Decimation of Triangle Meshes

William J. Schroeder, Jonathan A. Zarge[*], *William E. Lorensen*

General Electric Company Corporate Research and Development
[*]ConSolve, Inc

ABSTRACT

Computer graphics applications routinely generate geometric models consisting of large numbers of triangles. We present an algorithm that significantly reduces the number of triangles required to model a physical or abstract object. The algorithm makes multiple passes over an existing triangle mesh, using local geometry and topology to remove vertices that pass a distance or angle criterion. The holes left by the vertex removal are patched using a local triangulation process. The decimation algorithm has been implemented in a general scientific visualization system as a general network filter. Examples from volume modeling and terrain modeling illustrate the results of the decimation algorithm.

Keywords: computer graphics, geometric modeling, medical imaging, terrain modeling, volume modeling

1 INTRODUCTION

The polygon remains a popular graphics primitive for computer graphics application. Besides having a simple representation, computer rendering of polygons is widely supported by commercial graphics hardware and software. However, because the polygon is linear, often thousands or millions of primitives are required to capture the details of complex geometry. Models of this size are generally not practical since rendering speeds and memory requirements are proportional to the number of polygons. Consequently applications that generate large polygonal meshes often use domain-specific knowledge to reduce model size. There remain algorithms, however, where domain-specific reduction techniques are not generally available or appropriate.

One algorithm that generates many polygons is *Marching Cubes* [10]. *Marching Cubes* is a brute force surface construction algorithm that extracts isodensity surfaces from volume data, producing from one to five triangles within voxels that contain the surface. Although originally developed for medical applications, *Marching Cubes* has found more frequent use in scientific visualization where the size of the volume data sets are much smaller than those found in medical applications. A large computational fluid dynamics volume could have a finite difference grid size of order 100 by 100 by 100, while a typical medical computed tomography or magnetic resonance scanner produces over 100 slices at a resolution of 256 by 256 or 512 by 512 pixels each. Industrial computed tomography, used for inspection and analysis, has even greater resolution, varying from 512 by 512 to 1024 by 1024 pixels. For these sampled data sets, isosurface extraction using *Marching Cubes* can produce from 500k to 2,000k triangles. Even today's graphics workstations have trouble storing and rendering models of this size.

Other sampling devices can produce large polygonal models: range cameras, digital elevation data, and satellite data. The sampling resolution of these devices is also improving, resulting in model sizes that rival those obtained from medical scanners.

This paper describes an application independent algorithm that uses local operations on geometry and topology to reduce the number of triangles in a triangle mesh. Although our implementation is for the triangle mesh, it can be directly applied to the more general polygon mesh. After describing other work related to model creation from sampled data, we describe the triangle decimation process and its implementation. Results from two different geometric modeling applications illustrate the strengths of the algorithm.

2 RELATED WORK

The decimation algorithm applies to discrete modeling: the synthesis, analysis and manipulation of objects contained within sampled data. Approaches to synthesizing these objects can be either adaptive or filter-based.

Adaptive techniques produce more primitives in selected areas. For example, Fowler [7] creates triangulated irregular networks (TIN) of terrain by finding ridges and channels, performing a Delaunay triangulation of these features and then adaptively adding points from the dense elevation grids. In implicit modeling, Bloomenthal [2] produces isosurfaces from implicit models by adaptively evaluating the implicit equations as long as the surface intersects his sampling cubes. In finite element mesh generation, the CATFEM system [6] uses octree techniques to create 3D finite elements directly from volume samples, generating more elements in areas of fine detail. Deformable models [11, 15] use an initial surface model that is repeatedly deformed to fit the implicit surface that exists within a sampled volume. The original model resolution controls the number of primitives in the final, deformed model. Fitting techniques approximate a surface with one or more primitives using error criteria to measure the goodness of fit. Schmitt [13] starts with rough bi-cubic patch approximations to sample data, then subdivides those patches that are not sufficiently close to the underlying samples. Recent work by DeHaemer [4] extends this

Author address:
GE CRD, KW/C211, 1 River Road, Schenectady, NY 12345
schroeder@crd.ge.com

ACM-0-89791-479-1/92/007/0065 $01.50

work to reduce the number of polygons in a polygonal mesh. Turk [16] uses a re-triangulation technique that introduces new points onto a polygonal mesh, and then discards the old points to create a new mesh.

Filter-based techniques start with a large number of samples or primitives and remove or replace samples to reduce model size. Two naive approaches are sub-sampling and averaging. Sub-sampling uses every n^{th} point in the data to reduce the size of the data, while averaging resamples the data using neighboring points.

