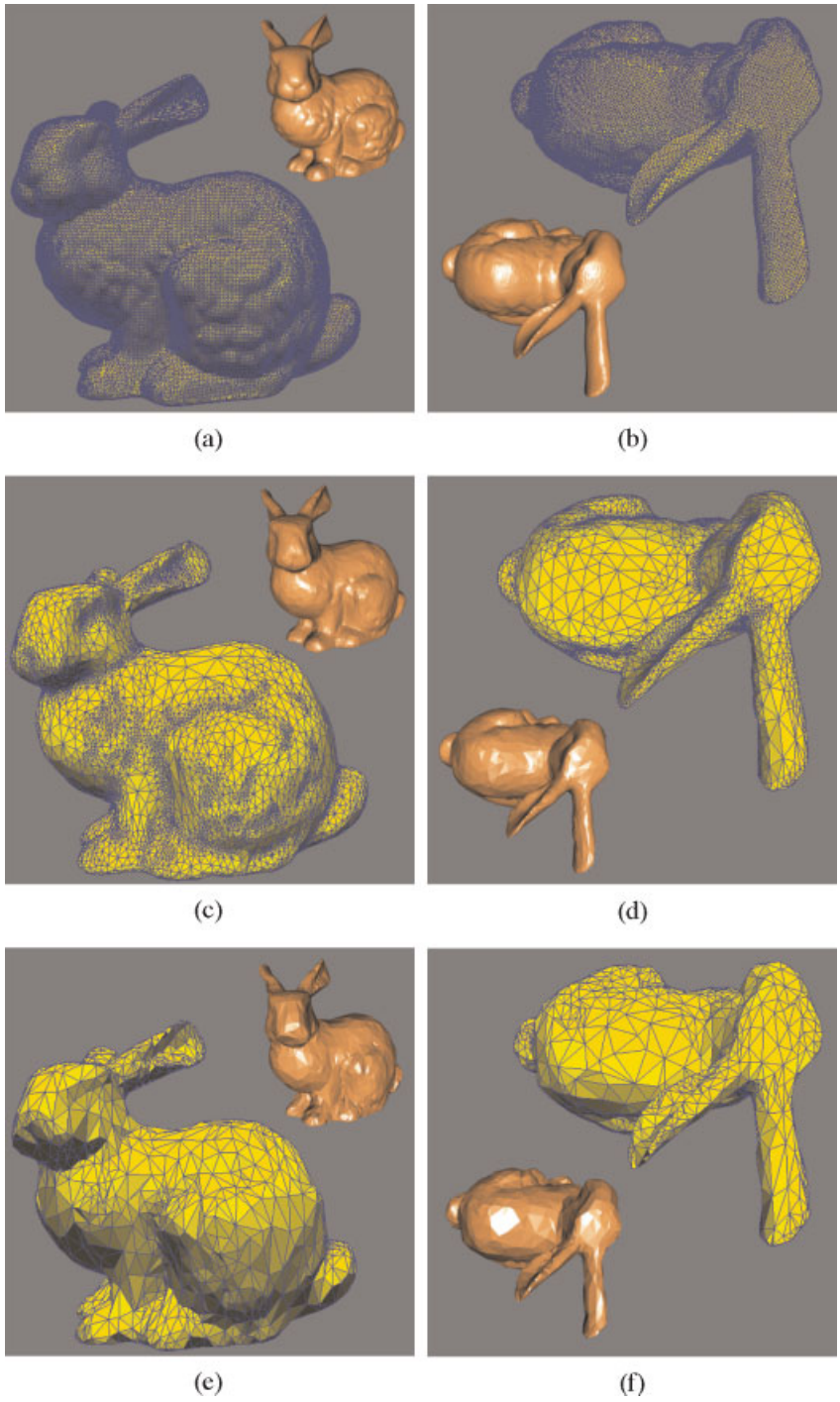


Figure 11. Result of simplification and corresponding optimality histogram: Bunny model with 69K triangles. Flat shaded. (a, b) High-resolution model (original mesh). (c, d) Middle-resolution model (20% of the original). (c, f) Low-resolution model (5% of the original).

