Open access • Journal Article • DOI:10.1145/1061347.1061350

Quadric-based simplification in any dimension — Source link

Michael Garland, Yuan Zhou

Institutions: University of Illinois at Urbana–Champaign

Published on: 01 Apr 2005 - ACM Transactions on Graphics (ACM)

Topics: Quadric, Surface (mathematics), Metric (mathematics), Generalization and Dimension (graph theory)

Quadric-Based Simplification in Any Dimension

MICHAEL GARLAND and YUAN ZHOU
University of Illinois at Urbana–Champaign

We present a new method for simplifying simplicial complexes of any type embedded in Euclidean spaces of any dimension. At the heart of this system is a novel generalization of the quadric error metric used in surface simplification. We demonstrate that our generalized simplification system can produce high quality approximations of plane and space curves, triangulated surfaces, tetrahedralized volume data, and simplicial complexes of mixed type. Our method is both efficient and easy to implement. It is capable of processing complexes of arbitrary topology, including non-manifolds, and can preserve intricate boundaries.

Categories and Subject Descriptors: I.3.5 [**Computer Graphics**]: Computational Geometry and Object Modeling—*surface and object representations*

Additional Key Words and Phrases: quadric error metric, surface simplification, edge contraction

1. INTRODUCTION

As the size of the geometric datasets commonly used in graphics systems has continued to grow very rapidly, automatic simplification techniques have become a critical tool for efficiently managing this abundance of data. This is true across a broad spectrum of applications. Surface models produced by scanning and reconstruction systems are almost always inefficiently over-sampled. Archival GIS databases, such as nationwide road networks or political boundaries, are often stored at a precision far higher than necessary or desirable for display to the user. Scientific simulations can easily produce gigabytes of data that cannot be efficiently visualized without a significant reduction in the raw data size. Finally, in resource constrained environments, it is vital to use as few geometric elements as possible, even though the original models may have been designed at a high resolution.

Over the past several years, a multitude of simplification techniques have been developed, focusing on various problem domains. In particular, most relevant work has been focused on three distinct types of geometric data: plane/space curves, triangulated surfaces, and tetrahedralized volumes. However, comparatively little effort has been directed towards algorithmic techniques that can successfully simplify *all* such data. And with one notable exception [Popović and Hoppe 1997], the problem of simplifying objects of mixed dimension (e.g., containing both lines and triangles) has remained unaddressed.

In this paper, we present a new simplification method that can produce approximations of simplicial complexes of any type embedded in Euclidean spaces of arbitrary dimension. In particular, we demonstrate that our technique can be applied to plane curves, space curves, surfaces, tetrahedral volume data, and mixed complexes. Our approach is both flexible and efficient, producing high quality approximations of its input data. It makes no restriction on the topological type

Email: {garland,yuanzhou}@cs.uiuc.edu Address: 201 North Goodwin Ave., Urbana, IL 61801

Technical Report No. UIUCDCS-R-2004-2450, June 2004.

of the input, and can easily process non-manifold objects. Our system can also faithfully preserve any number of per-vertex scalar fields defined over the input complex. All of this can be implemented cleanly and easily. Indeed, by using the generic programming features of the C++ language, we have been able to produce a single templatized simplification code to handle all input types. At the heart of our approach is a novel generalization of the well-known quadric error metric [Garland and Heckbert 1997; Garland 1999b].

2. BACKGROUND

Unless noted otherwise, we assume that we are given a manifold simplicial complex $M = (V, F)$, composed of a set of d-simplices F and a set of vertices V embedded in the Euclidean space E^n. The complex M defines a surface of dimension d and co-dimension $n - d$. Our goal will be to produce another d-manifold that faithfully preserves the shape of M using far fewer simplices.

We make no restrictive assumptions about the topological complexity of the input M. It may be of arbitrary genus, and it may have any number of boundaries. We will however generally assume that M is a *pure complex* — that no $(d - i)$-simplex occurs in M except as a subsimplex of a d-simplex. A simplex that is not a subsimplex of any other simplex is referred to as a *maximal simplex*. Thus, a pure d-complex is one in which all the maximal simplices are of dimension d. In order to streamline our discussion, we are also restricting our attention to manifolds. However, as we shall see the techniques we describe can be easily applied to non-manifold (§5.3) and mixed (§6.5) complexes as well. Even extensions to non-simplicial meshes (§8.6) are possible.

In practice, we are primarily interested in complexes composed of either edges, triangles, or tetrahedra. This corresponds to the kinds of geometric models most frequently used in graphics and visualization applications: plane/space curves, polygonal surfaces, and irregular volume grids.

2.1 Related Work

Over the years, many methods have been developed for the automatic simplification of simplicial models. Details on many of them can be found in one of a number of available surveys [Cignoni et al. 1998; Garland 1999a; Luebke et al. 2002]. In this section, we review only the work most relevant to our own.

2.1.1 *Surface Simplification.* The earliest general methods for the automatic simplification of triangulated surfaces were vertex clustering [Rossignac and Borrel 1993], surface re-tiling [Turk 1992], vertex decimation [Schroeder et al. 1992], and mesh optimization [Hoppe et al. 1993]. Of the numerous algorithms developed since, many of the most effective have been based on iterative edge contraction. Particularly relevant methods include progressive meshes [Hoppe 1996], quadric-based simplification [Garland and Heckbert 1997], and memoryless simplification [Lindstrom and Turk 1998].

Since typical graphics applications use additional non-geometric surface attributes, especially color and texture, methods have also been developed for preserving these attributes during the simplification process. Hoppe [1996] included explicit attribute terms in the error metric that his algorithm seeks to minimize. Garland

and Heckbert [1998], and later Hoppe [1999], extended the quadric error metric to account for attribute error. Erikson and Manocha [1999] also use an attribute error formulation that reduces to a point-wise quadric error term. Cohen *et al.* [1998] developed a method focused on minimizing parametric distortion during simplification. Others have pursued solutions based on resampling of attributes from the original surface onto the output approximation [Maruya 1995; Soucy et al. 1996; Cignoni et al. 1998].

All existing surface simplification are fundamentally heuristic in nature. Optimal height field approximation, and by extension surface approximation, is known to be NP-Hard [Agarwal and Suri 1994]. However, Heckbert and Garland [1999] have shown that, in the limit of infinitesimal triangulations, their quadric-based method produces triangles with the optimal aspect ratio required for L_2-optimal approximation [Nadler 1986; D'Azevedo 1991].

2.1.2 *Curve Simplification.* The study of curve simplification has a much longer history than surface simplification. There is of course a great deal of mathematical theory surrounding the problems of continuous function approximation. Of more relevance to us here is the existing work on approximation of piecewise-linear curves with similar curves containing fewer vertices. A good overview of curve simplification methods can be found in the survey by Weibel [1997], and synopses of the methods discussed here are provided by Heckbert and Garland [1997].

Unlike more general approximation problems, optimal curve approximations can be found in polynomial time. Representative methods include the work of Imai *et al.* [1988] and Ihm *et al.* [1991]. Unfortunately, the complexity of such methods can rise to $O(n^3)$ for general space curves. This makes them rather impractical for very large datasets.

Perhaps the most widely used algorithm for curve simplification is a simple refinement algorithm, commonly referred to as the Douglas–Peucker algorithm. This algorithm begins with some minimal approximation, normally a single line segment from the first to last vertex. This segment is split at the point on the original curve which is furthest from the approximation. Each of the two new subsegments can be recursively split until the approximation meets some termination criteria. This is evidently a rather natural algorithm for curve approximation, since it was independently invented by a number of people [Ramer 1972; Duda and Hart 1973; Douglas and Peucker 1973; Baumgart 1974; Turner 1974; Pavlidis 1977; Ballard 1981]. Hershberger and Snoeyink [1994] discuss the efficient implementation of the Douglas–Peucker algorithm, specifically with an eye towards reducing its worst case complexity. Decimation algorithms, which in essence are the Douglas–Peucker refinement algorithm in reverse, have also been developed [Boxer et al. 1993; Leu and Chen 1988].

2.1.3 *Tetrahedral Simplification.* By comparison to the curve and surface domains, relatively few methods have been developed for simplifying tetrahedral data.

Renze and Oliver [1996] proposed a scheme based on iterative vertex removal. However, they did not suggest any error metric to guide the ordering of the removal operations. This approach is also complicated by the fact that the hole left by a vertex removal is not necessarily tetrahedralizable without adding Steiner

points [Ruppert and Seidel 1992]. Van Gelder *et al.* [1999] implemented vertex removal via half-edge contractions and suggested a local density metric to guide the selection of vertices to remove.

Trotts *et al.* [1998; 1999], Cignoni *et al.* [2000], and Chopra and Meyer [2002] have all developed contraction-based simplification algorithms. Trotts *et al.* propose an error metric that provides a conservative bound on the maximum deviation of the data field. However, this accuracy comes at the expense of very long running times. Cignoni *et al.* propose a variety of possible error metrics, including an estimate of the field error, a gradient difference heuristic, and a quadric error metric. Chopra and Meyer base the importance of a tetrahedron on the scalar range within the tetrahedron.

Popović and Hoppe [1997] generalized the progressive mesh representation [Hoppe 1996] to the more general case of simplicial complexes. While their technique is suitable for use on tetrahedral meshes, their error metrics focus only on the geometry of the mesh, and do not consider possible data fields attached to the mesh. Staadt and Gross [1998] suggested a similar extension of progressive meshes that takes existing data fields into account. They estimate approximation error by measuring the change in data gradients resulting from a contraction, and couple this with mesh quality measures to preserve total mesh volume and generate roughly equilateral tetrahedra.

3. ALGORITHMIC FRAMEWORK

Our simplification framework is based on the well-known approach of iterative edge contraction. The contraction $j \to i$ of the edge (i, j) will replace all occurrences of vertex j with vertex i. The vertex j along with any degenerate simplices that were incident on both i and j are then removed. The simplification process consists of iteratively selecting and contracting an edge of least cost. In general, we also typically wish to compute a new position $\mathbf{v}_i$ for the remaing vertex that will minimize some notion of distortion. This is the role of our error metric: to (1) rank the cost of edges for contraction and (2) to select the position $\mathbf{v}_i$.

To begin the process, we select the set of *candidate edges* that will be considered for contraction. Nominally, this will be the set of all edges of the initial complex M. The user may, at their discretion, eliminate certain edges from consideration (e.g., to prevent simplification of boundaries). It can also be useful to consider edges that are not part of the complex M; for example, this can allow the algorithm to aggregate separate connected components. Such *virtual edges* might be chosen from the Delaunay complex [Popović and Hoppe 1997] or based on the proximity of their endpoints [Garland and Heckbert 1997]. While such extensions are trivially supported, we will not make use of them here. All examples shown here were simplified by choosing the set of candidate edges to be exactly the edges of M.

We divide the simplification process into two distinct phases. During the *initialization* phase, all the candidate edges are ranked according to their cost. We maintain these candidates in a heap, using their cost as the key. Following initialization is the *iteration* phase. For each iteration, we select a minimum cost edge (from the top of the heap) and contract it. All edges connected to the remaining vertex

must have their costs recomputed and the heap must be updated appropriately[1].

3.1 Preventing Mesh Inversion

Any particular edge contraction may cause the mesh to become locally inverted. In other words, the orientation of certain simplices — as defined by their signed content[2] — may become reversed. To prevent this, we check whether any particular edge contraction would change the orientation of a simplex. Any such contraction is disallowed.

(a) Input mesh (b) no inversion check (c) with inversion check

Fig. 1. Planar mesh of Lake Superior simplified with (b) and without (c) the appropriate mesh inversion check. Allowing mesh inversion can obviously result in disastrous approximations.

Preventing inversion by explicit orientation checks is quite common. It has been used for planar graphs [Ciarlet and Lamour 1996], triangular meshes [Garland and Heckbert 1997; Hoppe 1996; Ronfard and Rossignac 1996], and tetrahedral volumes [Staadt and Gross 1998; Gelder et al. 1999; Trotts et al. 1999; Cignoni et al. 2000; Chopra and Meyer 2002]. In our quadric-based system, it is in planar regions where mesh inversions are most likely to occur. Figure 1 demonstrates what can happen if mesh inversions are not prevented.

3.2 Topological Preservation

It is often desirable to allow genus reduction during simplification, as excess topological features often make models unnecessarily complex. However, there are also applications in which it is preferable to guarantee that the topology of the mesh being simplified is never changed. In order to achieve this, we can use the link condition formulated by Dey *et al.* [1999]. Any contraction that fails the link condition check would change the topology of the complex, and is therefore rejected.

The two most common cases of the link condition can be easily summarized as follows. For 1-manifold complexes, the edge (i, j) may be contracted iff at least one of i and j is of degree 2 and they share no common neighbor. In the case of a 2-manifold complex, let N_i and N_j be the set of vertices adjacent to vertex i and j, respectively. Similarly, let N_{ij} be the set of all vertices k such that (i, j, k) is a triangle in M. The edge (i, j) can be contracted iff $N_i \cap N_j = N_{ij}$ and no two vertices in $N_i \cap N_j$ are adjacent.

[1]The heap implementation used must therefore support efficient UPDATE-KEY operations.

[2]*Content* is the generalization of volume: lengths of segments, areas of triangles, volumes of tetrahedra, and hypervolumes of d-simplices.

For the results shown in this paper, we have not used the link condition check. Therefore, the simplification system was allowed to, and at times did, alter the topology of the model being simplified.

3.3 Tolerance Guarantees and Intersection Avoidance

Like most simplification heuristics, our error metric does not provide any guaranteed bounds on the Hausdorff or RMS error of the final approximation. Neither does it detect or prevent global self-intersections of the surface. There are, however, some applications where it is important to provide enforcable error bounds or to guarantee that self-intersections are never created. In such cases, it is best to couple iterative contraction methods with envelope methods. In particular, we recommend either simplification envelopes [Cohen et al. 1996] or permission grids [Zelinka and Garland 2002].

