Error-bounded Reduction of Triangle Meshes with Multivariate Data

Chandrajit L. Bajaj
Daniel R. Schikore

Department of Computer Sciences
Purdue University
West Lafayette, IN 47907
bajaj@cs.purdue.edu
drs@cs.purdue.edu

Abstract

Interactive visualization is complicated by the complexity of the objects being visualized. Sampled or computed scientific data is often dense, in order to capture high frequency components in measured data or to accurately model a physical process. Common visualization techniques such as isosurfacing on such large meshes generate more geometric primitives than can be rendered in an interactive environment. Geometric mesh reduction techniques have been developed in order to reduce the size of a mesh with little compromise in image quality. Similar techniques have been used for functional surfaces (terrain maps) which take advantage of the planar projection. We extend these methods to arbitrary surfaces in 3D and to any number of variables defined over the mesh by developing a algorithm for mapping from a surface mesh to a reduced representation and measuring the introduced error in both the geometry and the multivariate data. Furthermore, through error propagation, our algorithm ensures that the errors in both the geometric representation and multivariate data do not exceed a user-specified upper bound.

Keywords: Visualization, Surface on Surface, Mesh Reduction, Decimation.

1 Introduction

There are many tradeoffs in visualizing scientific data. Accuracy of representation and display can be critically important. This factor tends to cause scientific meshes to become very large, in order to accurately represent the underlying data. Interactivity in visualization can greatly enhance the user experience, however real-time interaction with large meshes designed for accuracy is frequently not possible. It is often the case that only a small amount of accuracy can be sacrificed for the sake of increased interactivity with the data, without rendering the visualization useless for interpretation.

Isosurfacing is a common technique for visualizing surfaces in volumetric data [9]. Large computational meshes of very small elements can generate millions of triangles through traditional isosurfacing techniques [15]. A common technique for dealing with the large number of triangles which cannot be rendered interactively is to compute a reduced model in which large triangles replace groups of small triangles which are nearly co-planar.

Related work on planar meshes with scalar data at the nodes aims to reduce the size of the mesh required to represent the scalar field to a defined level of fidelity [2, 3, 4, 8, 12, 16, 17, 19]. By taking advantage of the special case presented by a height map, various algorithms have been developed which create triangular surface approximations while maintaining a user-defined bound on the introduced error.

In visualizing scientific data, it is quite often not only geometry that the user is interested in, but data values defined on the surface as well [1, 10, 11]. Geometric mesh-reduction algorithms may not be well suited to this consideration. Likewise, algorithms developed for data defined in a 2D domain consider only errors introduced in the data, as the mesh domain remains the same through mesh simplification. If one is interested in viewing how a particular variable acts on a geometric surface or a constant surface of another variable, mesh reduction based only on the geometry or based only on the data can destroy the additional information present.

We present an algorithm which produces from an arbitrary surface mesh a reduced model in which errors in both the geometric representation and any number of scalar variables defined at the nodes of the surface mesh are bounded by a user-specified level. A novel method for error propagation between various resolutions of meshes is developed which guarantees error bounds not only between intermediate triangulations but bounds the errors incurred from all steps of mesh simplification. Developing a mesh reduction scheme which is driven by mesh quality, and not mesh resolution, is a major driving factor. The error representations used allow intuitive user guidance of or complete automation of the mesh reduction process. Reduced meshes may be nested, allowing smooth interpolation between several levels of detail. Results on surface meshes related to nuclear and mechanical physics demonstrate that the algorithm provides a significant reduction in mesh size while sacrificing little in the quality of the display, even in the presence of multivariate data.

2 Related Work

Mesh reduction is a general category of techniques designed to remove redundant information from a mesh. There has been related work in areas of reduction of 2D meshes based on data values at the vertices (height maps), as well as reduction of triangular meshes in 3D based on geometric constraints.

There has been a great deal of work by Geographical Information Systems (GIS) researchers interested in reducing the complexity of dense Digital Elevation Models (DEMs). A common technique is to extract key points of data from the originally dense set of points, and compute a Delaunay triangulation [2, 3, 4, 12, 17, 19]. Silva, et. al[16] use a greedy method for inserting points into an initially sparse mesh. A survey by Lee [8] reviews methods for computing reduced meshes by both point insertion and point deletion.

