Constructing Hierarchies for Triangle Meshes

Tran S. Gieng, Bernd Hamann, *Member, IEEE*, Kenneth I. Joy, *Member, IEEE*, Gregory L. Schussman, *Member, IEEE*, and Issac J. Trotts

Abstract—We present a method to produce a hierarchy of triangle meshes that can be used to blend different levels of detail in a smooth fashion. The algorithm produces a sequence of meshes $\mathcal{M}_0, \mathcal{M}_1, \mathcal{M}_2, ..., \mathcal{M}_n$, where each mesh $\mathcal{M}_i$ can be transformed to mesh $\mathcal{M}_{i+1}$ through a set of triangle-collapse operations. For each triangle, a function is generated that approximates the underlying surface in the area of the triangle, and this function serves as a basis for assigning a weight to the triangle in the ordering operation and for supplying the points to which the triangles are collapsed. The algorithm produces a limited number of intermediate meshes by selecting, at each step, a number of triangles that can be collapsed simultaneously. This technique allows us to view a triangulated surface model at varying levels of detail while insuring that the simplified mesh approximates the original surface well.

Index Terms—Mesh simplification, triangle meshes, level-of-detail representation, shape approximation, multiresolution.

1 Introduction

The rapid increase in the power of computer systems, coupled with the increasing sophistication of geometric-modeling operations, the increasing precision of computational simulations, and the development of state-of-the-art imaging systems, can now produce models with incredible detail. The most critical and fundamental research problem encountered in the visualization of these complex models is the development of methods for storing, approximating, and rendering very large data sets. The problem is to develop different representations of a data set, each of which can be substituted for the complete set depending on the requirements of the visualization technique. The data set may be represented by a few points or by several million points if necessary, with each of the data sets containing the essential features of the original data. A hierarchical or *multiresolution* representation [1], [2], [3], [4] allows the study of large-scale features by considering the data set at a coarse resolution and the study of small-scale features by considering the data set at a fine resolution.

Using a multiresolution data representation allows near real-time "traveling" through the data domain. The rendering algorithm can use a coarse resolution for image generation while the model moves, and can add detail over time by "adding" finer levels when the model is stationary—the progressive refinement approach to image generation. The rendering algorithm can also simultaneously display various levels of detail from the representation, changing among the levels "continuously." This level-of-detail problem is of great importance in visualization and computer graphics. Multiresolution techniques are useful for real-time data access to support telepresence and teleoperations, and to enhance accessibility of the data sets in real-time decision processes.

We introduce a method to produce a hierarchical representation of large, unstructured triangle meshes. Given an initial mesh $\mathcal{M}_0$, our algorithm reduces the number of triangles through a series of triangle-collapse operations. A triangle is selected from the mesh and removed by collapsing it to a point (see Fig. 1). A weight is assigned to each triangle and is used as the criterion to select triangles to be collapsed. This weight is partially based on a curvature measure determined by the principal curvatures of a function that approximates the surface in the area of each triangle. To insure that the new mesh accurately approximates the underlying surface, we use the approximating surface to supply the point to which the triangle is collapsed. This enables us to develop a simple algorithm, based upon a triangle-collapse strategy, which remains faithful to the underlying surface.

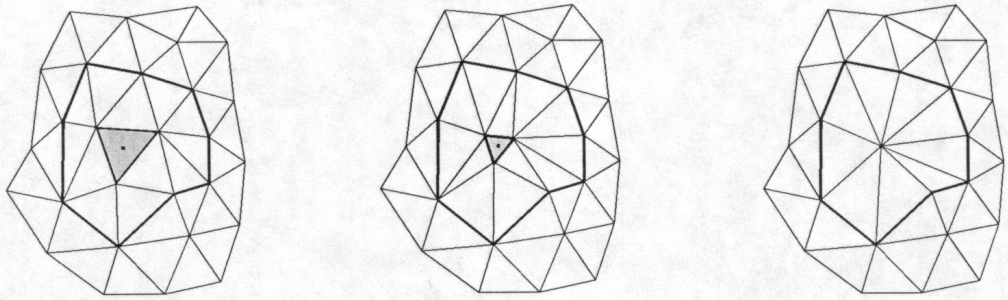
In a given mesh, we can identify a number of triangles that can be collapsed simultaneously, and this allows our algorithm to output a sequence of meshes $\mathcal{M}_0, \mathcal{M}_1, \mathcal{M}_2, ..., \mathcal{M}_n$ with the property that $\mathcal{M}_i$ can be smoothly collapsed to $\mathcal{M}_{i+1}$. By collapsing a relatively large number of triangles in an intermediate triangulation simultaneously, we achieve significant memory savings. Thus, the transition from mesh $\mathcal{M}_i$ to $\mathcal{M}_{i+1}$ is characterized by collapsing many triangles in parallel—instead of collapsing just one. The sequence of meshes, along with the triangle-collapse operations, can be used to create a smooth visual transition between levels in the hierarchy.

In Section 2, we discuss related work. We examine the topology of the triangle-collapse operation in Section 3. We define what it means for a triangle to be "collapsible," and exhibit the effect of the triangle-collapse operation on the mesh. In Section 4, we construct a function that approximates the underlying surface in the area of a triangle. This approximating surface is used in two ways:

1) to define the point to which a triangle will collapse, and
2) to assign weights to the triangles.

- *T.S. Gieng is with the Department of Computer Science, California Institute of Technology, Pasadena, CA 91125. E-mail: gieng@cs.caltech.edu.*
- *B. Hamann, K.I. Joy, G.L. Schussman, and I.J. Trotts are with the Department of Computer Science, University of California, Davis, CA 95616-8562 and with the Center for Image Processing and Integrated Computing (CIPIC), University of California, Davis., CA 95616-8553. E-mail: {hamann, joy, schussman, trotts}@cs.ucdavis.edu.*

For information on obtaining reprints of this article, please send e-mail to: tvcg@computer.org, and reference IEEECS Log Number 106597.



(a) (b) (c)

Fig. 1. Collapsing a triangle: The shaded triangle is selected from the mesh in (a), collapsed toward the centroid of the triangle in (b), creating a new mesh in (c) which has four fewer triangles than the original mesh.

The algorithm assigns a weight to each triangle which determines its priority in the mesh reduction process. In Section 5, we give a complete description of the algorithm which generates a sequence of triangle-collapse operations and a sequence of meshes. The calculation of the factors that make up the weights is discussed in Section 6. Results of the algorithm's use are given in Section 7.

2 Related Work

Algorithms that simplify meshes are an important component of research in visualization and computer graphics. State-of-the-art graphics workstations, which now allow the display of many thousands of shaded or textured polygons at interactive frame rates, are not sufficient to render the truly massive data sets containing millions of triangles. Rendering systems today require real-time level-of-detail representations to adaptively display the models at constant frame rates, or to continuously vary the detail on the model depending on the camera position.

Three classes of algorithms exist that directly pertain to our work and that deal with the triangle meshes directly: algorithms that simplify the mesh by removing vertices; algorithms that simplify the mesh by removing edges; and algorithms that simplify the mesh by removing faces.

Schroeder et al. [5] have developed an algorithm that simplifies the mesh by removing vertices. Vertices are identified through a distance-to-plane criterion, where an average plane is formed through a vertex and its adjacent vertices. If the vertex is within a specified distance of the average plane, it can be deleted; otherwise, it is retained. Removing a vertex from the mesh creates a hole that must be retriangulated, and several strategies may be used. Schroeder et al. use a recursive loop splitting procedure to generate a triangulation of the hole.

Renze and Oliver [6] have published a similar algorithm which is extended to meshes containing n-dimensional simplices. Rather than a loop splitting procedure, they fill the hole by using an unconstrained Delaunay triangulation algorithm.

Hoppe [7], [8] and Popovic and Hoppe [9] describe a progressive-mesh representation of a triangle mesh. This is a continuous-resolution representation based upon an edge-collapse operation which selects and collapses individual edges of the mesh. Hoppe et al. [10] formulate the data reduction problem in terms of a mesh optimization problem, ordering the edges according to an energy minimization function. Each edge is placed in a priority queue by the expected energy cost of the edge collapse. As edges are collapsed, the priorities of the edges in the neighborhood of the transformation are recomputed and reinserted into the queue. The result is an initial coarse representation of the mesh, and a linear list of edge-collapse operations, each of which can be regenerated to generate finer representations of the mesh. The geometrically continuous edge-collapse operation allows the development of a smooth visual transition between various levels of the representation.

Other edge-collapse algorithms have been described by Zia and Varshney [11], who use the constructed hierarchy for view-dependent simplification of models, and Garland and Heckbert [12], who utilize quadratic error metrics for fast calculation of the hierarchy.

Hamann [13] has developed an algorithm that attempts to simplify the mesh by removing triangles. His algorithm removes triangles from a mesh by first ordering the triangles according to the curvature at their vertices (see [14]). The curvature values are precomputed based on the original triangulated mesh. Triangles are then inserted into a priority queue and removed iteratively. Modified triangles receive new curvatures at their vertices and are inserted back into the priority queue. The user can specify a percentage of triangles to be removed or an error tolerance.

Each of these algorithms focuses on an individual vertex, edge, or triangle in the mesh. Our algorithm has a similar focus, but creates a hierarchy of meshes, not just a list of elements. The hierarchy of meshes can be used to create a continuous-reduction algorithm that enables us to smoothly vary the detail over the set of meshes.

Others have used different approaches to address these problems. Turk [15] uses a retiling strategy to place new points on the original mesh. These points are then retriangulated, resulting in a new coarser mesh that represents the original data set. Cohen et al. [16] use a generalization of offset surfaces to produce a hierarchy of inner and outer meshes that surround the original mesh. Lindstrom et al. [17] address the case of triangulations in the form of uniformly gridded height fields. With this specialized structure, it is possible to segment the grid into a quad-tree

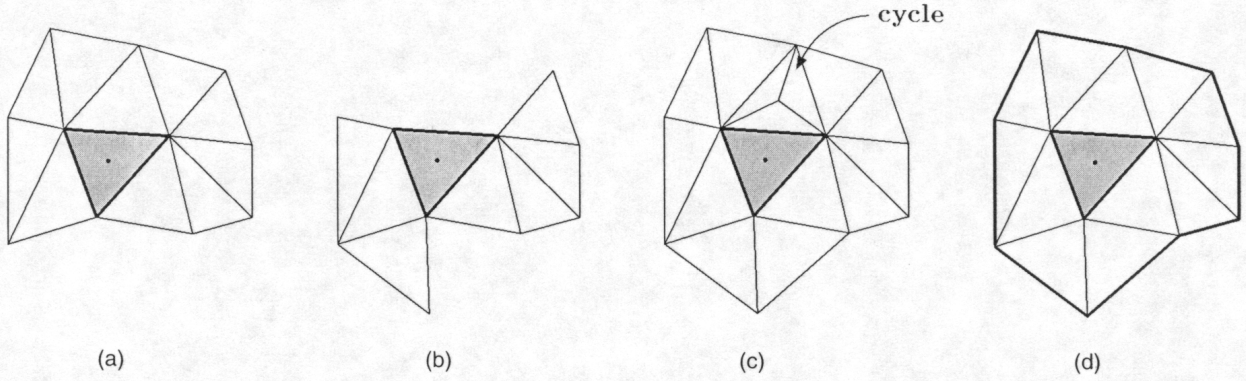


Fig. 2. Stencils of triangles. (a) The shaded triangle has a connected acyclic stencil. (b) The shaded triangle has a disconnected acyclic stencil. (c) The shaded triangle has a connected cyclic stencil. (d) The shaded triangle has a complete stencil. In the last case, the stencil boundary polygon is outlined in bold. We note that the boundary polygon of the triangle in (a) contains a vertex of the triangle, and therefore the stencil is not complete.

representation and then provide triangle reduction within the quad-tree nodes.