4. STANDARD ERROR METRIC

We begin by briefly reviewing the derivation of the standard quadric error metric for triangulated 2-manifolds [Garland and Heckbert 1997; Garland 1999b].

Suppose that we are given a point $\mathbf{p}$ on a 2-manifold surface embedded in E^3. Let $\mathbf{n} = [\, a\ b\ c\,]^\mathsf{T}$ be the unit vector normal to the surface at $\mathbf{p}$. The tangent plane of the surface at $\mathbf{p}$ is thus $(\mathbf{x}-\mathbf{p})^\mathsf{T}\mathbf{n} = 0$. From this, we can easily express $Q_\mathbf{p}(\mathbf{x})$ — the squared distance of any point $\mathbf{x} \in E^3$ to the tangent plane at $\mathbf{p}$ — as:

$$\begin{aligned} Q_\mathbf{p}(\mathbf{x}) &= \left((\mathbf{x}-\mathbf{p})^\mathsf{T}\mathbf{n}\right)^2 && (1)\\ &= \left((\mathbf{x}-\mathbf{p})^\mathsf{T}\mathbf{n}\right)\left(\mathbf{n}^\mathsf{T}(\mathbf{x}-\mathbf{p})\right) && (2)\\ &= (\mathbf{x}-\mathbf{p})^\mathsf{T}\left(\mathbf{n}\mathbf{n}^\mathsf{T}\right)(\mathbf{x}-\mathbf{p}) && (3)\\ &= (\mathbf{x}-\mathbf{p})^\mathsf{T}\mathbf{A}(\mathbf{x}-\mathbf{p}) && (4) \end{aligned}$$

where $\mathbf{A}$ is the outer product matrix

$$\mathbf{n}\mathbf{n}^\mathsf{T} = \begin{bmatrix} a^2 & ab & ac \\ ab & b^2 & bc \\ ac & bc & c^2 \end{bmatrix} \qquad (5)$$

Equation 4 is the key form of our error metric that we will revisit shortly in order to generalize our algorithm.

For actual use in our algorithm, we rewrite Equation 4 as the quadratic form

$$Q(\mathbf{x}) = \mathbf{x}^\mathsf{T}\mathbf{A}\mathbf{x} + 2\mathbf{b}^\mathsf{T}\mathbf{x} + c \qquad (6)$$

where

$$\mathbf{A} = \mathbf{n}\mathbf{n}^\mathsf{T} \qquad \mathbf{b} = -\mathbf{A}\mathbf{p} \qquad c = \mathbf{p}^\mathsf{T}\mathbf{A}\mathbf{p} \qquad (7)$$

Notice that $\mathbf{A}$ is a 3×3 symmetric positive semi-definite matrix, $\mathbf{b}$ is a 3-vector, and c is a scalar. Since the function $Q(\mathbf{x})$ is entirely determined by these three values, we identify the function Q with the triple

$$Q = (\mathbf{A}, \mathbf{b}, c) \qquad (8)$$

and use the term *quadric* to refer to this triple.

Since $Q(\mathbf{x})$ is a quadratic form, it achieves a minimum when $\nabla Q(\mathbf{x}) = 0$. Thus the point $\mathbf{x}^*$ for which $Q(\mathbf{x})$ is minimal is the solution to the linear system

$$\mathbf{A}\mathbf{x}^* = -\mathbf{b} \tag{9}$$

Since the matrix $\mathbf{A}$ is positive semidefinite, Cholesky decomposition is the preferred approach to solving this linear system [Press et al. 1992]. Assuming that $\mathbf{A}$ is not singular, the error of this optimal point will be:

$$Q(\mathbf{x}^*) = \mathbf{b}^\mathsf{T}\mathbf{x}^* + c = -\mathbf{b}^\mathsf{T}\mathbf{A}^{-1}\mathbf{b} + c \tag{10}$$

Note that these are both instances of well-known formulae for the minimum of a positive definite quadratic form [Strang 1988, pg. 347]. In the case when $\mathbf{A}$ is singular, it can be advantageous to perform singular value decomposition so as to project vertices onto the solution space [Lindstrom 2000b]. However, in the context of our iterative edge contraction algorithm, we prefer to simply pick the vertex $\mathbf{v}_i$ minimizing $Q(\mathbf{v}_i)$.

5. GENERALIZING THE ERROR METRIC

Having reviewed the standard quadric error metric for 2-manifold surfaces in E^3, let us consider the fully general case. As we will see, the key to our approach is a reorganization of the error metric to focus on tangent vectors rather than perpendicular normals. This very natural reformulation — an idea particularly common in differential and Riemannian geometry — results in a quadric metric that effortlessly generalizes to surfaces of dimension other than 2.

Suppose that we are given a d-manifold M embedded in E^n. At any point $\mathbf{p}$ on the surface, we can construct an orthonormal basis $\{\mathbf{e}_1, \mathbf{e}_2, \ldots, \mathbf{e}_n\}$ such that $\{\mathbf{e}_1, \ldots, \mathbf{e}_d\}$ span the tangent space at $\mathbf{p}$. Consequently, we know that $\{\mathbf{e}_{d+1}, \ldots, \mathbf{e}_n\}$ must all be orthogonal to the tangent space at $\mathbf{p}$.

Consider any point $\mathbf{x} \in E^n$. By applying the generalized Pythagorean Theorem, we know that we can write the squared distance of $\mathbf{x}$ from $\mathbf{p}$ in terms of its squared distance along each of our basis directions:

$$\|\mathbf{x} - \mathbf{p}\|^2 = \sum_{i=1}^{n} \left((\mathbf{x} - \mathbf{p})^\mathsf{T}\mathbf{e}_i\right)^2 \tag{11}$$

Furthermore, note that we can rewrite this equation in the following way:

$$\|\mathbf{x} - \mathbf{p}\|^2 = \sum_i (\mathbf{x} - \mathbf{p})^\mathsf{T} \left(\mathbf{e}_i \mathbf{e}_i^\mathsf{T}\right) (\mathbf{x} - \mathbf{p}) \tag{12}$$

$$= (\mathbf{x} - \mathbf{p})^\mathsf{T} \left(\sum_i \mathbf{e}_i \mathbf{e}_i^\mathsf{T}\right) (\mathbf{x} - \mathbf{p}) \tag{13}$$

which obviously implies that $\sum \mathbf{e}_i \mathbf{e}_i^\mathsf{T} = \mathbf{I}$.

Now let us consider the intuitive foundation of the quadric error metric. Imagine that we want to move a point $\mathbf{p}$ on the manifold from its current position to some new position $\mathbf{x}$. The quadric error $Q(\mathbf{x})$ should reflect the geometric error introduced by doing so. Our standard definition of the error, reviewed in the previous section, is to measure the squared distance of $\mathbf{x}$ to the tangent plane of

$\mathbf{p}$. Thus, we seek to allow the local movement of $\mathbf{p}$ within its tangent plane, but to penalize its movement away from the tangent plane.

Fundamentally, applying our error construction amounts to selecting a set of directions $\{\mathbf{e}_i\}$ along which we want to allow the point $\mathbf{p}$ to move without penalty. As we have just discussed, these are exactly the tangent directions $\{\mathbf{e}_1, \ldots, \mathbf{e}_d\}$ at $\mathbf{p}$. Thus, we want our error $Q(\mathbf{x})$ to measure the squared distance of $\mathbf{x}$ from $\mathbf{p}$ in the affine subspace orthogonal to the tangent space at $\mathbf{p}$. Since we have assumed that we have a basis $\{\mathbf{e}_{d+1}, \ldots, \mathbf{e}_n\}$ for this orthogonal space, we could simply measure this error by selectively including terms from the full summation (13) to get:

$$Q_{\mathbf{p}}(\mathbf{x}) = (\mathbf{x}-\mathbf{p})^{\mathsf{T}} \left(\sum_{i=d+1}^{n} \mathbf{e}_i \mathbf{e}_i^{\mathsf{T}} \right) (\mathbf{x}-\mathbf{p}) \tag{14}$$

However, a completely equivalent, but much more convenient, formulation would involve *subtracting* squared distances in the tangent space from the full squared distance:

$$Q_{\mathbf{p}}(\mathbf{x}) = (\mathbf{x}-\mathbf{p})^{\mathsf{T}} \left(\mathbf{I} - \sum_{i=1}^{d} \mathbf{e}_i \mathbf{e}_i^{\mathsf{T}} \right) (\mathbf{x}-\mathbf{p}) \tag{15}$$

This yields our generalized error metric, providing the definition for the *fundamental quadric* of a point $\mathbf{p}$ on some given manifold M. Thus, the fundamental quadrics that we compute are always of the form:

$$\mathbf{A} = \mathbf{I} - \sum_{i=1}^{d} \mathbf{e}_i \mathbf{e}_i^{\mathsf{T}} \qquad \mathbf{b} = -\mathbf{A}\mathbf{p} \qquad c = \mathbf{p}^{\mathsf{T}}\mathbf{A}\mathbf{p} \tag{16}$$

Note that this definition possesses the following crucial property: it is dependent only on the dimension of the manifold M. It is independent of the dimension in which M is embedded (i.e., the co-dimension of M). And except for our construction of the matrix $\mathbf{A}$, it has exactly the same form as the standard quadric metric (Equation 7).

Let us briefly consider the properties of the matrix $\mathbf{A}$. First, it is obviously symmetric, since the outer products $\mathbf{e}_i\mathbf{e}_i^{\mathsf{T}}$ are all symmetric. The tangent basis vectors $\mathbf{e}_1, \ldots, \mathbf{e}_d$ all lie in the null space of $\mathbf{A}$. The orthogonal vectors $\mathbf{e}_{d+1}, \ldots, \mathbf{e}_n$ are all eigenvectors of $\mathbf{A}$, and all with eigenvalue $\lambda = 1$. Thus $\mathbf{A}$ is positive semi-definite, as all its non-zero eigenvalues are 1.

5.1 Fundamental Quadrics of Simplices

The quadric definition given above is phrased in the rather general sense of examining the tangent space of a surface at a point $\mathbf{p}$. However, we are typically interested in the more specific case where M is a simplicial complex. In this case, we wish to define the fundamental quadric of a single simplex.

Any given non-degenerate d-simplex σ with corners $\mathbf{u}_0, \mathbf{u}_1, \ldots, \mathbf{u}_d$ defines a unique tangent d-plane. We can characterize this tangent hyperplane by choosing any point $\mathbf{p}$ contained within the simplex and d orthonormal tangent vectors $\mathbf{e}_1, \ldots, \mathbf{e}_d$ that all lie within the plane. This allows us to define the fundamental quadric $Q_{\mathbf{p}}$ at the point $\mathbf{p}$. We wish to define the fundamental quadric Q_σ of the simplex σ in such a

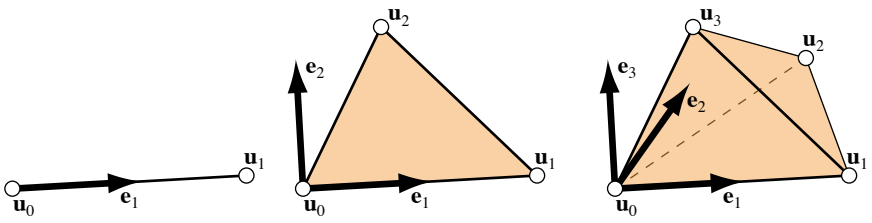


Fig. 2. Our standard orthonormal tangent frames for the three most common simplex types: edges, triangles, and tetrahedra.

way that:

$$Q_\sigma(\mathbf{x}) = \int_{\mathbf{p}\in\sigma} Q_\mathbf{p}(\mathbf{x}) \tag{17}$$

Since all points $\mathbf{p} \in \sigma$ have the same tangent plane, this means that the fundamental quadric is simply

$$Q_\sigma = \omega_\sigma Q_\mathbf{p} \tag{18}$$

where ω_σ is the content (or hypervolume) of σ and $\mathbf{p}$ is any point on its interior. Lindstrom [2000a] provides a nice synopsis of the formalization of simplex hypervolumes using the exterior product.

By convention, we choose $\mathbf{p}$ to be the barycenter of the simplex $\mathbf{p} = \frac{1}{d+1}\sum_i \mathbf{u}_i$ and we construct our orthonormal tangent vectors by a Gram-Schmidt orthogonalization [Strang 1988] of the edge vectors $(\mathbf{u}_i - \mathbf{u}_0)$. Figure 2 illustrates this construction for the 3 simplex types of most interest to us.

5.1.1 *Aggregate Quadrics.* For a given region N, we can define an aggregate quadric as a sum over its constituent simplices

$$Q_N = \sum_{\sigma\in N} Q_\sigma \tag{19}$$

Because we have chosen to weight the individual simplicial quadrics by content (18), this aggregate quadric will provide a discrete approximation of the integrated quadric error

$$Q_N(\mathbf{x}) = \int_{\mathbf{p}\in N} Q_\mathbf{p}(\mathbf{x})\, dN \tag{20}$$

It also has the important property that the error metric is invariant when the surface tessellation is changed without altering the geometry [Garland 1999b].

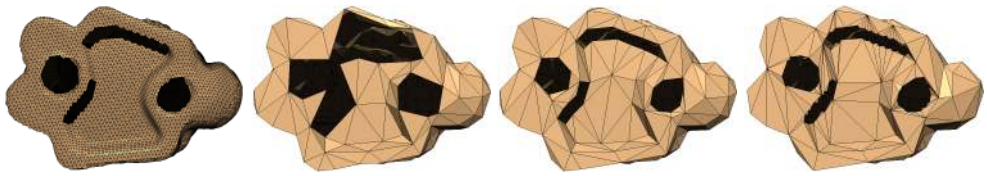
As with the standard quadric metric, we note that the quadratic form $Q_N(\mathbf{x})$ is positive semi-definite [Garland 1999b, §4.1.1]. Consider any eigenvector $\mathbf{x} \neq 0$ of Q_N. Since it is an eigenvector, there is some eigenvalue λ for which $Q_N(\mathbf{x}) = \lambda\mathbf{x}$. And because $Q_N(\mathbf{x})$ measures the sum of squared distances of $\mathbf{x}$ to a set of hyperplanes, we know that $Q_N(\mathbf{x}) \geq 0$. Given that $\mathbf{x} \neq 0$, we can conclude that $\lambda \geq 0$. Thus, since the eigenvalues of Q_N are non-negative, it must by positive semi-definite.

