Smooth Hierarchical Surface Triangulations

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

Center for Image Processing and Integrated Computing
Department of Computer Science
University of California, Davis 95616-8562

Abstract

We present a new method to produce a hierarchical set 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 point to which the triangles are collapsed. 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.

CR Categories and Subject Descriptors: I.3.3 [Computer Graphics]: Picture/Image Generation - Viewing Algorithms; I.3.6 [Computer Graphics]: Methodology and Techniques - Interation Techniques.

Additional Keywords: mesh simplification, triangle meshes, level-of-detail representation, shape approximation.

1 INTRODUCTION

The most critical and fundamental research problem encountered in the visualization of complex models is the development of methods for storing, approximating, and rendering the very large data sets that define them. The problem is to develop different representations of the 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 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.

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 Figure 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 surface approximate to supply the point to which the triangle is collapsed.

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 operation, can be used to create a smooth visual transition between levels in the hierarchy of triangle meshes.

In Section 2, we discuss the related work in mesh reduction. We examine the triangle-collapse operation in Section 3. Here 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. The calculation of the weights is discussed in Section 5. In Section 6, we give a complete description of the algorithm which generates a sequence of triangle-collapse operations and a sequence of meshes. Implementation issues are discussed in Section 7 and results of the algorithm's use are given in Section 8.

2 RELATED WORK

Three classes of algorithms exist that are used to reduce the number of triangles in a mesh:

- algorithms that simplify the mesh by removing vertices [13, 14].
- algorithms that simplify the mesh by removing edges [9], and
- algorithms that simplify the mesh by removing faces [8]

*{gieng,hamann,joy,schussma,trotts}@cs.ucdavis.edu

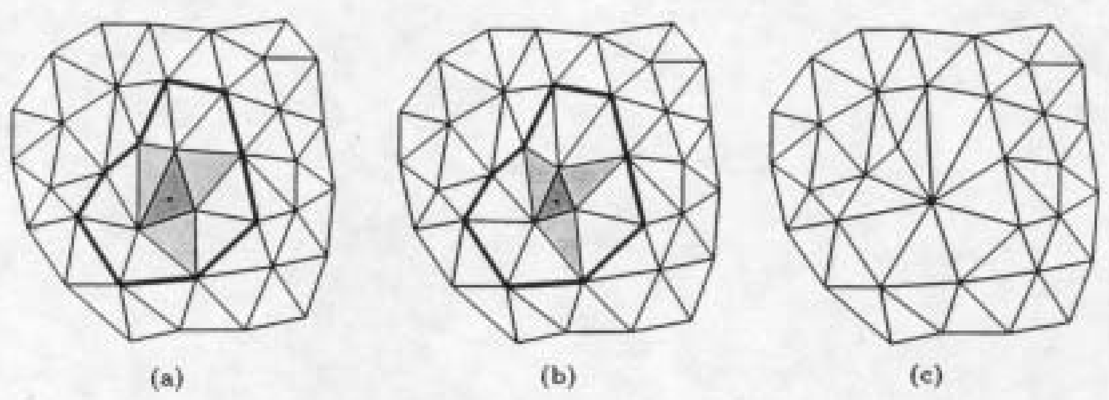
Figure 1: Collapsing a triangle: The shaded triangle is selected from the mesh in (a), collapsed toward the centroid of the triangle in (b), creating a new mesh in (c) which has four fewer triangles than the original mesh.

The problems of vertex-removal are characterized by the algorithm of Schroeder et al. [14]. In this algorithm, 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 re-triangulated, and several strategies may be used.

Hoppe [9] describes a continuous-resolution representation of a mesh, based upon an edge-collapse operation. The mesh reduction problem is formulated in terms of a optimization problem [10], ordering the edges according to an energy minimization function. 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.

Hamann [8] has developed an algorithm that simplifies the mesh by removing triangles. Triangles are selected for removal by first ordering them according to the principal curvatures at their vertices (see [7]). The curvature values are pre-computed based on the original triangulated mesh. Triangles are then inserted into a priority queue and removed iteratively. Modified triangles have new curvatures calculated 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.

Our algorithm is also based upon a triangle-removal strategy but creates a hierarchy of meshes, not just a hierarchy of triangles. These meshes can be used to create a continuous-reduction algorithm that enables us to vary the level of detail over the set of meshes and to blend the levels in a continuous-resolution representation of the data set.