The bulk of published work on reducing the number of primitives for modeling addresses the two-dimensional approximation of curves with line segments. Dunham [5] compares nine techniques for the piecewise linear approximation of 2D planar curves. These algorithms seek approximations that satisfy a uniform error criterion. The points produced by each algorithm all lie on the digitized curves. Recent work [8] uses dynamic programming to approximate 3D space curves. Kalvin et. al. [9] describe a technique called *Adaptive Face Merging* that removes co-planar polygons. They report substantial polygon reduction for binary voxel data sets.

3 THE DECIMATION ALGORITHM

The fundamental goal of the decimation algorithm is to reduce the total number of triangles in a triangle mesh, while preserving as accurately as possible important features. Here we define a triangle mesh to be a collection of triangles in three-space, joined along common edges and vertices. Typically the topology of the mesh is 2-manifold [17], but non-manifold forms are possible and must be treated by the algorithm.

Any reduced mesh must meet two requirements [14]. First, the reduced mesh must preserve the original topology of the mesh, including non-manifold forms. Second, the decimated mesh must form a good geometric approximation to the original mesh. Optionally, the vertices of the decimated mesh can be a subset of the original vertices. Hence new vertices are never created, instead relatively unimportant vertices (and associated triangles) are removed from the mesh, forming new approximations to the original. This optional requirement, although not essential to forming an effective approximation to the original mesh, is useful in practice because it provides a way to use the auxiliary vertex data such as normals or texture coordinates.

3.1 OVERVIEW

The decimation algorithm is simple. Multiple passes are made over all vertices in the mesh. During a pass, each vertex is a candidate for removal and, if it meets the specified decimation criteria, the vertex and all triangles that use the vertex are deleted. The resulting hole in the mesh is patched by forming a local triangulation. The vertex removal process repeats, with possible adjustment of the decimation criteria, until some termination condition is met. Usually the termination criterion is specified as a percent reduction of the original mesh (or equivalent), or as some maximum decimation value. The three steps of the algorithm are:

1. characterize the local vertex geometry and topology,
2. evaluate the decimation criteria, and
3. triangulate the resulting hole.

3.2 CHARACTERIZING LOCAL GEOMETRY / TOPOLOGY

The first step of the decimation algorithm characterizes the local geometry and topology for a given vertex. The outcome of this process determines whether the vertex is a potential candidate for deletion, and if it is, which criteria to use.

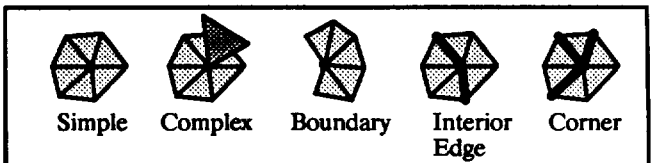

Figure 1. Vertex classifications.

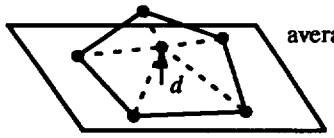

Figure 2. Distance to plane.

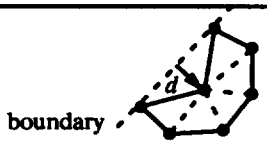

Figure 3. Distance to edge.

Each vertex may be assigned one of five possible classifications: simple, complex, boundary, interior edge, or corner vertex. Examples of each type are shown in Figure 1.

A simple vertex is surrounded by a complete cycle of triangles, and each edge that uses the vertex is used by exactly two triangles. If the edge is not used by two triangles, or if the vertex is used by a triangle not in the cycle of triangles, then the vertex is complex. These are non-manifold cases.

A vertex that is on the boundary of a mesh, i.e., within a semi-cycle of triangles, is a boundary vertex.

A simple vertex can be further classified as an interior edge or corner vertex. These classifications are based on the local mesh geometry. If the dihedral angle between two adjacent triangles is greater than a specified *feature angle*, then a *feature edge* exists. When a vertex is used by two feature edges, the vertex is an interior edge vertex. If one or three or more feature edges use the vertex, the vertex is classified a corner vertex.

Complex vertices are not deleted from the mesh. All other vertices become candidates for deletion.

3.3 EVALUATING THE DECIMATION CRITERIA

The characterization step produces an ordered loop of vertices and triangles that use the candidate vertex. The evaluation step determines whether the triangles forming the loop can be deleted and replaced by another triangulation exclusive of the original vertex. Although the fundamental decimation criterion we use is based on vertex distance to plane or vertex distance to edge, others can be applied.