5.1.2 *Vertex Quadrics.* We define the fundamental quadric for a vertex by assigning to it a corresponding region $N_{\mathbf{v}}$ over which we will integrate point-wise quadrics. Ideally, we would construct a geodesic Voronoi diagram, assigning each point on the surface to its closest vertex. The quadric for the vertex $\mathbf{v}$ would then be the weighted sum of the simplices intersecting the region $N_{\mathbf{v}}$, with weights corresponding to the fraction of the simplex's area contained within $N_{\mathbf{v}}$.

In practice, explicitly computing such regions can be very costly, and yields little improvement in accuracy. It can also complicate our construction as the set of points closer to $\mathbf{v}$ than any other vertex may extend beyond its 1-ring in the presence of obtuse angles [Meyer et al. 2003]. Therefore, we adopt a much simpler approach. The fundamental quadric of a vertex is the weighted sum of the fundamental quadrics of the maximal simplices incident on it. Each simplex is divided equally between its corners, making the weight for each individual corner $\omega_\sigma/(d+1)$.

5.2 Boundary Preservation

When using quadric-based simplification methods, it is critically important to use additional error terms that will help to preserve any open boundaries. Without any such terms, the core error metric may cause serious degradation in all boundaries. Garland and Heckbert [1997; 1998] solved this problem by introducing perpendicular planes along open boundaries of the surface. However, this construction does not generalize in any convenient way to higher dimensions.



(a) Input (b) $w_b = 0$ (c) $w_b = 10$ (d) $w_b = 1000$

Fig. 3. Simplification of the Stanford bunny to 1000 vertices with various choices of the boundary penalty factor w_b.

One of the very real benefits of our generalized formulation of the quadric error metric is that it trivially provides a generic form for arbitrary boundary terms. Given a simplex of any dimension, we can compute its fundamental quadric, and scale this by a large user-specified penalty factor[3]. This quadric is then added into the fundamental vertex quadrics like any other. See Figure 3 for an example of the effect of the boundary error term. With no boundary penalty, the geometry of the boundary is changed considerably. With increasing penalty factors, the boundary is preserved more exactly.

[3]All the results in this paper are generated with a penalty factor of 10.

5.3 Extension to Non-manifolds

One of the important features of the original quadric-based surface simplification algorithm [Garland and Heckbert 1997] is that it can process manifold and non-manifold surfaces with equal ease. Up to this point, our discussion has focused exclusively on manifold surfaces. However, the quadric error metric presented here extends to non-manifolds in the same way as the original.

Notice that for simplicial meshes we have defined the error metric in terms of fundamental quadrics defined for simplices. Within a single simplex, the surface must necessarily be manifold. The fact that a vertex neighborhood may be non-manifold does not alter the quadric construction in any way; we sum over its adjoining simplices exactly as we would in the manifold case.

5.4 Complexes with Scalar Attributes

We may frequently wish to simplify complexes with associated per-vertex scalar data. Following the approach taken by Garland and Heckbert [1998], and later Hoppe [1999], we can easily extend our generalized quadric metric to handle this situation.

Suppose that we are given a simplicial complex where each vertex has a position $\mathbf{x}_i \in E^d$ and associated attributes $\mathbf{s}_i \in R^k$. We view the surface as a manifold embedded in E^{d+k}, where each vertex is represented by a vector $\mathbf{v}_i = [\mathbf{x}_i\ \mathbf{s}_i] \in E^{d+k}$. To prevent bias between components due to different numerical ranges, we rescale the various coordinate and scalar components to lie within the unit hypercube prior to simplification.

Having lifted the manifold from E^d into E^{d+k}, the simplification algorithm itself is unchanged. All of the quadric definitions given above remain the same. Only the co-dimension of the manifold has changed.

5.4.1 *Analysis of Attribute Error.* Now let us consider in what sense this extended quadric measurs the error of the attribute field. For the purposes of our analysis, we assume that we are given a d-manifold complex embedded in E^d with a single piecewise linear scalar field defined over it. Over a given simplex, the scalar field $\phi : E^d \to R$ is a linear function. To simplify our discussion, we assume without loss of generality, that $\phi(0) = 0$ and that hence we can represent the scalar field in the form $\phi(\mathbf{x}) = \mathbf{g}^\mathsf{T}\mathbf{x}$. The fundamental quadric Q for this simplex will measure the squared distance of any point $(\mathbf{x}, s)$ in E^{d+1} to the d-plane $\mathbf{g}^\mathsf{T}\mathbf{x} - s = 0$.

We begin by defining the following basic measures of the error resulting from assigning scalar value s to a point $\mathbf{x}$.

$$\text{Field error} \quad \epsilon_F(\mathbf{x}, s) = |\phi(\mathbf{x}) - s|^2 \tag{21}$$

$$\text{Space error} \quad \epsilon_S(\mathbf{x}, s) = \|\mathbf{x}' - \mathbf{x}\|^2 \tag{22}$$

where $\mathbf{x}'$ is the closest point to $\mathbf{x}$ for which $\phi(\mathbf{x}') = s$. The field error ϵ_F is a typical measure of function approximation error. The space error ϵ_S is related more closely to the geometric deformation of the isosurfaces of the field Cignoni *et al.* [2004]. We also note that, because $\mathbf{x}'$ is the *closest* such point, the vector $(\mathbf{x}' - \mathbf{x})$ must be parallel to $\mathbf{g}$.

Given the plane equation above, it is easy to see that the squared distance of the

point $(\mathbf{x}, s)$ to the plane will be:

$$Q(\mathbf{x}, s) = \frac{(\mathbf{g}^\mathsf{T}\mathbf{x} - s)^2}{\mathbf{g}^\mathsf{T}\mathbf{g} + 1} = \frac{(\phi(\mathbf{x}) - s)^2}{\mathbf{g}^\mathsf{T}\mathbf{g} + 1} = \frac{\epsilon_F}{\mathbf{g}^\mathsf{T}\mathbf{g} + 1} \tag{23}$$

The term in the denominator is necessary to account for the fact that $(\mathbf{g}, -1)$ is not a unit vector. Given that we know $(\mathbf{x}' - \mathbf{x})$ is parallel to $\mathbf{g}$, we can express the squared length of $\mathbf{g}$ as follows:

$$\mathbf{g}^\mathsf{T}\mathbf{g} = \|\mathbf{g}\|^2 = \frac{|\phi(\mathbf{x}') - \phi(\mathbf{x})|^2}{\|\mathbf{x}' - \mathbf{x}\|^2} = \frac{\epsilon_F}{\epsilon_S} \tag{24}$$

This allows us to rewrite the equation for the quadric error of the point $(\mathbf{x}, s)$ as:

$$Q(\mathbf{x}, s) = \frac{\epsilon_F}{\frac{\epsilon_F}{\epsilon_S} + 1} = \frac{\epsilon_S \epsilon_F}{\epsilon_S + \epsilon_F} \tag{25}$$

Thus, we can see that the quadric error provides a pointwise measure of error combining both field error and space error. During simplification, the error assigned to a vertex will consequently be a sum of such terms, one for each of the associated simplices.

6. APPLYING THE GENERALIZED METRIC

We can apply the generalized quadric error metric developed in the last section to the simplification of simplicial complexes of any dimension embedded in Euclidean spaces of any dimension. However, there are a select few cases which are of by far the most relevance to applications in computer graphics and visualization: polylines, triangulated surfaces, and tetrahedralized volumes.

In each of these cases, the actual process of simplification looks nearly identical to that used in standard surface simplification methods. The system begins with an initialization phase where we compute a fundamental quadric for each vertex: the weighted sum of the quadrics of the incident maximal simplices plus the weighted sum of any penalty quadrics from any adjacent boundaries. Each edge is ranked according to its cost of contraction, determined by the quadrics of its endpoints. These costs are used to organize all edges into a heap-based priority queue. Once initialization is complete, the algorithm enters an iterative phase. Until it reaches the target number of output simplices, it selects the edge of least cost from the heap, contracts it, and updates the costs of the surrounding remaining edges.

6.1 Triangulated Surfaces

Let us begin by briefly discussing the application of our generalized metric to triangulated surface simplification. As we will see, the result is essentially equivalent to the standard quadric-based simplification algorithm [Garland 1999b; Garland and Heckbert 1997].

For each triangle σ of the mesh, we can compute its barycenter $\mathbf{p}$ and two orthonormal tangent vectors $\mathbf{e}_1, \mathbf{e}_2$. This allows us to define the quadric $Q_\mathbf{p}$ for the tangent plane at $\mathbf{p}$:

$$\mathbf{A} = (\mathbf{I} - \mathbf{e}_1\mathbf{e}_1^\mathsf{T} - \mathbf{e}_2\mathbf{e}_2^\mathsf{T}) \qquad \mathbf{b} = -\mathbf{A}\mathbf{p} \qquad c = \mathbf{p}^\mathsf{T}\mathbf{A}\mathbf{p} \tag{26}$$

Mathematically, this is completely equivalent to the standard quadric error metric (Equation 7). From here, we can also define the area-weighted fundamental quadric Q_σ (Equation 18) for the triangle σ. The fundamental quadric for a vertex $\mathbf{v}$ will be $1/3$ the sum of the fundamental quadrics of its surrounding faces. Recall that we perform this area-weighting to achieve mesh-invariant error (§5.1.1). In the case of triangulated surfaces, it is possible to express the more accurate Voronoi areas in closed form [Meyer et al. 2003], but only in the absence of obtuse angles.

Having computed the fundamental quadric for each vertex, we must also apply the appropriate boundary error terms. We begin by collecting the set of boundary edges (if any). As each edge is of course a simplex, we apply the same quadric construction for the boundary as for the interior triangles. We compute the point $\mathbf{p}$ that is the barycenter of the edge and a unit vector $\mathbf{e}_1$ along the edge.

$$\mathbf{A} = (\mathbf{I} - \mathbf{e}_1\mathbf{e}_1^\mathsf{T}) \qquad \mathbf{b} = -\mathbf{A}\mathbf{p} \qquad c = \mathbf{p}^\mathsf{T}\mathbf{A}\mathbf{p} \tag{27}$$

We weight this quadric appropriately and also scale it by a large penalty factor. Adding each boundary quadric into the fundamental quadrics for the edge's endpoints completes the initialization phase of the algorithm.

This boundary term is a slight modification of the one originally proposed by Garland and Heckbert [1997]. For a given boundary edge, their algorithm would construct a plane through that edge perpendicular to the triangle adjoining the edge. In contrast, our approach is equivalent to computing two mutually perpendicular planes through the edge, without reference to the adjoining face. For surface boundary edges having only one adjacent face, the end result is entirely equivalent. However, our error term is slightly different when used to enforce the preservation of user-defined curves where each edge may have more than one adjacent face. In this case, our boundary term is equivalent to that proposed by Kho and Garland [2003]. In practice, we find the difference between these slightly differing approaches negligible. The advantage of the formulation presented here is that it trivially extends to higher dimensional simplices, while the perpendicular plane construction used by Garland and Heckbert does not generalize nicely.

Figure 4 shows an example of the surface approximations generated by our prototype implementation. Note that even with only 10,000 vertices, almost all of the original shape has been preserved. The approximation using a mere 1000 vertices has lost most of the textural features of the surface, but it retains essentially all of the larger scale features (e.g., fingers, ears, and pectoral plates).

6.2 Surfaces with Attributes

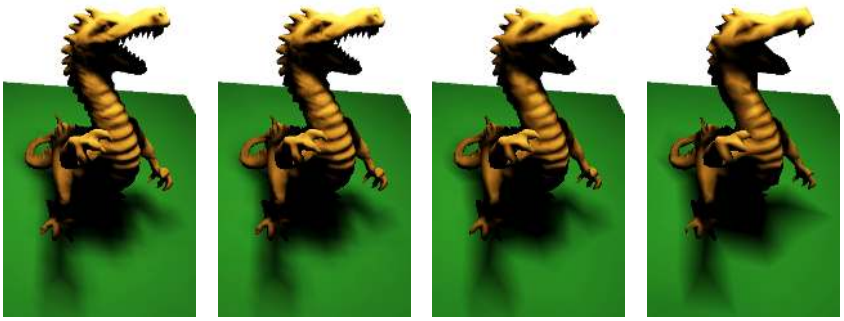
Garland and Heckbert [1998], and later Hoppe [1999], developed extensions of the purely geometric surface metric to the domain of surfaces with attributes. We have already outlined the general approach in Section 5.4. Here, we consider the specific case where we are given a triangulated surface where each vertex has a position $\mathbf{x}_i \in E^3$ and associated attributes $\mathbf{s}_i \in R^k$. For example, one common case would be surface meshes with per-vertex color, in which case each $\mathbf{s}_i$ might be an RGB 3-vector.

One of the primary benefits of the generalized error metric we have presented is that it is entirely independent of the co-dimension of the surface being simplified. Therefore, the quadric definition given in Equation 26 and used for normal

(a) 172,974 vertices (b) 10,000 vertices (c) 1000 vertices

Fig. 4. Sample approximations of an Armadillo Man (original at left). The approximations were generated in 12.5 and 12.8 seconds for the 10,000 and 1000 vertex approximations, respectively. All are flat-shaded to highlight the structure of the surface mesh.

triangulated surface simplification, can be used directly for simplification in this higher-dimensional space. The boundary error equations are unchanged as well. The area used for weighting the fundamental quadrics remains the usual area of the triangles in E^3. Only the specific dimensions of the matrices and vectors involved in the equations is actually changed in any way.



(a) 25,418 v. (b) 5000 vert. (c) 1500 vert. (d) 750 vert.