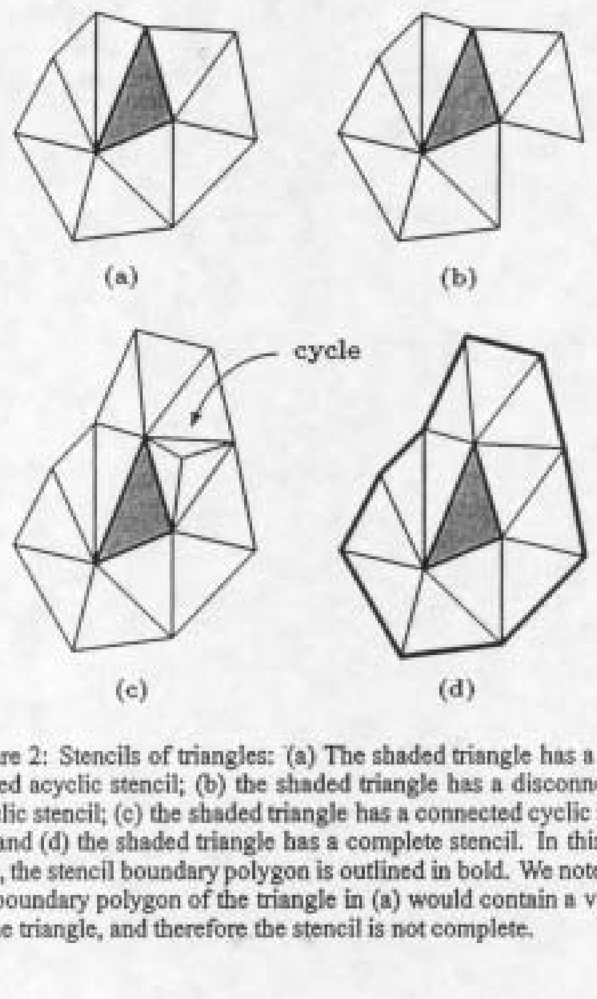

Figure 2: Stencils of triangles: (a) The shaded triangle has a connected acyclic stencil; (b) the shaded triangle has a disconnected acyclic stencil; (c) the shaded triangle has a connected cyclic stencil; and (d) the shaded triangle has a complete stencil. In this last case, the stencil boundary polygon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

3 TRIANGLE-COLLAPSE OPERATIONS

In this context, our surface is 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.

3.1 Stencils

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 such that T_i shares a vertex with T (see Figure 2). 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 $T_{i_2}, T_{i_3}, ..., T_{i_{k-1}}$ exist such that T_{i_j} and $T_{i_{j+1}}$ are neighbors[1] for $j = 1, ..., k-1$.

Three triangles T_1, T_2, and T_3 form a *cycle* in the mesh if they are pairwise neighbors. Each triangle of a cycle must have a vertex of valence three. A stencil S is called *cyclic* if there is a cycle in the stencil, otherwise the stencil is called *acyclic*. We note that a cycle can only exist in the stencil if the cycle contains a neighboring triangle of T[2]. Cycles can be eliminated in the original mesh by edge

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

[2] If the cycle does not contain a neighbor of T, then it will not be in the

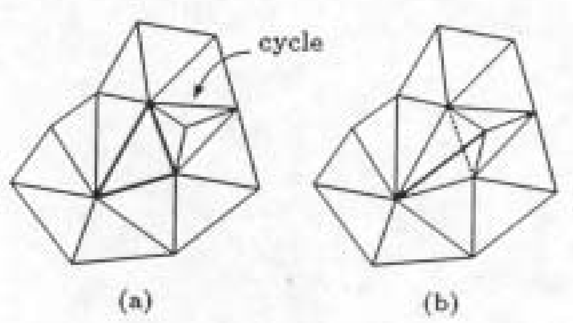

Figure 3: Edge swapping to remove cycles in the stencil: (a) the selected triangle contains a cycle in the stencil; and (b) the cycle is removed by swapping the common edge between the triangle and its neighboring triangle that belongs to the cycle.

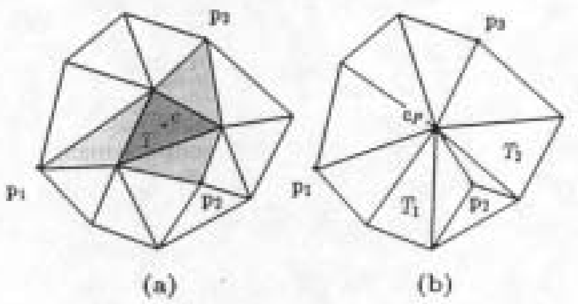

Figure 4: Introducing cycles into the triangulation: (a) The vertex $\mathbf{p}_2$ has valence four, and when triangle T is collapsed in (b), a cycle is introduced in the stencils of the triangles T_1 and T_2.

swapping (see Figure 3) where 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 such that a polygon describing the boundary of the stencil is obtained (the *stencil boundary polygon*). 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 Figure 2.