Our algorithm is based on a triangle-removal strategy that creates a hierarchy of meshes, not just a hierarchy of triangles. These meshes can be used to perform a continuous level-of-detail rendering which enables us to smoothly blend the various levels. This paper is an expansion of our work in [18]. We now provide full details of the topology considerations in the triangle-collapse operation, present an algorithm to handle meshes with boundaries, and describe additional factors used to order the triangles for collapse.

3 Triangle-Collapse Operations

In the rendering context, we define a surface to be a piecewise linear surface defined by a mesh of triangles. We require that the triangle mesh be connected and that each edge in the mesh be shared by at most two triangles. Meshes should not be self-intersecting—that is, no triangle of the mesh should have an intersection with the interior of another triangle.

Our objective is to collapse triangles in the mesh. If we examine Fig. 1, we see that collapsing a triangle affects other triangles in the area. These affected triangles, defining the *stencil*, are modified in the triangle-collapse operation.

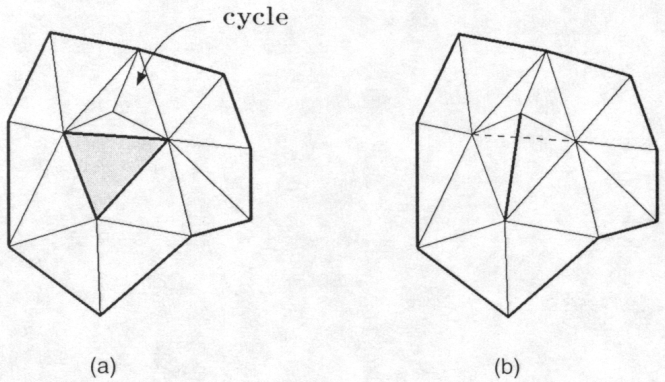


Fig. 3. Edge swapping to remove cycles in the stencil. (a) The selected triangle contains a cycle in the stencil. (b) The cycle is removed by swapping the common edge between the triangle and the neighboring triangle that belongs to the cycle.

To determine if a triangle is "collapsible," we must examine the stencil and its effect under the collapse operation.

3.1 Stencils of Triangles

If we are to collapse a given triangle T, it is the triangles surrounding T that influence the resulting mesh after the collapse. This set of triangles, the *stencil* S_T of T, is the set of triangles T_i, where $T_i \neq T$ and T_i shares a vertex with T (see Fig. 1: The stencil of the collapsing triangle is outlined in bold). The three triangles of the stencil that share an edge with T, the *neighbors* of T, are eliminated in the collapse operation, and the remaining triangles of the stencil are modified, stretching them to a common vertex. The stencil is called *connected* if, for each pair of triangles T_{i_1} and T_{i_k} in the stencil, a sequence of triangles $T_{i_2}, T_{i_3}, \ldots, T_{i_{k-1}}$ exist in the stencil such that T_{i_j} and $T_{i_{j+1}}$ are neighbors[1] for $j = 1, \ldots, k - 1$. Fig. 2 shows examples of connected and disconnected stencils.

Three triangles T_1, T_2, and T_3 form a *cycle* in the mesh if they are pairwise neighbors (see Fig. 2c). Each triangle of a cycle must have a vertex of valence three. A stencil S is called *cyclic* if it contains a cycle, otherwise, it is called *acyclic*. To be in the stencil, a cycle must contain a neighbor triangle of T. Cycles can be eliminated in the original mesh by edge swapping (see Fig. 3). Repeated swapping may be necessary to eliminate the three cycles that could possibly occur in the stencil.

If a triangle T has a connected acyclic stencil S_T, we can order the triangles of the stencil and obtain a polygon, the *stencil boundary polygon*, that describes the outer boundary of the stencil. If this polygon contains no vertices of the original triangle T, the stencil is called *complete*. Examples of various triangles and their stencils are shown in Fig. 2.

We can ensure that complete stencils are defined on the boundary by adding a single vertex, a point-at-infinity, to the mesh. This point, $\mathbf{p}_\infty$, is then connected to each vertex on the boundary of the mesh (see Fig. 4).

1. A neighboring triangle of T shares an edge with T.

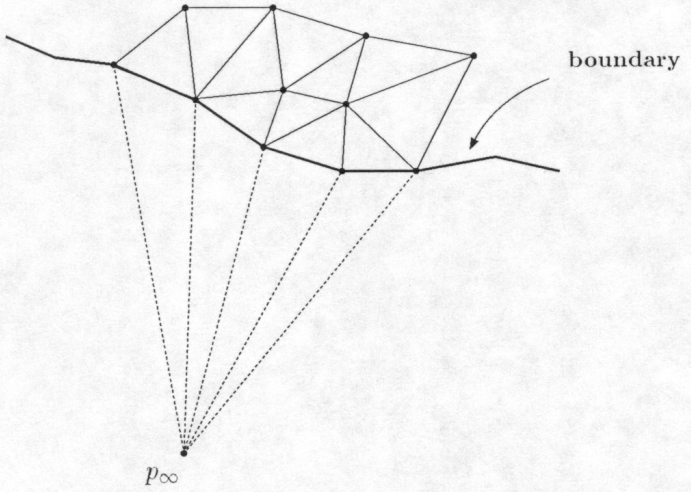


Fig. 4. A single point-at-infinity is added to the mesh and connected to each vertex on the boundary. This enables us to associate complete stencils to triangles on the boundary.

3.2 Collapsing a Triangle

Collapsing a triangle and removing it from the mesh affects both the triangle itself and the triangles of its stencil. As can be seen in Fig. 5, when a triangle is collapsed, the triangle and its neighbors are eliminated from the mesh. The triangle is replaced by a single point, which is connected to all points of the stencil boundary polygon, creating a new triangulation of the region. Geometrically, this transition is smooth; topologically, it is "discontinuous" when the three vertices eventually become one.

To make this specific, consider a triangle T and let its neighboring triangles be denoted by T_{N_1}, T_{N_2}, and T_{N_3}. For each i, let $\mathbf{p}_{N_i}$ be the vertex of T_{N_i} that is not a vertex of T (see Fig. 6). If $\mathbf{p}_1$, $\mathbf{p}_2$, and $\mathbf{p}_3$ are the vertices of the triangle, and if we denote $\mathrm{val}(\mathbf{p})$ to be the valence[2] of the point $\mathbf{p}$ in the triangulation, then after the triangle T is collapsed, the following holds:

- We have removed four triangles from the mesh—T, T_{N_1}, T_{N_2}, and T_{N_3}.
- From the n triangles in the stencil, the area is now retriangulated with $n-4$ triangles. These triangles all share a new vertex $\mathbf{c}$, which has valence equal to

$$\mathrm{val}(\mathbf{p}_1) + \mathrm{val}(\mathbf{p}_2) + \mathrm{val}(\mathbf{p}_3) - 9.$$

- Each point $\mathbf{p}_{N_i}$ is a vertex of one fewer triangle: Its valence is reduced by one.
- The Euler-Poincaré characteristic [19] is preserved. Consider a triangle T whose stencil contains v vertices, e edges, and t triangles. After the triangle is collapsed, the resulting set of triangles will have $v-2$ vertices, $e-6$ edges, and $t-4$ triangles. The Euler-Poincaré characteristic of both sets of triangles is the same:

2. The number of edges that emanate from a vertex define its valence.

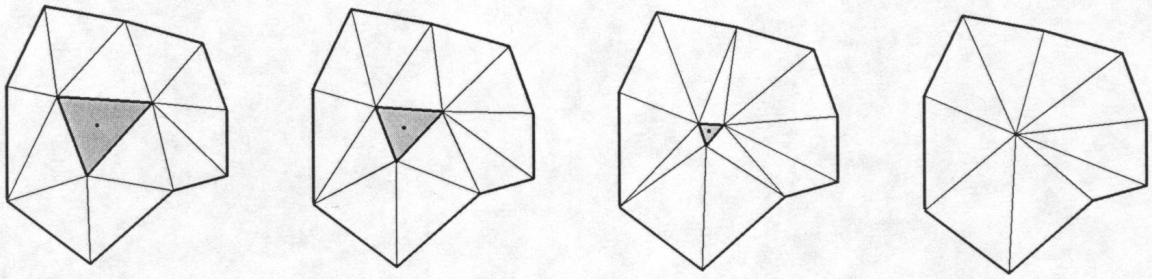


Fig. 5. A triangle and its stencil undergoing the collapse operation. As the triangle is collapsed, a new mesh is created containing four fewer triangles. Geometrically, this transition is smooth; topologically, it is "discontinuous."

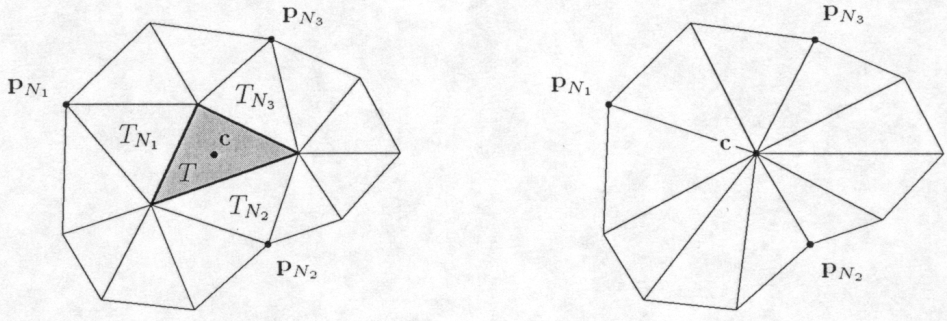


Fig. 6. Topology of the triangulation before the collapse (a); and after the collapse (b). In the process, the triangles T, T_{N_1}, T_{N_2}, and T_{N_3} are eliminated from the triangulation. The new triangulation has a vertex at $\mathbf{c}$, and the valence of each of the vertices $\mathbf{p}_{N_1}, \mathbf{p}_{N_2}$, and $\mathbf{p}_{N_3}$ is reduced by one.