Fig. 5. Simplifying a dragon lit with a radiosity solution. The original model (at left) has 1 RGB color per vertex. No further shading is being performed.

An example of attribute simplification is shown in Figure 5. Every vertex of the original model has an associated RGB color that was computed via radiosity simulation. Each approximation is being rendered by Gouraud shading of the per-vertex RGB colors, without any additional display-time lighting. Note the preservation

of geometric features such as the scales and teeth, the open boundary in the background, and color features such as the shadow on the ground plane. The time required to produce these approximations ranged from 1.58 seconds (5000 vertices) to 1.78 seconds (750 vertices).

Here we observe that the equations used by Garland and Heckbert [1998] to define their extended quadric definition are essentially identical to our own generalized metric. However, their work provided no useful definition for boundary error in the attribute space E^{3+k}. More importantly, we show here that our basic approach can be trivially extended much farther to encompass simplification of any simplicial complex.

6.3 Plane and Space Curves

Now let us consider the case of plane and space curves represented as a collection of line segments. Each segment of the curve is a 1-simplex with endpoints $(\mathbf{v}_i, \mathbf{v}_j)$. The orthonormal tangent vector to this simplex is simply the unit edge vector

$$\mathbf{e}_1 = \frac{\mathbf{v}_j - \mathbf{v}_i}{\|\mathbf{v}_j - \mathbf{v}_i\|} \tag{28}$$

Again, we select the point $\mathbf{p}$ to be the barycenter $\frac{1}{2}(\mathbf{v}_i + \mathbf{v}_j)$, and can thus formulate the fundamental quadric for the line segment as:

$$\mathbf{A} = (\mathbf{I} - \mathbf{e}_1\mathbf{e}_1^\mathsf{T}) \qquad \mathbf{b} = -\mathbf{A}\mathbf{p} \qquad c = \mathbf{p}^\mathsf{T}\mathbf{A}\mathbf{p} \tag{29}$$

Notice that this is exactly the same as the definition used above for boundary error on a triangulated surface. This illustrates one of the important, and very convenient, properties of our generalized metric: boundary error quadrics for a k-complex are entirely equivalent to the fundamental quadrics of a $(k-1)$-complex. In the planar case, this definition of quadrics for segments is equivalent to the planar metrics used by Garland [1999b] and Lindstrom [2000a] to analyze the behavior of the quadric error metric.

In the case of curves, the boundary will be determined by the endpoints of the curve (if any). For these 0-simplices, we can also easily define the quadric

$$\mathbf{A} = \mathbf{I} \qquad \mathbf{b} = -\mathbf{p} \qquad c = \mathbf{p}^\mathsf{T}\mathbf{p} \tag{30}$$

which has the effect of measuring the squared distance of any point $\mathbf{x}$ to the end point $\mathbf{p}$.

Figure 6 shows an example common to GIS applications. The input is a planar piecewise-linear curve outline of the state of Virginia composed of 6817 vertices. Using our prototype implementation, we generated several approximations of this input, all in 0.11 seconds or less. Notice how the major features of the curve are faithfully preserved, even after aggressive simplification. Note in particular how the major river basins along the eastern coastline are still present, albeit in a rather abstract form, even in the approximation containing only 20 vertices. And the relatively tiny peninsula in the extreme south-eastern corner is well-represented even at the 100 vertex level.

In Figure 7, we see an example of simplifying a rather densely tessellated spiral figure. Note that the central vertex of the spiral is non-manifold, and that each of the 20 spiral arms ends in a boundary vertex. As we would expect, the initial input

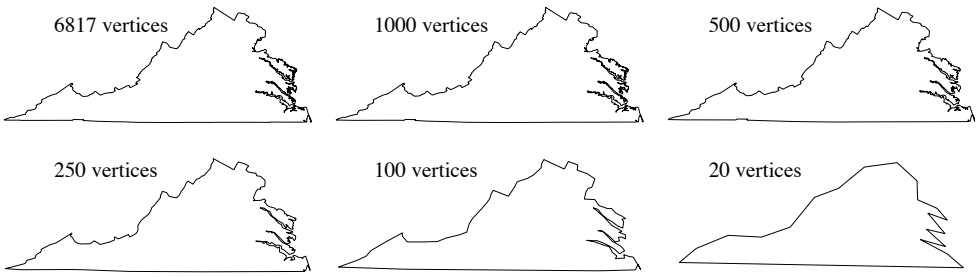
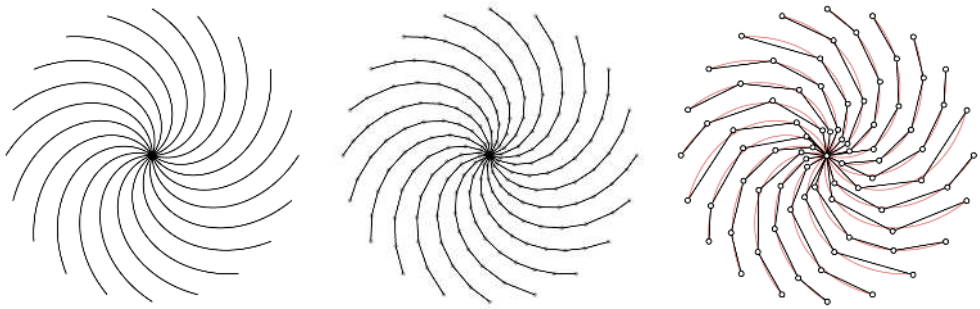


Fig. 6. Approximation of the (planar) boundary of the state of Virginia. Running times range from 0.06 to 0.11 seconds.



(a) Orig. 20,001 vertices (b) 201 vertices (c) 81 vertices

Fig. 7. Simplification of a non-manifold plane curve with many boundary vertices.

containing 1000 vertices per arm can be drastically simplified without substantially altering the shape of the curve. Our 201 vertex approximation, generated in 0.36 seconds, is nearly identical to the input. Even with only 81 vertices (running time of 0.37 seconds), the shape is recognizably the same, although the figure has lost some of its symmetry. More importantly, notice that the boundary vertices are unmoved — the original curve is drawn underneath in red to highlight this. Furthermore, we would ideally expect each of the 20 arms to have 4 vertices in this case. We see that our system has come quite close to this ideal, placing 4 vertices on most arms, and never using fewer than 3 nor more than 5 vertices on any arm.

6.4 Tetrahedralized Volume Data

Now let us consider the case of a tetrahedralized volume defining one or more scalar fields over a portion of E^3. We assume that the domain mesh is a simplicial complex where each vertex has a position $\mathbf{x}_i \in E^3$ and a data vector $\mathbf{s}_i \in R^k$. Furthermore, we assume that the data field is extended throughout each tetrahedron by linear interpolation of the data vectors at its corners.

Our approach to processing such a volume mesh is based on the treatment of attributes outlined in Section 5.4. We view the mesh as a 3-manifold embedded in

E^{3+k} where each vertex is a vector $\mathbf{v}_i = [\mathbf{x}_i\ \mathbf{s}_i]$ combining a position $\mathbf{x}_i$ and scalar attributes $\mathbf{s}_i$. Again, we also rescale the various coordinate and scalar components, prior to simplification, so that they lie within the unit hypercube.

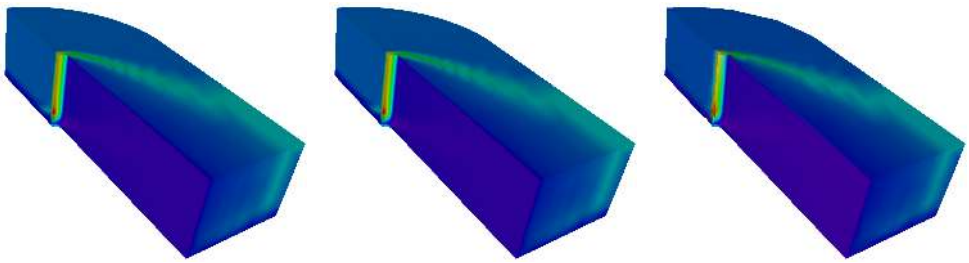
Since we are simply working with a 3-complex embedded in a Euclidean space (of dimension $3 + k$), we can directly apply our generalized quadric construction. We compute the barycenter $\mathbf{p}$ and 3 orthonormal tangent vectors, all of which lie in E^{3+k}. We can then directly formulate the fundamental quadric of a tetrahedron as:

$$\mathbf{A} = \mathbf{I} - \sum_{i=1}^{3} \mathbf{e}_i \mathbf{e}_i^\mathsf{T} \qquad \mathbf{b} = -\mathbf{A}\mathbf{p} \qquad c = \mathbf{p}^\mathsf{T}\mathbf{A}\mathbf{p} \tag{31}$$

For all practical purposes, every tetrahedral mesh will have one or more boundary surfaces. It is obviously essential that the shape of these surfaces be preserved as well as possible during simplification of the volume. As we did with surfaces and curves, we address this problem by introducing additional boundary error quadrics.

Consider a vertex attached to one or more boundary triangles. Intuitively, we want to allow this vertex to move along the boundary surface, but penalize its movement away from this surface. Given a particular boundary triangle, we can construct a quadric for that triangle in the same manner as outlined in Section 6.2:

$$\mathbf{A} = \mathbf{I} - \mathbf{e}_1 \mathbf{e}_1^\mathsf{T} - \mathbf{e}_2 \mathbf{e}_2^\mathsf{T} \qquad \mathbf{b} = -\mathbf{A}\mathbf{p} \qquad c = \mathbf{p}^\mathsf{T}\mathbf{A}\mathbf{p} \tag{32}$$

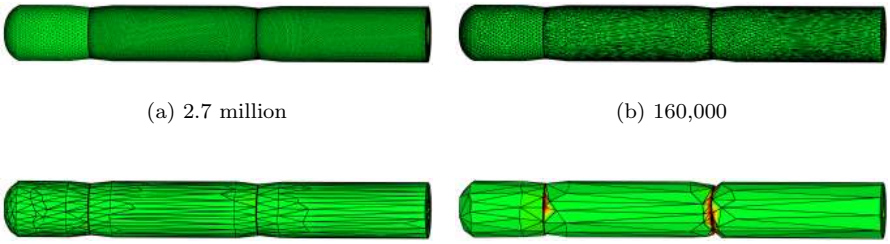


(a) Original (b) 44,969 tetrahedra (c) 4960 tetrahedra

Fig. 8. Approximations of the Blunt Fin mesh composed on 224,874 tetrahedra. Here we see only 1 of 5 scalar fields drawn over the boundary of the domain.

Figure 8 shows the approximations produced by our system on the Blunt Fin dataset. This CFD simulation models airflow over a flat plate with a blunt fin coming up out of the plate. The original hexahedral mesh contains 5 data fields. We cut each hexahedron into 6 tetrahedra, yielding a mesh of 224,874 tetrahedra. Thus, our method treats this mesh as a 3-manifold embedded in a Euclidean space of dimension 8. The field shown in Figure 8 is total energy. These approximations were produced in 31.6 seconds. All the significant features of the data field are preserved in both sample approximations. In particular, note the fidelity of the field along the leading edge of the fin, where the field gradient is quite high. We

also see that the shape of the domain boundary is well-preserved even at the rather coarse 4960 tetrahedron level.



(a) 2.7 million (b) 160,000

(c) 20,000 (d) 5000

Fig. 9. Approximations of a 2.7 million tetrahedron volume mesh, as seen from outside. The boundary surface is well preserved, even after extreme reduction.

In Figure 9, we see the boundary of a volume mesh modeling a Titan IV solid rocket. The solution over this mesh models the displacement of the rocket fuel as it burns. In this particular example, we are looking at the boundary of the volume (see Figure 23 for a better view of the solution data). The initial mesh is both very dense and very uniform, largely for reasons having to do with the solution process itself. However, the geometry of the boundary is very simple — it has large cylindrical patches — and the data field is essentially constant near the boundary. The approximations shown highlight the efficacy of our quadrics for preserving the boundary surface. Only at the most extreme level of simplification (a mere 5000 tetrahedra) has there been any significant change in the boundary geometry. And even at this level, we have a very acceptable approximation for interactive manipulation of the volume. Also notice how the coarser approximations have long and thin triangles that span the cylindrical segments of the boundary. While such triangle shapes are poorly suited for simulation, they are exactly what we desire for maintaining the integrity of the boundary with the fewest possible triangles.

6.5 Mixed Complexes

Up to this point, we have assumed that the complexes with which we are dealing are *pure* — namely that all simplices of dimension less than d occur as subsimplices of a d-simplex. We can, however, remove this assumption without any substantive change to our simplification system. We have already seen that we can formulate fundamental quadrics for any simplex. We have also defined the fundamental quadric of a vertex to be a sum over the fundamental quadrics of its adjacent simplices. There is no need for us to place any restriction on the dimension of these simplices, nor must we assume that they are all the same dimension. In other words, we handle mixed complexes in exactly the same way as a pure complex: at each vertex we sum over the quadrics of its adjoining maximal simplices and then perform iterative edge contraction.

The most common sort of mixed complex we encounter is one consisting of both triangles and edges (as in Figure 10). Each triangle, and each edge that is not contained in any triangle, contribute to the quadric at a vertex. Once these vertex quadrics are constructed, the iterative contraction phase is completely unchanged. Note that this corresponds directly with our treatment of boundary error terms. But now, instead of only considering edges along the surface boundary, we allow any arbitrary set of edges to be defined in the model. This ability to seamlessly process mixed simplicial complexes is a significant feature of our approach. With the exception of the Progressive Simplicial Complex work of Popović and Hoppe [1997], this level of generality is unique among simplification methods.

(a) Original (b) 5000 vertices (c) 1000 vertices

Fig. 10. Simplifying a horse in the winner's circle. The original model at left contains 96,966 triangles and 7000 edges, for a total of 55,485 vertices.