Geometric mesh reduction has been approached from several directions. In reduction of polygonal models, Turk [18] used point repulsion on the surface of a polygonal model to generate a set of vertices for retriangulation, computed a mutual tesselation as an intermediate form, and finally deleted the original vertices to give a reduced mesh. This method allows the user to specify how many vertices to place on the surface, in order to reduce the model to a desired resolution. Geometric detail was maintained by adjusting the repulsion at regions of high curvature. Schroeder, et al. [15] decimate dense polygonal meshes, generated by Marching Cubes[9], by deletion of vertices based on an error criteria, followed by local retriangulation with a goal of maintaining good aspect ratio in the resulting triangulation. Errors incurred from local retriangulation are not propagated to the simplified mesh, hence there is no global error control. Rossignac, et al.[14] uses clustering and merging of features of an object which are geometrically close, but may not be topologically connected. In this scheme, long thin objects may collapse to an edge and small objects may collapse to a point. Hamann[5] applies a similar technique in which triangles are considered for deletion based on curvature estimates at the vertices. Reduction may be driven by mesh resolution or, in the case of functional surfaces, root-mean-square error. He, et al. [6] perform mesh reduction by sampling and low-pass filtering an object. A multi-resolution triangle mesh is extracted from the resulting multi-resolution volume buffer using traditional isosurfacing techniques. Hoppe, et al. [7] perform time-intensive mesh optimization based on the definition of an energy function which balances the need for accurate geometry with the desire for compactness in representation. The level of mesh reduction is controlled by a parameter in the energy function which penalizes meshes with large numbers of vertices, as well as a spring constant which helps guide the energy minimization to a desirable result.

Neither the work on 2D data decimation nor 3D geometric decimation is directly applicable to arbitrary surface meshes with data. In this paper we describe a method for merging the methods of geometric mesh reduction for surfaces with 2D functional surface mesh reduction methods. The result allows error-bounded mesh reduction of geometric surfaces with multivariate data defined on the surface.

3 Reduction of Surfaces With Multivariate Data

Reduction of surface meshes with multivariate data requires combining and extending work in several areas. Our approach combines the efforts in geometric mesh reduction and planar reduction of meshes with data, as well as extending the reduction to multivariate data. The primary goal of planar mesh reduction is to reduce the number of elements in a planar mesh while maintaining an error bound on the value of a variable at all points in the domain. The goal of surface mesh reduction is to reduce the number of elements while remaining as close as possible to the original geometry of the mesh without violating the mesh topology. Combining the goals of the two methods, we aim to control the geometry and topology of a surface mesh, as well as the values of variables defined on the mesh.

3.1 Algorithm Overview

The algorithm for reduction follows the basic strategy of other “vertex deletion” schemes. From an initially dense surface mesh, vertices are considered as candidates for deletion. A candidate vertex is deleted if a valid retriangulation of the hole which results from deletion can be found. A valid retriangulation must maintain the topology of the original mesh, and the sum of the propagated errors and errors introduced from the deletion must be within user-specified bounds.

All vertices are candidates for removal, and may deleted so long as their removal does not violate the user-defined constraints. The error minimization scheme employed in retriangulation will automatically maintain edges in both the geometry and the data by choosing a retriangulation which is close to the original data. Note that we consider only the cases of meshes which are 2-manifold, which are common in scientific data. For non-manifold meshes, classification may be required in order to determine if a vertex should be considered for deletion.

3.2 Mapping between triangulations

In planar mesh simplification, errors are computed by mapping deleted vertices to the reduced surface using a projection in the direction orthogonal to the plane. In the more general case of a surface mesh, there is no identical mapping, and so we must define a mapping from a triangulation around a candidate vertex to a triangulation resulting from deletion of the candidate. This mapping will allow us to quantify errors in both the geometry and the data defined on the mesh.

Several criteria are important in developing a method for mapping between triangulations. First, a desirable mapping must not result in a triangle from the original mesh projecting to a degenerate line or point on the reduced triangulation. Such cases are considered undesirable because multiple points from one triangulation map to the same point in the reduced triangulation, which amounts to a singularity in the projection. A simple mapping which projects vertices to the nearest vertex in the reduced triangulation is not acceptable, as there may be many points which map to a single vertex. Instead we look for a mapping which provides a better sense of geometric continuity between the various triangulations. Intuitively, we want to use projections which require as little 'stretching' of the mesh as possible, and stretch equally in all directions rather than performing a skewed projection.

There are two types of candidate vertices, shown in figure 1, which are projected to the new triangulation using different criteria.