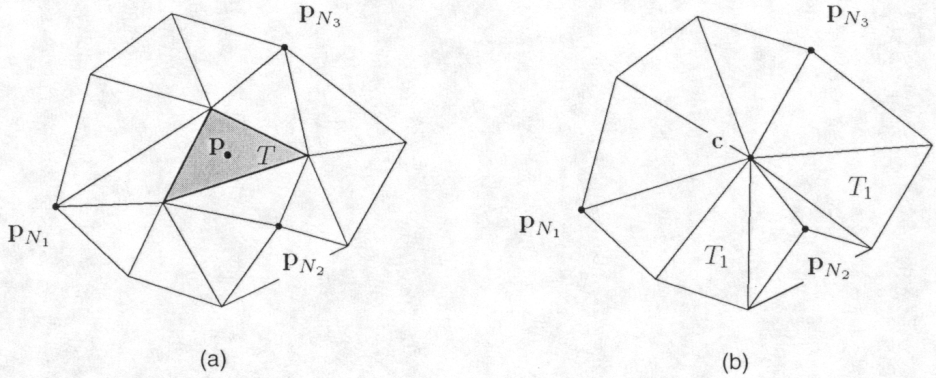


Fig. 7. Introducing cycles into the triangulation. (a) The vertex $\mathbf{p}_{N_2}$ has valence four, and when triangle T is collapsed in (b), a cycle is introduced in the stencils of the triangles T_1 and T_2.

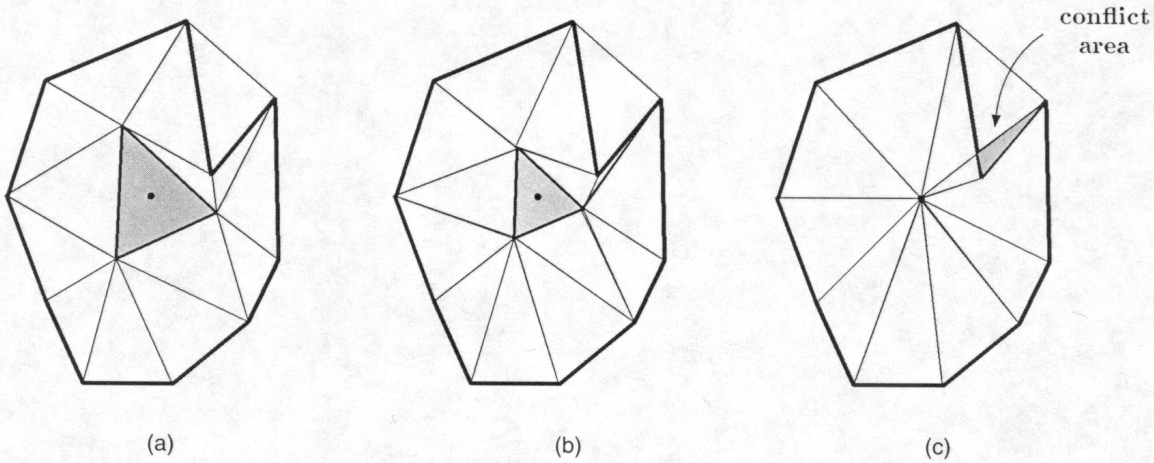


Fig. 8. Folds can occur in the modified triangles of the stencil if a star-shaped property does not hold. In this planar example, the triangle in (a) has a non-star-shaped stencil. As we collapse in (b), the potential conflict can be seen, and, in (c), the fold can be seen.

$$(t-4)-(e-6)+(v-2)=t-e+v.$$

The collapsing process can introduce triangles with cyclic stencils. If any of the points $\mathbf{p}_{N_1}$, $\mathbf{p}_{N_2}$, or $\mathbf{p}_{N_3}$ has valence four, collapsing the triangle T reduces the valence of this vertex by one—thus creating a vertex of valence three and a cycle. This cycle will cause three triangles of the mesh to have acyclic stencils if the cycle is in the interior of the mesh (see Fig. 7). We define a collapsible triangle as one that does not introduce additional cycles as a result of the collapse operation.

"Oddly shaped" stencils also pose a potential problem. If the stencil is "oddly shaped," the collapse process can potentially create folds in the resulting triangles (see Fig. 8). To avoid this problem, we require the stencil boundary polygon, when projected to the plane of the triangle, to be star-shaped with respect to the centroid $\mathbf{c}$ of the triangle. This implies that for any point $\mathbf{q}$ on the stencil boundary polygon, the line $\overline{\mathbf{cq}}$ must be contained in the interior of the polygon [19].

With these observations, we define a triangle T to be *collapsible* if

1) it has a complete stencil S_T;
2) the valence of the vertex of each neighboring triangle that is not a vertex of the triangle T is not equal to four; and
3) its projected boundary polygon is star-shaped with respect to the centroid of the triangle.

When collapsing triangles of a mesh, vertices change valences and triangles disappear. Thus, a triangle may not be collapsible in one level of the hierarchy, but may be collapsible in other levels. To collapse a triangle, we only need to identify the stencil boundary polygon and the point $\mathbf{c}$ to which the triangle is collapsed.

4 Approximating The Underlying Surface

It is our goal is to faithfully approximate the underlying surface of the triangulation—that is, stay as close as possible to the original mesh. For geometric models, it is the curvature of the model that has the primary effect on the mesh reduction process. In general, we would expect triangles to fit the underlying surface well in the areas where the curvature is low. Therefore, we should be most willing to reduce the number of triangles in the areas where the curvature of the underlying surface is low, and be least willing to do it when the curvature is high. In this section, we define a bivariate function that approximates the surface in the area of a triangle T. We use this surface in many ways: We approximate the principal curvatures of the surface by the principal curvatures of the graph of this function; we col-

lapse our triangles to points on this surface; and, we find the error between our triangles and their projections on this surface.

Let $\mathbf{p}_1, \mathbf{p}_2, \ldots, \mathbf{p}_n$ be the vertices of the triangles that make up the stencil of T, and let $\mathbf{c}$ be the centroid of T. We establish a local coordinate system in the plane of T whose origin is $\mathbf{c}$. This coordinate system uses any two orthonormal vectors $\vec{\mathbf{u}}$ and $\vec{\mathbf{v}}$, and uses the unit normal vector $\vec{\mathbf{n}}$ to the plane of the triangle T (see Fig. 9). For each vertex $\mathbf{p}_j$ of a stencil triangle, we convert the coordinates of $\mathbf{p}_j$ to coordinates in the local system defined by P. These coordinates can be calculated by first projecting $\mathbf{p}_j$ onto the plane P. This new point, $\mathbf{p}_j^P$, is obtained by

$$\mathbf{p}_j^P = \mathbf{p}_j - d_j\vec{\mathbf{n}},$$

where

$$d_j = \vec{\mathbf{n}} \cdot \left(\mathbf{p}_j - \mathbf{c}\right)$$

is the distance of the point $\mathbf{p}_j$ from the plane P. Projecting $\mathbf{p}_j^P$ onto the constructed axes defines its local coordinates u_j and v_j:

$$u_j = \vec{\mathbf{u}} \cdot \left(\mathbf{p}_j^P - \mathbf{c}\right) \text{ and}$$
$$v_j = \vec{\mathbf{v}} \cdot \left(\mathbf{p}_j^P - \mathbf{c}\right)$$

(see Fig. 10). If each $\mathbf{p}_j$ is converted into (u_j, v_j, d_j) in the new coordinate system, we can use the points (u_1, v_1, d_1), (u_2, v_2, d_2), ..., (u_n, v_n, d_n) to construct a least-squares, degree-two polynomial

$$f_T(u, v) = c_{2,0}u^2 + c_{1,1}uv + c_{0,2}v^2 + c_{1,0}u + c_{0,1}v + c_{0,0}, \quad (1)$$

which we use to approximate the original surface in the area of the triangle. We can substitute the coordinates of the points (u_j, v_j, d_j) into (1), defining the linear system

$$\begin{pmatrix} u_1^2 & u_1v_1 & v_1^2 & u_1 & v_1 & 1 \\ u_2^2 & u_2v_2 & v_2^2 & u_2 & v_2 & 1 \\ \vdots & \vdots & \vdots & \vdots & \vdots & 1 \\ u_n^2 & u_nv_n & v_n^2 & u_n & v_n & 1 \end{pmatrix} \begin{pmatrix} c_{2,0} \\ c_{1,1} \\ c_{0,2} \\ c_{1,0} \\ c_{0,1} \\ c_{0,0} \end{pmatrix} = U \begin{pmatrix} c_{2,0} \\ c_{1,1} \\ c_{0,2} \\ c_{1,0} \\ c_{0,1} \\ c_{0,0} \end{pmatrix} = \begin{pmatrix} d_1 \\ d_2 \\ \vdots \\ d_n \end{pmatrix}. \quad (2)$$

The resulting normal equations are

$$U^TU \begin{pmatrix} c_{2,0} \\ c_{1,1} \\ c_{0,2} \\ c_{1,0} \\ c_{0,1} \\ c_{0,0} \end{pmatrix} = U^T \begin{pmatrix} d_1 \\ d_2 \\ \vdots \\ d_n \end{pmatrix} \quad (3)$$

and, provided the determinant of $U^T U$ does not vanish,[3] this system can be solved and the coefficients of the function $f_T(u, v)$ be determined.

3. If the determinant does vanish, we consider additional points outside the stencil.

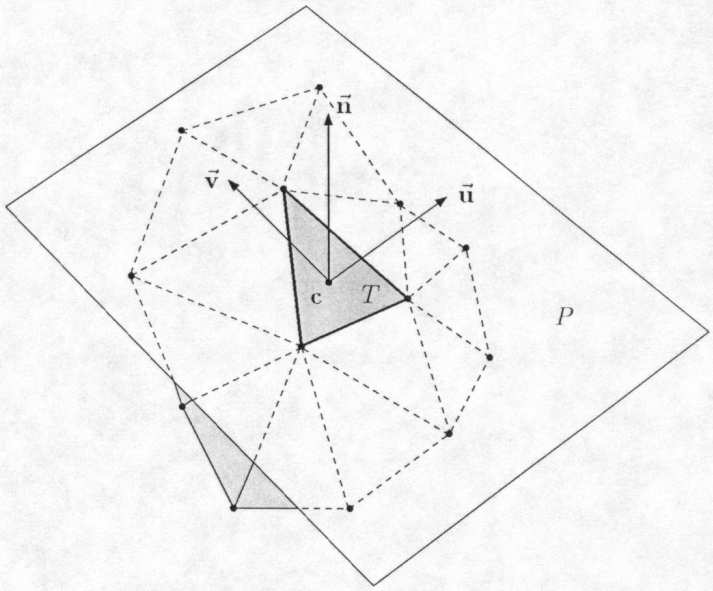


Fig. 9. Establishing a coordinate system in the plane P. Use the centroid $\mathbf{c}$ of the triangle T as the origin and use any two orthonormal vectors $\vec{\mathbf{u}}$ and $\vec{\mathbf{v}}$ in the plane P as basis vectors. The vectors $\vec{\mathbf{u}}, \vec{\mathbf{v}}$, and the unit normal $\vec{\mathbf{n}}$ form an orthonormal coordinate system.

