Survey of Polygonal Surface Simplification Algorithms

Paul S. Heckbert and Michael Garland

1 May 1997

School of Computer Science
Carnegie Mellon University
Pittsburgh, PA 15213

Multiresolution Surface Modeling Course
SIGGRAPH '97

This is a draft of a Carnegie Mellon University technical report, to appear. See http://www.cs.cmu.edu/∼ph for final version.

Send comments or corrections to the authors at: {ph,garland}@cs.cmu.edu

Abstract

This paper surveys methods for simplifying and approximating polygonal surfaces. A polygonal surface is a piecewise-linear surface in 3-D defined by a set of polygons; typically a set of triangles. Methods from computer graphics, computer vision, cartography, computational geometry, and other fields are classified, summarized, and compared both practically and theoretically. The surface types range from height fields (bivariate functions), to manifolds, to non-manifold self-intersecting surfaces. Piecewise-linear curve simplification is also briefly surveyed.

This work was supported by ARPA contract F19628-93-C-0171 and NSF Young Investigator award CCR-9357763.

Keywords: multiresolution modeling, surface approximation, piecewise-linear surface, triangulated irregular network, mesh coarsening, decimation, non-manifold, cartographic generalization, curve simplification, level of detail, greedy insertion.

Contents

1 Introduction

The simplification of surfaces has become increasingly important as it has become possible in recent years to create models of greater and greater detail. Detailed surface models are generated in a number of disciplines. For example, in computer vision, range data is captured using scanners; in scientific visualization, isosurfaces are extracted from volume data with the "marching cubes" algorithm; in remote sensing, terrain data is acquired from satellite photographs; and in computer graphics and computer-aided geometric design, polygonal models are generated by subdivision of curved parametric surfaces. Each of these techniques can easily generate surface models consisting of millions of polygons.

Simplification is useful in order to make storage, transmission, computation, and display more efficient. A compact approximation of a shape can reduce disk and memory requirements and can speed network transmission. It can also accelerate a number of computations involving shape information, such as finite element analysis, collision detection, visibility testing, shape recognition, and display. Reducing the number of polygons in a model can make the difference between slow display and real time display.

A variety of methods for simplifying curves and surfaces have been explored over the years. Work on this topic is spread among a number of fields, making literature search quite challenging. These fields include: cartography, geographic information systems (GIS), virtual reality, computer vision, computer graphics, scientific visualization, computer-aided geometric design, finite element methods, approximation theory, and computational geometry.

Some prior surveys of related methods exist, notably a bibliography on approximation [45], a survey of spatial data structures for curves and surfaces [106], and surveys of triangulation methods with both theoretical [6] and scientific visualization [89] orientations. None of these surveys surface simplification in depth, however.

The present paper attempts to survey all previous work on surface simplification and place the algorithms in a taxonomy. In this taxonomy, we intermix algorithms from various fields, classifying algorithms not according to the application for which they were designed, but according to the technical problem they solve. By doing so, we find great similarities between algorithms from disparate fields. For example, we find common ground between methods for representing terrains developed in cartography, methods for approximating bivariate functions developed in computational geometry and approximation theory, and methods for approximating range data developed in computer vision. This is not too surprising, since these are fundamentally the same technical problem. By calling attention to these similarities, and to the past duplication of work, we hope to facilitate cross-fertilization between disciplines.

Our emphasis is on methods that take polygonal surfaces as input and produce polygonal surfaces as output, although we touch on curved parametric surface and volume techniques. Our polygons will typically be planar triangles. Although surface simplification is our primary interest, we also discuss curve simplification, because many surface methods are simple generalizations of curve methods.

1.1 Characterizing Algorithms

Methods for simplifying curves and surfaces vary in their generality and approach – among surface methods, some are limited to height fields, for example, while others are applicable to general surfaces in 3-D. To systematize our taxonomy, we will classify methods according to the problems that they solve and the algorithms they employ. Below is a list of the primary characteristics with which we will do so:

Problem Characteristics

Topology and Geometry of Input: For curves, the input can be a set of points, a function $y(x)$, a planar curve, or a space curve. For surfaces, the input can be a set of points, samples of a height field $z(x, y)$ in a regular grid or at scattered points, a manifold[1], a manifold with boundary, or a set of surfaces with arbitrary topology (e.g. a set of intersecting polygons).

Other Attributes of Input: Color, texture, and surface normals might be provided in addition to geometry.

Domain of Output Vertices: Vertices of the output can be restricted to be a subset of the input points, or they can come from the continuous domain.

Structure of Output Triangulation: Meshes can be regular grids, they can come from a hierarchical subdivision such as a quadtree, or they can be a general subdivision such as a Delaunay or data-dependent triangulation.

Approximating Elements: The approximating curve or surface elements can be piecewise-linear (polygonal), quadratic, cubic, high degree polynomial, or some other basis function.

Error Metric: The error of the approximation is typically measured and minimized with respect to L_2 or L_∞ error[2]. Distances can be measured in various ways, e.g., to the closest point on a given polygon, or closest point on the entire surface.

Constraints on Solution: One might request the most accurate approximation possible using a given number of elements (e.g. line segments or triangles), or one might request the solution using the minimum number of elements that satisfies a given error tolerance. Some algorithms give neither type of guarantee, but give the user only indirect control over the speed/quality tradeoff. Other possible constraints include limits on the time or memory available.

Algorithm Characteristics

Speed/Quality Tradeoff: Algorithms that are optimal (minimal error and size) are typically slow, while algorithms that generate lower quality or less compact approximations can generally be faster.

Refinement/Decimation: Many algorithms can be characterized as using either *refinement*, a coarse-to-fine approach starting with a minimal approximation and building up more and more accurate ones, or *decimation*, a fine-to-coarse approach starting with an exact fit, and discarding details to create less and less accurate approximations.

1.2 Background on Application Areas

The motivations for surface simplification differ from field to field. Terminology differs as well.

[1] A *manifold* is a surface for which the infinitesimal neighborhood of every point is topologically equivalent to a disk. In a triangulated manifold, each edge belongs to two triangles. In a triangulated *manifold with boundary*, each edge belongs to one or two triangles.

[2] In this paper, we use the following error metrics: We define the L_2 error between two n-vectors $\mathbf{u}$ and $\mathbf{v}$ as $||\mathbf{u}-\mathbf{v}||_2 = \left[\sum_{i=1}^{n}(u_i - v_i)^2\right]^{1/2}$. The L_∞ error, also called the *maximum error*, is $||\mathbf{u}-\mathbf{v}||_\infty = \max_{i=1}^{n}|u_i - v_i|$. We define the *squared error* to be the square of the L_2 error, and the *root mean square* or RMS error to be the L_2 error divided by $\sqrt{n}$. Optimization with respect to the L_2 and L_∞ metrics are called *least squares* and *minimax* optimization, and we call such solutions L_2–*optimal* and L_∞–*optimal*, respectively.

Cartography. In cartography, simplification is one method among many for the "generalization" of geographic information [86]. In that field, curve simplification is called "line generalization". It is used to simplify the representations of rivers, roads, coastlines, and other features when a map with large scale is produced. It is needed for several reasons: to remove unnecessary detail for aesthetic reasons, to save memory/disk space, and to reduce plotting/display time. The principal surface type simplified in cartography is, of course, the terrain. Map production was formerly a slow, off-line activity, but it is currently becoming more interactive, necessitating the development of better simplification algorithms.

The ideal error measures for cartographic simplification include considerations of geometric error, viewer interest, and data semantics. Treatment of the latter issues is beyond the scope of this study. The algorithms summarized here typically employ a geometric error measure based on Euclidean distance. The problem is thus to retain features larger than some size threshold, typically determined by the limits of the viewer's perception, the resolution of the display device, or the available time or memory.

Computer Vision. Range data acquired by stereo or structured light techniques (e.g. lasers) can easily produce millions of data points. It is desirable to simplify the surface models created from this data in order to remove redundancy, save space, and speed display and recognition tasks. The acquired data is often noisy, so tolerance of and smoothing of noise are important considerations here.

Computer Graphics. In computer graphics and the closely related fields of virtual reality, computer-aided geometric design, and scientific visualization, compact storage and fast display of shape information are vital. For interactive applications such as military flight simulators, video games, and computer-aided design, real time performance is a very important goal. For such applications, the geometry can be simplified to multiple levels of detail, and display can switch or blend between the appropriate levels of detail as a function of the screen size of each object [13, 52]. This technique is called *multiresolution modeling*. Redisplaying a static scene from a moving viewpoint is often called a *walkthrough*. For off-line, more realistic simulations such as special effects in entertainment, real time is not vital, but reasonable speed and storage are nevertheless important.

When 3-D shape models are transmitted, compression is very important. This applies whether the channel has very low bandwidth (e.g. a modem) or higher bandwidth (e.g. the Internet backbone). The rapid growth of the World Wide Web is spurring some of the current work in surface simplification.

Finite Element Analysis. Engineers use the finite element method for structural analysis of bridges, to simulate the air flow around airplanes, and to simulate electromagnetic fields, among other applications. A preprocess to simulation is a "mesh generation" step. In 2-D mesh generation, the domain, bounded by curves, is subdivided into triangles or quadrilaterals. In 3-D mesh generation, the domain is given by boundary surfaces. Surface meshes of triangles or quadrilaterals are first constructed, and then the volume is subdivided into tetrahedra or hexahedra. The criteria for a good mesh include both geometric fidelity and considerations of the physical phenomena being simulated (stress, flow, etc). To speed simulation, it is desirable to make the mesh as coarse as possible while still resolving the physical features of interest. In 3-D simulations, surface details such as bolt heads might be eliminated, for example, before meshing the volume. This community calls simplification "mesh coarsening".

Approximation Theory and Computational Geometry. What is called a terrain in cartography or a height field in computer graphics is called a bivariate function or a function of two variables in more theoretical fields. The goal in approximation theory is often to characterize the error in the limit as the mesh becomes infinitely fine. In computational geometry the goal is typically to find algorithms to generate approximations with optimal or near-optimal compactness, error, or speed or to prove bounds on these. Implementation of algorithms and low level practical optimizations receive less attention.

2 Curve Simplification

Curve simplification has been used in cartography, computer vision, computer graphics, and a number of other fields.

A basic curve simplification problem is to take a polygonized curve with n vertices (a chain of line segments or "polyline") as input and produce an approximating polygonized curve with m vertices as output. A closely related problem is to take a curve with n vertices and approximate it within a specified error tolerance.

Douglas-Peucker Algorithm. The most widely used high-quality curve simplification algorithm is probably the heuristic method commonly called the Douglas-Peucker[3] algorithm. It was independently invented by many people [99], [31], [30, p. 338], [5, p. 92], [125], [91, p. 176], [3]. At each step, the Douglas-Peucker algorithm attempts to approximate a sequence of points by a line segment from the first point to the last point. The point farthest from this line segment is found, and if the distance is below threshold, the approximation is accepted, otherwise the algorithm is recursively applied to the two subsequences before and after the chosen point. This algorithm, though not optimal, has generally been found to produce the highest subjective- and objective-quality approximations when compared with many other heuristic algorithms [85, 130]. Its best case time cost[4] is $\Omega(n)$, its worst case cost is $O(mn)$, and its expected time cost is about $\Theta(n \log m)$. The worst case behavior can be improved, with some sacrifice in the best case behavior, using a $\Theta(n \log n)$ algorithm employing convex hulls [54].

A variant of the Douglas-Peucker algorithm described by Ballard and Brown [4, p. 234] on each iteration splits at the point of highest error along the whole curve, instead of splitting recursively. This yields higher quality approximations for slightly more time. If this subdivision tree is saved, it is possible to dynamically build an approximation for any larger error tolerance very quickly [18].

A potential problem is that simplification can cause a simple polygon to become self-intersecting. This could be a problem in cartographic applications.

Faster or Higher Quality Algorithms. There are faster algorithms than Douglas-Peucker, but all of these are generally believed to have inferior quality [84]. One such algorithm is the trivial method of *regular subsampling* (known as the "nth-point algorithm" in cartography), which simply keeps every kth point of the input, for some k, discarding the rest. This algorithm is very fast, but will sometimes yield very poor quality approximations.

Least squares techniques are commonly used for curve fitting in pattern recognition and computer vision, but they do not appear to be widely used for that purpose in cartography.