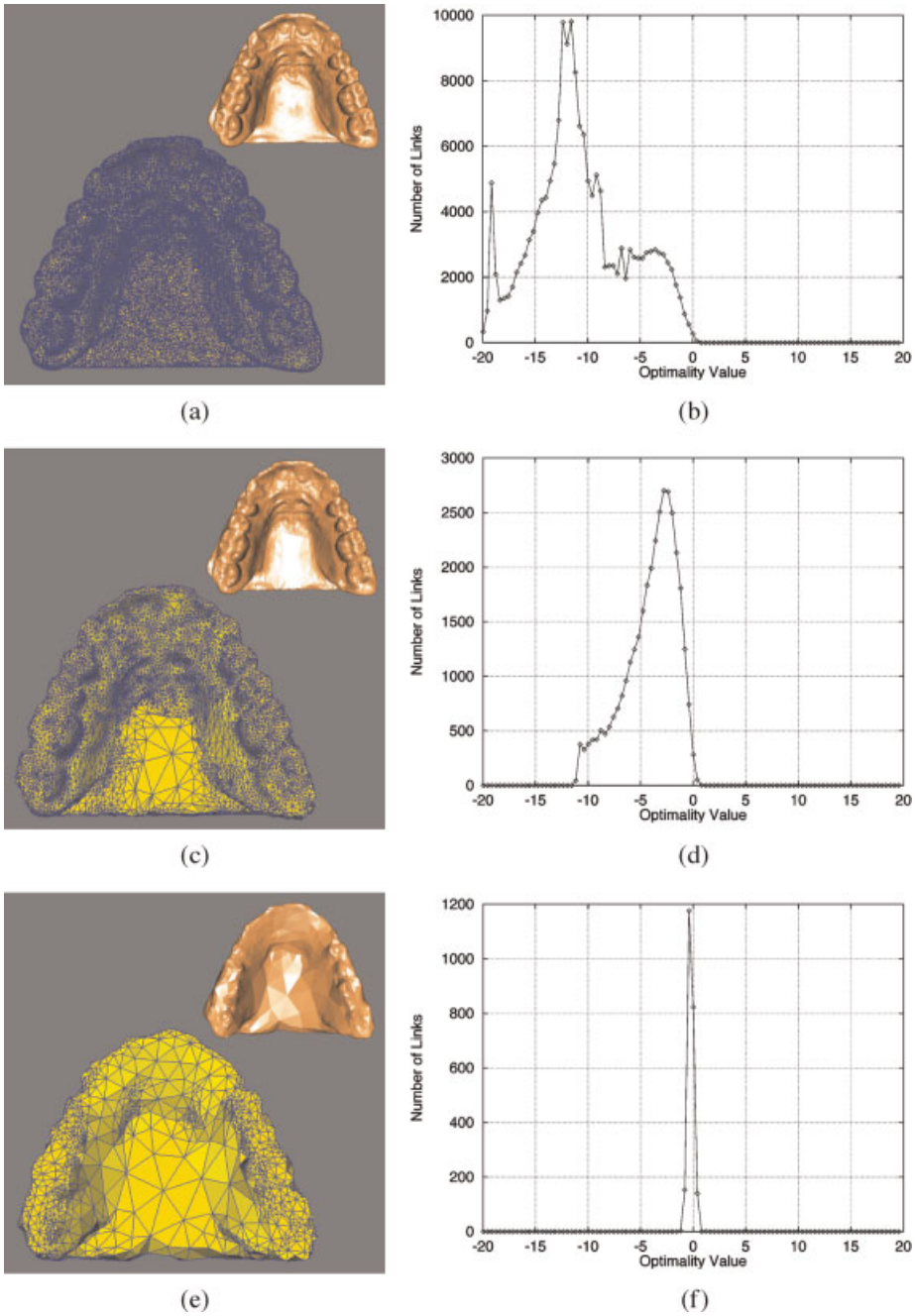


Figure 12. Result of simplification and corresponding optimality histogram: Teeth model with 117K triangles. Flat shaded. (a, b) High-resolution model (original mesh). (c, d) Middle-resolution Model (20% of the original). (e, f) Low-resolution model (2% of the original).

shown in the result at low resolution, the geometric features are also preserved well, yielding good visual quality. Figures 12 and 13 show the simplification results for the Teeth and Santa models, respectively. It is also shown that the mesh is simplified efficiently, without distorting geometrically important features on the model. The convergence of the optimality histograms is also illustrated.