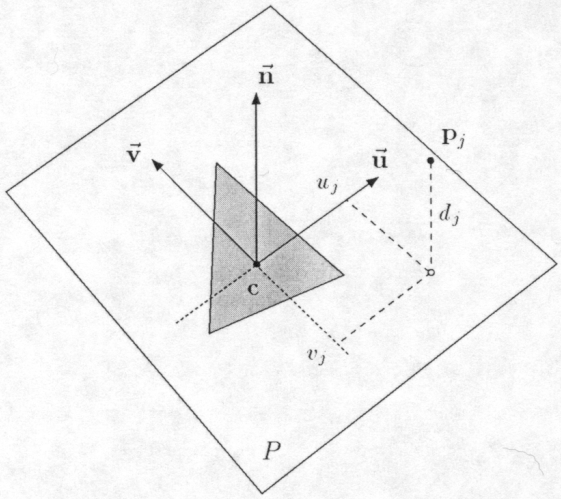


Fig. 10. To establish coordinates of the stencil points, each point is projected onto P. The coordinates u_j and v_j, along with the distance d_j from the plane, define the coordinates of the point in the local coordinate system.

4.1 Curvature Estimates

The two principal curvatures of the graph of $f_T(u, v)$ are

$$\kappa_1 = H + \sqrt{H^2 - K} \quad (4)$$

and

$$\kappa_2 = H - \sqrt{H^2 - K}, \quad (5)$$

where K is the *Gaussian curvature* of the surface at (u, v), and H is the *mean curvature* at (u, v) (see [20]). The Gaussian curvature is defined by

$$K = \frac{f_{uu}f_{vv} - f_{uv}^2}{\left(1 + f_u^2 + f_v^2\right)^2}, \quad (6)$$

and the mean curvature by

$$H = \frac{\left(1+f_u^2\right)f_{vv} - 2f_u f_v f_{uv} + \left(1+f_v^2\right)f_{uu}}{2\left(1+f_u^2+f_v^2\right)^{\frac{3}{2}}}. \quad (7)$$

In our case, $f_T(u, v)$ is a bivariate polynomial, and its partial derivatives are

$$\begin{aligned} f_u &= 2c_{2,0}u + c_{1,1}v + c_{1,0}, \\ f_v &= c_{1,1}u + 2c_{0,2}v + c_{0,1}, \\ f_{uu} &= 2c_{2,0}, \\ f_{vv} &= 2c_{0,2}, \text{ and} \\ f_{uv} &= c_{1,1}, \end{aligned}$$

defining the coefficients that we can substitute directly into (6) and (7) to obtain the Gaussian and mean curvatures of f_T at (u, v). These can then be substituted into (4) and (5) to obtain the two principal curvatures.

4.2 Calculating the Image of an Edge on the Approximating Surface

Given an edge of a triangle T defined by the local coordinates (u_0, v_0) and (u_1, v_1), using the equation of the approximating surface, we can express the equation of the triangle edge as

$$\begin{pmatrix} u(t) \\ v(t) \end{pmatrix} = \begin{pmatrix} u_0 + t(u_1 - u_0) \\ v_0 + t(v_1 - v_0) \end{pmatrix} = \begin{pmatrix} u_0 + ta_0 \\ v_0 + tb_0 \end{pmatrix},$$

where $a_0 = u_1 - u_0$ and $b_0 = v_1 - v_0$. Inserting this into (1), we obtain

$$\begin{aligned} f_T(u,v) &= f_T\big(u(t), v(t)\big) \\ &= c_{2,0}u^2(t) + c_{0,2}v^2(t) + c_{1,1}u(t)v(t) + c_{1,0}u(t) + c_{0,1}v(t) + c_{0,0} \\ &= c_{2,0}\big(u_0 + a_0t\big)^2 + c_{0,2}\big(v_0 + b_0t\big)^2 + c_{1,1}\big(u_0 + a_0t\big)\big(v_0 + b_0t\big) \\ &\quad + c_{1,0}\big(u_0 + a_0t\big) + c_{0,1}\big(v_0 + b_0t\big) + c_{0,0} \\ &= \big\{c_{2,0}a_0^2 + c_{0,2}b_0^2 + c_{1,1}a_0b_0\big\}t^2 \\ &\quad + \big\{2c_{2,0}u_0a_0 + 2c_{0,2}v_0b_0 + \big(u_0 + v_0a_0\big)c_{1,1} + c_{1,0}a_0 + c_{0,1}b_0\big\}t \\ &\quad + \big\{c_{2,0}u_0^2 + c_{0,2}v_0^2 + c_{1,1}u_0v_0 + c_{1,0}v_0 + c_{0,1}v_0 + c_{0,0}\big\} \\ &= At^2 + Bt + C, \end{aligned} \quad (8)$$

which is clearly a quadratic in t. Thus, for each edge of a triangle, we can calculate an associated parabola on the approximating surface.

4.3 The Error Between an Approximating Surface and a Triangle Edge

Given an edge of a triangle T defined by the local coordinates (u_0, v_0) and (u_1, v_1), we can calculate the maximum error between the local surface approximation and the edge. The parabola

$$f_T(t) = At^2 + Bt + C$$

from (8) has a local extremum for $t = \frac{-B}{2A}$, If $0 \le t \le 1$, then we define the maximum error e as

$$e = max\left\{\left|f_T(0)\right|, \left|f_T\left(\frac{-B}{2A}\right)\right|, \left|f_T(1)\right|\right\},$$

otherwise

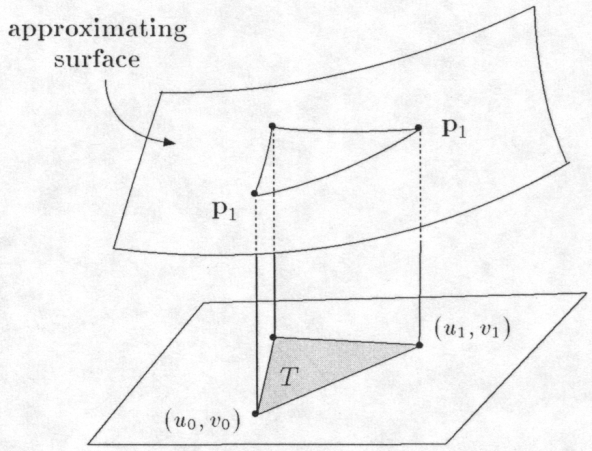


Fig. 11. On the approximating surface, the image of a triangle edge is a quadratic polynomial passing through $\mathbf{p}_0$ and $\mathbf{p}_1$.

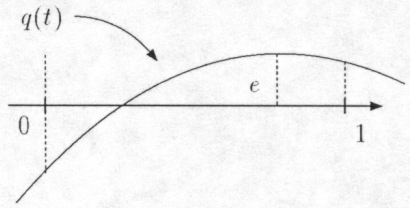


Fig. 12. The Maximum error e between an edge of a triangle and the approximating surface can occur at either at a local maximum between 0 and 1 or at the endpoints.

$$e = max\{\,|f_T(0)|\,,\ |f_T(1)|\,\}.$$

4.4 Calculating the Collapse Point

We use the approximating function f_T to determine the point to which a triangle T is to be collapsed. For triangles on the interior of the mesh, the function f_T is evaluated at $(u, v) = (0, 0)$ and the "collapse point" is defined to be

$$\mathbf{c} + f_T(0,\ 0)\vec{\mathbf{n}}, \quad (9)$$

where $\vec{\mathbf{n}}$ is the unit normal vector to the plane of the triangle. This is the point where the approximating surface intersects a line through the centroid $\mathbf{c}$ in the direction given by $\vec{\mathbf{n}}$ (see Fig. 13).

To preserve the boundary of the mesh in the various hierarchies, the triangles on the boundary must be collapsed carefully. Enumerating the types of triangles on the boundary (see Fig. 14), we obtain

1) A type-1 triangle has one edge on the boundary. To collapse this triangle, we find the approximating biquadratic surface, develop the quadratic polynomial from this edge, and use a point on the approximating polynomial that corresponds to the midpoint of the edge (see Fig. 15).
2) A type-2 triangle has a single vertex on the boundary. In this case, the triangle is collapsed to a point on the approximating surface that corresponds to the boundary vertex (see Fig. 16).

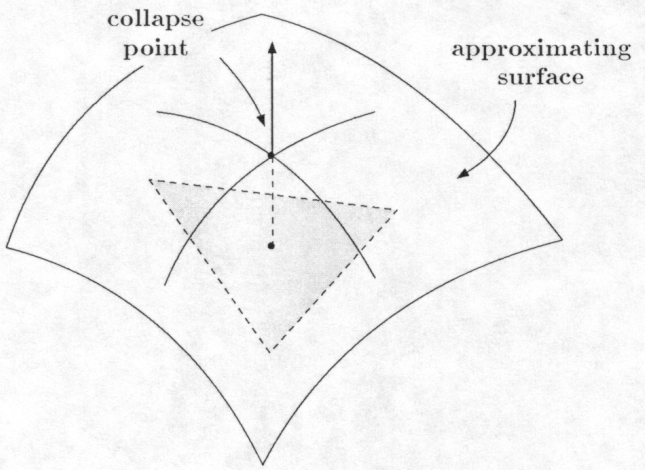


Fig. 13. To establish the point to which a triangle will collapse, we find the intersection of the line through the origin of the local coordinate system in the direction of the normal vector to the triangle, and the approximating surface.

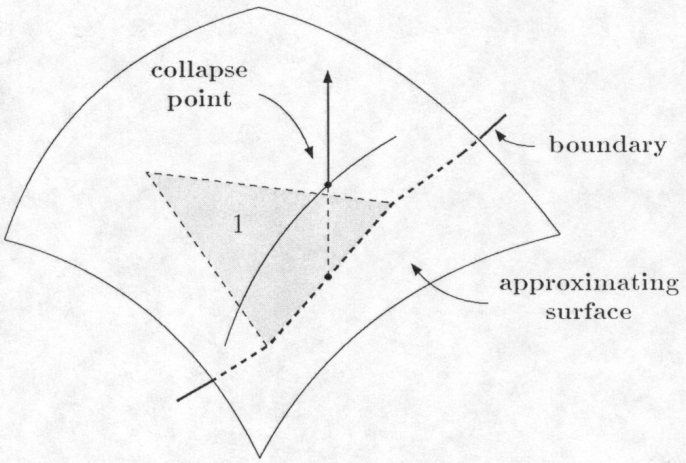


Fig. 15. Type-1 triangles are collapsed to a point that represents the edge of the mesh. This point can be found by finding the line that passes through the midpoint of the edge in the direction of the normal and intersecting this line with the approximating surface.

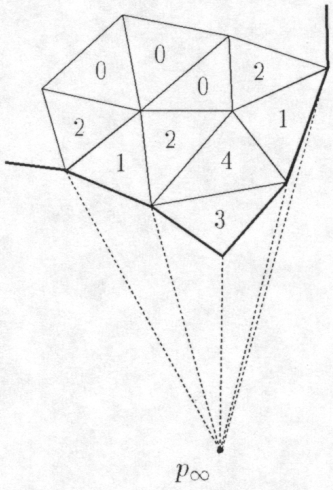


Fig. 14. Four types of boundary triangles can be identified in the mesh: 1) Triangles that have an edge on the boundary; 2) triangles that have a single vertex on the boundary; 3) triangles that have two edges on the boundary (representing corners); and 4) triangles that have two vertices on the boundary. Each triangle has stencil triangles that contain the point-at-infinity $\mathbf{p}_\infty$.