Polygonal Boundary Reduction. While the Douglas-Peucker algorithm and its variants are refinement algorithms, curves can also be simplified using decimation methods. Boxer *et al.* [8] describe two such algorithms for simplifying 2-D polygons. The first, due to Leu and Chen [75], is a simple decimation algorithm. It considers boundary arcs of 2 and 3 edges. For each arc, it computes the maximum distance between the arc and the chord connecting its endpoints. It then selects an independent set of arcs whose deviation is less than some threshold, and replaces them by their chords. The second algorithm is an improvement of this basic algorithm which guarantees that the approximate curve is always within some bounded distance from the original. They state that the running time of the simple algorithm is $\Theta(n)$, while the bounded-error algorithm requires $O(n + r^2)$ time where r is the number of vertices removed.

Optimal Approximations. Algorithms for optimal curve simplification are much less common than heuristic methods, probably because they are slower and/or more complicated to implement. In a common form of optimal curve simplification, one searches for the approximation of a given size with minimum error, according to some

[3] Pronounced, and later spelled, due to name change, "Poiker".

[4] A function is in $O(f(n))$ if it is less than or equal to $cf(n)$ as $n \to \infty$, for some positive constant c. "O" is used for upper bounds.
A function is in $\Theta(f(n))$ if it is between $c_1 f(n)$ and $c_2 f(n)$ as $n \to \infty$, for some positive constants c_1, c_2.
A function is in $\Omega(f(n))$ if it is greater than or equal to $cf(n)$ as $n \to \infty$, for some positive constant c. "Ω" is used for lower bounds.

definition of "error". Typically the output vertices are restricted to be a subset of the input vertices. A naive, exhaustive algorithm would have exponential cost, since the number of subsets is exponential, but using dynamic programming and/or geometric properties, the cost can be reduced to polynomial. The L_2–optimal approximation to a function $y(x)$ can be found in $O(mn^2)$ time, worst case, using dynamic programming. Remarkably, a slight variation in the error metric permits a much faster algorithm: the L_∞–optimal approximation to a function can be found in $O(n)$ time [63], using visibility techniques (see also [123, 91]). When the problem is generalized from functions to planar curves, the complexity of the best L_∞–optimal algorithms we know of jumps to $O(n^2 \log n)$ [63]. These methods use shortest-path graph algorithms or convex hulls. For space curves (curves in 3-D), there are $O(n^3 \log m)$ L_∞–optimal algorithms [62].

Asymptotic Approximation. In related work, McClure and de Boor analyzed the error when approximating a highly continuous function $y(x)$ using piecewise-polynomials with variable knots [82, 21]. We discuss only the special case of piecewise-linear approximations. They analyzed the asymptotic behavior of the L_p error of approximation in the limit as m, the number of vertices (knots) of the approximation, goes to infinity. They showed that the asymptotic L_p error with regular subsampling is proportional to m^{-2}, for any p. The L_p–optimal approximation has the same asymptotic behavior, though with a smaller constant. McClure showed that the spacing of vertices in the optimal approximation is closely related to the function's second derivative. Specifically, he proved that as $m \to \infty$, the density of vertices at each point in the optimal L_2 approximation becomes proportional to $|y''(x)|^{2/5}$. For optimal L_∞ approximations, the density is proportional to $|y''(x)|^{1/2}$. Also, as $m \to \infty$, all intervals have equal error in an L_p–optimal approximation.

The density property and the balanced error property described above can be used as the basis for curve simplification algorithms [82]. Although adherence to neither property guarantees optimality for real simplification problems with finite m, iterative balanced error methods have been shown to generate good approximations in practice [91, p. 181]. Another caveat is that many curves in nature do not have continuous derivatives, but instead have some fractal characteristics [80]. Nevertheless, these theoretical results suggest the importance of the second derivative, and hence curvature, in the simplification of curves and surfaces.

Summary of Curve Simplification. The Douglas-Peucker algorithm is probably the most commonly used curve simplification algorithm. Most implementations have $O(mn)$ cost, worst case, but typical cost of $\Theta(n \log m)$. An optimization with worst case cost of $O(n \log n)$ is available, however. Optimal simplification typically has quadratic or cubic cost, making it impractical for large inputs.

3 Surface Simplification

Surfaces are more difficult to simplify than curves. In the flight simulator field, lower level of detail models have traditionally been prepared by hand [17]. The results can be excellent, but the process can take weeks. Automatic methods are preferable for large and dynamic databases, however.

If only a single level of detail is needed, then in some cases, simplification can be obviated by simply avoiding generation of redundant data in the first place. In scientific visualization, for example, the marching cubes algorithm [90] is widely used. It generates many tiny triangles without testing for coplanarity between neighbors. A more sophisticated alternative is adaptive polygonization that subdivides finely only where the surface is highly curved [7]. In computer aided geometric design, when polygonizing parametric surfaces, rather than subdivide and polygonize a surface with a regular (u, v) grid, better results are often possible by subdividing adaptively based on curvature [10].

When simplification is needed however, one of the algorithms summarized below can be used.

Taxonomy of Surface Simplification Algorithms. We categorize algorithms at the highest level according to the class of surfaces on which they operate:

- height fields and parametric surfaces,

- manifold surfaces, and
- non-manifold surfaces.

Within each surface class we often group algorithms according to whether they work by refinement or decimation. Within the subclasses, methods are generally listed chronologically. We have attempted to be fairly comprehensive, so consequently the good methods are described along with the bad. As we summarize algorithms, we list their computational complexities and quote empirical times, where known (of course, hardware, compilers, languages, and programming styles differ between individuals, so we must be careful when judging based on this information). Complexities are given in terms of n, the number of vertices in the input, and m, the number of vertices in the output. Typically, $m \ll n$.

3.1 Height Fields and Parametric Surfaces

Height fields and parametric surfaces are the simplest class of surfaces. Within this class of surfaces, we divide methods into the following six sub-classes: regular grid methods, hierarchical subdivision methods, feature methods, refinement methods, decimation methods, and optimal methods.

Regular grid methods are the simplest techniques, using a grid of samples equally and periodically spaced in x and y. The hierarchical subdivision methods are based on quadtree, k-d tree, and hierarchical triangulations using a divide and conquer strategy. They recursively subdivide the surface into regions, constructing a tree-structured hierarchy. The next four categories employ more general subdivision and triangulation methods, most commonly Delaunay triangulation. Feature methods select a set of important "feature" points in one pass and use them as the vertex set for triangulation. Refinement methods are essentially generalizations of the Douglas-Peucker algorithm from curves to surfaces, where intervals are replaced by triangles and splitting is replaced by retriangulating. They start with a minimal approximation and use multiple passes of point selection and retriangulation to build up the final triangulation. Decimation methods use an approach opposite that of refinement methods: they begin with a triangulation of all of the input points and iteratively delete vertices from the triangulation, gradually simplifying the approximation. Refinement methods thus work top-down, while decimation methods work bottom-up. The final category, "optimal methods" are distinguished more for their theoretical focus than for their method.

For many height field simplification tasks, the input is a height field and the output is a general triangulation, called a *triangulated irregular network*, or TIN, in cartography. A TIN is a mesh of triangles where height is a function of x and y: $H(x, y)$. Examples of a height field and general triangulation are shown in Figures 1 and 2.

3.1.1 Triangulation

Most polygonal surface simplification methods employ triangles as their approximating elements when constructing a surface. For height fields and parametric surfaces, there is a natural 2-D parameterization of the surface. Basic triangulation methods are described in a 2-D domain, or in a 3-D domain where height z is a function of x and y.

In general, the topology of the triangulation can be chosen using only the xy projections of the input points, or it can be chosen using the heights of the input points as well. The latter approach is called data-dependent triangulation [32].

The most popular triangulation method that does not use height values is Delaunay triangulation; it is a purely two-dimensional method. Delaunay triangulation finds the triangulation that maximizes the minimum angle of all triangles, among all triangulations of a given point set. This helps to minimize the occurrence of very thin *sliver* triangles. Delaunay triangulations have a number of nice theoretical properties that make them very popular in computational geometry. In a Delaunay triangulation, the circumscribing circle (circumcircle) of each triangle contains no vertices in its interior [71]. Delaunay triangulations of m points can either be computed whole, using divide-and-conquer or sweepline algorithms, or incrementally, by inserting vertices one at a time, updating the triangulation after each insertion [48]. The former approach has cost $O(m \log m)$, while the latter, incremental method has worst case cost of $O(m^2)$. Typical costs for the incremental approach are much better than quadratic, however.

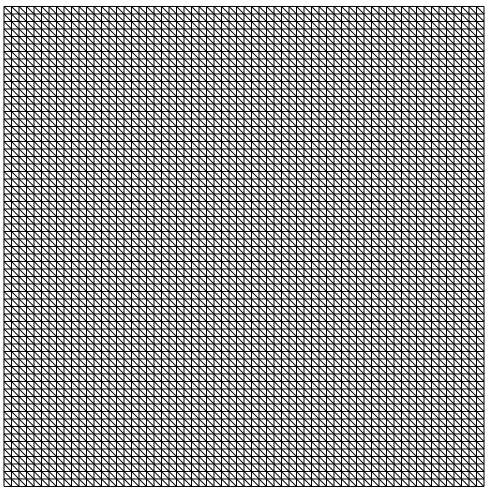

Figure 1: Top view of a regular grid triangulation of 65×65 height field.

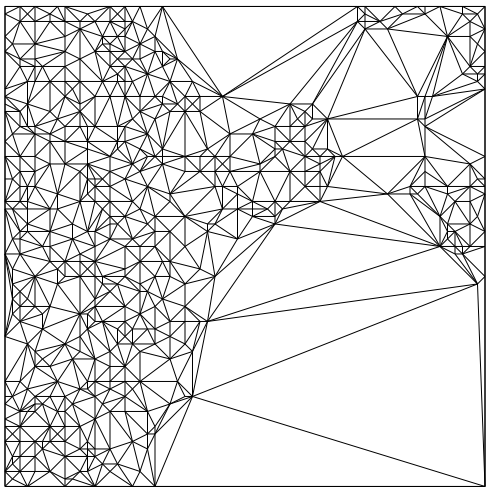

Figure 2: A triangulation using 512 vertices approximating the height field.

Sometimes equilateral triangles are not optimal, and maximization of the minimum angle is not the appropriate goal. Triangulation methods that attempt to optimize the approximation of z or other data associated with the triangulation are called *data-dependent triangulation* methods. Several researchers have shown that slivers can be good when the surface being approximated is highly curved in one direction, but not the other [102, 88, 20, 32]. Such slivers would not be generated by Delaunay triangulation, which minimizes slivers by tending to choose "fat" triangles.

3.1.2 Regular Grid Methods

The simplest method for approximation of surface grids is regular subsampling, in which the points in every kth row and column are kept and formed into a grid, and all other points are discarded. Regular grids are also known as uniform grids, and sometimes the term *DEM* (digital elevation model) is used in the specific sense of a regular grid terrain model. As with curves, regular subsampling is simple and fast, but low quality, since the points discarded might be the most important ones. The results are improved if a low pass filter [51] is run across the data before subsampling, but this still does not fix the basic problem with this method, its non-adaptive nature.

Kumler 94. An extensive comparison of regular grids (DEMs) and general triangulations (TINs), and the space/error tradeoffs between them, was done by Kumler [70]. He concluded, surprisingly, that for a given amount of storage space, regular grids approximate terrains better than general triangulations. His comparison seems biased against general triangulations, however, since he compares models of equal memory size, not equal rendering time, and the simplification algorithms he uses are not the best known [39]. Kumler assumes that general triangulations require three to ten times the memory of regular grids with the same number of vertices.

Pyramids. Regular subsampling can be done hierarchically, forming a pyramid of samples [131, 121]. Despite the wealth of research on hierarchical triangulations and TINs, pyramids are probably the most widely used type of multiresolution terrain model in the simulator commu-

nity [16, 17] and in the visualization/animation community [60], because of their simplicity and compactness.