The approximation error and running time of the mesh simplification are shown in Table 2. In Table 2, signed

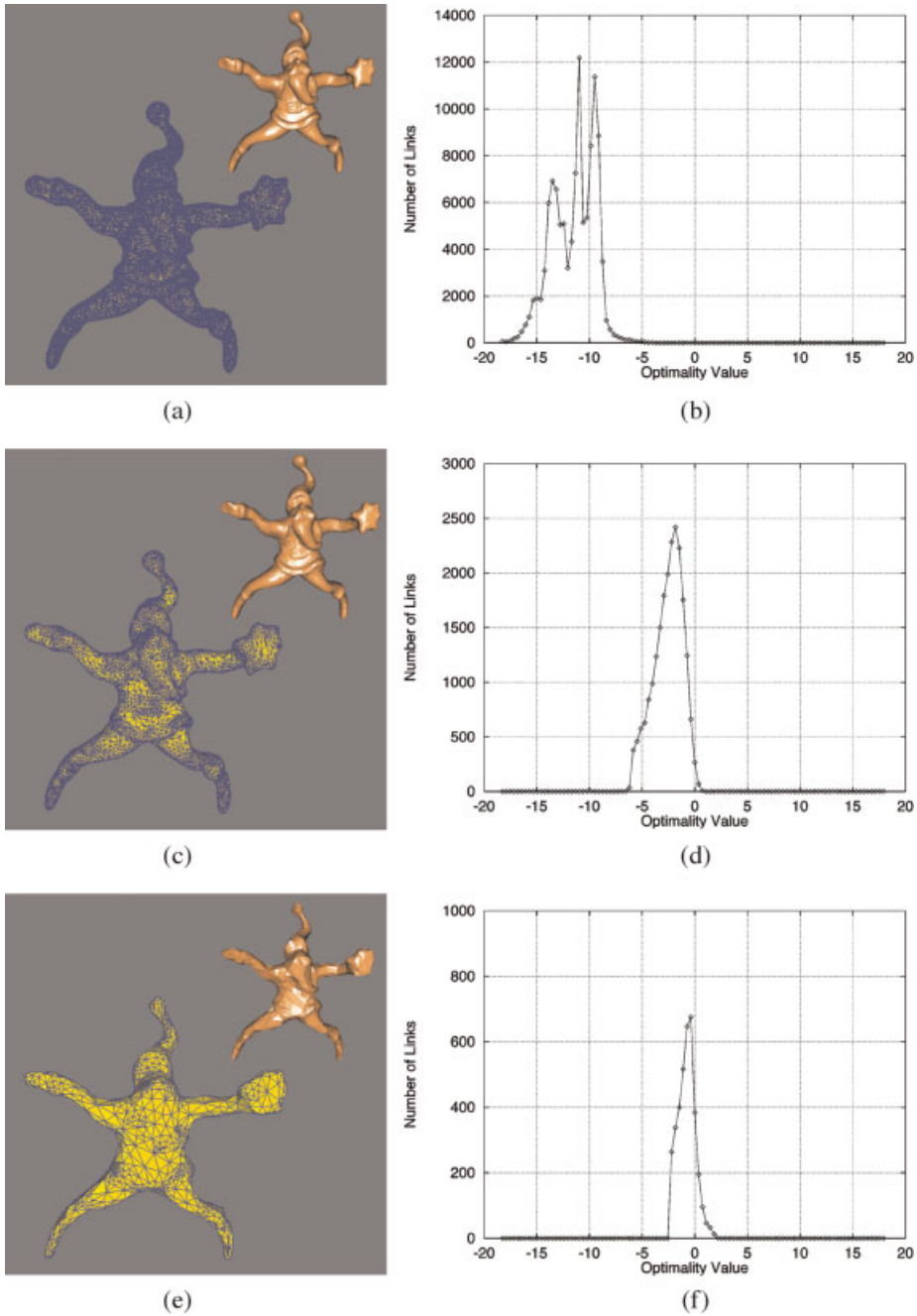


Figure 13. Result of simplification and corresponding optimality histogram: Santa model with 76K triangles. Flat shaded. (a, d) High-resolution model (original mesh). (b, e) Middle-resolution model (19% of the original). (c, f) Low-resolution model (3% of the original).