Simple vertices use the distance to plane criterion (Figure 2). An average plane is constructed using the triangle normals, $\vec{n}_i$, centers, $\vec{x}_i$, and areas A_i,

$$\vec{N} = \frac{\sum \vec{n}_i A_i}{\sum A_i}, \quad \vec{n} = \frac{\vec{N}}{|\vec{N}|}, \quad \vec{x} = \frac{\sum \vec{x}_i A_i}{\sum A_i}, \tag{1}$$

where the summation is over all triangles in the loop. The distance of the vertex $\vec{v}$ to the plane is then $d = |\vec{n} \cdot (\vec{v} - \vec{x})|$. If the vertex is within the specified distance to the average plane it may be deleted. Otherwise it is retained.

Boundary and interior edge vertices use the distance to edge criterion (Figure 3). In this case, the algorithm determines the distance to the line defined by the two vertices creating the

boundary or feature edge. If the distance to the line is less than d, the vertex can be deleted.

It is not always desirable to retain feature edges. For example, meshes may contain areas of relatively small triangles with large feature angles, contributing relatively little to the geometric approximation. Or, the small triangles may be the result of "noise" in the original mesh. In these situations, corner vertices, which are usually not deleted, and interior edge vertices, which are evaluated using the distance to edge criterion, may be evaluated using the distance to plane criterion. We call this edge preservation, a user specifiable parameter.

If a vertex can be eliminated, the loop created by removing the triangles using the vertex must be triangulated. For interior edge vertices, the original loop must be split into two halves, with the split line connecting the vertices forming the feature edge. If the loop can be split in this way, i.e., so that resulting two loops do not overlap, then the loop is split and each piece is triangulated separately.

3.4 TRIANGULATION

Deleting a vertex and its associated triangles creates one (simple or boundary vertex) or two loops (interior edge vertex). Within each loop a triangulation must be created whose triangles are non-intersecting and non-degenerate. In addition, it is desirable to create triangles with good aspect ratio and that approximate the original loop as closely as possible.

In general it is not possible to use a two-dimensional algorithm to construct the triangulation, since the loop is usually non-planar. In addition, there are two important characteristics of the loop that can be used to advantage. First, if a loop cannot be triangulated, the vertex generating the loop need not be removed. Second, since every loop is star-shaped [12], triangulation schemes based on recursive loop splitting are effective. The next section describes one such scheme.

Once the triangulation is complete, the original vertex and its cycle of triangles are deleted. From the Euler relation [12] it follows that removal of a simple, corner, or interior edge vertex reduces the mesh by precisely two triangles. If a boundary vertex is deleted then the mesh is reduced by precisely one triangle.

4 IMPLEMENTATION

The decimation algorithm has been implemented as a filter in our object-oriented LYMB/VISAGE visualization environment. Usually we apply the algorithm repeatedly to eliminate vertices and triangles from a mesh until a specified reduction threshold is achieved. The decimation is controlled by slowly adjusting the distance and feature angle criterion. It is also possible to limit the total number of iterations, as well as modify other parameters such as the triangulation aspect ratio. We often specify an initial distance of zero to first remove triangles within strictly planar regions.

Two major challenges were addressed to create a successful implementation of the decimation algorithm. First, the data structures had to be carefully crafted since the size of the data (i.e., millions of triangles) demands both efficient access to and storage of data. Second, the triangulation algorithm was designed to be simple and efficient, and to take advantage of the particular characteristics of the triangulation process.

It should be noted that this algorithm, while expressly described with triangle meshes in mind, is directly applicable to polygon meshes. Only minor modifications need be made in the implementation of the data structures and loop evaluation.

4.1 DATA STRUCTURES

The data structure must contain at least two pieces of information: the geometry, or coordinates, of each vertex, and the definition of each triangle in terms of its three vertices. In addition, because ordered lists of triangles surrounding a vertex are frequently required, it is desirable to maintain a list of the triangles that use each vertex.

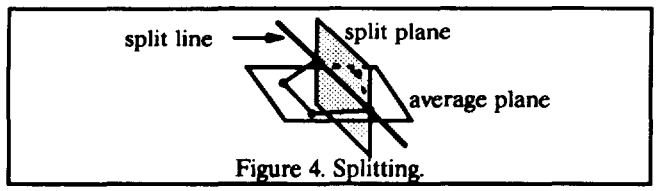

Figure 4. Splitting.

Although data structures such as Weiler's radial edge [17] or Baumgart's winged-edge data structure [1] can represent this information, our implementation uses a space-efficient vertex-triangle hierarchical ring structure. This data structure contains hierarchical pointers from the triangles down to the vertices, and pointers from the vertices back up to the triangles using the vertex. Taken together these pointers form a ring relationship. Our implementation uses three lists: a list of vertex coordinates, a list of triangle definitions, and another list of lists of triangles using each vertex. Edge definitions are not explicit, instead edges are implicitly defined as ordered vertex pairs in the triangle definition.