Figure 10 shows a simple example of this process. The original model contains 55,485 vertices (48,485 on the horse and 8000 in the rings). The approximations shown were generated in 3.3 seconds. As in the pure surface and pure curve case, we see results that preserve the original shape with a high degree of fidelity. In addition, the performance impact of treating this model as a mixed complex is essentially negligible. The total time required to simplify the horse and rings separately is 3.24 seconds.

6.6 Flat Meshes

Our focus up to this point has been on preserving the geometry of the model being simplified. The error metric we have outlined is based on measuring and minimizing distances from tangent hyperplanes. However, this tells us little if the entire model is flat (i.e., all tangent planes are identical).

Consider again the model shown in Figure 1. This is a planar triangle mesh of the interior of Lake Superior and contains 2894 vertices. It was made with the Triangle mesh generation software [Shewchuk 1996]. As we can clearly see in Figure 1, the triangle aspect ratios of the simplified mesh are quite extreme. However, we can easily remedy this by adding an additional *vertex distribution* quadric of the form:

$$\mathbf{A} = \mathbf{I} \qquad \mathbf{b} = -\mathbf{p} \qquad c = \mathbf{p}^{\mathsf{T}}\mathbf{p} \tag{33}$$

(a) 1000 (b) 500 (c) 250 (d) 100

Fig. 11. A planar mesh of the interior of Lake Superior is reduced from an initial size of 2894 vertices down to 100.

to each vertex. This penalizes the movements of vertices, and has the effect of biasing the algorithm to preserve the existing vertex distribution. By using this additional error term, we can produce much better approximations of flat meshes, as demonstrated in Figure 11. However, let us stress that these meshes are not necessarily suitable for numerical simulation; they do not enforce the strict angle guarantees provided by Triangle.

Positional error terms of this sort are quite natural and have been previously proposed in a number of contexts. Garland [Garland 1999b] recommended their use for preserving important feature points (e.g., the tips of horns) and they are a key component of the user-guided simplification system of Kho and Garland [2003]. Lindstrom [Lindstrom and Turk 1998] proposed a similar error term for triangle shape optimization, later demonstrating that it tends to produce roughly equilateral triangles and minimizes the triangle compactness measure suggested by Guéziec [1995; 1996]. Erikson and Manocha [1999] used an error term of this sort to measure attribute error in their GAPS simplification system.

7. RESULTS

We have implemented an experimental simplification system based on the generalized error metric that we have presented. We refer to this implementation as GSlim, since it is a generalized version of the earlier QSlim system [Garland 1999b]. In this section, we analyze the performance of this software on a variety of simplicial meshes of varying dimension. Unless noted otherwise, all experiments were run on a 1 GHz Pentium III system.

For all practical purposes, the performance of our newer GSlim system on triangulated models is essentially identical to that of the earlier QSlim software. As the performance of QSlim is well-documented [Garland and Heckbert 1997; Garland 1999b], we provide only a very brief analysis of its most important characteristic: its appealing combination of performance and approximation quality. We devote most of our attention to curves, volumes, and mixed complexes. These represent novel applications of the quadric error metric, whose performance has not been previously documented.

7.1 Triangulated Surfaces

Figure 12 shows simplification time as a function of approximation error for several simplification systems. Running times reflect performance on an R4400 SGI Indigo2, and the error values were measured using the Metro [Cignoni et al. 1998]

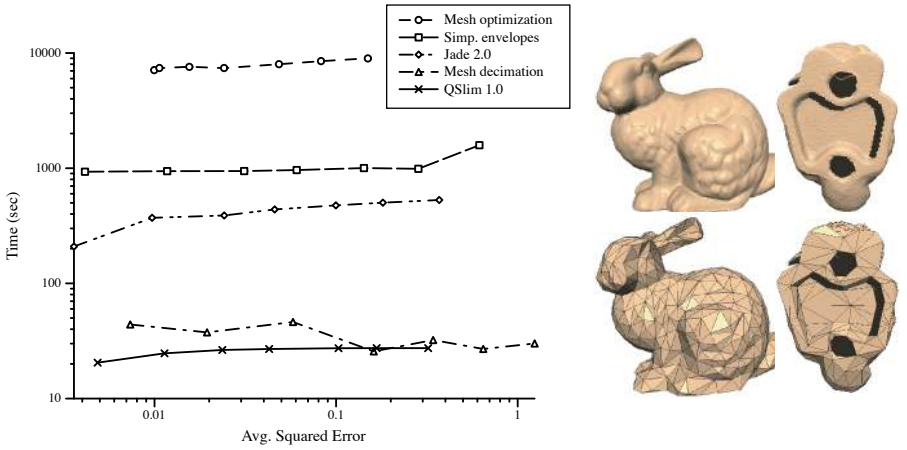


Fig. 12. Analysis of tradeoff between error and running time for simplification of the Stanford Bunny. Sample output from our GSlim system is shown at right.

comparison tool. The data itself is taken from the survey of Cignoni *et al.* [1998]. Corresponding points on the graph represent approximations with roughly identical vertex counts: 50%, 25%, 10%, 5%, 2%, 1%, and 0.5% of the original (from left to right). A good way to read this graph is to consider a particular vertical line. This will tell you, for a given error, how long it will take to produce an approximation with that error. As we would expect, mesh optimization [Hoppe et al. 1993] is by far the most expensive algorithm tested, requiring 1–3 orders of magnitude more time than the other algorithms. However, at low approximation levels, it also produces the best results. The next two algorithms below mesh optimization are the the simplification envelopes algorithm [Cohen et al. 1996], and the JADE vertex decimation system [Ciampalini et al. 1997]. By comparison to mesh optimization, these algorithms provide a significant savings in running time, but produce approximations with higher error at higher levels of simplification. The final two algorithms are vertex decimation [Schroeder et al. 1992], and an early version of the QSlim system [Garland and Heckbert 1997]. Both are substantially faster than the other algorithms. But while the error of the approximations generated by QSlim is quite competitive with the other algorithms, the error resulting from vertex decimation grows quite rapidly. In fact, at the 0.5% reduction level, vertex decimation produces the highest error, while only mesh optimization produces a lower error than QSlim. This example demonstrates quite clearly the mix of speed and approximation quality provided by quadric-based simplification. It is able to rapidly produce approximations whose error is competitive with significantly more expensive methods.

7.2 Plane Curves

We have already seen two examples of the performance of our GSlim system on plane curves in Figures 6 and 7. In both cases, we see that our algorithm is able to faithfully preserve the shape of the curves, even after very aggressive simplification.

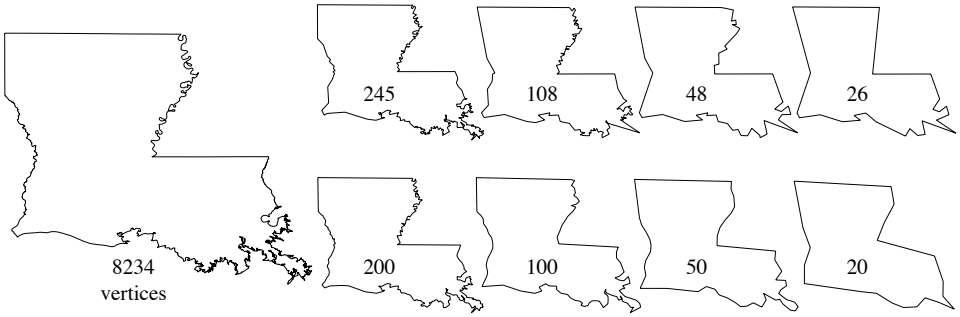
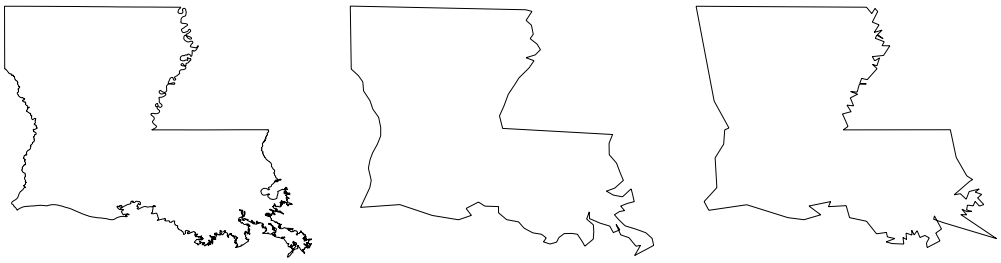


Fig. 13. Sample approximations of the state of Louisiana using Douglas–Peucker (top) and our GSlim system (bottom).



(a) Original (b) GSlim (c) Douglas–Peucker

Fig. 14. Detailed view of 100 vertex approximations of Louisiana.

Figure 7 also demonstrates both the success of our boundary error formulation and the application of our system to a non-manifold 1-complex.

In Figure 13 we look at a qualitative comparison between our system and the Douglas–Peucker algorithm, the most common line simplification method in general use. Our implementation of the Douglas–Peucker method is based on the work of Hershberger and Snoeyink [1994]. From a purely visual standpoint, we can clearly see that our GSlim system produces higher quality approximations. In particular, note how the large peninsula in the south-eastern corner of the state has degenerated to a spike in the 100 vertex approximation produced by the Douglas–Peucker method while retaining much of its overall shape in the approximation produced by our algorithm.

Figure 14 provides a more detailed view of the 100 vertex approximations produced by both systems. Here we can clearly see that our method accurately preserves the overall shape of the curve while allowing small details to disappear. In contrast, the Douglas–Peucker method winds up preserving many minor features while allowing the overall shape of the curve to deteriorate.

In order to quantitatively assess relative approximation quality, we generated a

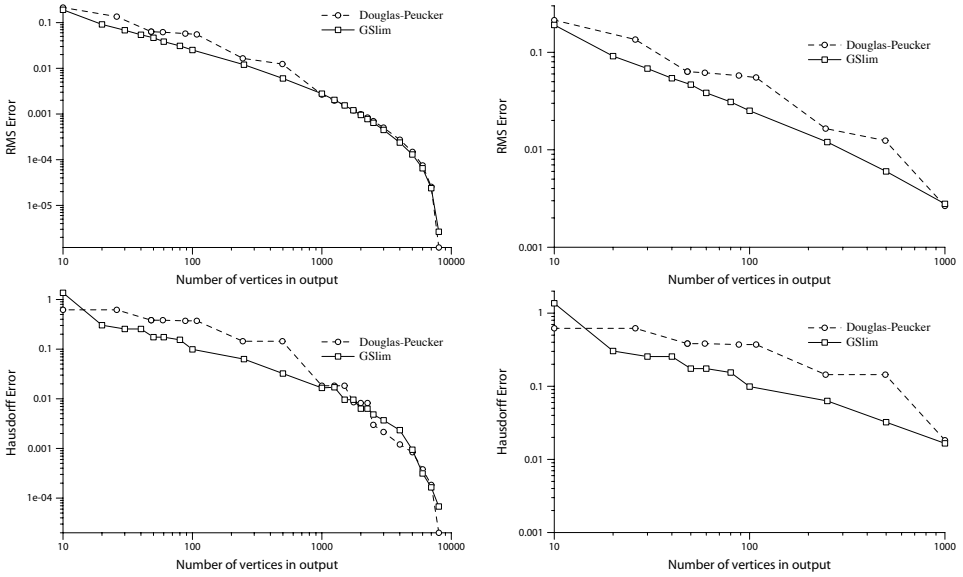


Fig. 15. Measuring error for Louisiana approximations. At left is the error data over all levels of approximation. At right, the data for moderate reduction levels is excluded.

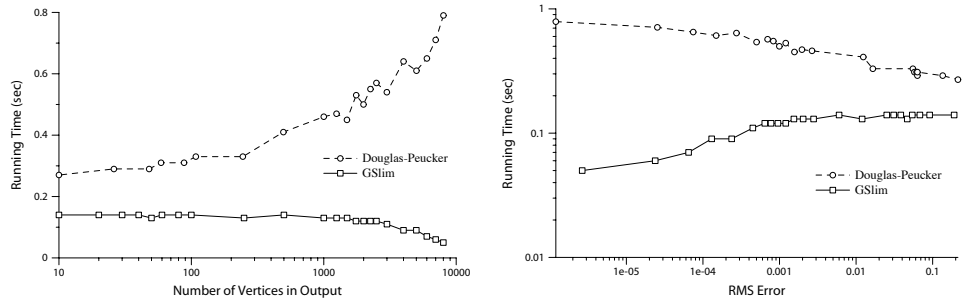
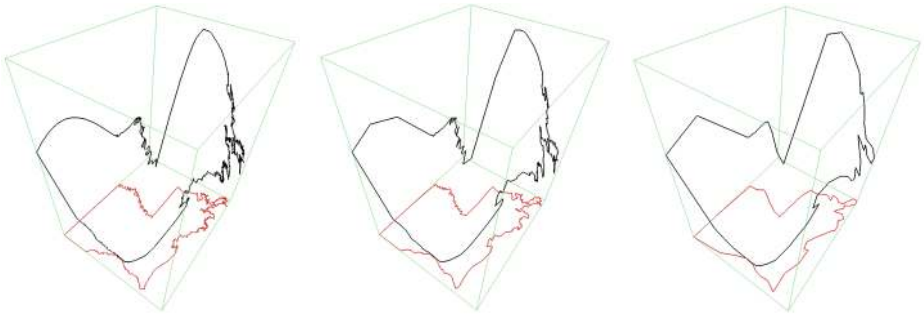


Fig. 16. Simplification times for Louisiana approximations, both in terms of output size and approximation error.

series of approximations with both systems and measured their error with respect to the input. Figure 15 shows the results, both for RMS (L_2) error and for maximum (L_∞) error. At relatively modest levels of reduction (up to around 1000 vertices in this case) both systems produce approximations with very low error. It is at more aggressive levels, as shown on the right, where we see a clear distinction between the methods. As was suggested by our visual analysis of Figure 13, we see that our quadric-based method produces consistently lower errors than the Douglas–Peucker method.