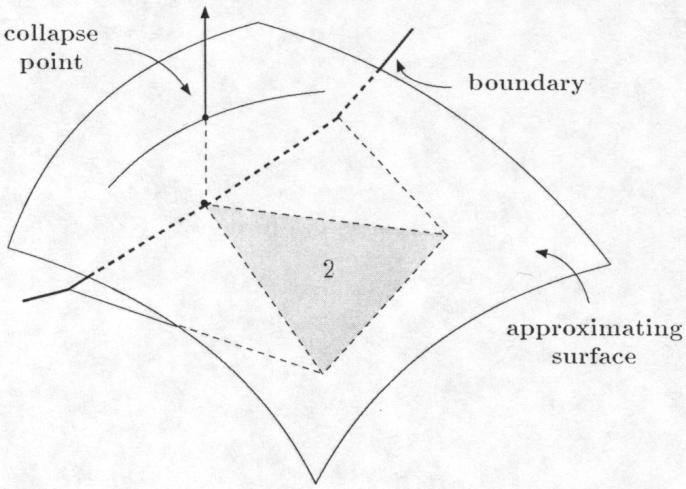


Fig. 16. Type-2 triangles are collapsed to the point on the approximating surface that corresponds to the triangle boundary vertex.

3) A type-3 triangle has two edges on the boundary. This implies a "corner vertex" and the triangle is collapsed to the vertex shared by the two boundary edges.
4) A type-4 triangle has two vertices on the boundary but no edges. We find these triangles in two situations:
   - They are neighbors of a corner triangle (a type-3 triangle). The triangles have a cycle in their stencil—the "corner triangle" and two triangles containing the point-at-infinity—and so they are not collapsible.
   - The two vertices touch two boundary curves of the mesh. If we collapse these triangles, the boundary of the mesh will be compromised.

   In general, triangles with more than one vertex, and no edge on the boundary are not collapsible. If such a triangle were collapsed, the boundary could not be preserved. Our algorithm may not collapse these type-4 triangles directly, but allows them to be neighbors of collapsing triangles.

5 The Mesh Reduction Algorithm

With the triangle-collapse operation, the mesh reduction algorithm is straightforward to describe. Given an initial triangle mesh $\mathcal{M}_0$, we calculate a weight for each triangle T of the mesh and place the triangle on a priority queue—ordered by increasing weight. The weight assigned to T represents the effect that the collapse of T will have on the mesh. We then repeatedly perform the following steps:

1) A triangle T is removed from the front of the queue, collapsed, and a new mesh is generated.
2) For each modified triangle of the stencil of T, recalculate the weight of the triangle and reposition it in the queue.

This process is repeated until a coarse mesh is generated with a specified number of triangles.

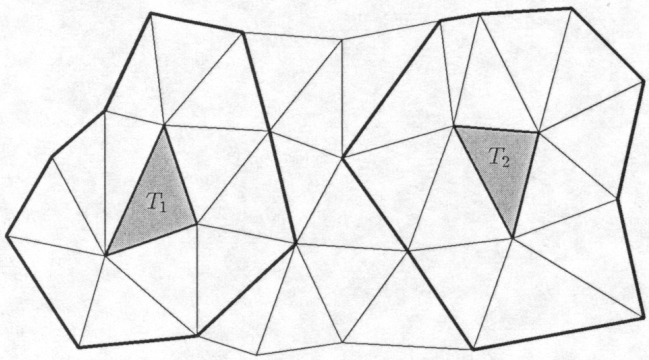


Fig. 17. Several triangles of the mesh can be collapsed simultaneously. The triangle stencils of T_1 and T_2 must not intersect.

The algorithm generates a series of triangle-collapse operations $C_0, C_1, C_2, \ldots, C_m$ and a sequence of meshes $\mathcal{M}_0, \mathcal{M}_1, \mathcal{M}_2, \ldots, \mathcal{M}_m$, each of which differs from the previous mesh by one triangle collapse. Since each of the triangle-collapse operations is reversible, we can store the coarse mesh $\mathcal{M}_m$ and the sequence $C_m, C_{m-1}, \ldots, C_1$ (in similar way to [10]) and create desired meshes of various resolutions by reversing the triangle-collapse operations—"expanding the vertices to triangles"—in sequence.

We can make a straightforward modification to this algorithm. Instead of collapsing just a single triangle in a mesh, the algorithm identifies a certain percentage of triangles that can be collapsed in parallel—recognizing that two triangles can be collapsed simultaneously if their stencils do not overlap (see Fig. 17).

In this case, we remove a triangle T from the queue if the following two conditions hold:

- T has a weight less than a specified value.
- If $T_0, T_1, T_2, \ldots, T_k$ represents the triangles already removed form the mesh, then the stencil of T cannot intersect the stencil of T_i, for $i = 0, \ldots k$.

Once the sequence of triangles $T_0, T_1, T_2, \ldots, T_k$ has been selected, the triangles are collapsed and a new mesh $\mathcal{M}_1$ is generated. The weights of each of the triangles in the stencils of T_i, $i = 0, \ldots, k$, are recalculated, and the triangles are repositioned in the queue. A new sequence of triangles is then selected from the queue, and a mesh $\mathcal{M}_2$ is created. This process continues until a coarse mesh is obtained.

The result is a sequence of triangle collapse operations $C_{0,0}, C_{1,0}, \ldots, C_{i,j} \ldots, C_{m,n}$ and meshes $\mathcal{M}_0, \mathcal{M}_1, \mathcal{M}_2, \ldots$ with the property that any mesh $\mathcal{M}_j$ can be transformed to mesh $\mathcal{M}_{j+1}$ by simultaneously performing the triangle collapses $C_{.j}$. Since each of these operations is reversible, we can store the coarse mesh $\mathcal{M}_n$, and the reversed set of triangle-collapse operations $C'_n, C'_{n-1}, \ldots, C_0 C_{m,n}, \ldots, C_{i,j} \ldots, C_{0,0}$ with "markers" indicating which collapses can be done simultaneously. This approach leads to a significantly smaller number of triangulation levels in the final hierarchy.

The sequence of meshes allows a smooth blending algorithm to be implemented by defining a partial triangle-collapse operation between consecutive meshes. If we define a parameter t, $0 \le t \le 1$, we can define a triangle mesh $\mathcal{M}_i(t)$, with the property that $\mathcal{M}_i(0) = \mathcal{M}_i$ and $\mathcal{M}_i(1) = \mathcal{M}_{i+1}$. $\mathcal{M}_i(t)$ is constructed by "partially collapsing" all selected triangles of $\mathcal{M}_i$: If T is a selected triangle of $\mathcal{M}_i$, with $\mathbf{p}_1$, $\mathbf{p}_2$, and $\mathbf{p}_3$ as its vertices, and $\mathbf{c}$ is the point to which the triangle is collapsing, then the mesh $\mathcal{M}_i(t)$ is the result of linearly interpolating $\mathbf{p}_i$ and $\mathbf{c}$—that is,

$$\mathbf{p}_1(t) = \mathbf{p}_1 + t(\mathbf{c} - \mathbf{p}_1),$$
$$\mathbf{p}_2(t) = \mathbf{p}_2 + t(\mathbf{c} - \mathbf{p}_2),$$

and

$$\mathbf{p}_3(t) = \mathbf{p}_3 + t(\mathbf{c} - \mathbf{p}_3).$$

The meshes $\mathcal{M}_i(t)$ provide a geometrically continuous method to vary smoothly between different levels in the hierarchy—despite the fact that two levels are topologically different.

Fig. 18 illustrates the smooth transitions possible between the meshes. Fig. 18a shows a portion of the mesh $\mathcal{M}_0$ on a torus data set. The triangles that have been selected for collapse are shown in yellow, and the stencils of these triangles are shown in blue. Fig. 18b shows the mesh $\mathcal{M}_0(0.75)$, where the triangles have been partially collapsed.

6 Triangle Weights

The algorithm assigns a weight to each triangle of the original mesh. The weights are used to determine the order in which triangles of the mesh will be collapsed. The triangles with the smallest weight should be removed first.

The weight of a triangle T depends on four criteria:

1) the absolute curvature[4] $\kappa(T)$ of the approximating function in the area about T;
2) a shape measure $\alpha(T)$, which assigns a higher weight to triangles that are near-equilateral;
3) a topological measure $V(T)$, which penalizes triangles that will produce high-valence vertices when collapsed;
4) an error measure $E(T)$ that indicates the error between the collapsed stencil of a triangle and its approximating surface; and
5) the triangle area $A(T)$.

Weights are defined by

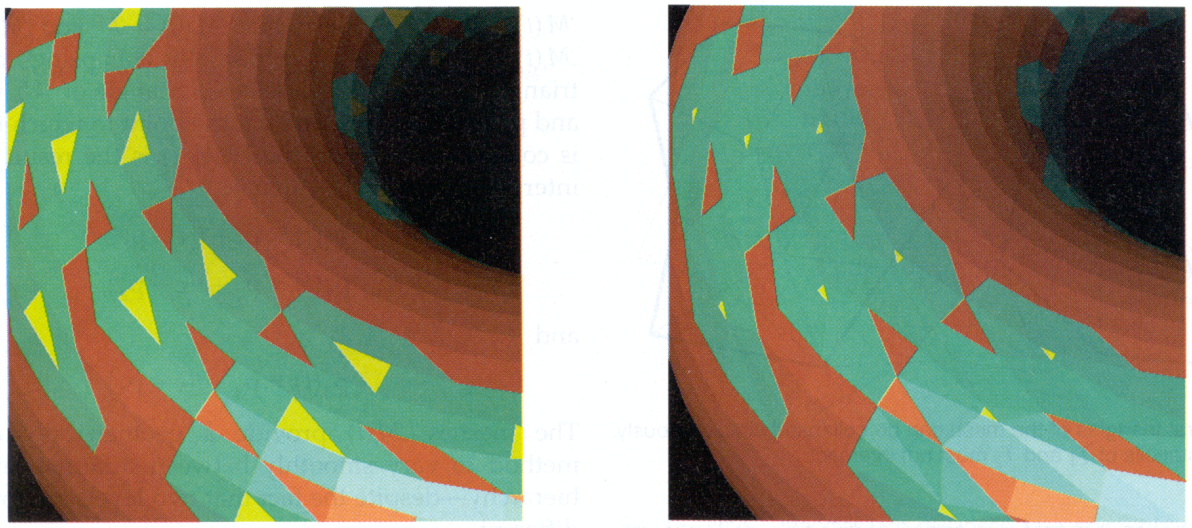
$$W(T) = A(T)\,(w_\kappa \kappa(T) + w_\alpha \alpha(T) + w_v V(T) + w_e E(T)),$$

where w_κ, w_α, w_v, and w_e are user-specified weights, with $0 \le w_\kappa, w_\alpha, w_v, w_e \le 1$. Triangles with small weight will have the least impact on the mesh when collapsed and are collapsed first.

The curvature weight $\kappa(T)$ is the absolute curvature of the graph of the approximating function $f_T(u, v)$ of T at $u = v = 0$, normalized by the maximum absolute curvature observed in the data set. When we multiply this weight by the area of the triangle, large triangles in areas of high curvature have large weights and small triangles in flat areas have small weights.