4.2 TRIANGULATION

Although other triangulation schemes can be used, we chose a recursive loop splitting procedure. Each loop to be triangulated is divided into two halves. The division is along a line (i.e., the split line) defined from two non-neighboring vertices in the loop. Each new loop is divided again, until only three vertices remain in each loop. A loop of three vertices forms a triangle, that may be added to the mesh, and terminates the recursion process.

Because the loop is non-planar and star-shaped, the loop split is evaluated using a split plane. The split plane, as shown in Figure 4, is the plane orthogonal to the average plane (Eqn. 1) that contains the split line. In order to determine whether the split forms two non-overlapping loops, the split plane is used for a half-space comparison. That is, if every point in a candidate loop is on one side of the split plane, then the two loops do not overlap and the split plane is acceptable. Of course, it is easy to create examples where this algorithm will fail to produce a successful split. In such cases we simply indicate a failure of the triangulation process, and do not remove the vertex or surrounding triangle from the mesh.

Typically, however, each loop may be split in more than one way. In this case, the best splitting plane must be selected. Although many possible measures are available, we have been successful using a criterion based on aspect ratio. The aspect ratio is defined as the minimum distance of the loop vertices to the split plane, divided by the length of the split line. The best splitting plane is the one that yields the maximum aspect ratio. Constraining this ratio to be greater than a specified value, .e.g., 0.1, produces acceptable meshes.

Certain special cases may occur during the triangulation process. Repeated decimation may produce a simple closed surface such as a tetrahedron. Eliminating a vertex in this case would modify the topology of the mesh. Another special case occurs when "tunnels" or topological holes are present in the mesh. The tunnel may eventually be reduced to a triangle in cross section. Eliminating a vertex from the tunnel boundary then eliminates the tunnel and creates a non-manifold situation.

These cases are treated during the triangulation process. As new triangles are created, checks are made to insure that duplicate triangles and triangle edges are not created. This preserves the topology of the original mesh, since new connections to other parts of the mesh cannot occur.

5 RESULTS

Two different applications illustrate the triangle decimation algorithm. Although each application uses a different scheme to create an initial mesh, all results were produced with the same decimation algorithm.

5.1 VOLUME MODELING

The first application applies the decimation algorithm to isosurfaces created from medical and industrial computed tomography scanners. *Marching Cubes* was run on a 256 by 256 pixel by 93 slice study. Over 560,000 triangles were required to model the bone surface. Earlier work [3] reported a triangle reduction strategy that used averaging to reduce the number of triangles on this same data set. Unfortunately, averaging applies uniformly to the entire data set, blurring high frequency features. Figure 5 shows the resulting bone isosurfaces for 0%, 75%, and 90% decimation, using a decimation threshold of 1/5 the voxel dimension. Figure 6 shows decimation results for an industrial CT data set comprising 300 slices, 512 by 512, the largest we have processed to date. The isosurface created from the original blade data contains 1.7 million triangles. In fact, we could not render the original model because we exceeded the swap space on our graphics hardware. Even after decimating 90% of the triangles, the serial number on the blade dovetail is still evident.

5.2 TERRAIN MODELING

We applied the decimation algorithm to two digital elevation data sets: Honolulu, Hawaii and the Mariner Valley on Mars. In both examples we generated an initial mesh by creating two triangles for each uniform quadrilateral element in the sampled data. The Honolulu example illustrates the polygon savings for models that have large flat areas. First we applied a decimation threshold of zero, eliminating over 30% of the co-planar triangles. Increasing the threshold removed 90% of the triangles. Figure 7 shows the resulting 30% and 90% triangulations. Notice the transitions from large flat areas to fine detail around the shore line.

The Mars example is an appropriate test because we had access to sub-sampled resolution data that could be compared with the decimated models. The data represents the western end of the Mariner Valley and is about 1000km by 500km on a side. Figure 8 compares the shaded and wireframe models obtained via sub-sampling and decimation. The original model was 480 by 288 samples. The sub-sampled data was 240 by 144. After a 77% reduction, the decimated model contains fewer triangles, yet shows more fine detail around the ridges.

6 CONCLUSIONS

The decimation algorithm significantly reduces the number of triangles required to model an object to a given level of detail. Using local topological and geometric operations, the algorithm makes multiple passes over a triangle mesh, removing vertices and triangulating the resulting holes until user-specified decimation criteria are satisfied. The three step algorithm affords the opportunity to experiment with other data structures, surface approximation metrics, and triangulation schemes. For example, the first step of the decimation could be modified to allow the user to tag some vertices as not-removable. Also, other non-geometric vertex data such as scalar quantities could be used to control the decimation.