3.1.3 Hierarchical Subdivision Methods

Hierarchical subdivision methods construct a triangulation by recursively subdividing a surface. They are the adaptive form of pyramids. The hierarchical pattern of subdivision, even if not stored explicitly in the data structure, forms a tree, each node of which has no more than one parent. With Delaunay triangulation and other general triangulation algorithms, the topology is not hierarchical, because a triangle might have multiple parents. Hierarchical subdivision methods are generally fast, simple, and they facilitate multiresolution modeling. In perspective scenes where nearby portions of the terrain require more detail than distant regions, the hierarchy facilitates rendering at adaptive levels of detail. Nearby portions are drawn at at a fine level, while distant regions are drawn at a coarse level. The penalty for their simplicity and speed is that hierarchical subdivision methods typically yield poorer quality approximations than more general triangulation methods.

Quadtrees and k-d Trees. Terrains and parametric surfaces are easily simplified using adaptive quadtree and k-d tree subdivision methods [106, 105]. DeHaemer and Zyda used quadtree and k-d tree splitting at the point of maximum error within each cell to approximate general 3-D surfaces described by a grid [27]. Taylor and Barrett described a similar method for terrains [122]. Von Herzen and Barr discuss a method for crack-free adaptive triangulation of parametric surfaces using quadtrees [128]. Gross, Gatti, and Staadt have used wavelets to construct quadtree approximations of height fields [44]. With an unoptimized implementation, they were able to simplify a 256×256 terrain in about 2 seconds on an SGI Indy.

Gómez-Guzmán 79. A *quaternary triangulation* method for height field approximation was proposed by Gómez and Guzmán [42]. In their method, each triangle is recursively subdivided into four subtriangles until a maximum error tolerance is met. To subdivide each triangle, a "significant" point near the midpoint of each edge is chosen (in some unspecified way), and the triangle is split into four nearly congruent triangles (Figure 3). Since the new vertices are not constrained to lie on the edges, however, the surface develops unsightly cracks, rendering the method unsuitable for most purposes.

De Floriani-Falcidieno-Nagy-Pienovi 84. In 1984, De Floriani *et al.* published a hierarchical *ternary triangulation* method in which points are inserted in triangle interiors and each triangle is split into three subtriangles by adding edges to its vertices [23]. No edge swapping is done (Figure 4). Consequently, all of the initial edges remain in the triangulation forever, most notably the diagonal across the entire grid rectangle, leading to spurious knife-edge ridges and valleys through the terrain. The flaws of this method make it unacceptable.

Schmitt 85. Schmitt and Gholizadeh simplified a grid with rectangular topology in 3-D using a triangulated surface [112]. Their method is similar to that of Faugeras *et al.* [34], described later. Having the input points in a grid allows the partition of points into triangles to be done in a two dimensional parametric space. The method begins with a small number of triangles and repeatedly splits those triangles whose associated input points are above the error tolerance. Triangles are subdivided into 2–4 subtriangles by splitting one, two, or three edges of the triangle. Triangle splitting is done in no particular order. They report that simplifying a grid of $n = 288 \times 360$ points down to about $m = 3,500$ points takes 1.5 hours on a DEC VAX 780. The computational complexity of their algorithm is $O(mn)$.

Scarlatos-Pavlidis 92a. The hierarchical triangulation algorithm for height fields developed by Scarlatos and Pavlidis employs a recursive triangulation approach [108, 107]. Their method begins with a minimal triangulation (typically two triangles) as level of detail 0. Error tolerances for each level of detail in the tree are specified by the user. To create level i from level $i-1$, the point of highest error along each triangle edge and in each triangle interior is found, those points with error above the threshold for level i are taken as new vertices, and each triangle is retriangulated using one of five simple subdivision templates (Figure 5). Passes of vertex selection and retriangulation

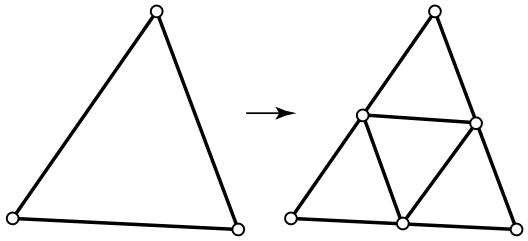

Figure 3: Quaternary triangulation.

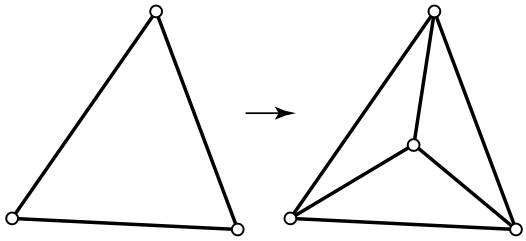

Figure 4: Ternary triangulation.

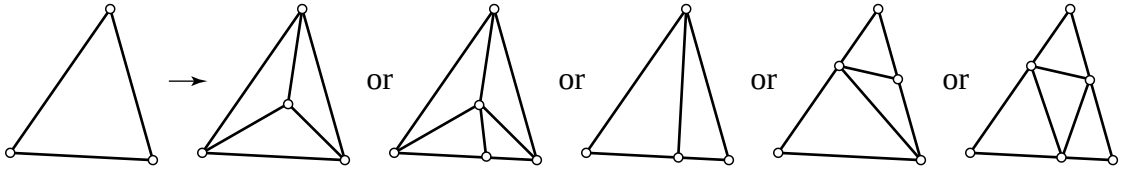

Figure 5: Subdivision templates for Scarlatos and Pavlidis' hierarchical triangulation.

for level i are repeated until no more candidates for that level are found. All levels of the hierarchy are retained in the data structure, facilitating adaptive display at any desired detail level. In their analysis, Scarlatos and Pavlidis suggest that the cost of the algorithm is $O(n \log n)$. Our analysis of their algorithm is that their expected case is $O(n \log m)$, but if the hierarchy is very unbalanced, the worst case cost is $O(mn)$.

De Floriani-Puppo 92. A similar method was developed by De Floriani and Puppo [26]. The triangle subdivision is more general, however. To subdivide a triangle for a given level in the hierarchy, a curve approximation algorithm [4] is used to add new vertices along the edges, then additional points are inserted in the interior of the triangle until the error threshold is met throughout the triangle, and the interior of the triangle is retriangulated using Delaunay triangulation. The method appears to have nearly identical flexibility and speed compared to Scarlatos and Pavlidis' method [108], but it will probably yield slightly better simplification for a given error threshold.

3.1.4 Feature Methods

A simple, intuitive approach to height field simplification is to make one pass over the input points, ranking each of them using some "importance" measure, to select the most important points as the vertex set, and construct a triangulation of these points. Typically, Delaunay triangulation is used. Feature methods are quite popular in cartography. Overall, our conclusion is that their quality relative to many of the other methods is inferior, so we only survey them briefly here.

Important points, also known as "features" or "critical points" and the edges between them, often known as "break lines", include such topographic features as peaks, pits, ridges, and valleys. The philosophy of many of the feature approaches is that some knowledge about the nature of terrains is essential for good simplification [129, 108]. In a feature approach, the chosen features become the vertex set, and the chosen break lines (if any) become edges in a constrained triangulation [6]. The most commonly used feature detectors are 2×2 and 3×3 linear or nonlinear filters, sometimes followed by a weeding process that discards features that are too close together,

such as a sequence of points along a ridge line. Such approaches were employed by Peucker-Douglas and Chen-Guevara [92, 12]. Some methods examine larger neighborhoods of points in an attempt to measure importance more globally.

Southard 91. One of the more interesting feature methods is Southard's [120]. He uses the Laplacian as a measure of curvature. The rank of each point's Laplacian is computed within a moving window, analogous to a median filter in image processing, and all points whose rank is below some threshold are selected. This is an improvement over the selection criteria of Peucker-Douglas and Chen-Guevara cited earlier, because it is less susceptible to noise and high frequency variations, but unfortunately, Southard's ranking approach tends to distribute points roughly uniformly across the domain, wasting points and leading to inferior approximations, in many cases. After computing the Delaunay triangulation of the selected points, Southard performs a data-dependent retriangulation, swapping edges where that would reduce the sum of the absolute errors along the edges in the triangulation.

3.1.5 Refinement Methods

Refinement methods are multi-pass algorithms that begin with a minimal initial approximation, on each pass they insert one or more points as vertices in the triangulation, and repeat until the desired error is achieved or the desired number of vertices is used. For input data in a rectangular grid, the minimal approximation is two triangles; for other topologies, the initial approximation might be more complex. Incremental methods are typically used to maintain the triangulation as refinement proceeds.

To choose points, importance measures much like those of the feature methods can be used. Whereas feature methods typically use importance measures that are independent of the approximation, in refinement algorithms, the importance of a given point is usually a measure of the error between it and the approximation. For a height field, the most common metric for the error is simply the maximum absolute value of the vertical error, the L_∞ norm. This is the error measure most closely related to the Douglas-Peucker algorithm.

Greedy Insertion. We call refinement algorithms that insert the point(s) of highest error on each pass *greedy insertion* algorithms, "greedy" because they make irrevocable decisions as they go [15], and "insertion" because on each pass they insert one or more vertices into the triangulation. Methods that insert a single point in each pass we call *sequential greedy insertion* and methods that insert multiple points in parallel on each pass we call *parallel greedy insertion*. The words "sequential" and "parallel" here refer to the selection and re-evaluation process, not to the architecture of the machine. Many variations on the greedy insertion algorithm have been explored over the years; apparently the algorithm has been reinvented many times.

Fowler-Little 79. In 1979, Fowler and Little published a hybrid algorithm that uses an initial pass of feature selection using 2×2 filters to "seed" the triangulation, followed by multiple passes of parallel greedy insertion [37]. On each of these latter passes, for each triangle, the point with highest error, or *candidate point*, is found, and all candidate points whose error is above the requested threshold are inserted into the triangulation. (When the point of highest error falls on an edge, they expand their search for the candidate to a sector of the triangle's circumcircle, a quirk unique to their algorithm.)

Fowler and Little discussed two methods for finding candidates. In their exhaustive search method, the error at each input point is computed and tested against the highest error seen so far for that triangle. In the initial passes of a greedy insertion method, the triangles are big, necessitating the testing of many points, but in later passes the triangles shrink and less testing per triangle is required. As a way to speed the selection of candidates, they propose an alternative method using hill-climbing, in which a test point is initialized to the center of the triangle, and it repeatedly steps to the neighboring input point of highest error until it reaches a local maximum, where it becomes the candidate. This latter method can be much faster, especially for the initial passes, but it would also yield poorer quality approximations in many cases, because the hill climbing might fail to find the global maximum within the

triangle. Unfortunately, Fowler and Little did not show a comparison of the results of the two methods, and did not analyze the speed of their algorithm. An approach similar to Fowler and Little's was very briefly described by Lee and Schachter [72].

De Floriani-Falcidieno-Pienovi 83. In 1983, De Floriani *et al.* presented a sequential greedy insertion algorithm [24, 25]. Their method is purer than Fowler and Little's: it does not seed the triangulation using feature points, and it inserts a single point on each pass, not multiple points. Consequently, the quality of its approximations can be higher than Fowler and Little's. The point inserted in each pass is the point of highest absolute error from the input point set. To find this point they apparently visit all input points on each pass, computing errors. Their paper says that their algorithm has worst case cost of $O(n^2)$, but too few details of the algorithm or its data structures are provided to verify this. We will refer to this paper and algorithm as "DeFloriani83".

De Floriani 89. In later work, De Floriani published an algorithm to build a "Delaunay pyramid" [22], a hierarchy of Delaunay triangulations, using a variant of her 1983 greedy insertion algorithm to construct each level of the pyramid. Her 1989 paper describes the greedy insertion algorithm in greater detail than her earlier papers ([24, 25]).

Each triangle stores the set of input points it contains and the error of its candidate point. On each pass, the set of triangles is scanned to find the candidate of highest error, this point is inserted using incremental Delaunay triangulation, and the candidates of all the triangles in the modified region are recomputed. Recomputing the candidate of a triangle requires calculating the error at each point in the triangle's point set.

De Floriani states that the worst case time cost to create a complete pyramid of all n points is $O(n^2)$. We believe that the expected time cost of her algorithm, to select and triangulate m points, is $O(n \log m + m^2)$ (compare to Algorithm III in [40]).

Because point set traversal is used, rather than triangle scan conversion [36], this algorithm is not limited to input points in a regular grid, as are most height field approximation algorithms. The price of this generality is speed; the inner loops of a set traversal method cannot be optimized as much as those of a scan conversion approach.