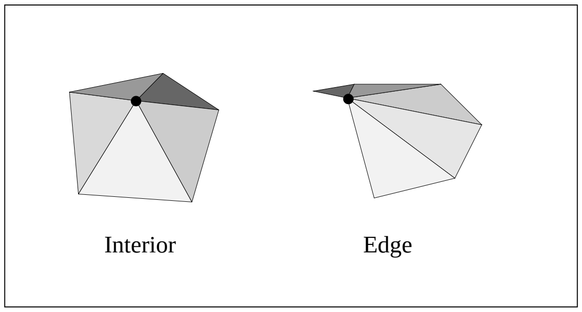


Figure 1: Types of candidate vertices

3.2.1 Interior Edges

Consider a candidate edge for retriangulation which spans the hole being triangulated. The edge is mapped to the original surface in the direction of the average plane of the hole. The result is a piecewise continuous series of line segments lying on the original surface, as shown in figure 2. The dark edges are the projections of the retriangulation onto the original surface. In order to ensure equal stretching, the segments, when projected back into the new triangulation, maintain proportional lengths.

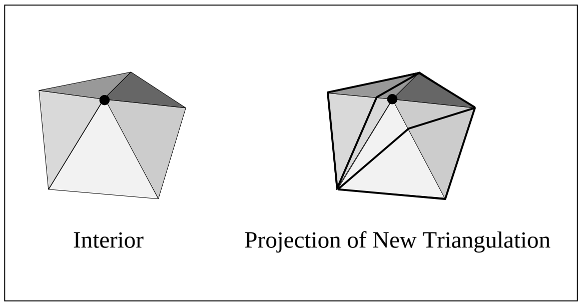


Figure 2: Mapping for an interior vertex

3.2.2 Edge vertices

In the case of an edge vertex, there is a special case for the edge which results from removal of the vertex. This edge is projected to the boundary edge of the original triangulation. Again, the segments along the edge and their projections in the plane maintain proportional lengths. Shown in figure 3, the dark edges represent the retriangulation of the hole projected onto the original surface.

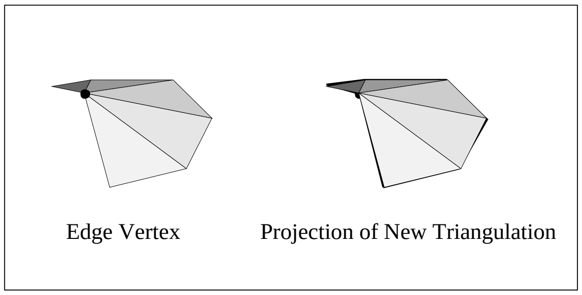


Figure 3: Mapping for an edge vertex

3.2.3 Topological checks

Simple projection methods do not guarantee a topologically correct retriangulation. Turk [18] and Schroeder, et al. [15] consider two necessary topological checks. First, a projection in the direction of the average normal may cause the surface to fold over on itself, changing the surface topology. This folding manifests itself by projections of edges which intersect on the original surface or which intersect the original surface in multiple places. Another topological problem results in areas in which the deletion of a vertex causes a narrow channel in the surface to close. This is another case which is easy to detect by examining the adjacent triangles for all vertices in the hole being triangulated. If a triangle exists with all three vertices lying on the outer edge of the hole, deletion of the candidate vertex may result in a topological change in the surface, and therefore the candidate vertex is not deleted. These two cases are easily identified, preventing a topologically incorrect mesh due to mesh simplification.

A more subtle topological error may occur when two different lobes of the surface pass very near one another. A resulting local triangulation near one lobe may result in triangles which are pierced by the other lobe. We provide a means to ensure that this does not occur by computing a feature size for a mesh, which is defined as the minimum distance between two separate lobes of the surface. By limiting the geometric error bound to be within the feature size, we guarantee that no piercing of the mesh will occur. For increased mesh reduction, this method could be replaced by bounding box computations at each vertex, in order to guarantee that retriangulation will not generate a topological error. The last two topological considerations are illustrated in figure 4.

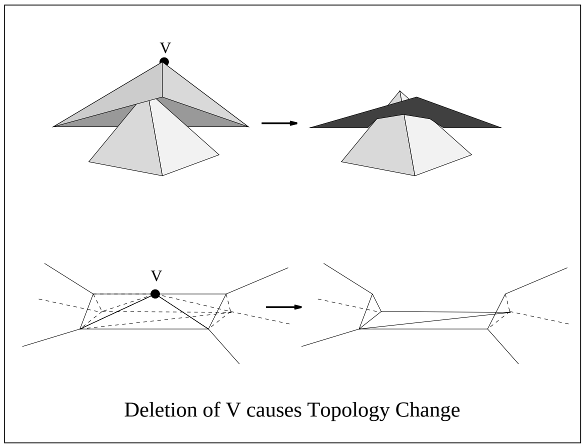


Figure 4: Topology Considerations

3.3 Computing Error Bounds

The previously defined mapping from a triangulation around a candidate vertex to a retriangulation permits us to compute errors introduced over the surface. The mutual projection segments the triangulation into pieces within which the variables and the geometry all vary linearly, as shown in figure 5. Thus, it suffices to compute errors at the intersections of the projected edges in order to compute an upper bound on the error in each triangle of the new mesh.