and unsigned L^1 and unsigned L^∞ metrics are used to calculate the deviation between current mesh and original model. The error is computed for every vertex in the reference mesh and the distance to the corresponding point on the simplified mesh surface is accumulated. The distance would have a positive value when it is located inside the simplified mesh. The error is then normalized to the diagonal length of the bounding box. As shown, the approximation error is quite small (almost negligible) for both the middle and low resolution.

Experimental data		Approximation error			Running time
		Signed mean	Unsigned mean	Unsigned max.	
Bunny	Middle resolution	−0.000345	0.001109	0.062146	611
	Low resolution	−0.000729	0.003105	0.061851	649
Teeth	Middle resolution	0.000575	0.001194	0.076936	921
	Low resolution	−0.001444	0.007309	0.076781	976
Santa	Middle resolution	−0.000063	0.000781	0.062846	718
	Low resolution	−0.001365	0.003185	0.062654	754

Table 2. Approximation error and running time (in seconds)

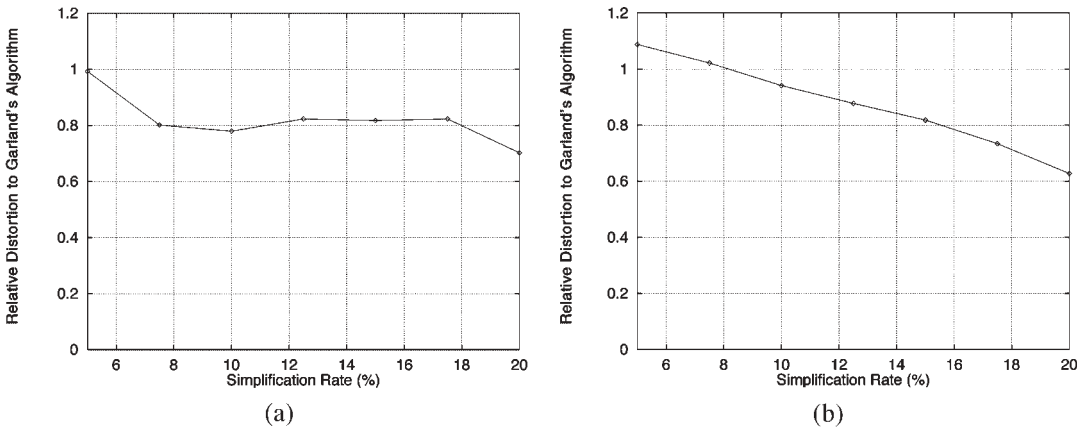


Figure 14. Comparison of relative distortion to reference 6. Middle–low range is examined. (a) Unsigned maximum error. (b) Unsigned mean error.

The simplification results are also compared with the result of the well-known algorithm by Garland and Heckbert.[6] The comparison of approximation error is shown in Figure 14, in which the ratio of error in the middle–low resolution range is plotted. In both cases of the unsigned max and unsigned mean metric, it is observed that the proposed approach produces a much smaller error than reference 6 in most cases. The difference becomes small as it goes to the lower resolution. This tendency is predicted by the characteristics of shape-adaptive meshing. Shape adaptiveness is clearer in the middle resolution, as shown in Figure 11. This confirms that shape-adaptiveness is the main source that reduces the approximation error, compared with the conventional approach. Note that reference 6 follows the shape adaptively to some extent. However, since reference 6 does not consider the directional shape explicitly, the effect is very small, as shown in Figure 15.

Although the proposed approach outperforms reference 6 in accuracy and visual fidelity, reference 6 is faster than the proposed approach. The running time reaches several minutes, as shown in Table 2, while reference 6 achieves 1–2 minutes for the same models. An explanation is that we need to compute the optimality value of the links which are involved in a specific mesh operation during each iteration. Although it is not computationally heavy, it is still more expensive than simple matrix computation proposed in reference 6. Note that reference 6 is one of the fastest among existing mesh simplification algorithms.