Heller 90. Heller explored a hybrid technique that he called "adaptive triangular mesh filtering" [53, p. 168]. This technique is much like Fowler and Little's. The principal difference is that the features are chosen not with a fixed-size local filter but by checking a variable-sized neighborhood to determine if each point is a local extremum within some height threshold. This feature selection method, while more expensive than Fowler and Little's, probably yields higher quality approximations.

His insertion method is sequential, like that of DeFloriani83. He optimizes the algorithm by storing the set of candidates, one candidate from each triangle, in a heap[5]. Below is an excerpt of Heller's brief explanation of his algorithm [53, p. 168]:

> The [insertion] of a point requires a local retriangulation which consists of swapping all necessary triangles, and readjusting the [importances] of all affected points. It is clear that the time for retriangulation is proportional to the number of readjusted points and the logarithm of the number of queued points. It is, therefore, advisable to start the process with as many [feature] points as possible.

Due to his optimizations, Heller's algorithm is probably faster than most others of comparable quality, such as DeFloriani83, but unfortunately, beyond the statements quoted above he does not analyze the speed of his algorithm theoretically or empirically. It appears that the expected complexity of the greedy insertion portion of his algorithm is $O((m+n)\log m)$, like Algorithm III in [40].

Schmitt-Chen 91. In order to segment computer vision range data into planar regions, Schmitt and Chen use a two stage process called *split-and-merge* [110, 91]. The

[5] Christoph Witzgall has also employed a heap. Personal communication. 1994.

splitting stage is a form of greedy insertion with Delaunay triangulation similar to DeFloriani83. The merging stage joins together adjacent regions with similar normals, in the process destroying the triangulation, but yielding a segmentation of the image. Their splitting stage approximated a height field with $n = 256^2$ points using about $m = 3,060$ vertices in 67 seconds on a DEC VAX 8550.

Scarlatos-Pavlidis 92a and De Floriani-Puppo 92. The hierarchical triangulation methods of Scarlatos-Pavlidis [108] and De Floriani-Puppo [26] discussed earlier are analogous to greedy insertion in many ways, although their triangulations are quite different. Their techniques will typically use more triangles to achieve a given error than sequential greedy insertion with Delaunay triangulation, but on the other hand, they have the advantage of a hierarchy.

Rippa 92. Rippa generalized the greedy insertion algorithm of DeFloriani83 to explore data-dependent triangulation and least squares fitting [101].

In place of incremental Delaunay triangulation, Rippa's algorithm computes a data-dependent triangulation using a version of Lawson's local optimization procedure [71], repeatedly swapping edges around a new vertex until the global error reaches a local minimum. He tested two definitions of global error. The first is a purely geometric measure: the sum of the absolute values of the angles between normals of all pairs of adjacent triangles in the triangulation, and the second is a simple L_2 measure: the sum of squares of absolute vertical errors over all input points.

From experiments with Delaunay and data-dependent triangulation on several smooth, synthetic functions, Rippa concluded that data-dependent triangulation usually yields more accurate approximations using a given number of vertices than Delaunay triangulation. The angle criterion performed well in most cases, so he mildly recommended it over both the L_2 criterion and Delaunay triangulation. Rippa observed that the L_2 criterion occasionally allowed long, extremely thin sliver triangles that did not fit the surface well to enter and remain in the triangulation. The algorithm failed to eliminate such triangles because they were so thin that they contained no input points, and hence they contributed zero error to the L_2 measure.

The angle criterion also made poor choices in some cases, so Rippa tried a hybrid scheme that on each pass compares the errors resulting from Delaunay triangulation and data-dependent triangulation with the angle criterion, and updates using the one with the smaller global error. The hybrid scheme generated high quality approximations more consistently than the other methods that Rippa tested. Unfortunately, the hybrid is less elegant, and it appears slower than the other methods. Margaliot and Gotsman reported an error measure yielding a better fit than the angle criterion [81].

Rippa also explored least squares methods that approximate the input points instead of interpolating them. The (x, y) coordinates of the vertices are frozen, but their heights are allowed to vary, and the combination of heights that minimizes the global sum of squared errors is found. This involves solving a large, sparse, $m \times m$ system of linear equations. He found that high quality results could be achieved fairly efficiently, on low-noise data, if the least-squares fitting was done as a post-process to greedy insertion. His empirical tests on simple functions showed that least squares fitting roughly halved the error of the standard interpolative methods. Overall, Rippa's methods appear expensive (data-dependent triangulation, particularly so) but the resulting approximations are higher quality than those of simpler sequential greedy insertion methods. The least squares technique appears to be particularly effective at improving the approximation.

Rippa tested his algorithm on rather small height fields and did not discuss computational costs of data-dependent triangulation much.

Polis-McKeown 93. Polis and McKeown explored a somewhat parallel variation of the greedy insertion method [95]. Their basic algorithm, in each pass, computes the absolute error at each input point. The set of points of maximal absolute error is found, and these are inserted into the triangulation, one at a time, rejecting any that are within a tolerance distance of vertices already in the triangulation (see paper for details). This method might insert multiple points per triangle, unlike the greedy insertion algorithms previously discussed. It would typically insert fewer points per pass than Fowler

and Little's algorithm, however.

Several practical issues in the creation of large terrain models for simulators are raised by Polis and McKeown. To facilitate dynamic loading of the terrain as a viewer roams, many display programs require that terrain databases be broken into small square blocks or "load modules". This necessitates extra care along block boundaries to avoid cracks between polygons. Polis and McKeown also proposed *selective fidelity*, in which regions of the terrain could be assigned error weights according to their visual importance, their likelihood of being seen, or some other criterion. Thus, for example, for a tank simulator, one might weight navigable valleys more than inaccessible mountain slopes.

Polis and McKeown tried a data-dependent triangulation method involving summing the squares of errors along all edges of the triangulation [94], much like Southard's method. They found Delaunay triangulation to be preferable to data-dependent triangulation, however, because the former was much faster [95].

Polis and McKeown's algorithm appears to have an expected cost of $O(mn)$ (like Algorithm I in [40]). They reported a compute time of 18 hours to select $m = 76,500$ points total from an $n = 1,979^2$ terrain broken into 36 tiles on a DECstation 5000. Speed was not the major issue for them, however, since they were creating their TINs offline. They later optimized their algorithm to select $m = 50,000$ points from a terrain of $n = 8,966,001$ points in 89 minutes on a DEC Alpha [93].

Franklin 93. Franklin has released code for a sequential greedy insertion algorithm (PL/I code from 1973, C code from 1993) [38]. His algorithm is quite similar to DeFloriani83, but optimized in a manner similar to De Floriani's Delaunay pyramid method ([22]). With each triangle, Franklin stores a candidate pointer, and he updates only the candidates of new or modified triangles on each pass. He stores an array of input points with each triangle, as in [22], so the algorithm is more general but typically slower than a comparable surface simplification algorithm limited to height fields.

Between his two implementations, Franklin has experimented with several triangulation methods: swapping an edge if it reduces the maximum error of the approximation, swapping an edge if it has shorter length, and Delaunay triangulation.

Unfortunately, Franklin has not published his results and conclusions. By comparison to De Floriani's Delaunay pyramid algorithm and Algorithm III of [40], we conclude that the expected cost of Franklin's algorithm is $O(n \log m + m^2)$. Franklin's program can select $m = 100$ points from an $n = 257^2$ height field in 7 seconds on an SGI Indy.

Puppo-Davis-DeMenthon-Teng 94. Puppo *et al.* explored terrain approximation algorithms for the Connection Machine that are parallel both in the computer architecture sense and also in the greedy insertion sense [98]. Their algorithm is much like that of DeFloriani83, except they insert all candidate points with error above the requested threshold on each pass, like Fowler and Little. They found that the number of points inserted on each pass grew exponentially, so the number of passes required to insert m points would typically be $\Theta(\log m)$. On a Thinking Machines CM-2 with 16,384 processors, they reported compute times of 8 seconds to select $m = 379$ points from an $n = 128^2$ terrain [98], or 86 seconds to select $m = 2,933$ points from an $n = 512^2$ terrain [97].

The algorithm was parallelized by assigning each input point to a different logical processor. Most of the parallelization was straightforward, but parallel incremental triangulation required the use of special mutual exclusion techniques to handle simultaneous topology changes in neighboring triangles.

Puppo *et al.* implemented both sequential and parallel greedy insertion and concluded, surprisingly, that the latter is better. Our own experiments have indicated otherwise [40].

Chen-Schmitt 93. Chen and Schmitt explored a hybrid feature/refinement approach for triangulation of computer vision range data [11]. To best approximate the step and slope discontinuities that are common in range data, they first use edge detection to identify significant discontinuity features. These then become constraint curves during greedy insertion of additional vertices, using either con-

strained Delaunay or data-dependent triangulation. Chen and Schmitt found that data-dependent triangulation simplified better on surfaces with a preferred direction, such as cylinders.

Silva-Mitchell-Kaufman 95. A rather different approach to height field triangulation was proposed by Silva *et al.* [117]. We classify it here as a refinement method, although it is different in spirit from the previous methods. Their method uses *greedy cuts*, triangulating the domain from the perimeter inward, on each pass "biting" out of the perimeter the triangle of largest area that fits the input data within a specified maximum error tolerance. The method is thus a generalization of greedy visibility techniques for curve simplification [123, 63], and also a form of data-dependent triangulation. In a comparison with Franklin's greedy insertion algorithm, their unoptimized program was about two to four times slower, but produced triangulations of a given quality using fewer vertices. They reported running times of about 8 seconds to select $m = 1,641$ points from grids of $n = 120^2$ points on a one-processor SGI Onyx.

Garland-Heckbert 95. Our own work in height field simplification has explored fast and accurate variations of the greedy insertion algorithm [40, 39].

We explored two optimizations of the most basic greedy insertion algorithm (as in DeFloriani83). First, we exploited the locality of mesh changes, and only recalculated the errors at input points for which the approximation changed, and second, we used a heap to permit the point of highest error to be found more quickly. When approximating an n point grid using an m vertex triangulated mesh, these optimizations sped up the algorithm from an expected time cost of $O(mn)$ to $O((m+n)\log m)$. We were able to approximate an $n = 1024^2$ grid to high quality using 1% of its points in about 21 seconds on a 150 MHz SGI Indigo2.

We also explored a data-dependent greedy insertion technique similar to Rippa's method. We found an algorithm that yielded, in a fairly representative test, a solution with 88% the error of Delaunay greedy insertion at a cost of about 3–4 times greater. Source code for these algorithms is available.

In that paper, we propose several ideas for future work that could improve the performance of the greedy insertion algorithm in the presence of cliff discontinuities, high frequencies, and noise.

Arc/Info Latticetin. The geographic information system Arc/Info sold by the Environmental Systems Research Institute (ESRI) can approximate terrain grids. Its "Latticetin" command employs a hybrid feature/refinement approach that starts with a regular grid of equilateral triangles and refines it with parallel greedy insertion [70, 95].

3.1.6 Decimation Methods

In contrast to refinement methods, the decimation approach to surface simplification starts with the entire input model and iteratively simplifies it, deleting vertices, triangles, or other geometric features on each pass. The decimation approach is not so common for height field simplification; we will see far more decimation methods in the section on manifold simplification.

Lee 89. A "drop heuristic" method for simplifying terrains was proposed by Lee [73]. We call it a vertex decimation approach because on each pass it deletes a vertex. The algorithm takes the height field grid as input and creates an initial triangulation in which each 2×2 square between neighboring input points is split into two triangles [73]. On each pass, the error at each vertex is computed and the vertex with lowest error is deleted. The error at a vertex is found by temporarily deleting the vertex from the triangulation, doing a local Delaunay retriangulation, and measuring the vertical distance from the vertex to its containing triangle. The process continues until the error exceeds the desired level, or the desired number of vertices is reached. Deletion in a Delaunay triangulation can be done incrementally to avoid excessive cost [68].

The drop heuristic method yields high quality approximations, but its computational cost and memory cost appear very high. When Lee compared his algorithm to Chen and Guevara's method and to De Floriani's ternary triangulation method [23], he found, not surprisingly, that his method yielded superior results [74]. The drop heuristic

method is expensive because of the need to visit each vertex on every pass. Its memory cost is high because a triangulation with n vertices must be created[6].