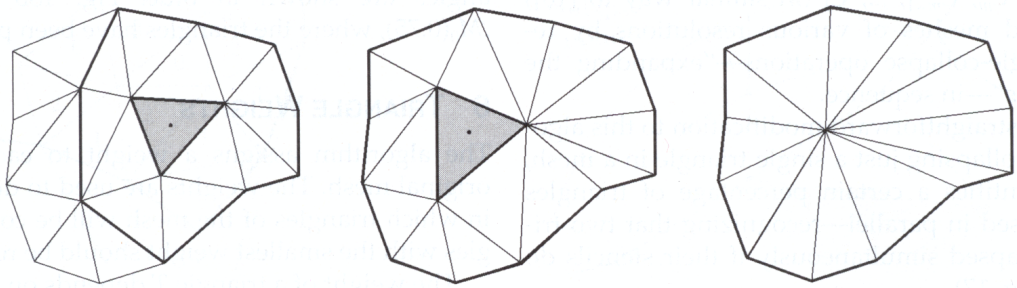
The angle weight $\alpha(T)$ is given by

4. We define the absolute curvature $\kappa(T)$ as the sum of the absolute values of the two principal curvatures, i.e., $\kappa(T) = |\kappa_1| + |\kappa_2|$.



(a) (b)

Fig. 18. Smooth transition between meshes.



(a) (b) (c)

Fig. 19. Producing vertices of high valence. If the dark-shaded triangle in (a) is collapsed, the mesh in (b) is produced. If the dark triangle in (b) is collapsed, the mesh in (c) is produced, which contains a vertex of valence 12.

$$\alpha(T) = 2\left(\left(\sum_{i=1}^{3} \cos\alpha_i\right) - 1\right),$$

where α_i, $i = 1, 2, 3$, are T's interior angles; $\alpha(T)$ ranges from zero for degenerate triangles to one for equilateral triangles. When multiplied by the area of the triangle, $\alpha(T)$ assigns a greater weight to large, equilateral triangles and a smaller weight to narrow, small triangles.

Triangles that have high-valence vertices have difficulty in passing the star-shaped requirement for collapsibility, and, in general, they do not approximate surfaces well. We seek to avoid these situations by adding a term that depends on the potential valence of the vertex to which the triangle will be collapsed. The topological term $V(T)$ penalizes triangles that produce vertices of high valence when collapsed (see Fig. 19). This term is given by

$$V(T) = \frac{|\text{val}(\mathbf{c}) - 6|}{m_v},$$

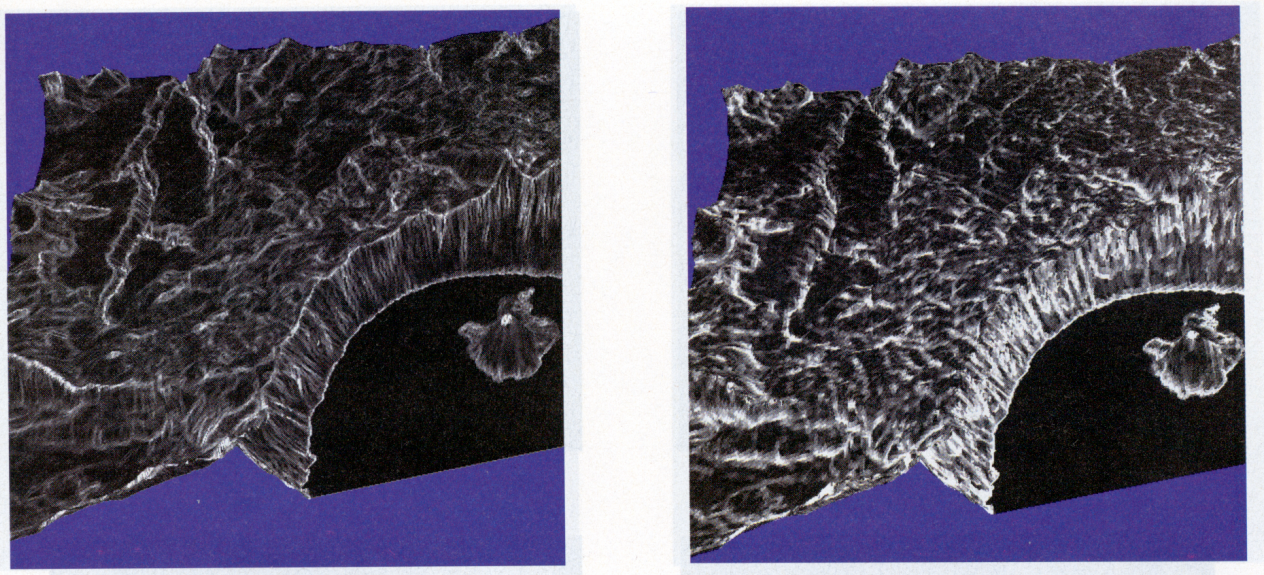
where m_v is chosen to be a maximum-valence normalizing factor and $\mathbf{c}$ is the point to which the triangle is collapsed. The algorithm considers vertices of valence six to be optimal.

The error weight $E(T)$ is based upon "predictive error." The triangle T is collapsed, and the error between the modified triangulation and the approximating surface (see Section 4.3) is calculated along the edges of the modified triangulation. $E(T)$ is the maximum edge error. When multiplied by the area of a triangle, large triangles whose modified stencil triangles fit the surface poorly receive large weights, and small triangles whose modified stencils fit the surface well receive small weights. Thus, if a triangle can be collapsed and the resulting retriangulation of the stencil is a good fit for the locally approximating surface, then the triangle will receive a low error weight.[5]

Fig. 20 illustrates curvature and error weights on a complex mesh. Fig. 20a shows a portion of the Crater-Lake data set textured with the curvature weight. Areas with high curvature are white, and areas with low curvature are dark. Fig. 20b shows the same data set textured with the value of the error weight.

The sequence of meshes and triangle-collapse operations provides an ancestral hierarchy for any triangle T of a mesh $\mathcal{M}_j$, which allows T to be associated with a set of vertices in the original mesh. T is either a triangle in both $\mathcal{M}_j$ and $\mathcal{M}_{j-1}$, or T was modified from a triangle T' in $\mathcal{M}_{j-1}$ through the collapse of a triangle T_C. In the first case, the ancestral

5. We note that this weight is not scale invariant. If the data set is rescaled the sequence of collapsed triangles may change.



(a) (b)

Fig. 20. Curvature and error weights on a portion of the Crater Lake data set.

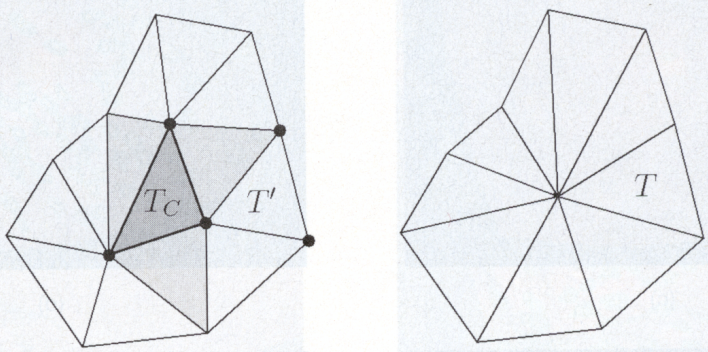


Fig. 21. The triangle T is the result of triangle T' after the collapse of T_C. The ancestral points of T are the union of the vertices of T' and T_C.

points associated with T in $\mathcal{M}_j$ are just the ancestral points of T in $\mathcal{M}_{j-1}$. In the second case, the ancestral points of T are the union of the ancestral points of T' and those of T_C (see Fig. 21). In this way, every triangle T has a set of ancestral points in the original mesh $\mathcal{M}_0$, which are the points that affect the construction of the vertices of T. When recalculating the weight of the triangle T, we can use points from the full ancestral hierarchy to determine the approximating surface. In this way, points of the original mesh are used to calculate the weights of the triangles and the new collapse points, reducing error accumulation.

TABLE 1
SUMMARIES OF THE MESH REDUCTION PROCESS FOR THE FOUR DATA SETS

Data Set	Level	Triangles	Percent	Fig.
Saddle	1	19,998	100.0%	22a
	10	6,322	31.6%	22c
	20	1,754	8.7%	22e
	30	474	2.4%	22f
Mount St. Helens	1	303,454	100.0%	23a
	10	96,022	31.6%	23c
	20	23,522	7.75%	23e
	30	7,400	2.45%	23f
Torus	1	5,000	100.0%	24a
	5	2,628	52,6%	24c
	10	1,300	27.6%	24e
	15	840	16.4%	24f
Crater Lake	1	318,542	100.0%	25a
	10	100,798	31.6%	25c
	20	28,058	8.8%	25e
	30	7,802	2.44%	25g

7 RESULTS

The algorithm that we have presented allows the representation of large triangular meshes at multiple levels of detail, requiring a relatively small number of triangulation levels to be stored. Our algorithm is based on the idea of collapsing a certain percentage of triangles in an intermediate mesh in a single step. This principle leads to significant reductions regarding storage requirements. Furthermore, it is possible to smoothly traverse the hierarchy "upward" and "downward."

We have applied our algorithm to several large triangulated models and have achieved very encouraging results. Table 1 summarizes the results of the data sets shown in the color plates.

Fig. 22 illustrates a saddle surface originally containing 19,998 triangles. We have illustrated the mesh at various lev-