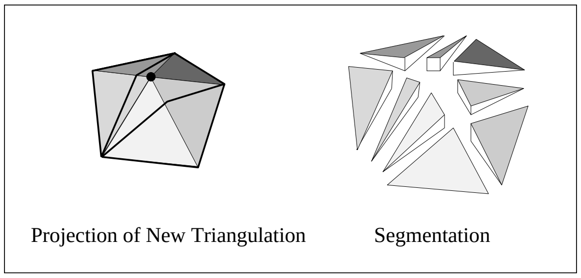


Figure 5: Segmentation of mutual projection

3.3.1 Geometric Error

Errors in the geometry are quantified by the signed distance spanned by the mapping from one triangulation to another. We use the convention that a displacement toward the “outside” (in the direction of the normal) of a mesh is a positive displacement, while displacement toward the “inside” is a negative displacement. In this way, we are able to compute the introduced geometric error incurred through retriangulation.

3.3.2 Data Errors

Computing the error in the variables is similarly computed at the intersections of the projected triangulation. In order to compute the values of the variables, linear interpolation is used along the edges which intersect. Errors are measured as the difference between the interpolated value in the old triangulation and the interpolated value in the new triangulation. Thus, a data value which interpolates to a lower value in the new triangulation introduces an error of positive magnitude, while a data value which is increased introduces an error of negative magnitude.

3.3.3 Error Propagation

We choose a simple scheme for propagation of error from one triangulation to the next. For the geometry and each scalar variable, we associate two variables with each triangular face. One variable is a current upper bound on the “positive” error accumulated, the other represents “negative” error. Given the accumulated errors on the faces surrounding a candidate vertex, we compute and add in the errors which are introduced through deleting the vertex. Figure 6 illustrates the process of computing an upper bound on the error along an introduced edge based on propagated error from the previous triangulation and introduced error caused by retriangulation.

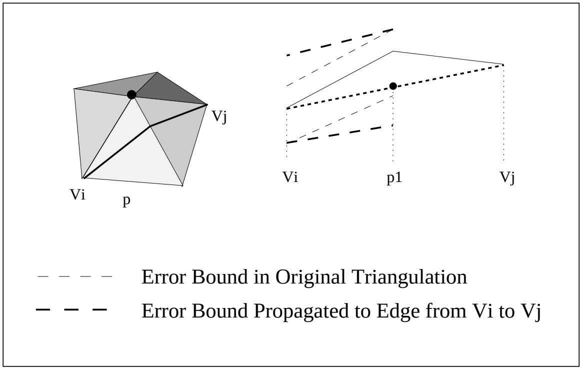

Vj
Vi
p
Vi
p1
Vj
Error Bound in Original Triangulation
Error Bound Propagated to Edge from Vi to Vj

Figure 6: Propagating error to a new triangulation

3.4 Retriangulation with Multivariate Control

Given the above methods for projecting a retriangulated surface to the previous triangulation, we now present an outline for an edge-flipping method for computing a triangulation which introduces as little error as possible.

We begin by computing a valid retriangulation in a greedy manner. Each edge is projected to the previous triangulation and errors are computed both for the geometry and each of the scalar variables on the mesh. Edges are examined in turn, and each edge is considered as a candidate to be flipped. If the flipping of an edge does not violate the topology of the surface, the flipped edge is projected and errors are computed. If the flipped edge has lower cost (defined below) than the candidate edge, the candidate is removed and the flipped edge is introduced. This process continues until no more edges are flipped.

After all edges are flipped, errors can be determined for each triangle in the resulting triangulation. Again, the linear definition of the mesh and variables allows us to bound the error over a triangle by the error bound on the edges which make up the triangle. The only exception is the triangle which maps to the vertex being deleted. Additional error is incurred at this point which has not yet been accounted for. We project the candidate vertex into the triangle, and compute introduced errors in a manner similar to the method described for computing introduced errors at intersections on the surface. The propagated error for the deleted point is the maximum of the propagated errors for the triangles it belongs to, as there are points from all original triangles which project to this triangle. As before, the propagated and introduced error are combined and propagated to the new triangulation.