We have successfully applied the algorithm to two visualization areas: volume and terrain modeling. We expect that some surface-based analysis techniques, such as boundary element methods or radiosity, will also benefit from the model reductions we have achieved. Here, the computational reduction will be even more significant, since the complexity of analysis is often more than linear with the number of primitives.

7 ACKNOWLEDGEMENTS

Joe Ross, GE Aircraft Engines, supplied the turbine blade data. Lee Moore, Webster Research Center, Xerox Corporation, provided the digital elevation data to the UseNet community through anonymous ftp access. The Mars data, also obtained from Xerox, is courtesy of the NASA Goddard's National Space Science Data Center (NSSDC), the US Geological Survey Astrogeology Division, Flagstaff, Arizona, and the NASA Ames Aerospace Human Factors Division Visualization for Planetary Exploration project.

REFERENCES

[1] Baumgart, B. G., "Geometric Modeling for Computer Vision," PhD Dissertation, Stanford University, August 1974.

[2] Bloomenthal, J., "Polygonalization of Implicit Surfaces," *Computer Aided Geometric Design*, Vol. 5, pp. 341-355, 1988.

[3] Cline, H. E., Lorensen, W. E., Ludke, S., Crawford, C. R., and Teeter, B. C., "Two Algorithms for the Three Dimensional Construction of Tomograms," *Medical Physics*, Vol. 15, No. 3, pp. 320-327, June 1988.

[4] DeHaemer, M. J., Jr. and Zyda, M. J., "Simplification of Objects Rendered by Polygonal Approximations," *Computers & Graphics*, Vol. 15, No. 2, pp 175-184, 1992.

[5] Dunham, J. G., "Optimum Uniform Piecewise Linear Approximation of Planar Curves," *IEEE Trans. on Pattern Analysis and Machine Intelligence*, Vol. PAMI-8, No. 1, pp. 67-75, January 1986.

[6] Finnigan, P., Hathaway, A., and Lorensen, W., "Merging CAT and FEM," *Mechanical Engineering*, Vol. 112, No. 7, pp. 32-38, July 1990.

[7] Fowler, R. J. and Little, J. J., "Automatic Extraction of Irregular Network Digital Terrain Models," *Computer Graphics*, Vol. 13, No. 2, pp. 199-207, August 1979.

[8] Ihm, I. and Naylor, B., "Piecewise Linear Approximations of Digitized Space Curves with Applications," in *Scientific Visualization of Physical Phenomena*, pp. 545-569, Springer-Verlag, June 1991.

[9] Kalvin, A. D., Cutting, C. B., Haddad, B., and Noz, M. E., "Constructing Topologically Connected Surfaces for the Comprehensive Analysis of 3D Medical Structures," *SPIE Image Processing*, Vol. 1445, pp. 247-258, 1991.

[10] Lorensen, W. E. and Cline, H. E., "Marching Cubes: A High Resolution 3D Surface Construction Algorithm," *Computer Graphics*, Vol. 21, No. 3, pp. 163-169, July 1987.

[11] Miller, J. V., Breen, D. E., Lorensen, W. E., O'Bara, R. M., and Wozny, M. J., "Geometrically Deformed Models: A Method for Extracting Closed Geometric Models from Volume Data," *Computer Graphics*, Vol. 25, No. 3, July 1991.

[12] Preparata, F. P. and Shamos, M. I., *Computational Geometry*, Springer-Verlag, 1985.

[13] Schmitt, F. J., Barsky, B. A., and Du, W., "An Adaptive Subdivision Method for Surface-Fitting from Sampled Data," *Computer Graphics*, Vol. 20, No. 4, pp. 179-188, August 1986.

[14] Schroeder, W. J., "Geometric Triangulations: With Application to Fully Automatic 3D Mesh Generation," PhD Dissertation, Rensselaer Polytechnic Institute, May 1991.

[15] Terzopoulos, D. and Fleischer, K., "Deformable Models," *The Visual Computer*, Vol. 4, pp. 306-311, 1988.

[16] Turk, G., "Re-Tiling of Polygonal Surfaces," *Computer Graphics*, Vol. 26, No. 3, July 1992.

[17] Weiler, K., "Edge-Based Data Structures for Solid Modeling in Curved-Surface Environments," *IEEE Computer Graphics and Applications*, Vol. 5, No. 1, pp. 21-40, January 1985.