Scarlatos-Pavlidis 92b. Scarlatos and Pavlidis explored a method for adjusting a triangulation in order to equalize the curvature of the input data within each triangle [109], extending McClure's and Pavlidis' earlier work [82, 91, 83]. Their algorithm takes an initial triangulation and applies three passes: shrinking triangles with high curvature, merging adjacent coplanar triangles, and swapping edges to improve triangle shape and fit. In tests, the method achieved little improvement when applied to the output of their hierarchical triangulation algorithm [108, 107]: in most cases, the method reduced the number of triangles, but it also increased the maximum error unless explicit error tests were added [109]. Curvature equalization was more successful at improving regular subsampling meshes [107, p. 89]. No unshaded pictures of the resulting meshes were given, however, so it is difficult to compare the quality of the results to other methods.

Scarlatos 93. In addition to the recursive subdivision method described earlier, Scarlatos also developed a vertex decimation method for constructing hierarchical triangulations [107]. The method begins with an initial triangulation and, to generate each level of the hierarchy, computes the "significance" of each vertex and deletes vertices in increasing order of significance until no more can be deleted. Significance is an (unspecified) function of the error between a vertex and a weighted average of its neighbors, and the degree of a vertex. The method is similar to that of Schroeder *et al.*, discussed later, except that Scarlatos' method is limited to height fields, and it takes more precautions to minimize error accumulation. Scarlatos reported a running time of 7.75 minutes to build a complete hierarchy for about $n = 5,900$ points on a VAX 8530.

Hughes-Lastra-Saxe 96. The simplification algorithm described by Hughes, Lastra, and Saxe [59] is targeted towards simplifying global illumination meshes resulting from radiosity systems. Consequently, the algorithm must simplify both the mesh geometry and the color values associated with each mesh vertex. They rejected a greedy insertion algorithm because of its inability to deal well with sharp discontinuities (i.e., shadow borders). Instead, they chose a combination of local vertex decimation and simplification envelopes as in [126, 14]. Interestingly, they chose to select vertices for removal at random rather than in order of increasing error. They claim that this provides more uniform meshes, which they believe to be advantageous. Their method also uses higher-order elements (quadratic, cubic, etc.) for reconstructing the surface, a possibility which most simplification methods ignore.

[6] We find that storing a triangulation with n vertices uses 5 to 100 times the memory of a height field of n points because of the extra adjacency information required.

3.1.7 Optimal Methods

The error of an optimal piecewise-linear, triangulated approximation to a smooth function of two variables has been analyzed in the limit as the number of triangles goes to infinity. Nadler showed that the L_2–optimal approximation has L_2 error proportional to m^{-1} [88].

Finding the optimal approximation of a grid or surface using triangulations of a subset of the input points could be done by enumerating all possible subsets and all possible triangulations, but this would take exponential time, and it would clearly be impractical. As with curves, certain problems in optimal surface approximation are well understood, while others are not. It is known that L_∞–optimal polygonal approximation of convex surfaces is NP-hard (requires exponential time, in practice) [19, 9]. This implies, of course, that L_∞–optimal approximation of height fields and more general surfaces (in the space of all triangulations) is also NP-hard, since they are a superset of convex surfaces. We do not know if there are polynomial time algorithms for optimal surface simplification using any other error metric (such as L_2), or within a more restricted class of triangulations. Even if some form of this problem permits an optimal algorithm with polynomial time, it would be surprising if it were as fast as the heuristic methods we have summarized above.

Polynomial time algorithms are known, however, for sub-optimal solutions with provable size and quality bounds. If the optimal L_∞ solution for a given error tolerance has m_o vertices, there is an $O(n^7)$ algorithm

to find an approximation with the same error using $m = O(m_o \log m_o)$ vertices [87, 1], but this is far too slow to be practical for large problems.

3.2 Manifold Surfaces

We now turn our attention from height fields and parametric surfaces to manifolds and manifolds with boundary. In general, the manifold can have arbitrary genus and be non-orientable[7] unless stated otherwise. Manifolds are more difficult to simplify than height fields or parametric surfaces because there is no natural 2-D parameterization of the surface. Delaunay triangulation is thus less easily applied. We group manifold simplification methods into two classes: refinement methods and decimation methods.

3.2.1 Refinement Methods

Faugeras-Hebert-Mussi-Boissonnat 84. Faugeras *et al.* developed a technique somewhat similar to De Floriani's 1984 algorithm, but it does not have persistent long edges, and it is applicable to the simplification of any 3-D triangulated mesh of genus 0, not just height fields [34]. The method begins with a pancake-like two-triangle approximation defined by three vertices of the input mesh. Associated with each triangle of the approximation is a set of input points. In successive passes, for each triangle of the approximation, the input point farthest from the triangle is found, and if the distance is above threshold, the triangle is split into 3–6 subtriangles by inserting new vertices at the interior point of highest error. Edges common to two subdivided triangles are split at their points of highest error (Figure 6). Splitting in this way eliminates the long edges of ternary triangulation.

During subdivision, each triangle's point set must be partitioned into 3–6 subsets. In methods that are limited to height fields, the partition of input points to subtriangles is done with simple projection and linear splitting. To partition point sets on a surface in 3-D, Faugeras *et al.* instead split using the shortest path along edges of the input mesh.

[7] A manifold is *orientable* if its two sides can be consistently labeled as "inside" and "outside". A Möbius strip is non-orientable.

The method simplified an $n = 2,000$ point model in 1 minute on a Perkin Elmer computer. The approximations generated were sometimes poor, however, and the method had particular problems with concavities [96]. A later subdivision data structure, the "prism tree", addressed these problems by recursively subdividing surface points into truncated pyramidal volumes [96].

Delingette 94. A related method for the simplification of orientable manifolds was developed by Delingette [28]. He fits surfaces to sets of 3-D points by minimizing an energy function which is a sum of an error term, an edge length term, and a curvature term. The algorithm starts with a mesh that is the dual to a subdivided icosahedron. It then iteratively adjusts the geometry, attempting to minimize the global energy [29]. After a good initial fit is achieved with this fixed topology, the mesh is refined. Regions of the mesh with high curvature, high local fit error, or elongated faces are subdivided and vertices migrate to points of high curvature [28]. Delingette reports that it takes 2 to 7 minutes to approximate a set of $n = 260,000$ points with a mesh of $m = 1,700$ vertices on a DEC Alpha. The method is much faster than the related method of Hoppe *et al.* [58], but it does not achieve comparable simplification, and it has a number of parameters that appear to require careful tuning.

Lounsbery-Eck-*et al.* 95. A two-stage method for multiresolution wavelet modeling of arbitrary triangulated polyhedra was developed by Lounsbery, Eck, *et al.* [76, 33]. The method is not limited to height fields or even to triangulated meshes with spherical topology; it can be applied to any triangulated manifold with boundary. The approach first constructs a base mesh which is a triangulated polyhedron with the same topology as the input surface. Geodesic-like distance measures are used in this step, reminiscent of the method of Faugeras *et al.*. It then uses repeated quaternary subdivision of the base mesh to construct a new mesh that approximates the input surface very closely. A multiresolution model of the new mesh is then built using wavelet techniques, after which an approximation at any desired error tolerance can be quickly generated. Eck *et al.* simplified a model with about $n = 35,000$ vertices to $m = 5,400$ vertices in 22 minutes of resampling

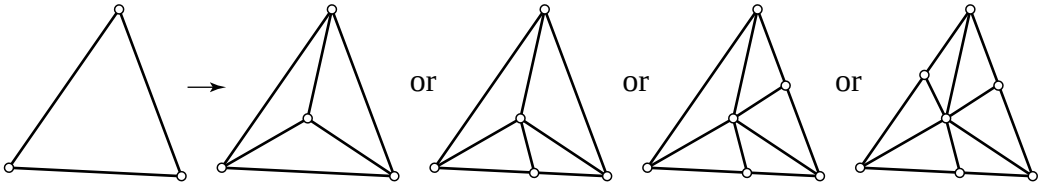

Figure 6: Subdivision pattern of Faugeras *et al.*.

plus 5 minutes of wavelet analysis/synthesis, on an SGI Onyx. The intermediate, approximating mesh had about twice as many vertices as the original.

While the approach is very attractive for interactive surface design and surface optimization, it may not be the best method for multiresolution modeling of static surfaces because of the cost of resampling. For the approximation of height fields, resampling is not needed, and simpler tensor product wavelet techniques could be used instead [79]. Another disadvantage is that the method does not resolve creases at arbitrary angles well, since the final mesh subdivides the triangles of the base mesh on a regular grid.

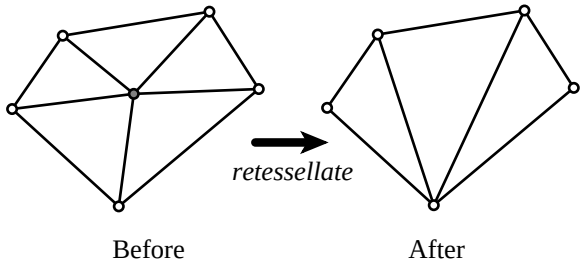

Figure 7: Vertex decimation. The target vertex and its adjacent triangles are removed. The resulting hole is then retessellated.

3.2.2 Decimation Methods

The next class of surface simplification algorithms we will consider is decimation methods: algorithms that start with a polygonization (typically a triangulation) and successively simplify it until the desired level of approximation is achieved. Most decimation algorithms fall into one of the following categories:

vertex decimation methods delete a vertex and retriangulate its neighborhood,

edge decimation methods delete one edge and two triangles, and merge two vertices,

triangle decimation methods delete one triangle and three edges, merge three vertices, and retriangulate the neighborhood, and

patch decimation methods delete several adjacent triangles and retriangulate their boundary.

Several variants of the decimation approach have been used for the problem of simplifying manifolds, particularly for thinning the output of isosurface polygonizers.

Kalvin 91. Kalvin *et al.* developed a two phase method to create surface models from medical data [65]. The first phase approximates a surface with tiny polygons using an algorithm similar to marching cubes [90], and the second phase then does patch decimation on the model by merging adjacent coplanar rectangles. Since it only merges precisely coplanar faces, the method does not allow control over the degree of simplification, so it is quite limited.

Schroeder-Zarge-Lorensen 92. Schroeder *et al.* developed a general vertex decimation algorithm primarily for use in scientific visualization [116]. Their method takes a triangulated surface as input, typically a manifold with boundary. The algorithm makes multiple passes over the data until the desired error is achieved. On each pass, all vertices that are not on a boundary or crease that have error below the threshold are deleted, and their surrounding polygons are retriangulated (see Figure 7). The error at

a vertex is the distance from the point to the approximating plane of the surrounding vertices. Note that errors are measured with respect to the previous approximation, not relative to the input points, so errors can accumulate (this flaw was fixed in later versions of the algorithm). Their paper demonstrated simplifications of models containing as many as 1,700,000 triangles. The computation time to simplify a model of $n = 400,000$ vertices to $m = 40,000$ vertices is about 14 minutes on an R4000 processor [115]. This method uses significant memory, like Lee's. To conserve memory, compact data structures were developed [115]. Source code for this algorithm is available [114].

Relative to Lee's method, the technique of Schroeder *et al.* is more general since it is not limited to height fields, it uses a less expensive and less accurate error measure, and it deletes multiple vertices per pass. Consequently, it is faster, but probably has lower quality.

Soucy and Laurendeau 92. To simplify manifolds with boundary, Soucy and Laurendeau also developed a vertex decimation algorithm [118, 119]. Their application was the construction of surface models from multiple range views. On each pass, the vertex with least error is deleted, and its neighborhood (the set of adjacent triangles) is retriangulated. The process stops when the error rises above a specified tolerance or the desired size of model is achieved.

To compute rigorous error bounds, a set of deleted vertices is stored with each triangle. We will call these points the *ancestors* of the triangle. To compute the error at a vertex, a temporary vertex deletion and retriangulation are done. The error of a vertex is a measure of the error that would result from its removal. More precisely, it is defined to be the maximum distance between either an ancestor from the neighborhood or the vertex itself to the retriangulated surface. Deletion of a non-boundary vertex is considered legal if the neighborhood triangles can be projected to 2-D without foldover.