3.2 Considerations when collapsing a triangle

As can be seen in Figure 1, as 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. The new mesh contains four fewer triangles. Geometrically, this transition is smooth; topologically, it is "discontinuous" when the three vertices eventually become one.

The collapsing process can introduce triangles with acyclic stencils. Cycles in the stencil occur only in neighboring triangles and each triangle in the cycle must have a vertex of valence three. If any of the points $\mathbf{p}_1$, $\mathbf{p}_2$, or $\mathbf{p}_3$ has valence four, collapsing the triangle T will result in reducing 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 (see Figure 4). We define a collapsible triangle as one that does not introduce additional cycles as a result of the collapsing step.

Collapsing a triangle affects all triangles in the stencil. If the stencil is "oddly shaped," this can potentially create folds in the resulting triangles. To avoid this problem, we require the stencil boundary

stencil.

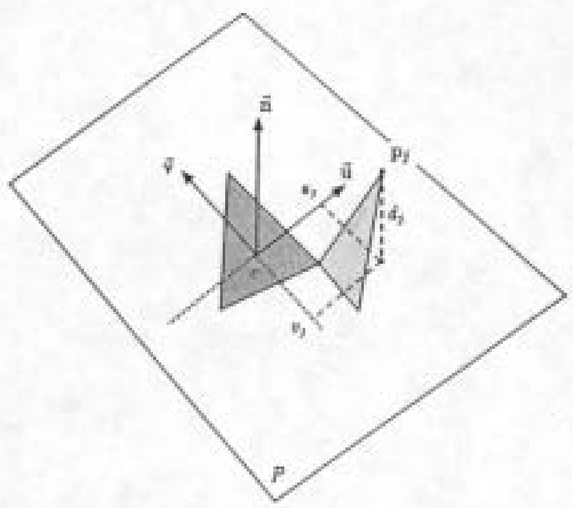

Figure 5: To establish coordinates of the stencil points, each point is projected onto P. The local coordinates u_j and v_j, along with the distance d_j from the plane, define the coordinates of the point in the local coordinate system.

polygon, when projected onto the plane of the triangle, to be star-shaped.

With these observations, we define a triangle T to be *collapsible* if (1) it has a complete stencil S_T; (2) the unique vertices of the neighboring triangles that are not part of the triangle T do not have valence four; and (3) its boundary polygon is star-shaped when projected to the plane of the triangle. With the collapsing operation, vertices change valences, and triangles disappear. Thus a triangle may not be collapsible in one mesh of the hierarchy, but may be collapsible in other meshes. To collapse a triangle, we only need identify the stencil boundary polygon and the point $\mathbf{c}_P$ to which the triangle is collapsed. For a complete discussion of the topological considerations of the collapsing operation, see [6].

4 APPROXIMATING THE UNDERLYING SURFACE

Let $\mathbf{p}_1, \mathbf{p}_2, ..., \mathbf{p}_n$ 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 $\mathbf{c}$, uses any two orthonormal vectors $\vec{u}$ and $\vec{v}$ in the plane of P, and uses the unit normal vector $\vec{n}$ to the plane of the triangle T. For each vertex $\mathbf{p}_j$, we denote its coordinates in the local coordinate system as (u_j, v_j, d_j) (see Figure 5). If each $\mathbf{p}_j$ can be transformed into (u_j, v_j, d_j) in the new coordinate system for $j = 1, 2..., n$, then 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

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

which will be used 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 equation (1), creating the linear system

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

The resulting normal equations are

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

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

4.1 Curvature estimates

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

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

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

where K is the *Gaussian curvature* of the surface at (u, v), and H is the *mean curvature* at (u, v) (see [4]). The Gaussian curvature is defined by

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

and the mean curvature by

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

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 equations (6) and (7) to obtain the Gaussian and mean curvatures of f_T at (u, v). These can then be substituted in to equations (4) and (5) to obtain the two principal curvatures.

We use this bivariate polynomial to determine both the curvatures of the underlying surfaces and to determine the point to which a triangle T is to be collapsed. The curvatures are evaluated at $(u, v) = (0, 0)$ and the "collapse point" is defined to be

$$\mathbf{c} + f_T(0, 0)\vec{\mathbf{n}}. \qquad (8)$$

This is the point where the approximating surface intersects a line through the centroid $\mathbf{c}$ in the direction given by $\vec{\mathbf{n}}$.