In addition to approximation error, it is of course also important to consider the running time of simplification systems. Figure 16 shows comparative running times, both as a function of output size and approximation error. As we can see, the



(a) Original (b) 250 vertices (c) 50 vertices

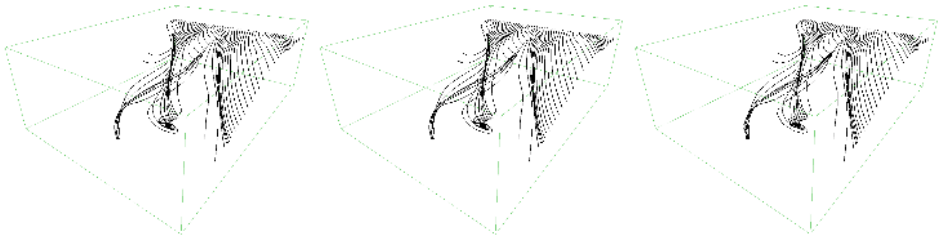
Fig. 17. Approximation of the Louisiana state outline lifted onto a sinusoidal surface. Drawn below the lifted curve is its projection back into the plane (in red).

running time of our GSlim system is consistently well below the Douglas–Peucker run time on this example. In fact, the slowest GSlim time (for the 10 vertex output) is faster than the fastest Douglas–Peucker time (also for the 10 vertex output). The graph at right shows running time as a function of approximation error. Our GSlim system is able to achieve any particular desired error threshold in as little as one tenth the time of the Douglas–Peucker method. This is a particularly important result. The Douglas–Peucker method is so widely used in practice because it is seen as providing good approximation quality while being quite fast on typical data. Our experiments clearly indicate that our generalized quadric-based method provides an even more attractive combination of efficiency and accuracy.

7.3 Space Curves

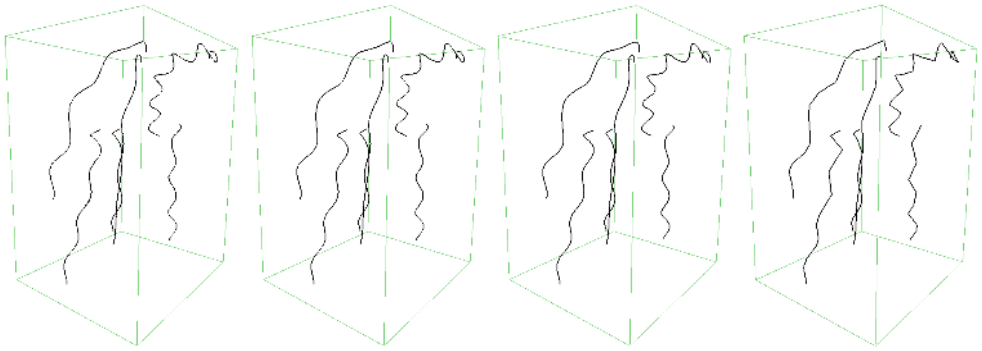
Our system is also able to handle space curves just as easily, and just as well, as plane curves. Figure 17 shows the outline of Louisiana lifted onto a sinusoidal surface of the form $z = (\sin \pi u)(\cos \pi v)$. For clarity, we also show both the bounding box of the curve and its projection back into the plane; however, these are both for visualization purposes alone, they are not part of the data. Visually, the approximations preserve the shape of the original curve very well. Measuring the RMS error of the approximations reveals nearly identical behavior to the plane curve case above, with a only a slightly higher absolute error range. This is of course exactly what we would expect, given that we have transitioned from a planar curve to a more complex space curve. There is little effect on the running time to produce these approximations. We measured only a 0.01 second increase in total running time with respect to the planar case.

Figure 18 shows a more complex example of space curves. The underlying dataset is a flow simulation of gas in a combustion chamber [Schroeder et al. 2003]. The space curve dataset is a set of streamlines extracted from this flow field, and contains a total of 28,623 vertices along 51 individual streamlines. Reducing this original model to 500 vertices required only 0.53 seconds. From Figure 18 we can see that



(a) Original (b) 5000 vertices (c) 500 vertices

Fig. 18. A set of 3-D streamline space curves containing 28,623 vertices is reduced down to 1.7% of its original size.



(a) Original (b) 5000 vert. (c) 500 vert. (d) 100 vert.

Fig. 19. Approximations of a set of 6 hair strands containing 75,737 vertices.

the shape of these streamlines is preserved quite well, even near the vortex feature in the middle. Measuring the error of these approximations confirms that they do indeed faithfully preserve the geometry of the original. The RMS errors of the 5000 and 500 vertex approximations are, respectively, 0.06% and 0.70% of the bounding box diagonal.

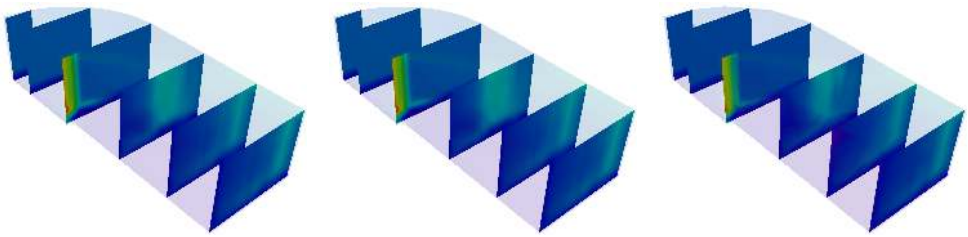
As a final example, consider the space curves shown in Figure 19. These are 6 strands of a human hair model — in this particular case, a fairly long and wavy head of hair [Yu 2001]. The original model contains 75,737 vertices. We produced approximations of 5000, 500, and 100 vertices. The overall structure of the hair is preserved fairly well at all levels of detail. Specifically, notice that the tips of the strands maintain both their position and direction even in the coarsest approximation.

7.4 Volume Data

Table I summarizes the overall performance of our system on 3 representative tetrahedral datasets. Times are reported in seconds and are split between the initialization phase (which is constant for a given input) and the iterative simplification phase (which is dependent on the output size). Even the largest dataset, containing 2.7 million tetrahedra, can be completely simplified in just over 6 minutes. More modest datasets in the range of 200,000 tetrahedra can be simplified in roughly 30 seconds.

Dataset	Input Tetrahedra	Init. Time (s)	Output Size		Iter. Time (s)
Blunt fin	224,874	12.95			
			44,969	20%	13.26
			22,483	10%	15.85
			11,235	5%	17.74
			4951	2.2%	18.65
LOP	616,050	34.86			
			144,000	23%	85.49
			72,000	11.7%	143.51
			36,000	5.8%	173.76
Titan IV	2,715,380	155.19			
			1,279,997	47%	96.54
			640,000	23.6%	144.38
			160,000	5.9%	190.96
			20,000	0.7%	212.61
			2500	0.09%	218.21

Table I. Simplification time for 3 representative volume models.

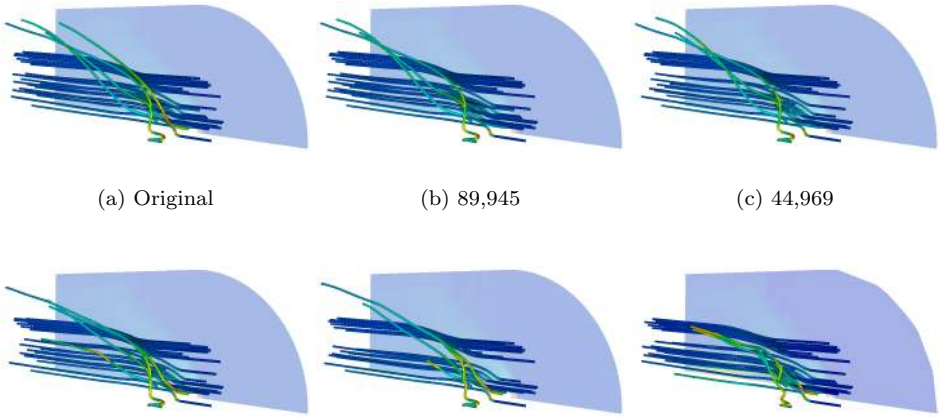


(a) Original (b) 44969 (c) 4960

Fig. 20. Slices through the interior of the Blunt Fin dataset reveal the internal structure of the data field.

We have already seen sample approximations of the Blunt Fin data set in Figure 8. In Figure 20 we see the same approximations. However, more of the internal structure of the data, as seen through several axis-parallel slices of the data, is visible. Just as in Figure 8, we see that the features of the data field are very

well-preserved. Note in particular the high data gradient (red–green) and the green plume visible in the third slice from the top.



(a) Original (b) 89,945 (c) 44,969

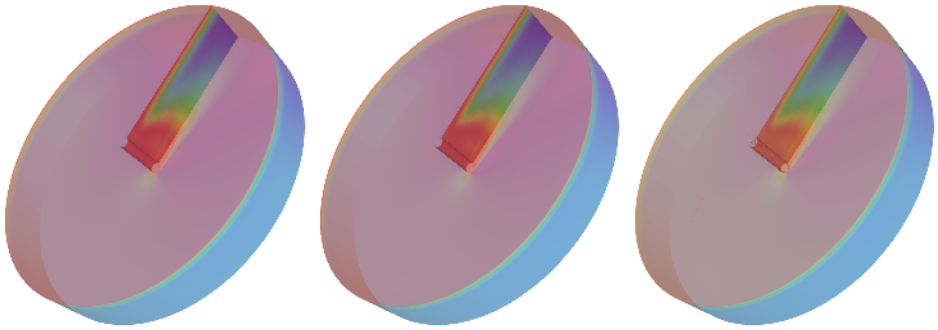
(d) 22,483 (e) 11,235 (f) 4960

Fig. 21. Tracing streamlines through approximations of the Blunt Fin data.

Figure 21 provides a noticeably more difficult test of approximation fidelity. Here we have traced a series of streamlines through the Blunt Fin flow field, colored according to the mass density scalar field. As we can see, even after an 80% reduction (c) of the data, the flow field still provides a fairly accurate approximation of the original data. By the time we reach the lowest level of detail (f) noticeable distortions have occurred, but this is of course to be expected. We can also see that the streamlines degrade fairly gracefully, an important requirement if we are to use these approximations in an adaptive level-of-detail system.

Our next example is the Liquid Oxygen Post dataset shown in Figure 22. This is also a fluid flow simulation having 5 data fields, producing a manifold in 8-D. The images plot the fluid density scalar field. This dataset is somewhat larger, containing 616,050 tetrahedra, and producing a 5.8% approximation takes just under 3.5 minutes. In this example we see some of the interior detail of the volume. Notice that the scalar field is preserved very well across all approximations. In particular, note the very fine detail visible as a red–blue color transition along the upper edge of the volume. This fine detail is preserved even in the coarsest approximation.

Finally, we come to the Titan IV rocket shown in Figures 9 and 23. This is a rather large dataset, containing 2.7 million tetrahedra. The data itself is a vector field recording the displacement of the solid rocket fuel as it burns. We treat this vector field as 3 separate scalar fields, producing a 3-manifold surface (with boundary) embedded in a 6-dimensional space. Figure 23 shows some of the internal detail of the same set of approximations shown in Figure 9. The finest approximation shown has only 160,000 tetrahedra (5.9% of the original), yet the volume's shape and data field are virtually unchanged. The second approximation, having



(a) 616,050 tetrahedra (b) 144,000 tetrahedra (c) 36,000 tetrahedra

Fig. 22. Approximations of the Liquid Oxygen Post volume dataset. The boundary surface has been made semi-transparent to show the internal structure of the volume.

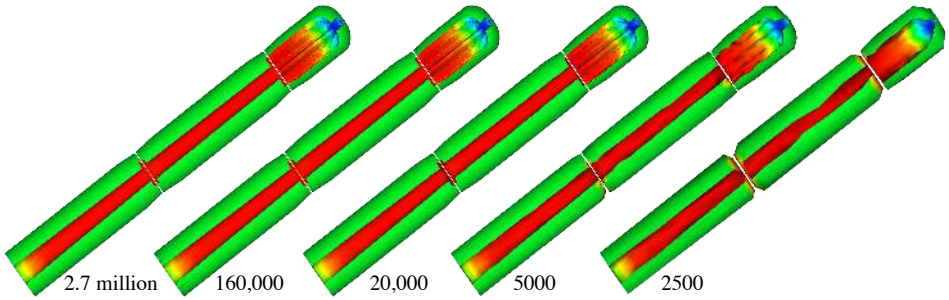


Fig. 23. Interior detail of the Titan IV rocket approximations shown in Figure 9.

20,000 tetrahedra, is still an excellent approximation of the original, although some minor artifacts are becoming evident. By the time we reach the 2500 tetrahedron approximation at the bottom, significant artifacts are of course present. However, as a temporary stand-in for interactive manipulation, or as a base mesh for progressive encoding, this would be entirely acceptable.

This compares quite favorably with existing volume simplification techniques. The system developed by Trotts *et al.* [1999] is possibly more accurate, but the price of this accuracy is substantially longer processing times. Other methods, such as TetFusion [Chopra and Meyer 2002] have tended to use simpler heuristics, which are often incapable of simplifying complex boundary surfaces. While capable of equaling or bettering our time efficiency, they appear to produce lower quality results at aggressive simplification levels. Prior methods have also tended to focus on single scalar fields, rather than data fields containing potentially many scalar components. Perhaps of equal importance is the fact the, being a generalization of a well-known surface simplification technique, it can enable the use of many surface-oriented techniques in the domain of tetrahedral meshes.

