Decimation of triangle meshes — **Source link** ↗

William J. Schroeder, Jonathan A. Zarge, William E. Lorensen

Institutions: General Electric

Related papers:

- Progressive meshes

- Surface simplification using quadric error metrics

- Mesh optimization

- Marching cubes: A high resolution 3D surface construction algorithm

- Re-tiling polygonal surfaces

Decimation of Triangle Meshes

William J. Schroeder
Jonathan A. Zarge
William E. Lorensen

General Electric Company
Schenectady, NY

1.0   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*. *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.

2.0   THE DECIMATION ALGORITHM

The goal of the decimation algorithm is to reduce the total number of triangles in a triangle mesh, preserving the original topology and a good approximation to the original geometry.

2.1   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.

2.2   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.

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 the figure below.

Simple   Complex   Boundary   Interior Edge   Corner
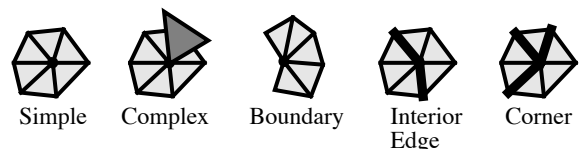
A simple vertex is surrounded by a complete cycle of

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.

2.3   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 (see figure below). If the vertex is within the specified distance to the average plane it may be deleted. Otherwise it is retained.

average plane
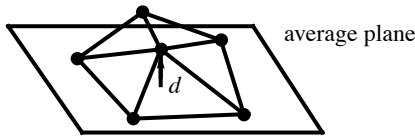
Boundary and interior edge vertices use the distance to edge criterion (figure below). In this case, the algorithm determines the distance to the line defined by the two vertices creating the boundary or feature edge. If the distance to the line is less than d, the vertex can be deleted.

boundary
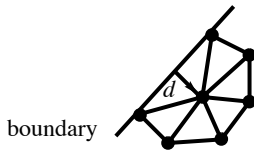
It is not always desirable to retain feature edges. For example, meshes may contain areas of relatively small triangles with large feature angles, contributing relatively little to the geometric approximation. Or, the small triangles may be the result of "noise" in the original mesh. In these situations, 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.

2.4   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, 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 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.

3.0   IMPLEMENTATION

3.1   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.

Although data structures such as Weiler's radial edge or Baumgart's winged-edge data structure 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.

3.2   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 ver-

tices 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 the figure below, is the plane orthogonal to the average plane 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.
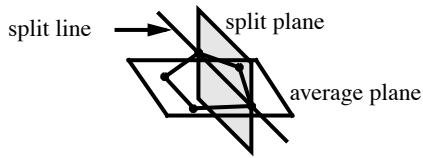
Typically, however, each loop may be split in more than one way. In this case, the best splitting plane must be selected. Although many possible measures are available, we have been successful using a criterion based on aspect ratio. The aspect ratio is defined as the minimum distance of the loop vertices 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.

4.0 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.

4.1 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 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
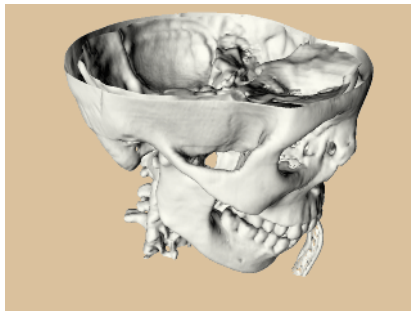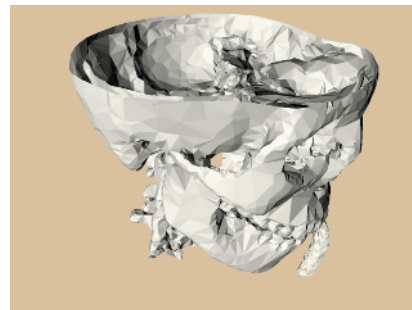

Full Resolution
(569K Gouraud shaded triangles)
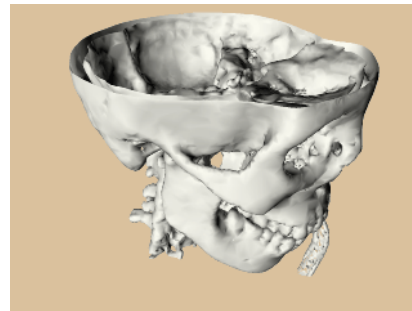
75% decimated
(142K Gouraud shaded triangles)
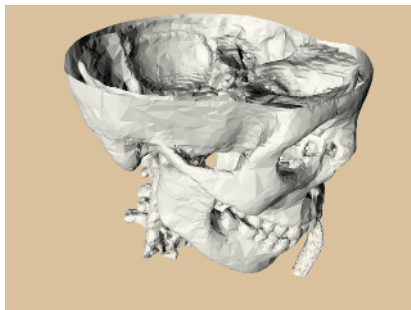
75% decimated
(142K flat shaded triangles)

90% decimated
(57K flat shaded triangles)

high frequency features. The first set of figures shows the resulting bone isosurfaces for 0%, 75%, and 90% decimation, using a decimation threshold of 1/5 the voxel dimension. The next pair of figures 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.

4.2 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. The next set of four figures 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. The last set of figures 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.