5 TRIANGLE WEIGHTS

Given a triangle T, we calculate its weight as

$$W(T) = A(T)\left(w_\kappa \kappa(T) + w_\alpha \alpha(T) + w_v V(T)\right),$$

[3]If the determinant does vanish, points in an "extended stencil" are considered.

where $A(T)$ is the area of T, $\kappa(T)$ is the absolute curvature[4] of the approximating function in the area about T, $\alpha(T)$ is a shape measure which assigns higher weights to triangles that are near-equilateral, and $V(T)$ is a topological measure which penalizes triangles that will produce high-valence vertices when collapsed. These quantities are combined through user-specified weights w_κ, w_α and w_v, with $0 \leq w_\kappa, w_\alpha, w_v \leq 1$ and $w_\kappa + w_\alpha + w_v = 1$. Those triangles with a small weight will have the least impact on the mesh when collapsed.

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 the largest weight, and small triangles in flat areas have the smallest.

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

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

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, this 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. 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 Figure 6). This term is given by

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

where m_v is chosen to be a maximum-valence normalizing factor and $\mathbf{c}_P$ is the point to which the triangle is collapsed.

6 MESH REDUCTION

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. Then iterate over the following procedure:

- A triangle T is removed from the front of the queue, collapsed, and a new mesh is generated.
- The triangles of the stencil of T that were modified have their weights recalculated and are reinserted into the queue.

We continue until a coarse mesh is generated with a specified number of triangles.

We This process generates a series of triangle-collapse operations $\mathcal{C}_0, \mathcal{C}_1, \mathcal{C}_2, ..., \mathcal{C}_m$ and a sequence of meshes $\mathcal{M}_0, \mathcal{M}_1, \mathcal{M}_2, ..., \mathcal{M}_m$, 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 $\mathcal{M}_n$ and the sequence $\mathcal{C}_n, \mathcal{C}_{m-1}, ..., \mathcal{C}_1$ (in similar way to [10]) and create desired meshes of various resolutions by reversing the triangle-collapse operations – "expanding the vertices into triangles" – in sequence.

We can make a straightforward modification to this algorithm that, instead of collapsing just a single triangle in a set, identifies a

[4]We define the absolute curvature κ_A as the sum of the absolute values of the principal curvatures $\kappa_A = |\kappa_1| + |\kappa_2|$.

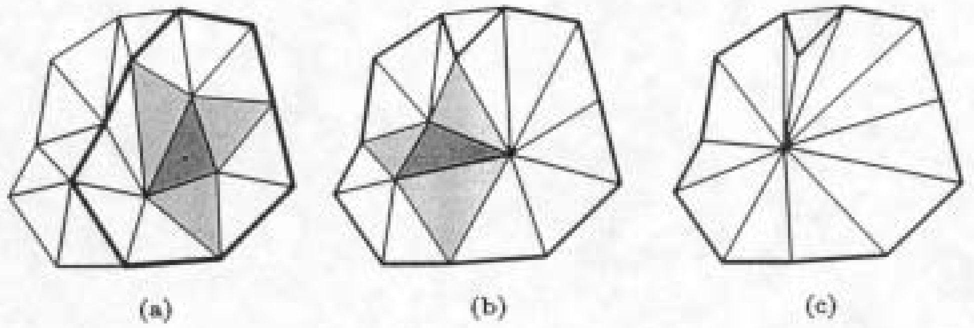

Figure 6: Producing vertices of high valence. If the dark-shaded triangle in (a) is collapsed, the mesh in (b) is produced. If the dark triangle in (b) is collapsed, the mesh in (c) is produced, which contains a vertex of valence 12.

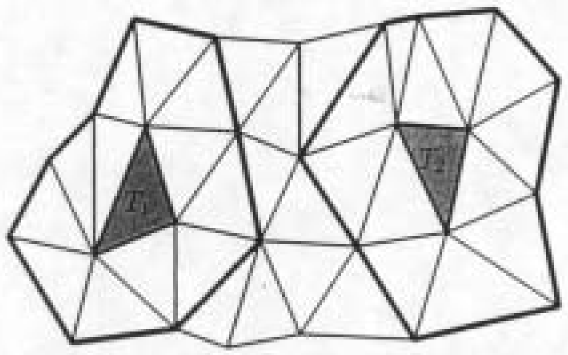

Figure 7: Several triangles of the mesh can be collapsed simultaneously. To qualify for this, the triangle stencils must not intersect.

certain percentage of triangles that can be collapsed in parallel. This algorithm recognizes that two triangles can be collapsed in simultaneously if their stencils do not overlap (see Figure 7).