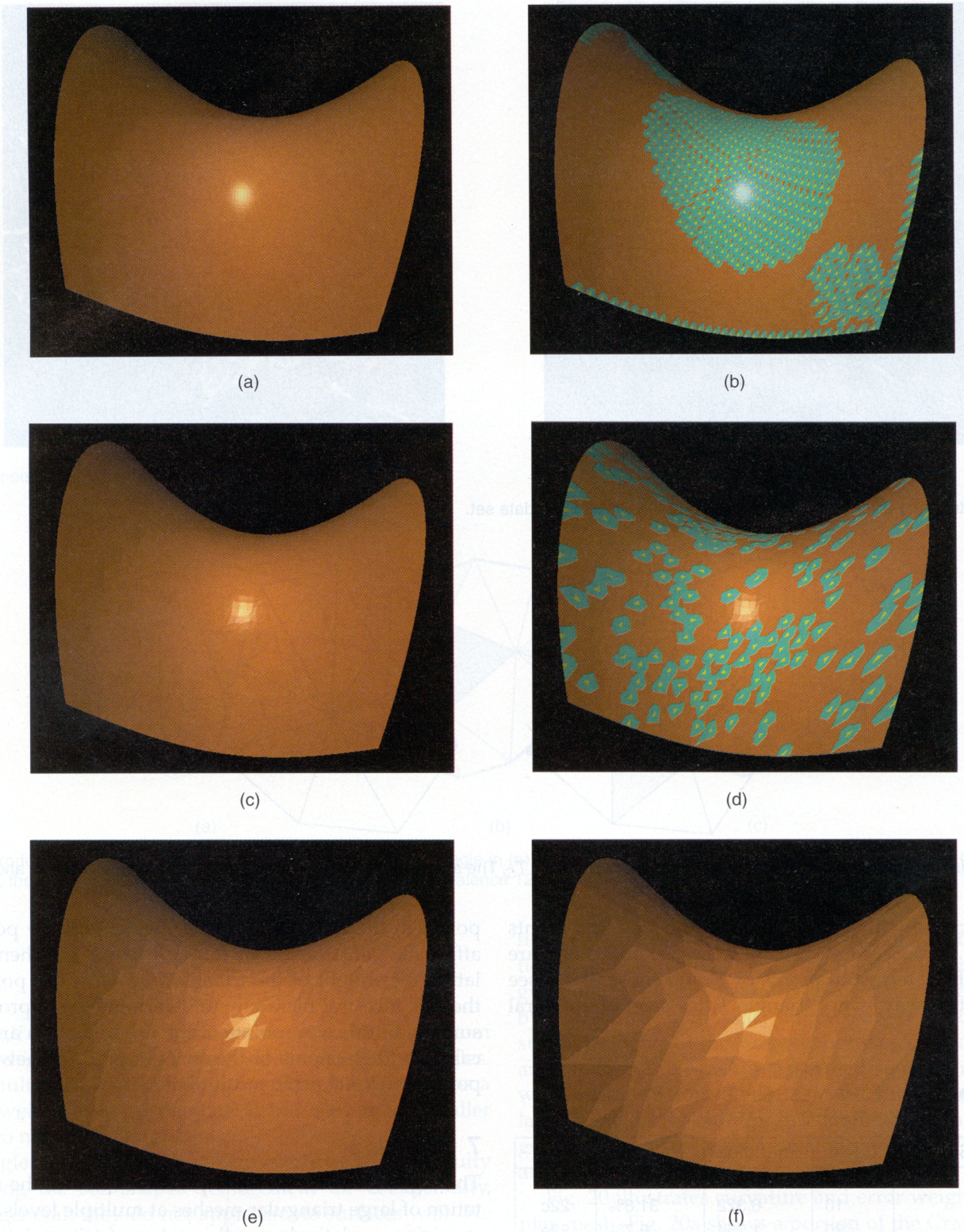


Fig. 22. Saddle surface.

els, and the triangles selected for collapse at level 1 and at level 10. Due to a user-defined limit on the number of triangles selected for each level, the algorithm initially selects triangles close to the saddle point. However, as the algorithm proceeds, the triangles selected for collapse are nearly uniformly distributed over the mesh. Even though the mesh at level 30 has only 2.4 percent of the original number of triangles, it still gives an adequate representation.

Fig. 23 is taken from the digital elevation model of Mount St. Helens, in the Cascade Range in the western United States. This model originally has 303,454 triangles. Fig. 23b shows the triangles and their stencils chosen for collapse in the initial level. We note that these are distributed in the nearly flat areas of the data set. At level 10, most of the triangles in the flat areas have been collapsed, and the area weight begins to force the algorithm to choose triangles in the highly curved areas of the mesh (Fig. 23d). At level 30, the reduced data set has less than 2.5 percent of the original number of triangles. While most of the detail has disappeared, this still is a very good representation of the Mount St. Helens region.

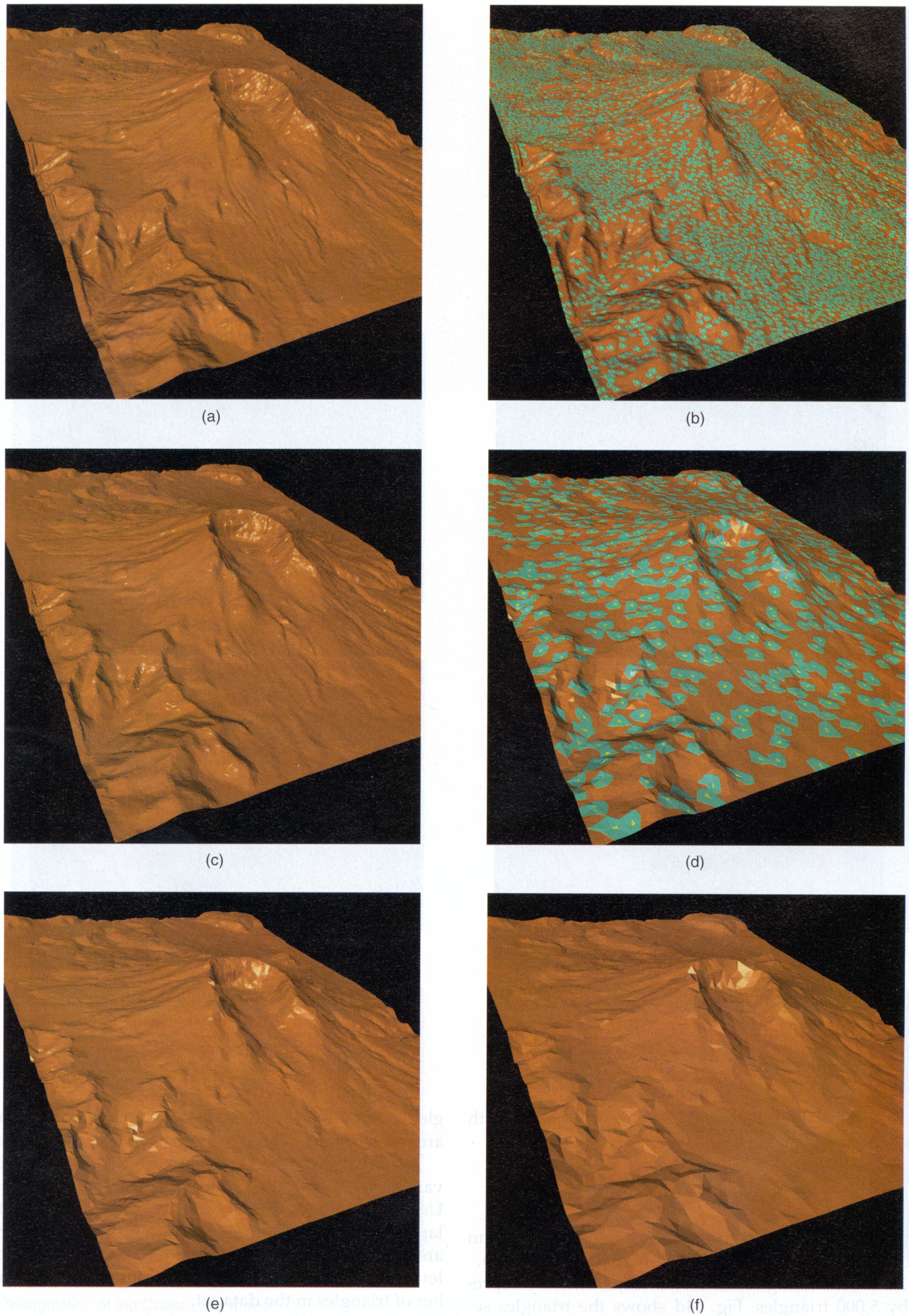


Fig. 23. Model of Mount St. Helens.

Fig. 24. Triangulation of a torus.

The images of Figs. 22 and 23 were generated with weights defined by

$$W(T) = A(T)\left(\frac{1}{3}\kappa(T) + \frac{1}{3}\alpha(T) + \frac{1}{3}V(T)\right).$$

Here, the curvature weight and the area have the maximum influence on the outcome of the mesh-reduction algorithm.

Fig. 24 shows triangulations of a torus, initially represented by 5,000 triangles. Fig. 24d shows the triangles selected for collapse in level 1, and Fig. 24d shows the triangles selected for collapse at level 5. Notice that the algorithm does not choose triangles on the equator, as the triangles there fit the resulting surface very well. Eventually, the area weight forces the algorithm to consider these triangles.

Fig. 25 shows the triangulation levels of the digital elevation model of the Crater Lake region in the western United States. Fig. 25b shows the triangles chosen for collapse in the initial level. Notice that the selected triangles are distributed in the flat areas of the data set. The mesh at level 30 contains less than 2.5 percent of the original number of triangles in the data set.

The image of Figs. 24 and 25 were generated with weights defined by

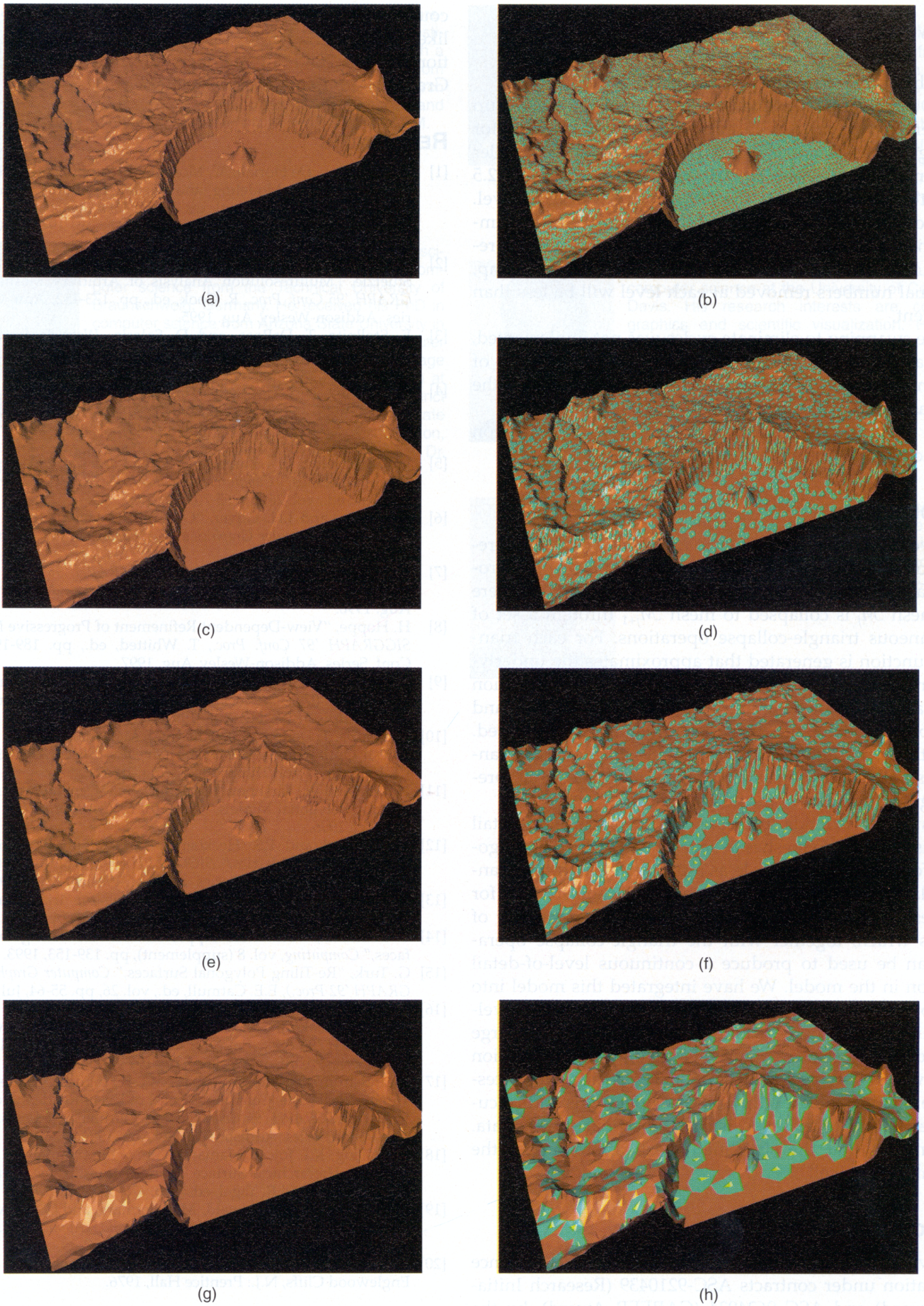


Fig. 25. Triangulation of the Crater Lake model.

$$W(T) = A(T)\left(\frac{1}{8}\kappa(T) + \frac{1}{8}\alpha(T) + \frac{3}{8}V(T) + \frac{3}{8}E(T)\right).$$

Here, the dominant factor is the predictive error weight.