With errors computed for each new triangle, we now determine whether the resulting triangulation exceeds any of the error bounds specified by the user. If the triangulation is valid, the candidate vertex and all neighboring triangles are deleted, and the new triangulation is added to the mesh.

3.4.1 Computing Costs

As we are considering errors in multivariate data as well as in geometry, there are many definitions of cost which may be considered. We suggest two methods for computing the cost between two points which project to each other:

maximum error - Using the maximum error as a measure of cost is a logical choice. We want to avoid situations in which error bounds for a triangle come near to the error parameters specified by the user, as this is likely to hinder further mesh reduction in this region.

maximum introduced error - By using the maximum introduced error rather than the maximum error, we take into consideration all error parameters rather than computing costs based only on the parameter which is nearest the user specified bounds. Consider the case in which the relative error in the geometry is near the limit, but errors in the variables are very low. One valid retriangulation may increase the geometric error only slightly, while errors in other variables increase dramatically, but remain within the limits. By computing costs using introduced errors, we bias the retriangulation toward configurations where all associated errors are allowed to increase slightly, but no error increases drastically more than the others.

4 Multiple Levels of Detail

A desirable trait for a mesh reduction algorithm is the ability to create multiple nested levels of detail for an object, which can be smoothly interpolated and blended. Such methods are frequently used in real-time rendering systems such as a flight simulator, so that objects which are far away and appear small to the user may be rendered using a coarse resolution model [13]. As the object nears the user, the model can be smoothly blended to a more detailed resolution. Such methods would also be very useful in a navigable visualization environment. We describe a method to generate nested sets of models and interpolate between them.

The error control in the mesh reduction described previously is driven by a mapping from one triangulation to another. Using this fact, we can develop a simple method for interpolating between multiple levels of triangulation.

The first step is to generate multiple nested representations for a surface mesh. In a vertex removal mesh reduction, this is easily accomplished by generating meshes in the order of the most detailed to the most simplified mesh. At each stage of the mesh reduction, we use the previous mesh as a starting point and further reduce the mesh. We are guaranteed that the lower resolution meshes will contain only vertices from the higher resolutions.

In order to properly interpolate between the representations, we need only to store a description of the mapping which was used to create the reduced representations. When a re-triangulation is projected to the original surface, the surface is segmented into linear pieces (Figure 5). When interpolating between a higher and lower resolution model, it is these linear pieces which are interpolated from one resolution to the other. When the interpolation reaches the higher or lower resolution, the pieces can be replaced with the exact resolution model.

5 Conclusions

We have described a method for computing simplified meshes for triangular surfaces with data, while maintaining upper bounds on the error introduced to the geometry and the multivariate data defined on the mesh. In addition, we have described a procedure by which a nested triangulation can be obtained, allowing smooth interpolation between meshes of varying resolution. The user-specified error bounds for geometry and data are intuitive, making it easy to guide or automate the mesh reduction process resulting in a mesh of desirable quality.

Several example surfaces illustrate the utility of the algorithm. Figure 7 demonstrates the mesh reduction on a constant surface of density in a pion collision simulation. The surface was extracted through isosurfacing of a 70 x 40 x 25 volume of data. Seven variables, including density, pressure, and component velocities were all interpolated to the isosurface. Reduced meshes were computed with relative error bounds of 3% and 6%. Three pseudocolored surfaces are shown, representing three of the variables in the mesh. The reduced meshes are 70% - 80% smaller than the original mesh, while maintaining the geometry and data on the mesh.

In figure 8, an isosurface was extracted from a 54 x 24 x 24 volume mesh from a projectile impact simulation. Six variables defined on the mesh were interpolated to the surface. Meshes of reduced resolution were computed with relative error bounds of 3% and 9%. Three pseudocolored surfaces show the values of three of the variables on the original mesh as well as the simplified meshes. Mesh simplification reduced the size of the meshes by 50% - 70%.

6 Acknowledgements

We are grateful to Lawrence Livermore National Lab for access to the pion collision and projectile impact data sets. This research was supported in part by NSF grants CCR 92-22467 and GER-9253915-02, AFOSR grants F49620-93-10138 and F49620-94-10080, and ONR grant N00014-91-1-0370.

References

[1] C. Bajaj and G. Xu. Modeling Scattered Function Data on Curved Surface. In J. Chen, N. Thalmann, Z. Tang, and D. Thalmann, editor, *Fundamentals of Computer Graphics*, pages 19 – 29, Beijing, China, 1994.