On the other hand, comparison in terms of the optimality histogram is shown in Figure 16. The final distribution of the optimality histogram is quite narrow for the proposed algorithm, while that of reference 6 is widely spread. This observation is also expected, since reference 6 does not consider the direction of local shape variation. Notice that the proposed algorithm provides a useful way to control the mesh density to a predefined level. We can set the target optimality value to be different from zero, and obtain a non-optimally simplified mesh with specified optimality margin.

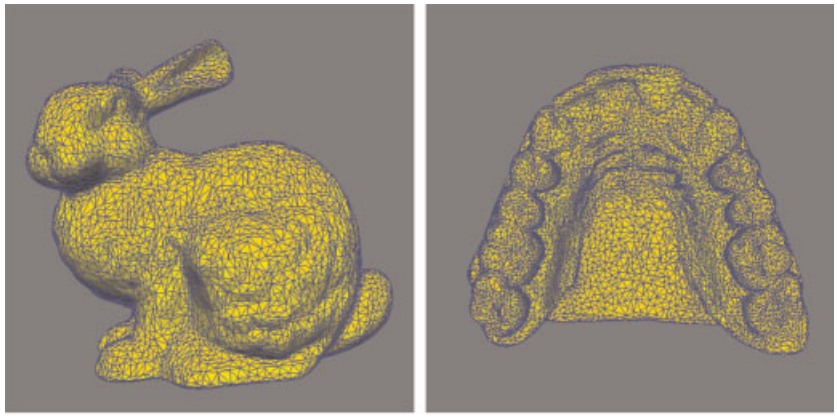


Figure 15. Result of Garland's simplification (20% of the original). (a) Bunny model. (b) Teeth model.