The initial preprocessing step of the algorithm sets up the hierarchy of meshes in approximately 15 minutes for the large data sets, and three to five minutes for the smaller ones. In all meshes, the algorithm attempted to select 2.5 percent of the triangles in the priority queue at each level. Since four triangles disappear with each collapse, this implies that, at most, 10 percent of the triangles will be removed at each level. However, due to the stencil overlap, the actual numbers removed at each level will be less than 10 percent.

Edge swapping to eliminate cycles was not implemented, as it was found that the algorithm will select neighbors of the stencil triangles for collapse. This eliminates most of the cycles as the algorithm progresses.

The pictures can be viewed in real time on a Silicon Graphics Onyx 2 computer system.

8 Conclusions

We have introduced an algorithm for the hierarchical representation of very large triangle meshes. The algorithm produces a sequence of meshes $\mathcal{M}_0$, $\mathcal{M}_1$, $\mathcal{M}_2$, ..., $\mathcal{M}_n$, where each mesh $\mathcal{M}_j$ is collapsed to mesh $\mathcal{M}_{j+1}$ through a set of simultaneous triangle-collapse operations. For each triangle, a function is generated that approximates the underlying surface in the area of the triangle, and this function serves as a basis for assigning weights to each triangle and for supplying the point to which triangles are collapsed. Using this representation allows us to display a large triangle mesh at various levels of detail in real time while preserving the geometry of the original mesh.

This work extends previous work on level-of-detail analysis for triangle meshes in several ways. First, our algorithm focuses on the triangle as a primitive—and the triangle-collapse operation as the primary reduction strategy for the mesh. Second, our algorithm produces a sequence of meshes which, together with the triangle-collapse operation, can be used to produce a continuous level-of-detail variation in the model. We have integrated this model into a prototype viewing system that supports interactive level-of-detail manipulation of complex models defined by large triangle meshes. Third, whenever we compute the location of a new vertex replacing a triangle, we consider the ancestral hierarchy created by the collapse operations and calculate the weights of triangles using the original surface data. This ensures that the simplified mesh approximates the original surface well.

Acknowledgments

This work was supported by the U.S. National Science Foundation under contracts ASC-9210439 (Research Initiation Award) and ASC 9624034 (CAREER Award), by the U.S. Office of Naval Research under contract N00014-97-1-0222 and the U.S. Army Research Office under contract ARO 36598-MA-RIP. The digital elevation models are courtesy of the United States Geological Survey. We would like to thank Eric LaMar for helping with the digital elevation models, and all members of the Visualization and Graphics Group of CIPIC for their help.

References

[1] P. Cignoni, L. De Floriani, C. Montoni, E. Puppo, and R. Scopigno, "Multiresolution Modeling and Visualization of Volume Data Based on Simplicial Complexes," *Proc. 1994 Symp. Volume Visualization*, A. Kaufman and W. Krueger, eds., pp. 19-26, ACM SIGGRAPH, Oct. 1994.

[2] M. Eck, T. DeRose, T. Duchamp, H. Hoppe, M. Lounsbery, and W. Stuetzle, "Multiresolution Analysis of Arbitrary Meshes," *SIGGRAPH '95 Conf. Proc.*, R. Cook, ed., pp. 173-182, Ann. Conf. Series, Addison-Wesley, Aug. 1995.

[3] E. Stollnitz, T. DeRose, and D. Salesin, *Wavelets for Computer Graphics: Theory and Applications*. San Francisco: Morgan Kaufmann, 1996.

[4] P. Schröder and W. Sweldens, "Spherical Wavelets: Efficiently Representing Functions on the Sphere," *SIGGRAPH '95 Conf. Proc.*, R. Cook, ed., pp. 161-172, Ann. Conf. Series, Addison-Wesley, Aug. 1995.

[5] W.J. Schroeder, J.A. Zarge, and W.E. Lorensen, "Decimation of Triangle Meshes," *Computer Graphics (SIGGRAPH '92 Proc.)*, E.E. Catmull, ed., vol. 26, pp. 65-70, July 1992.

[6] K.J. Renze and J.H. Oliver, "Generalized Unstructured Decimation," *IEEE Computer Graphics and Applications*, vol. 16, no. 6, pp. 24-32, Nov. 1996.

[7] H. Hoppe, "Progressive Meshes," *SIGGRAPH '96 Conf. Proc.*, H. Rushmeier, ed., pp. 99-108, Ann. Conf. Series, Addison-Wesley, Aug. 1996.

[8] H. Hoppe, "View-Dependent Refinement of Progressive Meshes," *SIGGRAPH '97 Conf. Proc.*, T. Whitted, ed., pp. 189-198, Ann. Conf. Series, Addison-Wesley, Aug. 1997.

[9] J. Popovic and H. Hoppe, "Progressive Simplicial Complexes," *SIGGRAPH '97 Conf. Proc.*, T. Whitted, ed, pp. 217-224, Ann. Conf. Series, Addison-Wesley, Aug. 1997.

[10] H. Hoppe, T. DeRose, T. Duchamp, J. McDonald, and W. Stuetzle, "Mesh Optimization," *Computer Graphics (SIGGRAPH '93 Proc.)*, J.T. Kajiya, ed., vol. 27, pp. 19-26, Aug. 1993.

[11] J.C. Zia and A. Varshney, "Dynamic View-Dependent Simplification for Polygonal Models," *Proc. IEEE Visualization '96*, pp. 327-334, Oct. 1996.

[12] M. Garland and P.S. Heckbert, "Surface Simplification Using Quadratic Error Metrics," *SIGGRAPH '97 Proc.*, T. Whitted, ed., pp. 209-216, Ann. Conf. Series, Addison-Wesley, Aug. 1997.

[13] B. Hamann, "A Data Reduction Scheme for Triangulated Surfaces," *Computer Aided Geometric Design*, vol. 11, pp. 197-214, 1994.

[14] B. Hamman, "Curvature Approximation for Triangulated Surfaces," *Computing*, vol. 8 (supplement), pp. 139-153, 1993.

[15] G. Turk, "Re-Tiling Polygonal Surfaces," *Computer Graphics (SIGGRAPH '92 Proc.)*, E.E. Catmull, ed., vol. 26, pp. 55-64, July 1992.

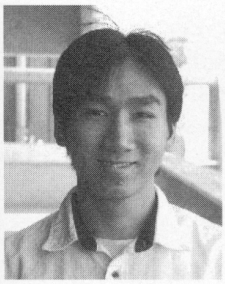
[16] J. Cohen, A. Varshney, D. Manocha, G. Turk, H. Weber, P. Agarwal, F.P. Brooks Jr., and W. Wright, "Simplification Envelopes," *SIGGRAPH '96 Conf. Proc.*, H. Rushmeier, ed., pp. 119-128, Ann. Conf. Series, Addison-Wesley, Aug. 1996.

[17] P. Lindstrom, D. Koller, W. Ribarsky, L.F. Hughes, N. Faust, and G. Turner, "Real-Time, Continuous Level of Detail Rendering of Height Fields," *SIGGRAPH '96 Conf. Proc.*, H. Rushmeier, ed., pp. 109-118, Ann. Conf. Series, Addison-Wesley, Aug. 1996.

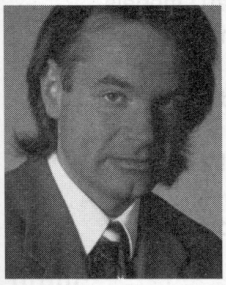
[18] T.S. Gieng, K.I. Joy, B. Hamann, G.L. Schussman, and I.J. Trotts, "Smooth Hierarchical Surface Triangulations," *Proc. Visualization '97*, H. Hagen and R. Yagel, eds., pp. 379-386, Oct. 1997.

[19] M. de Berg, M. van Kreveld, M. Overmars, and O. Schwarzkopf, *Computational Geometry: Algorithms and Applications*. Berlin: Springer-Verlag, 1997.

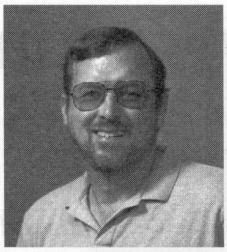
[20] M.P. do Carno, *Differential Geometry of Curves and Surfaces*. Englewood Cliffs, N.J.: Prentice Hall, 1976.



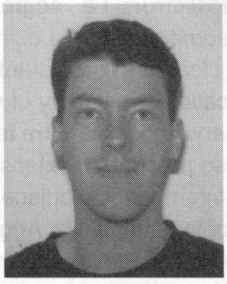
Tran Gieng is a graduate student in the Multiresolution Modeling Group at the California Institute of Technology. He graduated in 1997 with a BS in computer science and engineering from the University of California at Davis. His research interests are in the areas of mathematics and computer science. He is a member of the ACM.



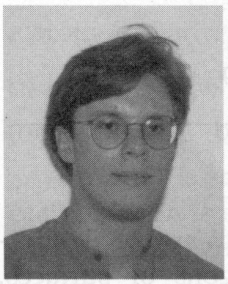
Bernd Hamann received a BS in computer science, a BS in mathematics, and an MS in computer science from the Technical University of Braunschweig, Germany. He received his PhD in computer science from Arizona State University in 1991. He is an associate professor of computer science and codirector of the Center for Image Processing and Integrated Computing (CIPIC) at the University of California, Davis, and an adjunct professor of computer science at Mississippi State University. His primary interests are visualization, computer-aided geometric design (CAGD), and computer graphics. Dr. Hamann is a member of the ACM, the IEEE, and SIAM.



Kenneth I. Joy received a BA and MA in mathematics from the University of California at Los Angeles and a PhD in mathematics from the University of Colorado in 1976. He is an associate professor of computer science at the University of California at Davis. He came to UC Davis in 1980 in the Department of Mathematics and was a founding member of the Computer Science Department in 1983. His primary interests are visualization, geometric modeling, and computer graphics. Dr. Joy is a member of the ACM, the IEEE, and SIAM.



Greg L. Schussman is a graduate student in computer science at the University of California, Davis. His research interests are computer graphics and scientific visualization. He is a member of the ACM and the IEEE.



Issac J. Trotts is pursuing a BS in computer science and a BS in mathematics at the University of California, Davis. His research interests include computer graphics, visualization, and computer-aided geometric design (CAGD). He is a member of the ACM.