[2] L. DeFloriani, B. Falcidieno, and C. Pienovi. Delaunay-based representation of surfaces defined over arbitrarily shaped domains. *Computer Vision, Graphics and Image Processing*, 32:127–140, 1985.

[3] L. De Floriani, B. Falcidieno, G. Nagy, and C. Pienovi. A hierarchical structure for surface approximation. *Computers and Graphics*, 8(2):183–193, 1984.

[4] R. J. Fowler and J. J. Little. Automatic extraction of irregular network digital terrain models. In *Computer Graphics (SIGGRAPH '79 Proceedings)*, volume 13(3), pages 199–207, August 1979.

[5] B. Hamann. A data reduction scheme for triangulated surfaces. In *Computer Aided Geometric Design*, volume 13, pages 197–214, 1994.

[6] T. He, L. Hong, A. Kaufman, A. Varshney, and S. Wang. Voxel based object simplification. In G. M. Nielson and D. Silver, editors, *Visualization '95 Proceedings*, pages 296–303, October 1995.

[7] 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.

[8] J. Lee. Comparison of existing methods for building triangular irregular networks. *Int. Journal of Geographical Information Systems*, 5(2):267–285, 1991.

[9] William E. Lorensen and Harvey E. Cline. Marching cubes: A high resolution 3D surface construction algorithm. In Maureen C. Stone, editor, *Computer Graphics (SIGGRAPH '87 Proceedings)*, volume 21, pages 163–169, July 1987.

[10] Nelson Max, Roger Crawfis, and Charles Grant. Visualizing 3d velocity fields near coutour surfaces. In *Visualization '94*, pages 248–255, October 1994.

[11] Gregory M. Nielson. Scattered data modeling. *IEEE Computer Graphics and Applications*, 13(1):60–70, January 1993.

[12] E. Puppo, L. Davis, D. DeMenthon, and Y.A. Teng. Parallel terrain triangulation. *Int. Journal of Geographical Information Systems*, 8(2):105–128, 1994.

1994), Computer Graphics Proceedings, Annual Conference Series, pages 381–395. ACM SIGGRAPH, ACM Press, July 1994. ISBN 0-89791-667-0.

[14] J. Rossignac and P. Borrel. Multiresolution 3d approximations for rendering complex scenes. In B. Falcidieno and T. L. Kunii, editors, *Modeling in Computer Graphics*, pages 455–465, 1993.

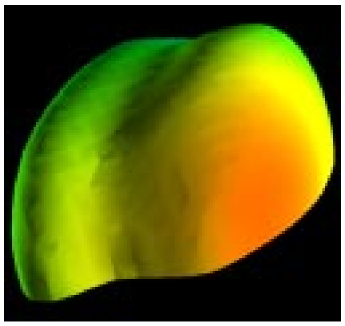
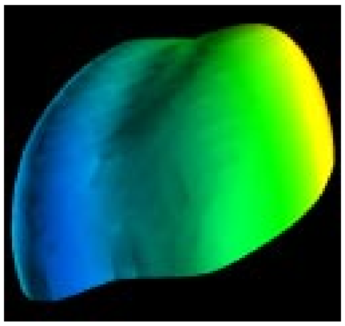
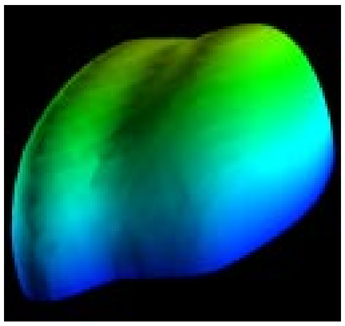
[15] William J. Schroeder, Jonathan A. Zarge, and William E. Lorensen. Decimation of triangle meshes. In Edwin E. Catmull, editor, *Computer Graphics (SIGGRAPH '92 Proceedings)*, volume 26(2), pages 65–70, July 1992.

[16] C. Silva, J. Mitchell, and A. Kaufman. Automatic generation of triangular irregular networks using greedy cuts. In G. M. Nielson and D. Silver, editors, *Visualization '95 Proceedings*, pages 201–208, October 1995.

[17] Victor J.D. Tsai. Delaunay triangulations in tin creation: an overview and a linear-time algorithm. *Int. Journal of Geographical Information Systems*, 7(6):501–524, 1993.