Therefore, we remove a set of triangles from the queue if these conditions hold:

- Each triangle has a weight less than a specified value.
- If we have removed triangles $T_0, T_1, T_2, ..., T_k$, then we can only remove a triangle T if it has a weight less than a specified value, and if the stencil of T does not intersect any of the stencils of the T_i, $i = 0, ..., k$.

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

The result of this procedure is a sequence of triangle collapse operations $\mathcal{C}_{0,0}, \mathcal{C}_{1,0}, ..., \mathcal{C}_{i,j} ..., \mathcal{C}_{m,n}$ and meshes $\mathcal{M}_0, \mathcal{M}_1, \mathcal{M}_2, ...$ with the property that any mesh $\mathcal{M}_j$ can be transformed to mesh $\mathcal{M}_{j+1}$ by simultaneously performing the edge collapses $\mathcal{C}_{\cdot,j}$. These are again stored as a coarse mesh $\mathcal{M}_n$, and the reversed set of triangle-collapse operations $\mathcal{C}'_n, \mathcal{C}'_{n-1}, ..., \mathcal{C}_0$ $\mathcal{C}_{m,n}, ..., \mathcal{C}_{i,j} ..., \mathcal{C}_{0,0}$ with "markers" indicating which collapses can be done simultaneously.

This approach leads to a significantly smaller number of triangulation levels in the final hierarchy and allows a smooth blending algorithm to be implemented by defining a partial triangle-collapse operation between consecutive meshes. If we define a parameter t, $0 \leq t \leq 1$, 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

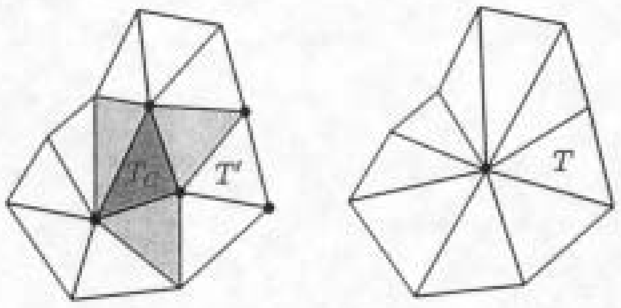

Figure 8: The triangle T is the result of triangle T' after the collapse of T_C. The ancestral points of T are the union of the vertices of T' and T_C.

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 $\mathbf{c}_P$ 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 $\mathbf{c}_P$ – that is,

$$\begin{aligned}\mathbf{p}_1(t) &= \mathbf{p}_1 + t(\mathbf{c}_P - \mathbf{p}_1),\\ \mathbf{p}_2(t) &= \mathbf{p}_2 + t(\mathbf{c}_P - \mathbf{p}_2), \text{ and}\\ \mathbf{p}_3(t) &= \mathbf{p}_3 + t(\mathbf{c}_P - \mathbf{p}_3).\end{aligned}$$

The meshes $\mathcal{M}_i(t)$ provide a geometrically continuous method to vary smoothly between different meshes in the hierarchy.

7 ANCESTRAL INFORMATION

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 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 Figure 8). 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 a triangle T in the stencil of a collapsed triangle, we use the ancestral points in the original mesh to calculate the approximating surface. These ancestral points are input to the approximating function calculation in Section 4, a new approximating function is constructed, and the collapse point is calculated by equation (8). With this procedure, we always use the vertices of the original mesh to calculate the weights of the triangles, minimizing the errors that could accumulate.

8 RESULTS

The algorithm that we have presented allows the representation of large triangular meshes at varying levels of detail, requiring a relatively small number of triangulation levels to be stored. Our algorithm is based on the idea of collapsing a large 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 "upwards" and "downwards".

We have applied our algorithm to several large triangulated models and have achieved very encouraging results[5].

- The skull data set of Figures 9–12 is the output of a marching-cubes algorithm [12][6]. and is represented by a hierarchy of 48 meshes. Figure 9 shows the complete data set where we have colored the the collapsing triangles and their stencils. Figure 10 shows the data set at level seven; Figure 11 shows the data set at level 16; and Figure 12 shows the data set at level 29. The last level contains less that five percent of the triangles of the original data set.

- The bunny data set of Figures 13–19 contains 69,668 triangles. In Figure 13, which shows the complete data set, we have colored the the collapsing triangles and their stencils. Various levels of detail along with their collapsing triangles and stencils are shown in Figures 14–16. These illustrations are chosen to represent meshes $\mathcal{M}_i(t)$, where t is a real number. Flat-shaded illustrations of the bunny data set at levels corresponding to Figures 13, 15, and 16 are shown in Figures 17–19.