To retriangulate, Soucy and Laurendeau first compute a constrained Delaunay triangulation in a 2-D projection, then this triangulation is improved using a version of Lawson's local optimization procedure [71] adapted to surfaces in 3-D. To update the data structures after retriangulation, first the ancestor lists are redistributed among the new triangles, then the error of each formerly neighboring vertex is updated.

We can relate the method to several of its precursors. Like Lee's method, this algorithm does vertex decimation by "one move lookahead", but unlike his technique, it is not limited to height fields. Like Faugeras *et al.* and De Floriani *et al.* (1989), it stores a point set with each triangle, but unlike those methods, it is a decimation algorithm, and it is more general: it can simplify any manifold with boundary.

Soucy and Laurendeau estimate the expected complexity of their algorithm to be $O\Big(n \log (n/(n-m))\Big)$. Their method appears to yield higher quality results than the method of Schroeder *et al.*, but it is slower and it uses more memory, since it maintains lists of all deleted points. A revised version of this algorithm is used in the *IMCompress* software sold by InnovMetric [64].

Turk 92. Another method for simplifying a manifold with boundary is due to Turk [124]. This algorithm is not a decimation method in the same sense as the previous methods, but we list it here because it also starts with a full triangulation and simplifies.

Turk's algorithm takes a triangulated surface as input, sprinkles a user-specified number of points on these triangles at random, and uses an iterative repulsion procedure to spread the points out nearly uniformly. The points remain on the surface as they move about. After these points are inserted into the original surface triangulation, the original vertices are deleted one by one, yielding a triangulation of the new vertices that has the same topology as the original surface. Turk also demonstrated an improved variant of this technique that groups points most densely where the surface is highly curved.

Turk's method appears to be best for smooth surfaces, since it tends to blur sharp features[8]. Overall, it appears that Turk's algorithm is quite complex and that it will yield results inferior in quality to the methods of Schroeder *et al.* or Soucy-Laurendeau.

[8] William Schroeder, SIGGRAPH '94 tutorial talk.

Hinker-Hansen 93. Hinker and Hansen developed a patch decimation algorithm for use in scientific visualization [55]. It is a one pass method that first finds patches of triangles with nearly parallel normal vectors, and then retriangulates each patch. The method has $O(nlogn)$ time cost in practice. A model with about $n = 510,000$ vertices was simplified to $m = 321,000$ vertices in 9 minutes on a CM-5. The method is "largely ineffective when faced with surfaces of high curvature", however [55]. It appears to work best on piecewise-ruled surfaces: those with zero curvature in at least one direction, such as cylinders, cones, and planes. Therefore the method is not as general as that of Schroeder *et al.* or Soucy-Laurendeau.

Hoppe-DeRose-Duchamp-McDonald-Stuetzle 93. Hoppe *et al.* developed an optimization-based algorithm for general 3-D surface simplification [58]. Their method takes a set of points and an initial, fine triangulated surface approximation to those points as input, and outputs a coarser triangulation of the points with the same topology as the input mesh. The method attempts to minimize a global energy measure consisting of three terms: a complexity term that penalizes meshes with many vertices, an error term that penalizes geometric distance of the surface from the input points, and a spring term that penalizes long edges in the triangulation. The method proceeds in three nested loops, the outermost one decreasing the spring constant, the middle one doing an optimization over mesh topologies, and the inner one doing an optimization over geometries. The topological optimization uses heuristics and random selection to pick an edge and either collapse it, split it, or swap it. The geometric optimization uses nonlinear optimization techniques to find the vertex positions that minimize the global error for a given topology. Topological changes are kept if they reduce the global error, otherwise they are discarded. In other words, the method makes repeated semi-random changes to the mesh, keeping those that allow better fit and/or a simpler mesh.

Unlike most general surface simplification methods, the method of Hoppe *et al.* does not constrain output vertices to be a subset of the input points. Their method appears to be less sensitive to noise in the input points than most other methods because of its freedom in choosing vertices and because the geometric error measure uses an L_2 norm, and not an L_∞ norm.

Their method is slow, but it is capable of very good simplifications. They simplified a mesh with $m_1 = 4,059$ vertices to $m_2 = 262$ vertices while fitting to $n = 16,864$ points in 46 minutes on a 1-processor DEC Alpha. They have released their code. Their algorithm yields higher quality approximations than that of Eck *et al.*, but it is slower [33].

Hamann 94. A triangle decimation method was explored by Hamann [49]. In this algorithm, triangles are deleted in increasing order of weight, where weight is the product of "equi-angularity" and curvature, roughly speaking. Thus, slivers and low curvature triangles are deleted first. The method appears rather complex, however, since second degree surface fitting is used to position the new vertices, and a number of geometric checks are required to prevent topological changes.

Kalvin 94. In later work, Kalvin and Taylor developed a patch decimation method called "superfaces" to simplify manifolds within a given error tolerance [66, 67]. The algorithm operates in a single pass. This pass consists of three phases. The first phase segments the surface into approximately planar patches. Each patch is found by picking a face at random and merging in adjacent faces until the patch's faces can no longer be fit by a plane within the error tolerance. Additional tests prevent degenerate or highly elongated patches from being created. The second phase simplifies the curves common to adjacent patches using the Douglas-Peucker algorithm. The third phase retriangulates the patches by subdividing them into star polygons and then triangulating each star polygon.

When a face is merged into a patch, the set of feasible approximating planes $ax + by + cz + 1 = 0$ of the patch must be updated. This set could be represented using linear programming, as a convex polytope in the 3-D (a, b, c) parameter space of planes, but the complexity of this data structure could grow quite large. Instead, Kalvin and Taylor use an ellipsoidal approximation that supports constant time updates and queries.

At a high level, this method is quite similar to Hinker-Hansen, in that it employs a single pass to find nearly coplanar sets and then retriangulates them. Hinker-

Hansen define patches based on angles between normal vectors, however, while Kalvin-Taylor define them based on distance-to-plane. Distance to plane is probably a better method for defining patches, since it is less sensitive to noise. Guéziec reports that Kalvin and Taylor's algorithm can simplify a model with about $n = 90,000$ vertices to $m = 5,000$ vertices in 3 to 5 minutes on an IBM RS6000.

Varshney 94. Using visibility techniques from computational geometry, Varshney developed a patch decimation algorithm for simplifying orientable triangulated manifolds with boundary [127, 126]. The method has bounded error. Instead of simplifying in a fast, greedy manner, as most other decimation methods do, it is much more brute force, exhaustively testing to find the largest triangle to insert on each pass.

First, the input surface is offset inwards and outwards by a tolerance distance ϵ to create two offset surfaces. All triangles defined by three vertices of the input surface are checked for validity by testing that they do not intersect either offset surface and that they do not overlap previously inserted triangles. On each pass of the algorithm, the valid triangle that "covers" the greatest number of previously uncovered input vertices is inserted, the old triangulation of this portion of the surface is deleted, and small triangles are added to fill the cracks between the old and the new. The algorithm generates good approximations when it works, but problems arise when the offset surfaces collide. So far, the method has not been demonstrated for simplifications below 30% of the input size, and it is very slow. The time costs of this algorithm and its variants range from $O(n^2)$ to $O(n^6)$[9]

Guéziec 95. Guéziec developed a method for simplifying orientable manifolds that employs edge decimation [46]. He defines the *edge collapse*, or edge contraction, operator to delete an edge and merge its two endpoints into a single vertex (Figure 8). Guéziec's algorithm orders edges by "importance" (in some unspecified way), and makes a single pass through the edges in increasing order of importance, doing edge collapses where legal.

[9]Personal communication, Pankaj K. Agarwal and Amitabh Varshney, 1995.

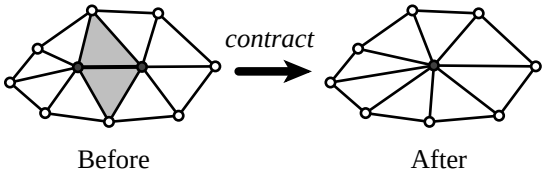

Figure 8: A simple edge contraction. The highlighted edge is contracted into a single point. The shaded triangles become degenerate and are removed during the contraction.

Testing legality entails most of the work required to do an edge collapse. The provisional new vertex is positioned to fit the old faces well and to preserve volume. During simplification, an error radius is associated with each vertex. By interpolating spheres with these radii across the surface, a error volume is defined. At any step during simplification, the error volume encloses the original surface. When an edge collapse is being considered, the error radius for the provisional new vertex is set so that the new error volume encloses the old error volume.

The collapse is considered legal if it meets four conditions: (1) the topology of the surface is preserved, (2) the normals of the modified faces change little, (3) the new triangles are well shaped (not slivers), and (4) the error radius for the new vertex is below an error threshold.

Use of the error volume could give the user local control of error tolerance at each vertex. No examples of this are shown in the paper, however.

Guéziec reports a time of 10 minutes to simplify a model with about $n = 90,000$ vertices to $m = 5,000$ vertices on an IBM RS6000 model 350. He says that Kalvin and Taylor's algorithm yields more compact approximations for small error tolerances, but that his algorithm performs better for large error tolerances, and that his triangles are better shaped. Closely related algorithms are illustrated with better pictures in another paper [47]. In that work, a model with about $n = 181,000$ vertices was simplified to $m = 26,000$ vertices in 53 minutes, on the same type of machine. The quality of the resulting meshes appears good.

Gourdon 95. Gourdon explored a method for simplifying orientable surface meshes resulting from surface reconstruction [43]. His algorithm differs from almost all other simplification algorithms in that it does not assume the surface mesh to be a triangulation. The algorithm is designed to preserve the Euler characteristic[10] of the model; this implies that the topology is preserved. Topological preservation is important for simplifying medical data, which is the focus of this technique. The algorithm iteratively removes edges based on an unspecified curvature criterion. Because the algorithm supports non-triangular facets, no retessellation is required after removing edges. Gourdon observes that simply removing a sequence of edges can lead to undesirable, irregular meshes. To control the regularity of the tessellation, he restricts the degree of vertices to be at most 6 and facet may have at most 12 edges. Following simplification, a "regularization" step is performed. Regularization attempts to improve the mesh by moving points to minimize an energy function, in this case the sum of squared edge lengths. A simple regularization step would move a vertex towards the barycenter of its neighbors. However, this can produce significant shrinkage of the surface. To avoid this, Gourdon uses a regularization step that moves the vertex towards the barycenter, but constrains the vertex to move parallel to the average plane of its neighbors.

Klein-Liebich-Strasser 96. The algorithm described by Klein, Liebich, and Strasser [69] is very similar to the method of Soucy and Laurendeau [119]. It simplifies an oriented manifold by iteratively removing a vertex a retriangulating the resulting hole using a constrained Delaunay triangulation. Each deleted vertex is linked to the closest face in the approximation. These links are used to compute the distance between the original and approximate surfaces. To select a vertex for removal, each vertex is tentatively removed and the additional error introduced by the removal is computed. The vertex which introduces the least error is selected for removal. After the vertex is removed, the links and projected additional errors within its neighborhood must be recomputed.

[10]The *Euler characteristic* of a model is defined as $\chi = F - E + V$ where F, E, and V are, respectively, the number of faces, edges, and vertices.

Algorri-Schmitt 96. Algorri and Schmitt developed an algorithm for simplifying closed, dense triangulations resulting from surface reconstruction [2]. Their algorithm begins with a pre-processing phase which smooths the initial mesh by swapping edges based on a G^1-continuity criterion as in [32]. After this initial smoothing, every edge whose dihedral angle exceeds some user-specified planarity threshold is classified as a *feature edge*. Each vertex is subsequently labeled according to its number of incident feature edges. An independent set of edges connecting "0" vertices is collected, and all the edges are collapsed simultaneously. This simplification phase is followed by a smoothing phase were all non-feature edges are considered for swapping based on the G^1-continuity criterion. If further simplification is desired, edges are reclassified and the process outlined above is repeated. Since only edges in mostly planar regions are selected for decimation, the basic step will not simplify "characteristic curves" (e.g., the edges of a cube) and there will always be a single vertex left in the midst of planar regions. Algorri and Schmitt describe additional iterative steps which simplify these cases separately from the basic step outlined above.