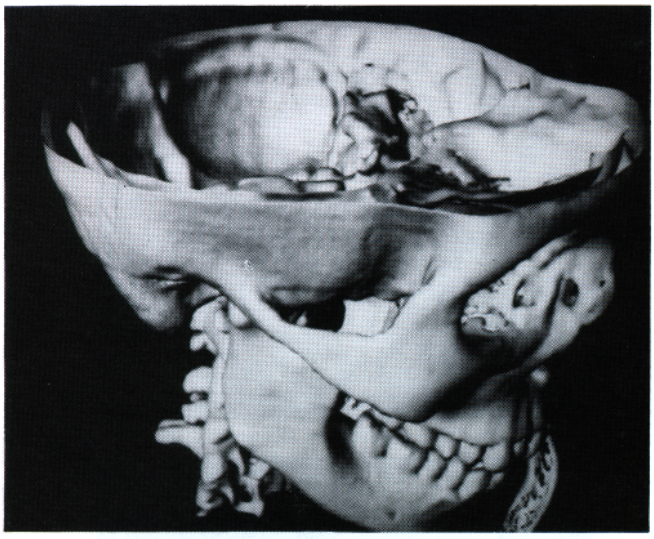

Figure 5a. Full resolution (569k Gouraud shaded triangles).

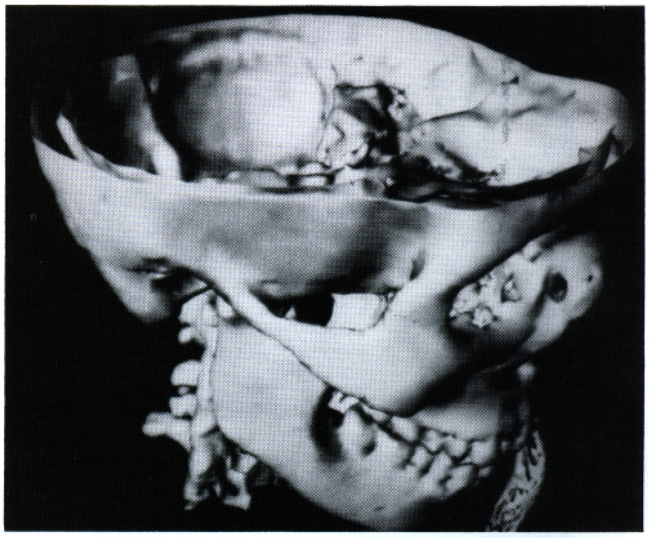

Figure 5b. 75% decimated (142k Gouraud shaded triangles).

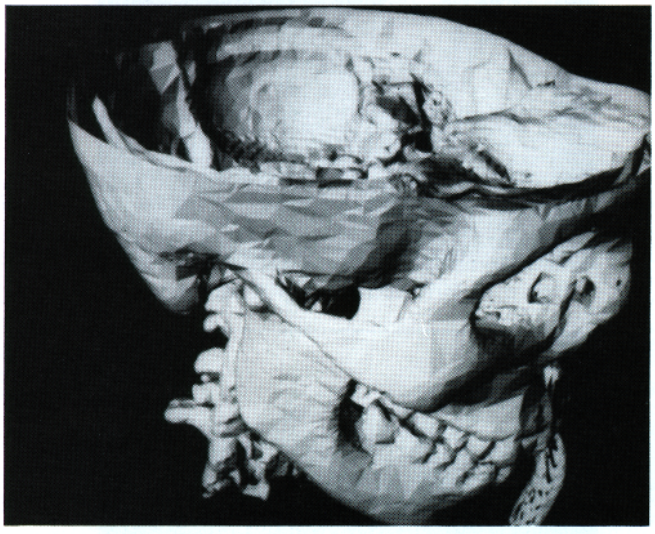

Figure 5c. 75% decimated (142k flat shaded triangles).

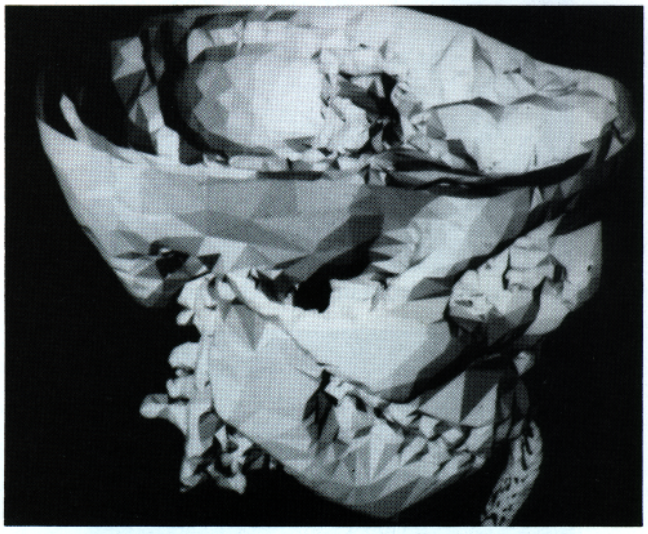

Figure 5d. 90% decimated (57k flat shaded triangles).