The pictures can be viewed in real time on a Silicon Graphics Indigo^2 system with a 150MHz R4400 processor and 128MB RAM. The initial preprocessing step of the algorithm sets up the hierarchy of meshes in 15 minutes for the skull data set and 20 minutes for the bunny data set. The illustrations were generated using weights of $w_\kappa = w_\alpha = w_v = \frac{1}{3}$. When removing a sequence of triangles from the queue the algorithm attempted to select 2.5% of the triangles to be collapsed simultaneously.

9 CONCLUSIONS

We have introduced a new algorithm for the hierarchical representation of very large triangle meshes. The algorithm generates a hierarchical set of meshes for a given triangular mesh. This 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 has extended 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. Finally, whenever we compute the location of a new vertex replacing a triangle, we consider the ancestral hierarchy created by the collapse operations calculate the weights of triangles using the original surface data. This ensures that the simplified mesh approximates the original surface well.

[5]For review purposes we have presented these images in a large format. For the proceedings, the images would be reduced in size to lie on a single page.

[6]This is the cause of the ridges along the skull in the data set.

10 ACKNOWLEDGMENTS

This work was supported by the National Science Foundation under contracts ASC-9210439 (Research Initiation Award) and ASC 9624034 (CAREER Award), by the Office of Naval Research under contract N00014-97-1-0222 and the Army Research Office under contract ARO 36598-MA-RIP. We would like to thank all members of the Visualization and Graphics Group of CIPIC for their help.

References

[1] Andrew Certain, Jovan Popović, Tony DeRose, Tom Duchamp, David Salesin, and Werner Stuetzle. Interactive multiresolution surface viewing. In Holly Rushmeier, editor, *SIGGRAPH 96 Conference Proceedings*, Annual Conference Series, pages 91–98. ACM SIGGRAPH, Addison Wesley, August 1996.

[2] J. H. Clark. Hierarchical geometric models for visible surface algorithms. *Communications of the ACM*, 19(10):547–554, October 1976.

[3] L. De Floriani and E. Puppo. Hierarchical triangulation for multiresolution surface description geometric design. *ACM Transactions on Graphics*, 14(4):363–411, October 1995.

[4] Manfredo P. do Carmo. *Differential Geometry of Curves and Surfaces*. Prentice-Hall, Englewood Cliffs, N.J., 1976.

[5] Matthias Eck, Tony DeRose, Tom Duchamp, Hugues Hoppe, Michael Lounsbery, and Werner Stuetzle. Multiresolution analysis of arbitrary meshes. In Robert Cook, editor, *SIGGRAPH 95 Conference Proceedings*, Annual Conference Series, pages 173–182. ACM SIGGRAPH, Addison Wesley, August 1995.

[6] Tran Gieng, Kenneth I. Joy, Bernd Hamann, Gregory Schussman, and Issac Trotts. Mesh reduction via collapsing triangles. Technical Report CSE-97-1, Department of Computer Science, University of California, Davis, January 1997.

[7] B. Hamann. Curvature approximation for triangulated surfaces. *Computing*, 8 (Supplement):139–153, 1993.

[8] B. Hamann. A data reduction scheme for triangulated surfaces. *Computer Aided Geometric Design*, 11:197–214, 1994.

[9] Hugues Hoppe. Progressive meshes. In Holly Rushmeier, editor, *SIGGRAPH 96 Conference Proceedings*, Annual Conference Series, pages 99–108. ACM SIGGRAPH, Addison Wesley, August 1996.

[10] Hugues Hoppe, Tony DeRose, Tom Duchamp, John McDonald, and Werner Stuetzle. Mesh optimization. In James T. Kajiya, editor, *Computer Graphics (SIGGRAPH '93 Proceedings)*, volume 27, pages 19–26, August 1993.

[11] Reinhard Klein, Gunther Liebich, and Wolfgang Straßer. Mesh reduction with error control. In *Proceedings of IEEE Visualization '96*, pages 311–318. IEEE, October 1996.

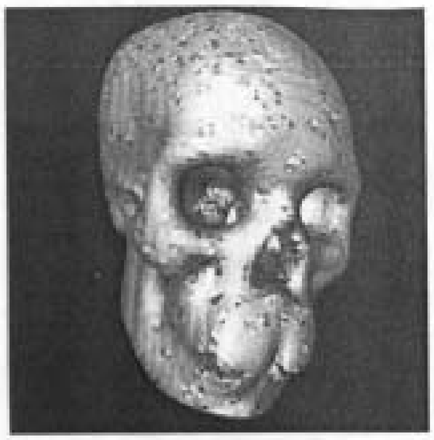

Figure 9: The original skull data set contains 52,744 triangles. The mesh along with the collapsing triangles and their stencils for this initial level are shown. The triangles to be collapsed in this first level of detail, and their stencils, are randomly colored.