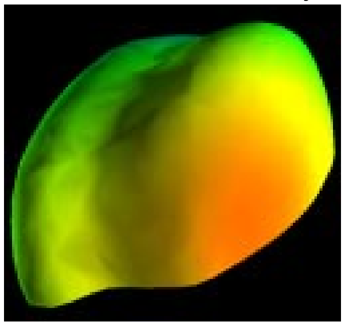
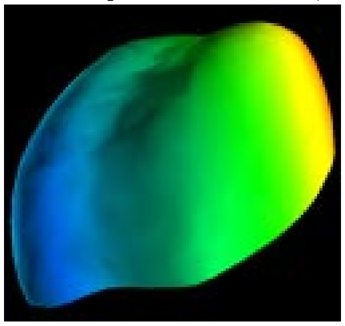
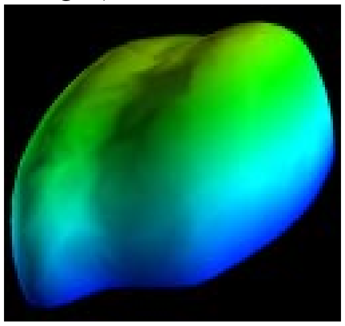
[18] Greg Turk. Re-tiling polygonal surfaces. In Edwin E. Catmull, editor, *Computer Graphics (SIGGRAPH '92 Proceedings)*, volume 26(2), pages 55–64, July 1992.

[19] J. Mark Ware and Christopher B. Jones. A multiresolution topographic surface database. *Int. Journal of Geographical Information Systems*, 6(6):479–496, 1992.

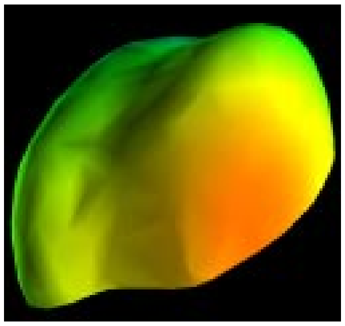
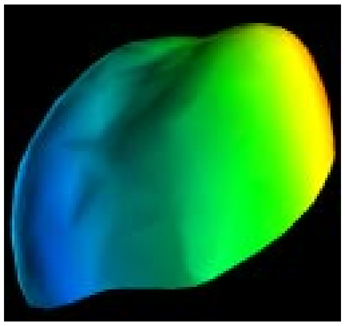
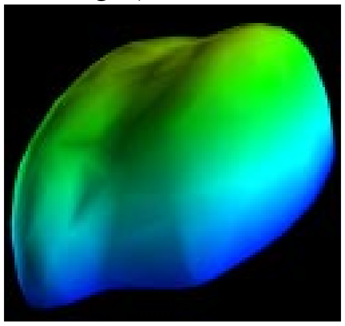
(a) (b) (c)

Density surfaces from pion collision data (3450 triangles)

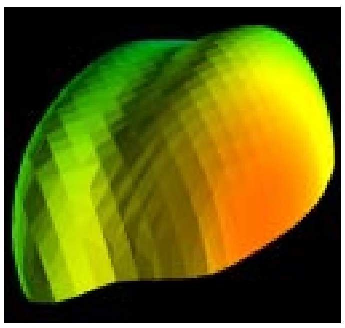
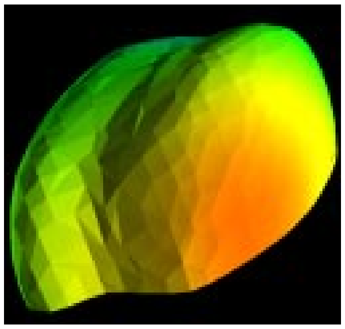
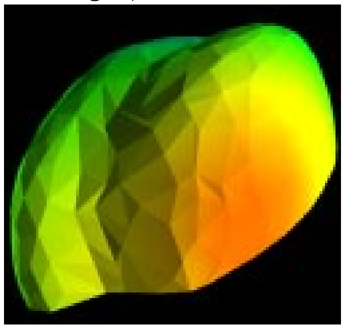
(d) (e) (f)

Reduced surfaces with 3% relative error bounds (1043 triangles)

(g) (h) (i)