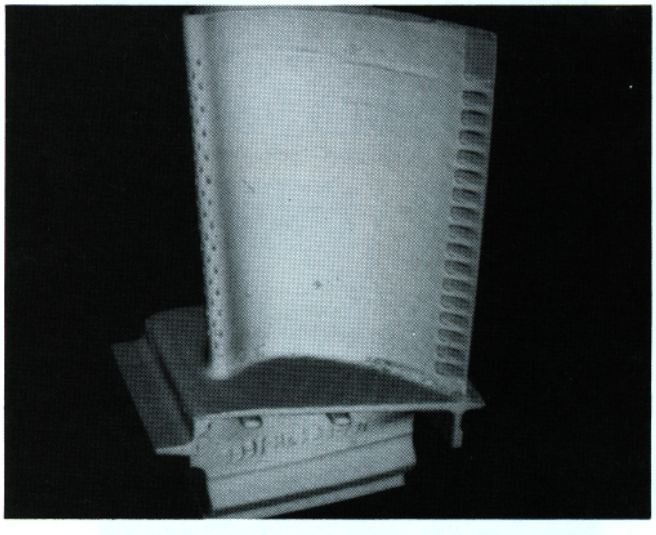

Figure 6a. 75% decimated (425k flat shaded triangles).

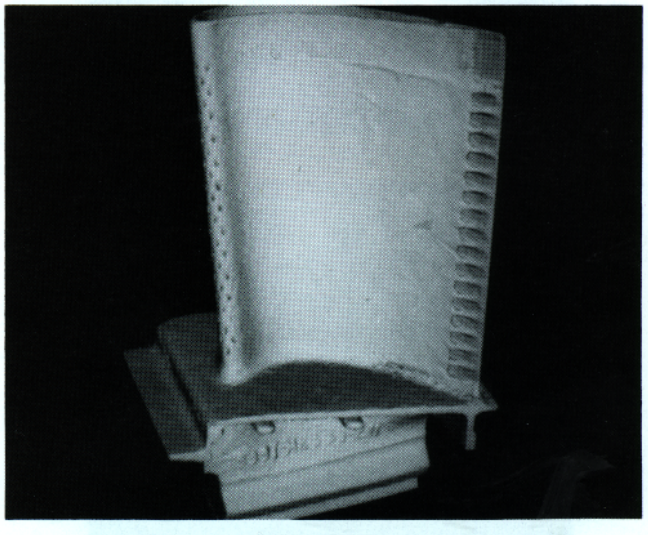

Figure 6b. 90% decimated (170k flat shaded triangles).

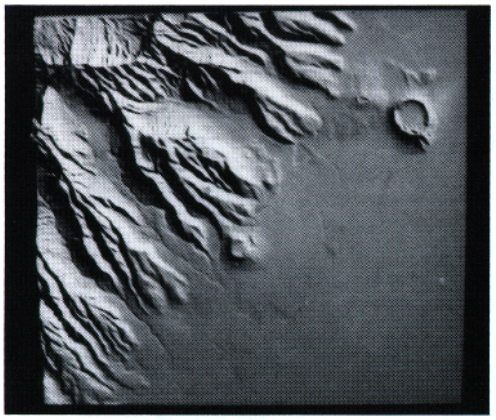

Figure 7a. 32% decimated (276k Gouraud triangles).

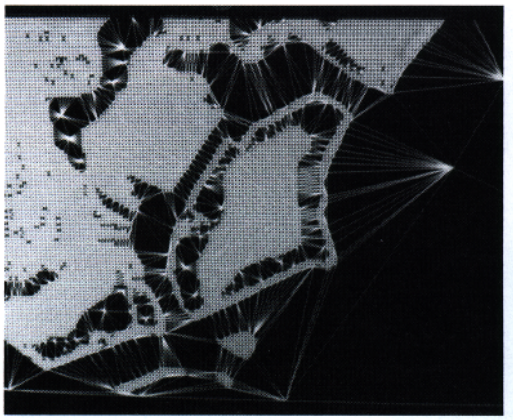

Figure 7b. 32% decimated, shore line detail.