Ronfard-Rossignac 96. Another algorithm based on edge collapse was described by Ronfard and Rossignac [103]. The fundamental observation underlying their algorithm is that each vertex in the original model lies at the intersection of a set of planes, in particular, the planes of the faces that adjoin the vertex. They associate a set of planes with each vertex; they call this set the *zone* of the vertex. A vertex's zone is initialized to be the set of planes of the adjoining faces. The error at a vertex is measured by the maximum distance between the vertex and the planes in its zone. When contracting an edge, the zone of the resulting vertex is the union of the zones of the original endpoints. The error of this resulting vertex characterizes the cost of contracting the edge. At each iteration, the edge of lowest cost is selected and contracted. The complexity of this algorithm would seem to be $O(n \log n)$.

Hoppe 96. The simplification algorithm presented by Hoppe [56] for the construction of progressive meshes is a simplified version of the algorithm of Hoppe *et al.* [58]. Rather than performing a more general search, it simply

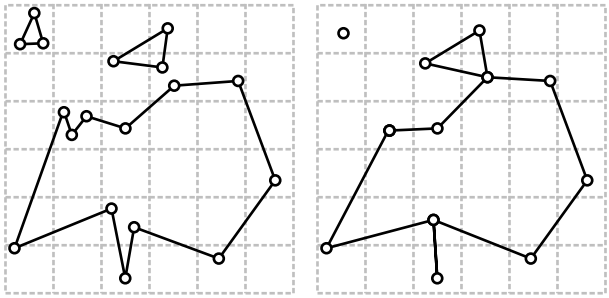

Figure 9: Uniform vertex clustering. Note the triangle which as collapsed to a single point, and the now dangling edge at the bottom. Also note how separate components have been joined together.

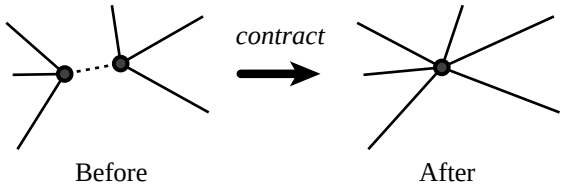

Figure 10: Pair contraction joining unconnected vertices. The dashed line indicates the two vertices being contracted together.

selects a sequence of edge contractions. The algorithm uses essentially the same error formulation of the earlier method, although it is augmented to handle surface attributes such as colors. Hoppe suggests that the resulting meshes are just as good, and perhaps even better, than the results of the more general mesh optimization algorithm.

3.3 Non-Manifold Surfaces

The most general class of surfaces is the non-manifold surface, which permits three or more triangles to share an edge, and permits arbitrary polygon intersections. Relatively few surface simplification algorithms can handle models of this generality.

Rossignac-Borrel 93. A very general technique for simplifying general 3-D triangulated models was described by Rossignac and Borrel [104]. They subdivide the object's bounding volume into a regular grid of boxes of user-specified size. All vertices are graded (or weighted) according to some scheme, and all vertices within each box are merged together into a new representative vertex. A simplified model is then synthesized from these representative vertices by forming triangles according to the original topology (see Figure 9). This method is extremely general, as it can operate on any set of triangles (not just manifolds), it can achieve arbitrary simplification levels, and it can even eliminate small objects or otherwise change the topology of a surface. Unfortunately, it does not preserve detail well [52]. When applied to height fields, it is roughly equivalent to blurring followed by regular subsampling. This software is being sold as part of IBM's "3D Interaction Accelerator" [61]. This method has been extended using octrees instead of regular grids [78].

Low-Tan 97. Low and Tan [77] developed a clustering algorithm that is intended to provide higher quality than the uniform clustering described by Rossignac and Borrel while maintaining its generality. Their first improvement was to suggest a better weighting criterion. More importantly, they replaced the uniform grid with a set of cluster cells. These cells can be any simple shape, such as cubes or spheres. Cells are centered around their vertex of highest weight. When a vertex falls within the intersection of multiple cells, it is placed in the cell whose center is closest. In addition to these algorithmic improvements, they improved the appearance of simplified models by rendering stray edges as thick lines whose area approximates the area of the original model in that region.

Garland-Heckbert 97. We have developed an algorithm for simplifying surfaces based on iterative vertex-pair contractions [41]. A pair contraction is a natural generalization of edge contraction (Figure 8) where the vertex pair need not be connected by an edge (see Figure 10). A 4×4 symmetric matrix $\mathbf{Q}_i$ is associated with each vertex $\mathbf{v}_i$. The error at the vertex is defined to be $\mathbf{v}^T\mathbf{Q}\mathbf{v}$, and when a pair is contracted, their matrices are added together to

form the matrix for the resulting vertex. We derive these matrices to calculate the sum of squared distances of the vertex to a set of planes (this is similar to the error metric of Ronfard and Rossignac [103]).

Our technique for tracking vertex error is quite efficient, and the algorithm is correspondingly fast. The quality of the approximations is similar to those of Ronfard and Rossignac, although the algorithm is more general in that it can join model components.

3.4 Related Techniques

We have focused on approximation of surfaces by polygons, but there has been related work in fitting curved surfaces to a set of points on a surface, and approximation of volumetric data. We include a partial survey.

Fitting a Curved Surface Model. Polygon models for curved surfaces can be bulky. More compact representations for surfaces are often possible using curved surface primitives such as piecewise-polynomial surfaces. The next class of models beyond piecewise-linear surfaces are surfaces with tangent continuity. Schmitt and others have developed adaptive refinement methods for fitting rectangular Bézier patches [113] and triangular Gregory patches [111] to a grid of points in 3-D. The latter method is superior to the former because it is better able to adapt to features at an angle to the grid. Another curved surface primitive, the subdivision surface, has been fit to points in 3-D by Hoppe *et al.*, with very nice results [57]. Piecewise quadratic surfaces have been fit to range data using least squares techniques [35].

Fitting to a Volume. A generalization of the feature approach to the approximation of volumes (scalar functions of three variables) was explored by Hamann and Chen [50]. They ranked points according to an estimate of the curvature of the function $f(x, y, z)$ at each point, and incrementally inserted vertices into a data-dependent tetrahedrization, in decreasing order of curvature, until a given error tolerance was met. The errors for data-dependent tetrahedrization were measured using L_2 or L_∞ norms on all points inside each tetrahedron. The surface decimation approach has also been generalized to tetrahedrizations [100].

4 Conclusions

Surface simplification is not as well understood as curve simplification. Whereas there appears to be fairly widespread agreement that one algorithm, Douglas-Peucker, does a high quality job of curve simplification at acceptable speeds, there is little agreement about the best approach for surface simplification. No thorough empirical comparison of surface simplification methods has been done analogous to the studies for curves ([85, 130]). Furthermore, surface simplification seems inherently much more difficult than curve simplification.

Why are surfaces so much harder? The biggest qualitative difference we observe is that curves inherently lend themselves to divide and conquer strategies like Douglas-Peucker, since splitting a curve at the point of highest error yields two curves, breaking the task into two smaller subtasks of the same type. Splitting a surface at the point of highest error is an ambiguous concept. Certain methods arbitrarily choose some way of splitting at a point, as with the hierarchical subdivision methods that split a triangle into three or more subtriangles; and other methods abandon the divide and conquer strategy and employ the more complex general triangulations.

Our purpose has been primarily to survey the existing methods, not to evaluate them, so we offer few conclusions here. Instead we hope that this survey, by collecting references and descriptions to the large body of work on this topic, will draw attention to similar lines of research in disparate fields, and facilitate future cross-fertilization.

5 Acknowledgements

We thank Frank Bossen, Marshall Bern, Jon Webb, and Anoop Bhattacharjya for their comments on drafts of this survey. The CMU Engineering & Science library has been very helpful in locating obscure papers.

6 References

[1] Pankaj K. Agarwal and Subhash Suri. Surface approximation and geometric partitions. In *Proc. 5th ACM-SIAM Sympos. Discrete Algorithms*, pages 24–33, 1994. (Also available as Duke U. CS tech report, ftp://ftp.cs.duke.edu/dist/techreport/1994/1994-21.ps.Z).

[2] María-Elena Algorri and Francis Schmitt. Mesh simplification. *Computer Graphics Forum*, 15(3), Aug. 1996. Proc. Eurographics '96.

[3] Dana H. Ballard. Strip trees: A hierarchical representation for curves. *Communications of the ACM*, 24(5):310–321, 1981.

[4] Dana H. Ballard and Christopher M. Brown. *Computer Vision.* Prentice Hall, Englewood Cliffs, NJ, 1982.

[5] Bruce G. Baumgart. *Geometric Modeling for Computer Vision.* PhD thesis, CS Dept, Stanford U., Oct. 1974. AIM-249, STAN-CS-74-463.

[6] Marshall Bern and David Eppstein. Mesh generation and optimal triangulation. Technical report, Xerox PARC, March 1992. CSL-92-1. Also appeared in "Computing in Euclidean Geometry", F. K. Hwang and D.-Z. Du, eds., World Scientific, 1992.

[7] Jules Bloomenthal. Polygonization of implicit surfaces. *Computer Aided Geometric Design*, 5:341–355, 1988.

[8] Laurence Boxer, Chun-Shi Chang, Russ Miller, and Andrew Rau-Chaplin. Polygonal approximation by boundary reduction. *Pattern Recognition Letters*, 14(2):111–119, February 1993.

[9] H. Brönnimann and M. T. Goodrich. Almost optimal set covers in finite VC-dimension. In *Proc. 10th Annual ACM Symp. on Computational Geometry*, pages 293–302, 1994.

[10] Edwin E. Catmull. *A Subdivision Algorithm for Computer Display of Curved Surfaces.* PhD thesis, Dept. of CS, U. of Utah, Dec. 1974.

[11] Xin Chen and Francis Schmitt. Adaptive range data approximation by constrained surface triangulation. In B. Falcidieno and T. Kunii, editors, *Modeling in Computer Graphics: Methods and Applications*, pages 95–113. Springer-Verlag, Berlin, 1993.

[12] Zi-Tan Chen and J. Armando Guevara. Systematic selection of very important points (VIP) from digital terrain model for constructing triangular irregular networks. In N. Chrisman, editor, *Proc. of Auto-Carto 8 (Eighth Intl. Symp. on Computer-Assisted Cartography)*, pages 50–56, Baltimore, MD, 1987. American Congress of Surveying and Mapping.

[13] James H. Clark. Hierarchical geometric models for visible surface algorithms. *CACM*, 19(10):547–554, Oct. 1976.

[14] Jonathan Cohen, Amitabh Varshney, Dinesh Manocha, Greg Turk, Hans Weber, Pankaj Agarwal, Frederick Brooks, and William Wright. Simplification envelopes. In *SIGGRAPH '96 Proc.*, pages 119–128, Aug. 1996. http://www.cs.unc.edu/∼geom/envelope.html.

[15] Thomas H. Cormen, Charles E. Leiserson, and Ronald L. Rivest. *Introduction to Algorithms.* MIT Press, Cambridge, MA, 1990.

[16] Michael A. Cosman, Allan E. Mathisen, and John A. Robinson. A new visual system to support advanced requirements. In *Proceedings of the 1990 Image V Conference*, pages 371–380. Image Society, Tempe, AZ, June 1990.

[17] Michael A. Cosman and Robert A. Schumacker. System strategies to optimize CIG image content. In *Proceedings of the Image II Conference*, pages 463–480. Image Society, Tempe, AZ, June 1981.

[18] Robert G. Cromley. Hierarchical methods of line simplification. *Cartography and Geographic Information Systems*, 18(2):125–131, 1991.

[19] Gautam Das and Michael T. Goodrich. On the complexity of approximating and illuminating three-dimensional convex polyhedra. In *Proc. 4th Workshop Algorithms Data Struct.*, Lecture Notes in Computer Science. Springer-Verlag, 1995. To appear.

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

[21] Carl de Boor. *A Practical Guide to Splines.* Springer, Berlin, 1978.

[22] Leila De Floriani. A pyramidal data structure for triangle-based surface description. *IEEE Computer Graphics and Appl.*, 9(2):67–78, March 1989.

[23] Leila De Floriani, Bianca Falcidieno, George Nagy, and Caterina Pienovi. A hierarchical structure for surface approximation. *Computers and Graphics*, 8(2):183–193, 1984.