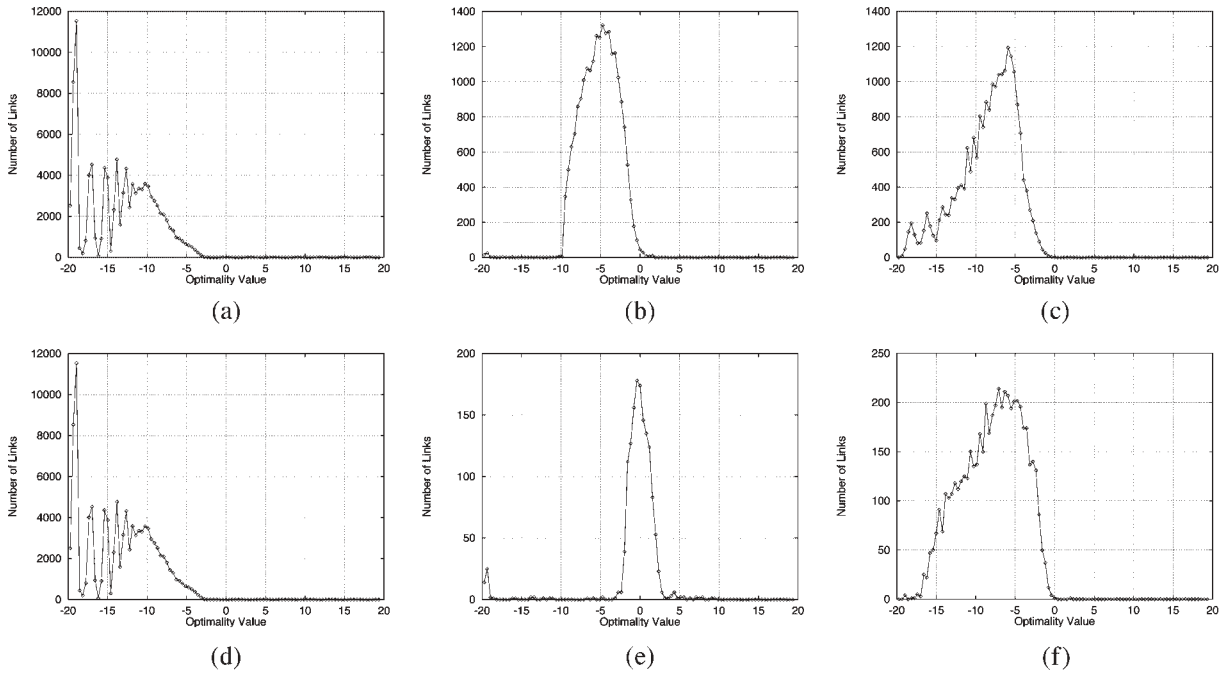


Figure 16. Comparison of optimality histogram with Garland's simplification result on Bunny model. (a) High-resolution model (original mesh). (b) Optimality histogram of middle-resolution model (proposed algorithm, 20% of the original). (c) Optimality histogram of middle-resolution model (Garland and Heckbert's algorithm, 20% of the original). (d) High-resolution model (original mesh). (e) Optimality histogram of low-resolution model (proposed algorithm, 5% of the original). (f) Optimality histogram of low-resolution model (Garland and Heckbert's algorithm, 5% of the original).

Conclusion

In this paper, an algorithm for adaptively simplifying 3-D mesh modelling was proposed, based on local edge optimality. The proposed approach utilizes the local directional curvature to compute the optimal length of a given point on the surface. Several mathematical formulations were developed for computing the optimal length from a curve, a cubic spline approximation of polyline and a circular approximation of polyline. For the circular approximation, we proposed an efficient

computation method employing a circular fitting and look-up table. In the experimental test, optimality maps were obtained for several error thresholds. The optimality map can be used to achieve shape-adaptive multiresolution modelling. Mesh simplification results demonstrated that the object details are preserved faithfully for a given error bound or a given number of triangles, since the triangle size can be controlled adaptively to the local shape.

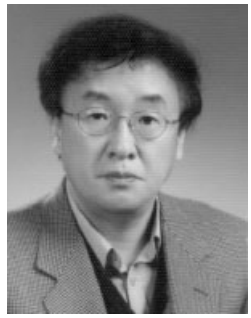
References

1. Schroeder W, Zarge J, Lorensen W. Decimation of triangular meshes. In *Proceedings of SIGGRAPH '92*, July 1996; pp 65–70.
2. Hoppe H, DeRose T, Duchamp T, McDonald J, Stuetzle W. Mesh optimization. In *Proceedings of SIGGRAPH '93*, August 1993; 19–26.
3. Eck M, DeRose T, Duchamp T, Hoppe H, Lounsbery M, Stuetzle W. Multiresolution analysis of arbitrary meshes. In *Proceedings of SIGGRAPH '95*, August 1995; 173–182.
4. Hoppe H. Progressive meshes. In *Proceedings of SIGGRAPH '96*, August 1996; 99–108.
5. Cohen J, Varshney A, Manocha D, Turk G, Weber H, Agarwal P, Brooks F, Wright W. Simplification envelopes. In *Proceedings of SIGGRAPH '96*, August 1996; 119–128.
6. Garland M, Heckbert P. Surface simplification using quadric error metrics. In *Proceedings of SIGGRAPH '97*, August 1997; 209–216.
7. Johnson AE, Hebert M. Control of polygonal mesh resolution for 3-D computer vision. *CVGIP: Graphical Models and Image Processing* 1998; **60**(4): 261–285.
8. Kobbelt L, Campagna S, Vorsatz J, Seidel H. Interactive multi-resolution modeling on arbitrary meshes. In *Proceedings of SIGGRAPH '98*, July 1998; 105–114.
9. Garland M, Heckbert P. Simplifying surfaces with color and texture using quadric error metrics. In *Proceedings of IEEE Conference on Visualization*, October 1998; 264–270.
10. Lindstrom P, Turk G. Evaluation of memoryless simplification. *IEEE Transactions on Visualization and Computer Graphics* 1999; **5**(2): 98–115.
11. Guéziec A. Locally toleranced surface simplification. *IEEE Transactions on Visualization and Computer Graphics* 1999; **5**(2): 168–189.
12. Guskov I, Sweldens W, Schröder P. Multiresolution signal processing for meshes. In *Proceedings of SIGGRAPH '99*, August 1999; 325–334.
13. Hoppe H. New quadric metric for simplifying meshes with appearance attributes. In *Proceedings of IEEE Conference on Visualization*, October 1999; 59–66.
14. Besl PJ, Jain RC. Segmentation through variable-order surface fitting. *IEEE Transactions on Pattern Analysis and Machine Intelligence* 1988; **10**(2): 167–192.
15. Hamann B. Curvature approximation for triangulated surfaces. *Computing* 1993; Suppl 8: 139–153.
16. Taubin G. Estimating the tensor of curvature of a surface from a polyhedral approximation. In *Proceedings of the IEEE International Conference on Computer Vision*, June 1995; 902–907.
17. Heckbert P, Garland M. Optimal triangulation and quadric-based surface simplification. *Computational Geometry* 1999; **14**(1–3): 49–65.
18. Mokhtarian F, Khalili N, Yuen P. Curvature computation on free-form 3-D meshes at multiple scales. *CVGIP: Image Understanding* 2001; **83**(2): 118–139.
19. Salomon D. *Computer Graphics and Geometric Modeling*. Springer: Berlin, 1999.
20. Weiss M. *Data Structures and Algorithm Analysis in C++* (2nd edn). Addison-Wesley: Reading, MA, 1999.
21. Cignoni R, Montani C, Scopigno R. A comparizon of mesh simplification algorithms. *Computers & Graphics* 1998; **22**(1): 37–54.