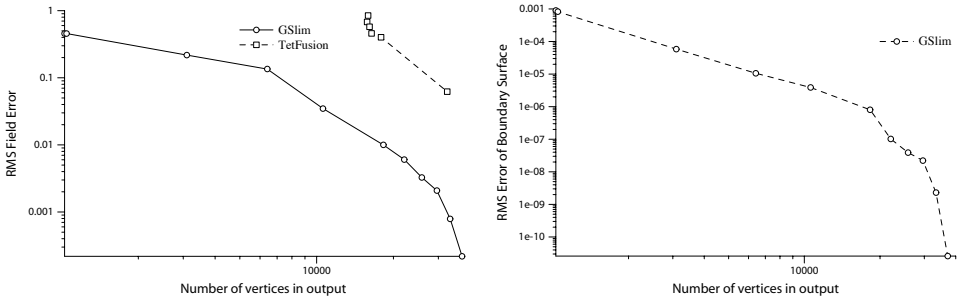


Fig. 24. Measurements of field error and boundary error for approximations of the Blunt Fin dataset. Note that the TetFusion boundary error is always 0.

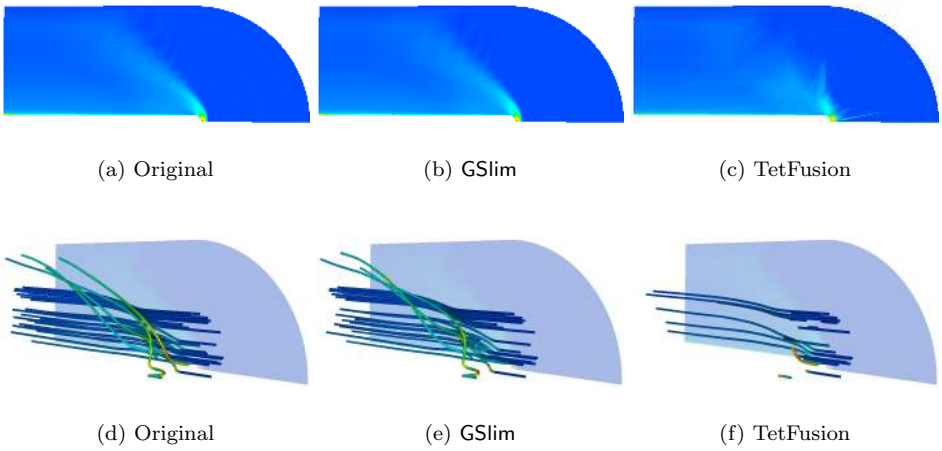
To provide a comparative measurement of the quality of our tetrahedral approximations, we implemented the TetFusion algorithm of Chopra and Meyer [2002]. We also extended their definition of scalar attribute error to encompass multiple scalar fields. Figure 24 graphs the error for approximations of the Blunt Fin data generated by both our GSlim system and our implementation of TetFusion. Running times for TetFusion were on the order of 200 seconds. The RMS (L_2) field error of the approximations generated by our method is substantially lower than those produced by TetFusion. Like several previous volume simplification systems, TetFusion conservatively avoids contracting boundary elements. Consequently, the boundary error of the approximations it produces is always 0. In contrast, GSlim simplifies the boundary, and therefore does introduce some amount of domain error. However, we can see that this error grows slowly. Because of its explicit preservation of the boundary, TetFusion is also unable to produce approximations at low levels of detail. In the experiment shown, we were unable to achieve reductions below 15,000 vertices (36% of original size).

Figure 25 provides a visual point of comparison between GSlim and TetFusion. The impact of the higher RMS error of the TetFusion approximations is immediately apparent. The scalar field is shown over a planar slice through the middle of the dataset. Despite using significantly more tetrahedra, there are quite obvious distortions in the field. As can be seen from the stream lines, the vector field has also been noticeably degraded.

7.5 Mixed Complexes

Now let us look at a more interesting example of a mixed complex, as shown in Figure 26. This illustrates a p53 tumor suppressor protein. The input model consists of a combination of triangles and edges. The two approximations shown were generated in 0.33 seconds (5000 vertices) and 0.42 seconds (1000 vertices). Because the original model was not extremely dense with respect to its shape complexity, it cannot be simplified as aggressively as typical surface scans. However, the 1000 vertex approximation uses only 10% of the original vertex count while maintaining a good approximation of the shape.

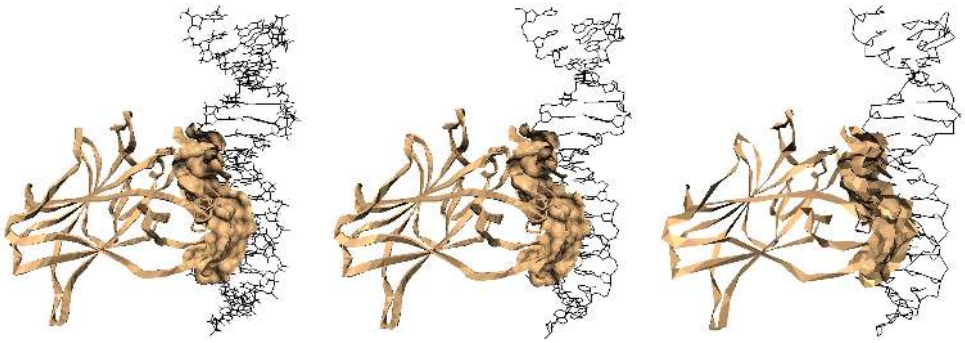
In Figure 27, we see a mixed complex extracted from the Liquid Oxygen Post dataset (see Figure 22). This model was constructed by first extracting a high-



(a) Original (b) GSlim (c) TetFusion

(d) Original (e) GSlim (f) TetFusion

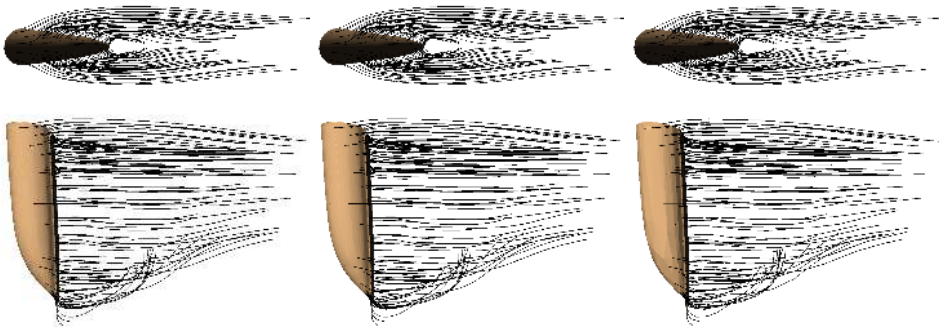
Fig. 25. Comparison of Blunt Fin approximations generated by GSlim (44,969 tetrahedra) and TetFusion (72,093 tetrahedra).



(a) Original (b) 5000 vertices (c) 1000 vertices

Fig. 26. Approximations of a protein model containing 10,710 vertices shared between 14,564 triangles and 1437 edges.

energy isosurface from the solution data. Then, a number of streamlines were seeded at selected vertices of the isosurface mesh. The result is a 25,438 vertex simplicial complex containing a mix of triangles and segments. This is clearly a non-manifold complex, as there are many vertices that are both part of the manifold isosurface and are the endpoints of the streamlines. Notice that the embedding of the model in space is also non-trivial, as several streamlines originating at vertices on the left cross through the isosurface as they flow to the right. The model also has numerous boundary simplices, both edges along the boundary of the isosurface and the vertices at the right-hand end of the streamlines; therefore, our generalized



(a) Original (b) 5000 (c) 1000

Fig. 27. Approximation of a simplicial complex containing 25,438 vertices. The model is an isosurface of the Liquid Oxygen Post data, with attached streamlines, and is pictured from both above and to the side.

boundary error term is particularly important in generating good approximations. Using our simplification system, we produced approximations of 5000 (19.7%) and 1000 (4%) vertices. The running times for simplification were 0.63 and 0.71 seconds, respectively. As before, we can see that the algorithm preserves the shape of the model quite well.

8. DISCUSSION

8.1 Contraction Primitives

The algorithm we have presented is based on the edge contraction operator. Other choices for the fundamental unit of simplification are of course possible. We could easily select simplices of higher dimension (i.e., faces or tetrahedra) as the atomic entities of contraction. We have chosen edge contraction because it provides the finest-grained control over the simplification process.

Ultimately, this choice is unimportant as the error metric we have presented is well-defined for any vertex contraction primitive. In broadest generality, we can select any subset of the available vertices and unify them in a single step. They need not even be connected [Garland and Heckbert 1997; Lindstrom 2000b]. Our error metric is still perfectly well-defined. The quadric for the remaining vertex is simply the sum of the vertices that were unified with it, and the method for finding the optimal vertex data is unchanged.

8.2 Extremely Large Meshes

Our method is based on iterative contraction. The number of iterations required to produce a particular approximation is linear in the number of vertices that must be removed from the model. For this reason, it is widely recognized that such iterative methods are not always the most effective choice for use on extremely large data. The rocket dataset shown in Figure 23, containing over 2.7 million tetrahedra, is probably near the point at which an alternative framework would make more sense. There is simply no need to explicitly step through all the iterations needed to get to

5000 tetrahedra, picking the best edge to contract at each of the many steps along the way. Fortunately, our generalized metric is equivalent in form to the standard quadric error metric. Therefore, it can trivially form the basis of an efficient clustering [Lindstrom 2000b; Shaffer and Garland 2001] or multiphase [Garland and Shaffer 2002] simplification system.

8.3 Implementation

We believe that one of the great strengths of our error formulation is its uniformity. A single simple summation (Equation 15) is used to construct quadrics for simplices of all dimensions, regardless of the dimension of the space in which they are embedded. Both geometry quadrics and boundary quadrics are treated in fundamentally the same manner.

This uniformity makes it possible to implement simplification algorithms in a very generic way. By making use of C++ templating features, our prototype system consists of a single code base. The error metric is parameterized by vector and matrix classes, and the simplification algorithm itself is parameterized by simplex type. This does away with the need for separate implementations for different model types. There is a single implementation that is specialized at compile-time[4] for the problem being solved.

8.4 Error Norm for Attributes

Our approach to dealing with mesh attributes involves embedding the geometric complex in a higher dimensional space. Once this is done, our error formulation makes no distinction between spatial and attribute sub-spaces. The same is true of the extended quadric metric proposed by Garland and Heckbert [1998]. Hoppe [1999] also made use of this conceptual embedding in a higher dimensional space. However, his method treats spatial and attribute coordinates differently — he finds quadric minima with respect to the spatial embedding and then computes new attributes via projection. While formulated in a slightly different way, Cignoni *et al.* [2000] follow essentially the same approach for scalar fields in tetrahedral meshes.

The difference between these approaches is fundamentally a question of choosing different measures of distance. Our approach always measures the distance of a point to the surface via perpendicular projection onto a local tangent plane. In contrast, the formulation used by Hoppe and Cignoni *et al.* measures attribute error by projection along the attribute axes. Which of these specific formulations is likely to produce the best results for a particular application is quite likely application-specific. However, our choice of perpendicular rather than axial distance measurement produces a formulation that is uniform in all dimensions. As just discussed, this provides significant tangible benefits from an implementation standpoint.

8.5 Higher Order Fields

The development of our error metric for attribute fields, both on surfaces and tetrahedral volumes, made the fundamental assumption that all data fields defined over the domain mesh were scalar fields. It is possible to extend the method to vector

[4]Due to limitations of our matrix library, the tetrahedral code has been manually specialized.

and tensor fields by treating each component of the vector/tensor as a separate scalar field. And as we have seen, this can perform reasonably well in practice. This corresponds to the treatment of RGB color vectors in the work of Garland and Heckbert [1998] and Hoppe [1999].

However, it is important to keep in mind that some care must be taken when treating higher-order data elements in this way. Given two data vectors $\mathbf{s}_i$ and $\mathbf{s}_j$, our method makes the fundamental assumption that the appropriate norm for computing their distance is the usual Euclidean L_2 norm $\|\mathbf{s}_i - \mathbf{s}_j\|_2$. It also assumes that data values can always be extended within simplices by linear interpolation of the values at its corners. The L_2 norm is frequently appropriate for vectors and matrix data — the L_2 norm of the components of a matrix is equivalent to the Frobenius norm of the matrix. However, component-wise linear interpolation may not always be appropriate.

8.6 Non-Simplicial Meshes

Our discussion in this paper has focused exclusively on simplicial complexes. This is frequently the most convenient mesh representation from the standpoint of simplification methods. However, there is nothing that intrinsically limits our approach to simplicial meshes alone. It can easily be used on non-simplicial meshes as well. All that is required to do this is a definition for the quadric of a non-simplicial cell and the necessary data structures to perform contractions on non-simplicial graphs.

(a) Original (b) 1000 (c) 250

Fig. 28. Simplifying a quad-dominant mesh of 6100 vertices.

The most common case of non-simplicial meshes is that of polygonal models containing non-triangular polygons. An example is shown in Figure 28. The central issue in simplifying such models is in defining the fundamental quadric for a polygon, as the polygons are not guaranteed to be planar. In this example, we have simply defined the fundamental quadric for non-planar polygons using a least-squares approximating plane. An alternative would be to explicitly integrate $Q_{\mathbf{p}}$ over a surface defined by the polygon. For example, a quadrilateral defines a bilinear patch, and one could define the quadric of the quadrilateral to be the integration of $Q_{\mathbf{p}}$ over the tangent planes of this bilinear patch.

9. CONCLUSION

We have presented a new generalized simplification system that can process simplicial complexes of any type embedded in any dimension. In particular, we have shown examples of its application to plane and space curves, surfaces, tetrahedral

volumes, and mixed complexes. Our system can successfully process complexes of any topological type, and supports non-manifold complexes as well. We have demonstrated that our method is both very efficient and capable of producing high quality approximations in a variety of domains. Indeed, the results of our system compare very favorably with those produced by other systems. Because of its efficiency, accuracy, and ease of implementation, we believe that it offers a very attractive and practical solution to the simplification problem.