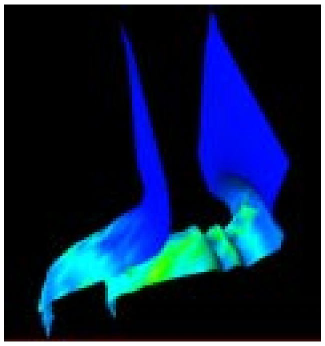
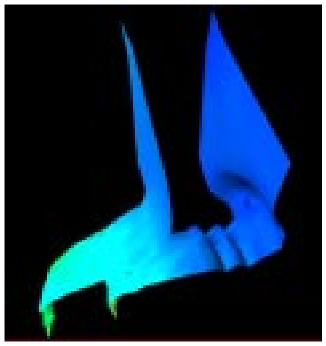
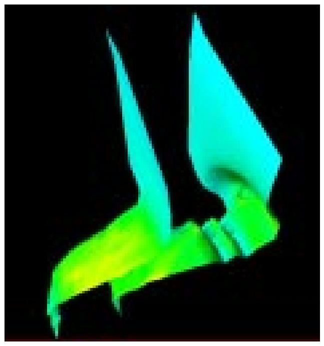
Reduced surfaces with 6% relative error bounds (660 triangles)

(j) (k) (l)

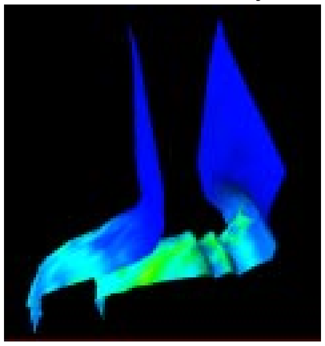
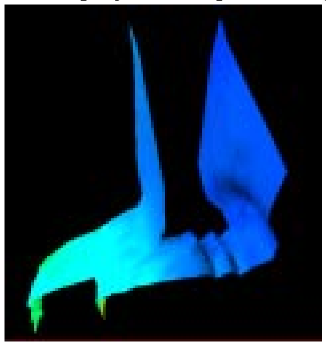
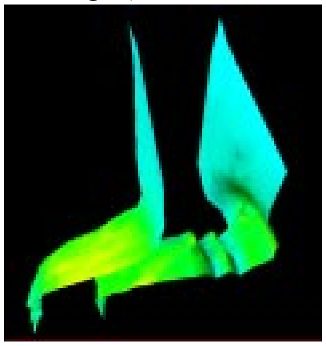
Flat shaded surfaces of (a), (d), and (g) reveal the tesselation

Figure 7: Original surface data with two levels of mesh reduction, shown for 3 variables

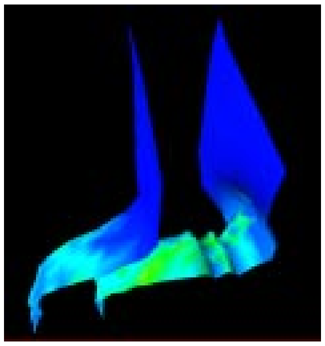
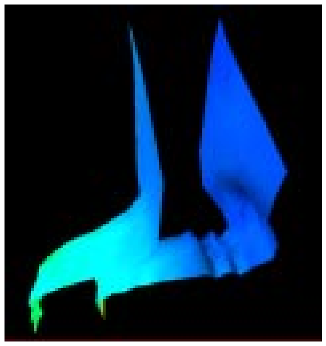
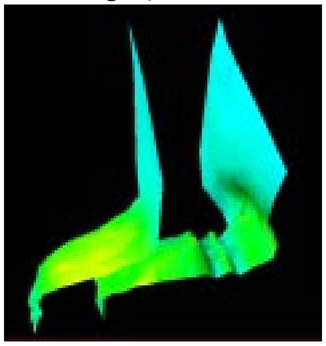
(a) (b) (c)

Density surfaces from projectile impact data (2869 triangles)

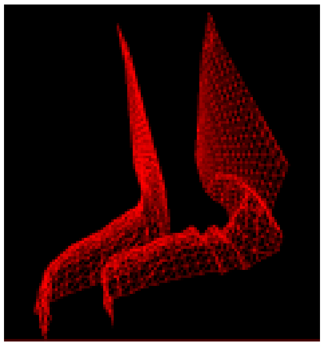
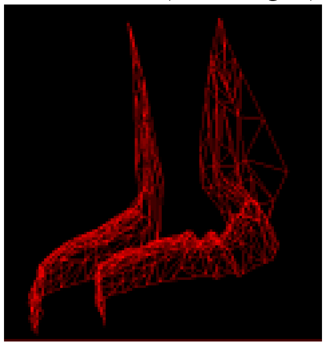
(d) (e) (f)

Reduced surfaces with 3% relative error bounds (1356 triangles)

(g) (h) (i)

Reduced surfaces with 9% relative error bounds (844 triangles)

(j) (k)

Wireframe surfaces of (a-c) and (g-i)

Figure 8: Original surface data with two levels of mesh reduction, shown for 3 variables