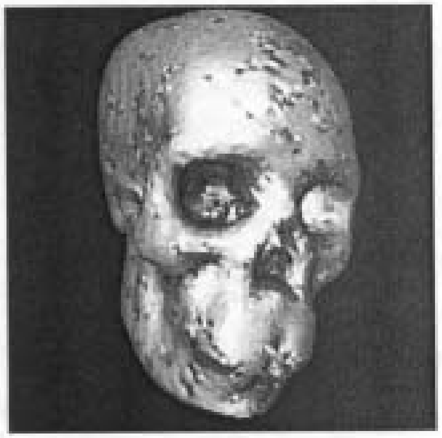

Figure 10: The skull data set at level-of-detail seven contains 25,196 triangles. The mesh along with the collapsing triangles and their stencils are shown.

[12] W. E. Lorensen and H. E. Cline. Marching cubes: a high resolution 3D surface construction algorithm. In M. C. Stone, editor, *Computer Graphics (SIGGRAPH '87 Proceedings)*, volume 21, pages 163–170, July 1987.

[13] Kevin J. Renze and James H. Oliver. Generalized unstructured decimation. *IEEE Computer Graphics & Applications*, 16(6):24–32, November 1996.

[14] William J. Schroeder, Jonathan A. Zarge, and William E. Lorensen. Decimation of triangle meshes. In Edwin E. Catmull, editor, *Computer Graphics (SIGGRAPH '92 Proceedings)*, volume 26, pages 65–70, July 1992.

[15] Greg Turk. Re-tiling polygonal surfaces. In Edwin E. Catmull, editor, *Computer Graphics (SIGGRAPH '92 Proceedings)*, volume 26, pages 55–64, July 1992.

[16] Julie C. Xia and Amitabh Varshney. Dynamic view-dependent simplification for polygonal models. In *Proceedings of IEEE Visualization '96*, pages 327–334. IEEE, October 1996.

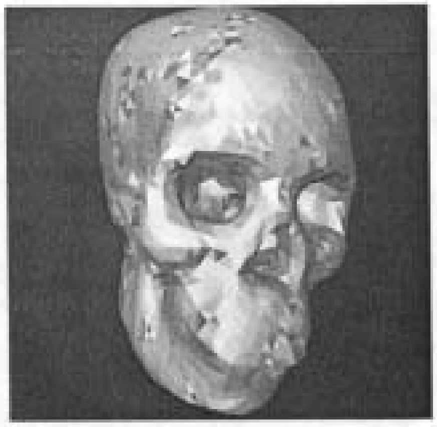

Figure 11: The skull data set at level-of-detail 16 contains 10,118 triangles. The mesh along with the collapsing triangles and their stencils are shown.

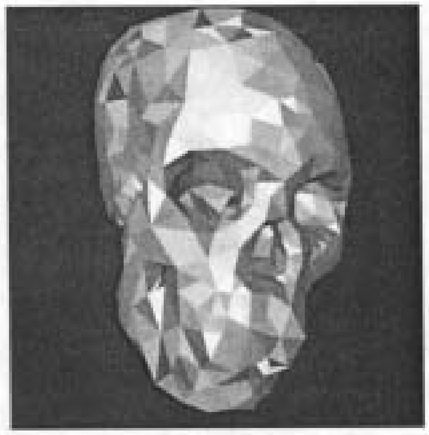

Figure 12: The skull data set at level-of-detail 29 contains 2,428 triangles. The mesh along with the collapsing triangles and their stencils are shown.

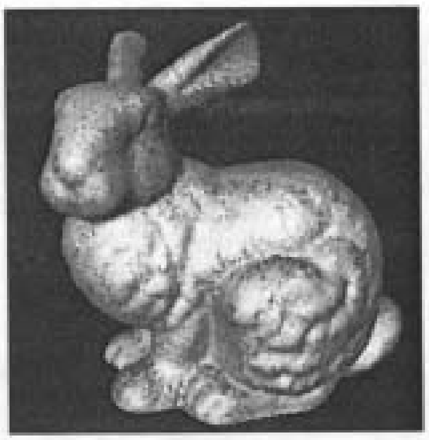

Figure 13: The original bunny data set contains 69,668 triangles. The mesh along with the collapsing triangles and their stencils for this initial level are shown.