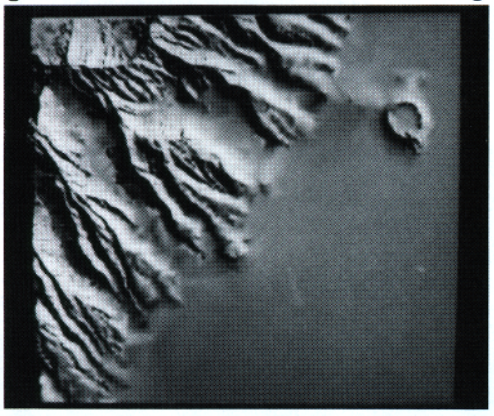

Figure 7c. 90% decimated (40k Gouraud triangles).

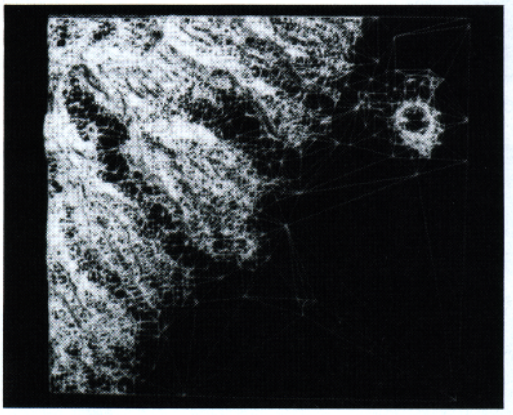

Figure 7d. 90% decimated (40k wireframe).

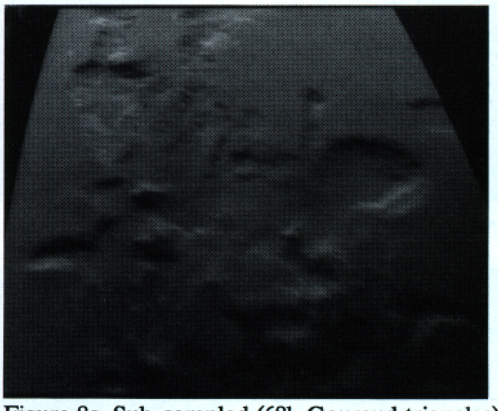

Figure 8a. Sub-sampled (68k Gouraud triangles).

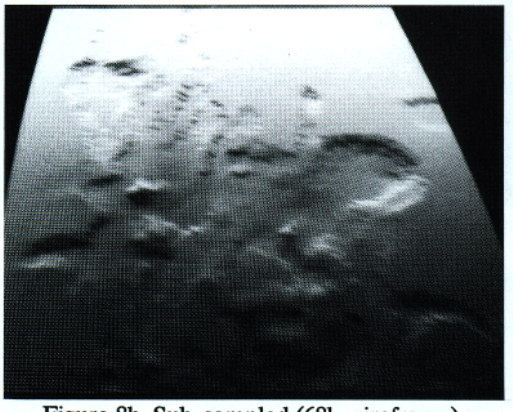

Figure 8b. Sub-sampled (68k wireframe).

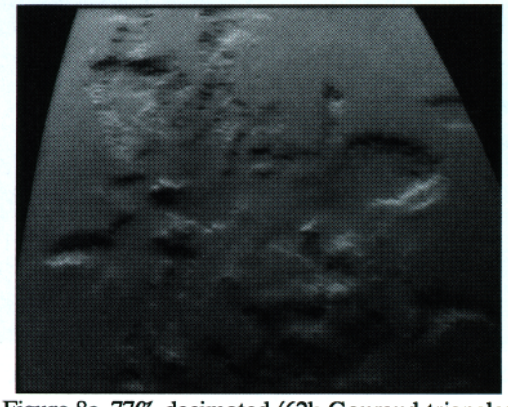

Figure 8c. 77% decimated (62k Gouraud triangles).

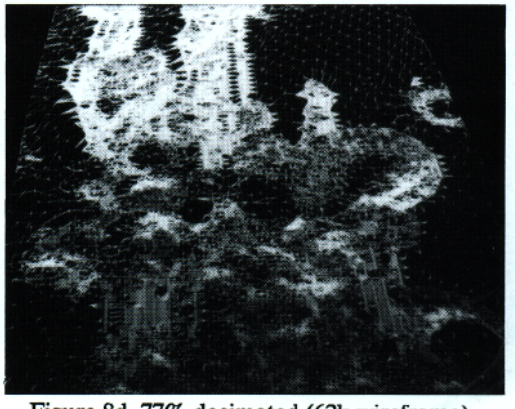

Figure 8d. 77% decimated (62k wireframe).