There remain a number of promising directions in which this work can be extended. One of the potential weaknesses of our system is its memory consumption, since the size of the quadric structures is quadratic in the dimension of the vertex data. For the kinds of data encountered in practice, this is not a problem, but it would be useful to explore the possibility of generalizing Hoppe's scheme for reducing the number of necessary coefficients [Hoppe 1999]. Even more dramatic memory reduction could be achieved by implementing a multiphase system [Garland and Shaffer 2002] built on top of our generalized metric. Both the standard quadric metric and our own generalized version fundamentally solve a least squares problem via the normal equations [Garland 1999b]. The method of normal equations is known to suffer from certain numerical instability. Ju *et al.* [2002] describe an alternative quadric formulation based on the much more stable QR factorization. Extending their work to our more generalized setting could be quite beneficial. Finally, Heckbert and Garland [1999] developed an analysis of the standard quadric metric demonstrating that, in the limit of infinitesimal triangles on smooth surfaces, it produces triangles of optimal aspect ratios. Extending this theoretical analysis to our own generalized method should yield valuable insight into its behavior.

ACKNOWLEDGMENTS

Many groups and individuals provided the example models used in this work. The armadillo man and bunny models are courtesy of the Stanford Graphics Lab. Damrong Guoy and the Center for Simulation of Advanced Rockets at UIUC for provided the Titan IV data. The Blunt Fin and Liquid Oxygen Post are publicly available from NASA's Advanced Supercomputing Division. The boundary curves of Virginia and Louisiana are taken from the USGS. Andrew Willmott provided the dragon radiosity solution. The streamline space curves were extracted from a dataset provided by Kitware Inc. as part of the Visualization ToolKit.

This research was funded in part by the National Science Foundation under grants CCR-0098170 and DMR-0121695.

REFERENCES

Agarwal, P. K. and Suri, S. 1994. Surface approximation and geometric partitions. In *Proc. 5th ACM-SIAM Sympos. Discrete Algorithms*. 24–33.

Ballard, D. H. 1981. Strip trees: A hierarchical representation for curves. *Communications of the ACM 24,* 5, 310–321.

Baumgart, B. G. 1974. Geometric modeling for computer vision. Ph.D. thesis, CS Dept, Stanford U. AIM-249, STAN-CS-74-463.

Boxer, L., Chang, C.-S., Miller, R., and Rau-Chaplin, A. 1993. Polygonal approximation by boundary reduction. *Pattern Recognition Letters 14,* 2 (Feb.), 111–119.

Chopra, P. and Meyer, J. 2002. TetFusion: An algorithm for rapid tetrahedral simplification. In *IEEE Visualization 2003*. 133–140.

Ciampalini, A., Cignoni, P., Montani, C., and Scopigno, R. 1997. Multiresolution decimation based on global error. *The Visual Computer 13,* 5, 228–246.

Ciarlet, P. and Lamour, F. 1996. Does contraction preserve triangular meshes? *Numerical Algorithms 13*, 201–223.

Cignoni, P., Costanza, D., Montani, C., Rocchini, C., and Scopigno, R. 2000. Simplification of tetrahedral meshes with accurate error evaluation. In *IEEE Visualization 2000*. 85–92.

Cignoni, P., De Floriani, L., Magillo, P., Puppo, E., and Scopigno, R. 2004. Selective refinement queries for volume visualization of unstructured tetrahedral meshes. *IEEE Transactions on Visualization and Computer Graphics 10,* 1, 29–45.

Cignoni, P., Montani, C., Rocchini, C., and Scopigno, R. 1998. A general method for preserving attribute values on simplified meshes. In *IEEE Visualization 98 Conference Proceedings*. 59–66,518.

Cignoni, P., Montani, C., and Scopigno, R. 1998. A comparison of mesh simplification algorithms. *Computers & Graphics 22,* 1, 37–54.

Cignoni, P., Rocchini, C., and Scopigno, R. 1998. Metro: Measuring error on simplified surfaces. *Computer Graphics Forum 17,* 2 (June), 167–74.

Cohen, J., Olano, M., and Manocha, D. 1998. Appearance-preserving simplification. In *Proceedings SIGGRAPH 98*. 115–122.

Cohen, J., Varshney, A., Manocha, D., Turk, G., Weber, H., Agarwal, P., Brooks, F., and Wright, W. 1996. Simplification envelopes. In *SIGGRAPH '96 Proc.* 119–128.

D'Azevedo, E. F. 1991. Optimal triangular mesh generation by coordinate transformation. *SIAM J. Sci. Stat. Comput. 12,* 4 (July), 755–786.

Dey, T. K., Edelsbrunner, H., Guha, S., and Nekhayev, D. V. 1999. Topology preserving edge contraction. *Publ. Inst. Math. (Beograd) (N.S.) 66*, 23–45.

Douglas, D. H. and Peucker, T. K. 1973. Algorithms for the reduction of the number of points required to represent a digitized line or its caricature. *The Canadian Cartographer 10,* 2 (Dec.), 112–122.

Duda, R. O. and Hart, P. E. 1973. *Pattern Classification and Scene Analysis*. Wiley, New York.

Erikson, C. and Manocha, D. 1999. GAPS: General and automatic polygonal simplification. In *Proc. Symposium on Interactive 3D Graphics '99*. 79–88,225.

Garland, M. 1999a. Multiresolution modeling: Survey & future opportunities. In *State of the Art Report*. Eurographics, 111–131.

Garland, M. 1999b. Quadric-based polygonal surface simplification. Ph.D. thesis, Carnegie Mellon University, CS Dept. Tech. Rept. CMU-CS-99-105.

Garland, M. and Heckbert, P. S. 1997. Surface simplification using quadric error metrics. In *Proceedings of SIGGRAPH 97*. ACM SIGGRAPH, 209–216.

Garland, M. and Heckbert, P. S. 1998. Simplifying surfaces with color and texture using quadric error metrics. In *IEEE Visualization 98 Conference Proceedings*. 263–269,542.

Garland, M. and Shaffer, E. 2002. A multiphase approach to efficient surface simplification. In *IEEE Visualization 2002 Conference Proceedings*. 117–124.

Gelder, A. V., Verma, V., and Wilhelms, J. 1999. Volume decimation of irregular tetrahedral grids. In *Computer Graphics International*. 222–.

Guéziec, A. 1995. Surface simplification with variable tolerance. In *Second Annual Intl. Symp. on Medical Robotics and Computer Assisted Surgery (MRCAS '95)*. 132–139.

Guéziec, A. 1996. Surface simplification inside a tolerance volume. Tech. rep., Yorktown Heights, NY 10598. Mar. IBM Research Report RC 20440, http://www.watson.ibm.com:8080/search_paper.shtml.

Heckbert, P. S. and Garland, M. 1997. Survey of polygonal surface simplification algorithms. In *Multiresolution Surface Modeling Course Notes*. ACM SIGGRAPH.

Heckbert, P. S. and Garland, M. 1999. Optimal triangulation and quadric-based surface simplification. *Computational Geometry: Theory and Applications 14,* 1–3 (November), 49–65.

HERSHBERGER, J. AND SNOEYINK, J. 1994. An $o(n \log n)$ implementation of the Douglas-Peucker algorithm for line simplification. In *Proc. 10th Annual Symposium on Computational Geometry*. ACM, 383–384.

HOPPE, H. 1996. Progressive meshes. In *Proceedings of SIGGRAPH 96*. ACM SIGGRAPH, 99–108.

HOPPE, H., DEROSE, T., DUCHAMP, T., MCDONALD, J., AND STUETZLE, W. 1993. Mesh optimization. In *SIGGRAPH '93 Proc.* 19–26.

HOPPE, H. H. 1999. New quadric metric for simplifying meshes with appearance attributes. *IEEE Visualization '99*, 59–66.

IHM, I. AND NAYLOR, B. 1991. Piecewise linear approximations of digitized space curves with applications. In *Scientific Visualization of Physical Phenomena*, N. M. Patrikalakis, Ed. Springer-Verlag, Tokyo, 545–569.

IMAI, H. AND IRI, M. 1988. Polygonal approximations of a curve – formulations and algorithms. In *Computational Morphology*, G. T. Toussaint, Ed. Elsevier Science, 71–86.

JU, T., LOSASSO, F., SCHAEFER, S., AND WARREN, J. 2002. Dual contouring of hermite data. *ACM Transactions on Graphics 21,* 3 (July), 339–346.

KHO, Y. AND GARLAND, M. 2003. User guided simplification. In *Proceedings of the Symposium on Interactive 3D Graphics*. ACM SIGGRAPH, 123–126.

LEU, J.-G. AND CHEN, L. 1988. Polygonal approximation of 2-D shapes through boundary merging. *Pattern Recognition Letters 7,* 4 (Apr.), 231–238.

LINDSTROM, P. 2000a. Model simplification using image and geometry-based metrics. Ph.D. thesis, Georgia Institute of Technology.

LINDSTROM, P. 2000b. Out-of-core simplification of large polygonal models. *Proceedings of SIGGRAPH 2000*, 259–262.

LINDSTROM, P. AND TURK, G. 1998. Fast and memory efficient polygonal simplification. In *Proceedings of IEEE Visualization 98*. 279–286,544.

LUEBKE, D., REDDY, M., COHEN, J. D., VARSHNEY, A., WATSON, B., AND HUEBNER, R. 2002. *Level of Detail for 3-D Graphics*. Morgan Kaufmann.

MARUYA, M. 1995. Generating a texture map from object-surface texture data. *Computer Graphics Forum 14,* 3, 397–405,506–507. Proc. Eurographics '95.

MEYER, M., DESBRUN, M., SCHRÖDER, P., AND BARR, A. H. 2003. Discrete differential-geometry operators for triangulated 2-manifolds. In *Visualization and Mathematics III*, H.-C. Hege and K. Polthier, Eds. Springer-Verlag, Heidelberg, 35–57.

NADLER, E. 1986. Piecewise linear best L_2 approximation on triangulations. In *Approximation Theory V*, C. K. Chui et al., Eds. Academic Press, Boston, 499–502.

PAVLIDIS, T. 1977. *Structural Pattern Recognition*. Springer-Verlag, Berlin.

POPOVIĆ, J. AND HOPPE, H. 1997. Progressive simplicial complexes. In *Proceedings of SIGGRAPH 97*. ACM SIGGRAPH, 217–224.

PRESS, W. H., TEUKOLSKY, S. A., VETTERLING, W. T., AND FLANNERY, B. P. 1992. *Numerical Recipes in C: The Art of Scientific Computing*, Second ed. Cambridge University Press.

RAMER, U. 1972. An iterative procedure for the polygonal approximation of plane curves. *Computer Graphics and Image Processing 1*, 244–256.

RENZE, K. J. AND OLIVER, J. H. 1996. Generalized unstructured decimation. *IEEE Computer Graphics & Applications 16,* 6 (Nov.), 24–32.

RONFARD, R. AND ROSSIGNAC, J. 1996. Full-range approximation of triangulated polyhedra. *Computer Graphics Forum 15,* 3 (Aug.). Proc. Eurographics '96.

ROSSIGNAC, J. AND BORREL, P. 1993. Multi-resolution 3D approximations for rendering complex scenes. In *Modeling in Computer Graphics: Methods and Applications*, B. Falcidieno and T. Kunii, Eds. Springer-Verlag, Berlin, 455–465. Proc. of Conf., Genoa, Italy, June 1993. (Also available as IBM Research Report RC 17697, Feb. 1992, Yorktown Heights, NY 10598).

RUPPERT, J. AND SEIDEL, R. 1992. On the difficulty of triangulating three-dimensional nonconvex polyhedra. *Discrete & Computational Geometry 7*, 227–253.

Schroeder, W., Martin, K., and Lorensen, B. 2003. *The Visualization Toolkit: An Object-Oriented Approach to 3D Graphics*, Third ed. Kitware, Inc.

Schroeder, W. J., Zarge, J. A., and Lorensen, W. E. 1992. Decimation of triangle meshes. *Computer Graphics (SIGGRAPH '92 Proc.) 26,* 2 (July), 65–70.

Shaffer, E. and Garland, M. 2001. Efficient adaptive simplification of massive meshes. In *Proceedings of IEEE Visualization 2001*. 127–134.

Shewchuk, J. R. 1996. Triangle: Engineering a 2D quality mesh Generator and Delaunay triangulator. In *Applied Computational Geometry: Towards Geometric Engineering*, M. C. Lin and D. Manocha, Eds. Lecture Notes in Computer Science, vol. 1148. Springer-Verlag, 203–222. From the First ACM Workshop on Applied Computational Geometry.

Soucy, M., Godin, G., and Rioux, M. 1996. A texture-mapping approach for the compression of colored 3D triangulations. *The Visual Computer 12,* 10, 503–514.

Staadt, O. G. and Gross, M. H. 1998. Progressive tetrahedralizations. In *IEEE Visualization 98 Conference Proceedings*. 397–402,555.

Strang, G. 1988. *Linear Algebra and its Applications*, Third ed. Harcourt Brace Jovanovich, San Diego.

Trotts, I. J., Hamann, B., and Joy, K. I. 1999. Simplification of tetrahedral meshes with error bounds. *IEEE Transactions on Visualization and Computer Graphics 5,* 3 (July-September), 224–237.

Trotts, I. J., Hamann, B., Joy, K. I., and Wiley, D. F. 1998. Simplification of tetrahedral meshes. In *IEEE Visualization '98*. 287–296.

Turk, G. 1992. Re-tiling polygonal surfaces. *Computer Graphics (SIGGRAPH '92 Proc.) 26,* 2 (July), 55–64.

Turner, K. J. 1974. Computer perception of curved objects using a television camera. Ph.D. thesis, U. of Edinburgh, Scotland.

Weibel, R. 1997. Generalization of spatial data: Principles and selected algorithms. In *Algorithmic Foundations of GIS*, M. V. Kreveld, J. Nievergelt, T. Roos, and P. Widmayer, Eds. Lecture Notes in Computer Science, vol. 1340. Springer-Verlag, Berlin, 99–152.

Yu, Y. 2001. Modeling realistic virtual hairstyles. In *Proceedings of Pacific Graphics*. 295–304.

Zelinka, S. and Garland, M. 2002. Permission grids: Practical, error-bounded simplification. *ACM Transactions on Graphics 21,* 2 (April), 207–229.