Authors' biographies:

In Kyu Park received the BS, MS, and PhD degrees from Seoul National University, Seoul, Korea in 1995, 1997, and 2001, respectively, all in electrical engineering and computer science. In September 2001, he joined the Samsung Advanced Institute of Technology, Yongin, Korea as a member of technical staff, where he has been involved in MPEG standardization activities. Dr Park's research interests include the joint area of 3-D computer vision, computer graphics, and multimedia application, especially 3-D shape reconstruction, image-based modeling and rendering, and multimedia database indexing/retrieval.

Sang Wook Lee received the BS degree in electronic engineering from Seoul National University, Seoul, Korea, in 1981, the MS degree in electrical engineering from Korea Advanced Institute of Science and Technology (KAIST), Seoul, Korea, in 1983, and the PhD degree in electrical engineering from the University of Pennsylvania in 1991. He is currently an associate professor of media technology at Sogang University, Seoul, Korea. He was an assistant professor in computer science and engineering at the University of Michigan (1994–2000), a postdoctoral research fellow and a research associate in computer and information Science at the University of Pennsylvania (1991–1994), a researcher at KAIST (1985–1986), a research associate at Columbia University (1984–1985), and a researcher at LG Telecommunication Research Institute (1983–1984). Dr Lee's major field of interest is computer vision and graphics, with an emphasis on physics-based vision, colour indexing and recognition, inverse rendering, 3-D modeling from range sensing. He received the outstanding paper award at European Conference in Computer Vision, Margherita Ligure, Italy, 1992, and was a program committee co-chair of the IEEE Workshop on Physics-Based Modeling in Computer Vision, June, Cambridge, Massachusetts, 1995, and the IEEE Workshop on Photometric Modeling for Computer Vision and Graphics, June, Fort Collins, Colorado, 1999.



Sang Uk Lee received the BS degree from Seoul National University, Seoul, Korea, in 1973, the MS degree from Iowa State University, Ames in 1976, and PhD degree from the University of Southern California, Los Angeles, in 1980, all in electrical engineering. From 1980 to 1981, he was with the General Electric Company, Lynchburg, VA, working on the development of digital mobile radio. From 1981 to 1983, he was a Member of Technical Staff, M/A-COM Research Center, Rockvill, MD. In 1983, he joined the Department of Control and Instrumentation Engineering at Seoul National University as an Assistant Professor, where he is now a Professor of the School of Electrical Engineering and Computer Science. Currently, he is also affiliated with the Automation and Systems Research Institute and the Institute of New Media and Communications at Seoul National University. His current research interests are in the areas of image and video signal processing, digital communication, and computer vision. He served as an Editor-in-Chief for the Transaction of the Korean Institute of Communication Science from 1994 to 1996. Dr Lee is currently a Member of the Editorial Board of the Journal of Visual Communication and Image Representation and an Associate Editor for IEEE Transactions on Circuits and Systems for Video Technology. He is a member of Phi Kappa Phi.