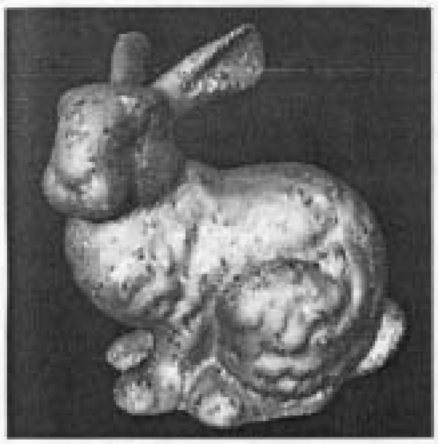

Figure 14: The bunny data set at level-of-detail 6.42 contains 39,996 triangles, some of which are partially collapsed. The mesh along with the collapsing triangles and their stencils are shown.

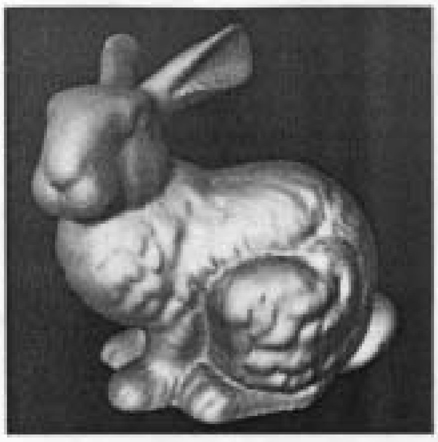

Figure 17: The original bunny data set using flat shading. The data set contains 69,668 triangles.

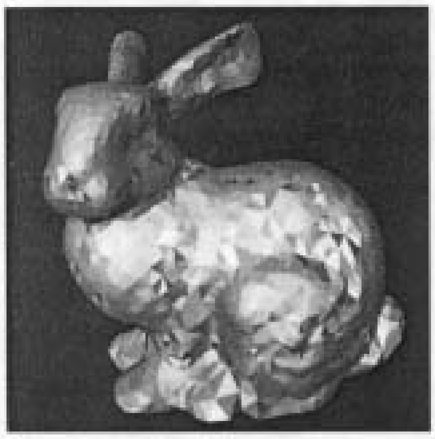

Figure 15: The bunny data set at level-of-detail 18.95 contains 10,408 triangles, some of which are almost completely collapsed. The mesh along with the collapsing triangles and their stencils are shown.

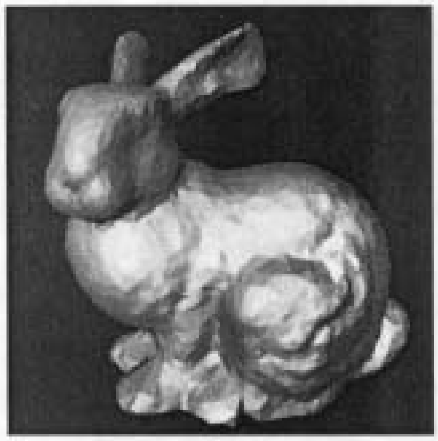

Figure 18: The bunny data set at level-of-detail 18.95 using flat shading. The mesh contains 10,408 triangles.

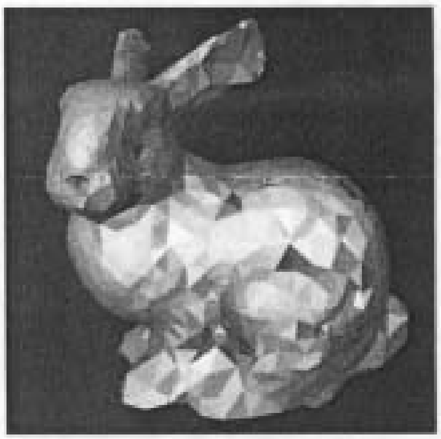

Figure 16: The bunny data set at level-of-detail 25.67 contains 4,944 triangles, some of which are partially collapsed. The mesh along with the collapsing triangles and their stencils are shown.

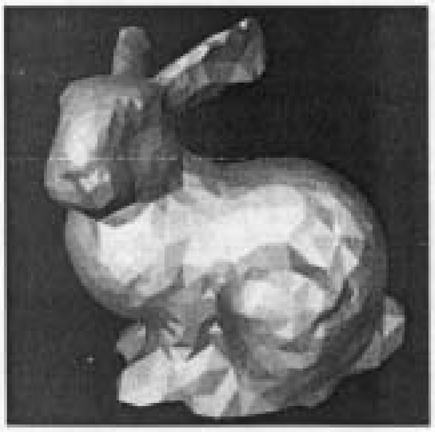

Figure 19: The bunny data set at level-of-detail 25.67 using flat shading. The mesh contains 4,944 triangles.