5.0 REFERENCES

[1] Baumgart, B. G., "Geometric Modeling for Computer Vision," Ph.D. 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.
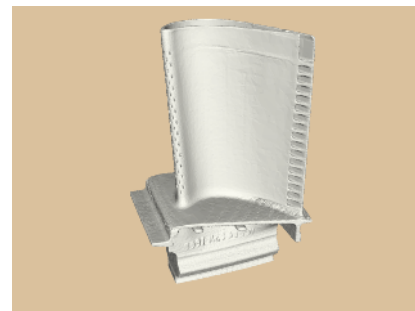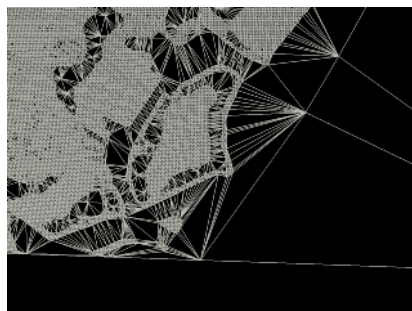
75% decimated
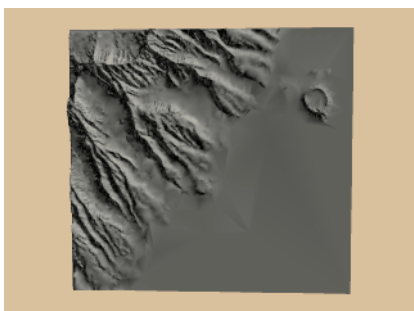(425K flat shaded triangles)
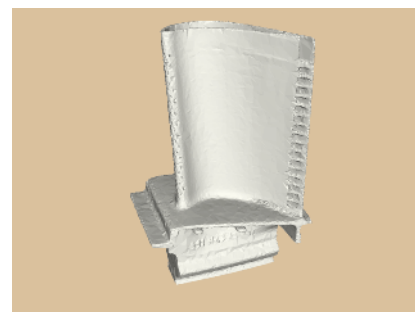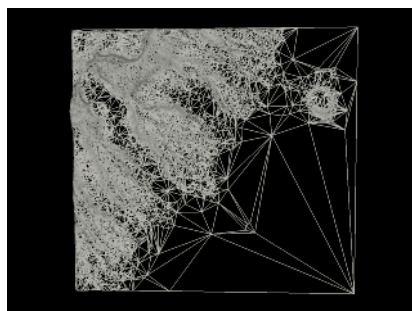

90% decimated
(170K flat shaded triangles)
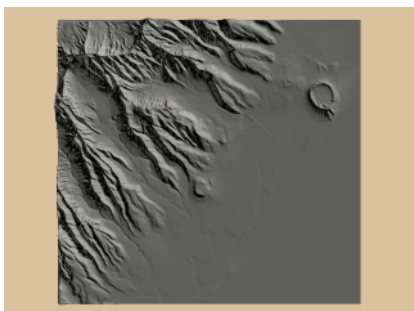
32% decimated
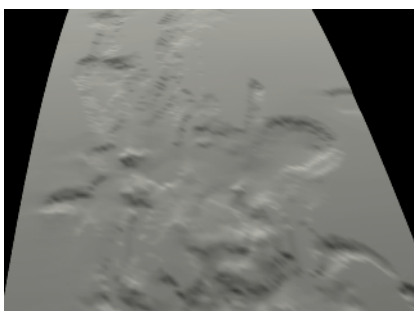(276K flat shaded triangles)
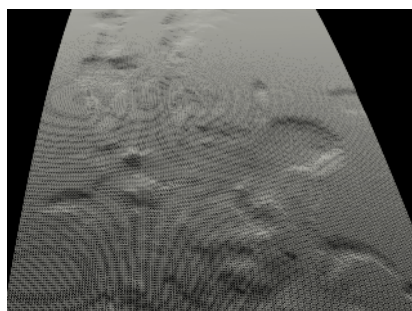
32% decimated
(shore line detail, wireframe)

90% decimated
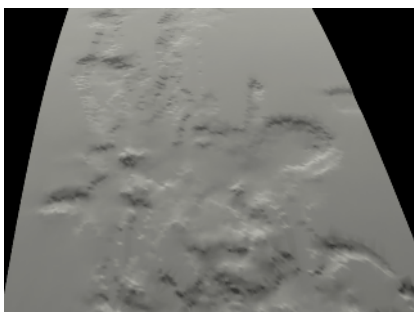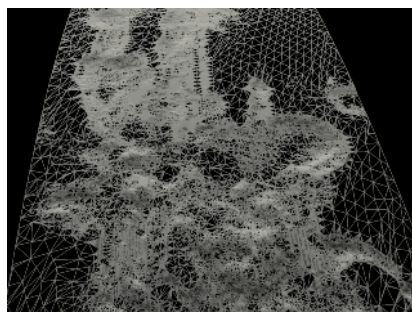(40K Gouraud shaded triangles)

90% decimated
(40K wireframe)

Sub-sampled
(68K Gouraud shaded triangles)

Sub-sampled
(68K wireframe)

77% decimated
(62K Gouraud shaded triangles)

77% decimated
(62K wireframe)