[24] Leila De Floriani, Bianca Falcidieno, and Caterina Pienovi. A Delaunay-based method for surface approximation. In *Eurographics '83*, pages 333–350. Elsevier Science, 1983.

[25] Leila De Floriani, Bianca Falcidieno, and Caterina Pienovi. Delaunay-based representation of surfaces defined over arbitrarily shaped domains. *Computer Vision, Graphics, and Image Processing*, 32:127–140, 1985.

[26] Leila De Floriani and Enrico Puppo. A hierarchical triangle-based model for terrain description. In A. U. Frank et al., editors, *Theories and Methods of Spatio-Temporal Reasoning in Geographic Space*, pages 236–251, Berlin, 1992. Springer-Verlag.

[27] Michael DeHaemer, Jr. and Michael J. Zyda. Simplification of objects rendered by polygonal approximations. *Computers and Graphics*, 15(2):175–184, 1991.

[28] Hervé Delingette. Simplex meshes: a general representation for 3D shape reconstruction. Technical report, INRIA, Sophia Antipolis, France, Mar. 1994. No. 2214, http://zenon.inria.fr:8003/epidaure/personnel/delingette/delingette.html.

[29] Hervé Delingette, Martial Hebert, and Katsushi Ikeuchi. Shape representation and image segmentation using deformable surfaces. *Image and Vision Computing*, 10(3):132–144, Apr. 1992.

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

[31] Richard O. Duda and Peter E. Hart. *Pattern Classification and Scene Analysis.* Wiley, New York, 1973.

[32] Nira Dyn, David Levin, and Shmuel Rippa. Data dependent triangulations for piecewise linear interpolation. *IMA J. Numer. Anal.*, 10(1):137–154, Jan. 1990.

[33] Matthias Eck, Tony DeRose, Tom Duchamp, Hugues Hoppe, Michael Lounsbery, and Werner Stuetzle. Multiresolution analysis of arbitrary meshes. In *SIGGRAPH '95 Proc.*, pages 173–182. ACM, Aug. 1995. http://www.cs.washington.edu/research/projects/grail2/www/pub/pub-author.html.

[34] Olivier Faugeras, Martial Hebert, P. Mussi, and Jean-Daniel Boissonnat. Polyhedral approximation of 3-D objects without holes. *Computer Vision, Graphics, and Image Processing*, 25:169–183, 1984.

[35] Olivier D. Faugeras, Martial Hebert, and E. Pauchon. Segmentation of range data into planar and quadratic patches. In *Proc. IEEE Intl. Conf. on Computer Vision and Pattern Recognition*, pages 8–13, June 1983.

[36] James D. Foley, Andries van Dam, Steven K. Feiner, and John F. Hughes. *Computer Graphics: Principles and Practice, 2nd ed.* Addison-Wesley, Reading MA, 1990.

[37] Robert J. Fowler and James J. Little. Automatic extraction of irregular network digital terrain models. *Computer Graphics (SIGGRAPH '79 Proc.)*, 13(2):199–207, Aug. 1979.

[38] W. Randolph Franklin. tin.c, 1993. C code, ftp://ftp.cs.rpi.edu/pub/franklin/tin.tar.gz.

[39] Michael Garland and Paul S. Heckbert. Fast triangular approximation of terrains and height fields. Submitted for publication.

[40] Michael Garland and Paul S. Heckbert. Fast polygonal approximation of terrains and height fields. Technical report, CS Dept., Carnegie Mellon U., Sept. 1995. CMU-CS-95-181, http://www.cs.cmu.edu/∼garland/scape.

[41] Michael Garland and Paul S. Heckbert. Surface simplification using quadric error metrics. In *SIGGRAPH '97 Proc.*, August 1997. To appear. http://www.cs.cmu.edu/∼garland/.

[42] Dora Gómez and Adolfo Guzmán. Digital model for three-dimensional surface representation. *Geo-Processing*, 1:53–70, 1979.

[43] Alexis Gourdon. Simplification of irregular surface meshes in 3D medical images. In *Computer Vision, Virtual Reality, and Robotics in Medicine (CVRMed '95)*, pages 413–419, Apr. 1995.

[44] Markus H. Gross, R. Gatti, and O. Staadt. Fast multiresolution surface meshing. In *Proc. IEEE Visualization '95*, July 1995. (Also ETH Zürich CS tech report 230, http://www.inf.ethz.ch/publications/tr.html).

[45] Eric Grosse. Bibliography of approximation algorithms. ftp://netlib.att.com/netlib/master/readme.html, link="catalog".

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

[47] André Guéziec and Robert Hummel. Exploiting triangulated surface extraction using tetrahedral decomposition. *IEEE Trans. on Visualization and Computer Graphics*, 1(4):328–342, 1995.

[48] Leonidas Guibas and Jorge Stolfi. Primitives for the manipulation of general subdivisions and the computation of Voronoi diagrams. *ACM Transactions on Graphics*, 4(2):75–123, 1985.

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

[50] Bernd Hamann and Jiann-Liang Chen. Data point selection for piecewise trilinear approximation. *Computer-Aided Geometric Design*, 11:477–489, 1994.

[51] Richard W. Hamming. *Digital Filters*. Prentice-Hall, Englewood Cliffs, NJ, 1983.

[52] Paul S. Heckbert and Michael Garland. Multiresolution modeling for fast rendering. In *Proc. Graphics Interface '94*, pages 43–50, Banff, Canada, May 1994. Canadian Inf. Proc. Soc. http://www.cs.cmu.edu/∼ph.

[53] Martin Heller. Triangulation algorithms for adaptive terrain modeling. In *Proc. 4th Intl. Symp. on Spatial Data Handling*, volume 1, pages 163–174, Zürich, 1990.

[54] John Hershberger and Jack Snoeyink. Speeding up the Douglas-Peucker line-simplification algorithm. In P. Bresnahan et al., editors, *Proc. 5th Intl. Symp. on Spatial Data Handling*, volume 1, pages 134–143, Charleston, SC, Aug. 1992. Also available as TR-92-07, CS Dept, U. of British Columbia, http://www.cs.ubc.ca/tr/1992/TR-92-07, code at http://www.cs.ubc.ca/spider/snoeyink/papers/papers.html.

[55] Paul Hinker and Charles Hansen. Geometric optimization. In *Proc. Visualization '93*, pages 189–195, San Jose, CA, October 1993. http://www.acl.lanl.gov/Viz/vis93_abstract.html.

[56] Hugues Hoppe. Progressive meshes. In *SIGGRAPH '96 Proc.*, pages 99–108, Aug. 1996. http://www.research.microsoft.com/research/graphics/hoppe/.

[57] Hugues Hoppe, Tony DeRose, Tom Duchamp, Mark Halstead, Hubert Jin, John McDonald, Jean Schweitzer, and Werner Stuetzle. Piecewise smooth surface reconstruction. In *SIGGRAPH '94 Proc.*, pages 295–302, July 1994. http://www.research.microsoft.com/research/graphics/hoppe/.

[58] Hugues Hoppe, Tony DeRose, Tom Duchamp, John McDonald, and Werner Stuetzle. Mesh optimization. In *SIGGRAPH '93 Proc.*, pages 19–26, Aug. 1993. http://www.research.microsoft.com/research/graphics/hoppe/.

[59] Merlin Hughes, Anselmo A. Lastra, and Edward Saxe. Simplification of global-illumination meshes. *Computer Graphics Forum*, 15(3):339–345, August 1996. Proc. Eurographics '96.

[60] Peter Hughes. Building a terrain renderer. *Computers in Physics*, pages 434–437, July/August 1991.

[61] IBM. IBM 3D Interaction Accelerator, 1995. Commercial software, http://www.research.ibm.com/3dix.

[62] Insung Ihm and Bruce Naylor. Piecewise linear approximations of digitized space curves with applications. In N. M. Patrikalakis, editor, *Scientific Visualization of Physical Phenomena*, pages 545–569, Tokyo, 1991. Springer-Verlag.

[63] Hiroshi Imai and Masao Iri. Polygonal approximations of a curve – formulations and algorithms. In G. T. Toussaint, editor, *Computational Morphology*, pages 71–86. Elsevier Science, 1988.

[64] InnovMetric. Commercial software, http://www.innovmetric.com.

[65] Alan D. Kalvin, Court B. Cutting, B. Haddad, and M. E. Noz. Constructing topologically connected surfaces for the comprehensive analysis of 3D medical structures. In *Medical Imaging V: Image Processing*, volume 1445, pages 247–258. SPIE, Feb. 1991.

[66] Alan D. Kalvin and Russell H. Taylor. Superfaces: Polyhedral approximation with bounded error. In *Medical Imaging: Image Capture, Formatting, and Display*, volume 2164, pages 2–13. SPIE, Feb. 1994. (Also IBM Watson Research Center tech report RC 19135).

[67] Alan D. Kalvin and Russell H. Taylor. Superfaces:polygonal mesh simplification with bounded error. *IEEE Computer Graphics and Appl.*, 16(3), May 1996. http://www.computer.org/pubs/cg&a/articles/g30064.pdf.

[68] Thomas Kao, David M. Mount, and Alan Saalfeld. Dynamic maintenance of Delaunay triangulations. Technical report, CS Dept., U. of Maryland at College Park, Jan. 1991. CS-TR-2585.

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

[70] Mark P. Kumler. An intensive comparison of triangulated irregular networks (TINs) and digital elevation models (DEMs). *Cartographica*, 31(2), Summer 1994. Monograph 45.

[71] Charles L. Lawson. Software for C^1 surface interpolation. In John R. Rice, editor, *Mathematical Software III*, pages 161–194. Academic Press, NY, 1977. (Proc. of symp., Madison, WI, Mar. 1977).

[72] D.T. Lee and Bruce J. Schachter. Two algorithms for constructing a Delaunay triangulation. *Intl. J. Computer and Information Sciences*, 9(3):219–242, 1980.

[73] Jay Lee. A drop heuristic conversion method for extracting irregular network for digital elevation models.

In *GIS/LIS '89 Proc.*, volume 1, pages 30–39. American Congress on Surveying and Mapping, Nov. 1989.

[74] Jay Lee. Comparison of existing methods for building triangular irregular network models of terrain from grid digital elevation models. *Intl. J. of Geographical Information Systems*, 5(3):267–285, July-Sept. 1991.

[75] J.-G. Leu and L. Chen. Polygonal approximation of 2-D shapes through boundary merging. *Pattern Recognition Letters*, 7(4):231–238, April 1988.

[76] Michael Lounsbery. *Multiresolution Analysis for Surfaces of Arbitrary Topological Type.* PhD thesis, Dept. of Computer Science and Engineering, U. of Washington, 1994. http://www.cs.washington.edu/research/projects/grail2/www/pub/pub-author.html.

[77] Kok-Lim Low and Tiow-Seng Tan. Model simplification using vertex-clustering. In *1997 Symposium on Interactive 3D Graphics*. ACM SIGGRAPH, 1997. To appear, http://www.iscs.nus.sg/∼tants/.

[78] David Luebke. Hierarchical structures for dynamic polygonal simplification. TR 96-006, Department of Computer Science, University of North Carolina at Chapel Hill, 1996.

[79] Stephane G. Mallat. A theory for multiresolution signal decomposition: The wavelet representation. *IEEE Trans. on Pattern Analysis and Machine Intelligence*, 11(7):674–693, July 1989.

[80] Benoit Mandelbrot. *Fractals – form, chance, and dimension*. Freeman, San Francisco, 1977.

[81] Michael Margaliot and Craig Gotsman. Approximation of smooth surfaces and adaptive sampling by piecewise-linear interpolants. In Rae Earnshaw and John Vince, editors, *Computer Graphics: Developments in Virtual Environments*, pages 17–27. Academic Press, London, 1995.

[82] Donald E. McClure. Nonlinear segmented function approximation and analysis of line patterns. *Quarterly of Applied Math.*, 33(1):1–37, Apr. 1975.

[83] Donald E. McClure and S. C. Shwartz. A method of image representation based on bivariate splines. Technical report, Center for Intelligent Control Systems, MIT, Mar. 1989. CICS-P-113.

[84] Robert B. McMaster. Automated line generalization. *Cartographica*, 24(2):74–111, 1987.

[85] Robert B. McMaster. The geometric properties of numerical generalization. *Geographical Analysis*, 19(4):330–346, Oct. 1987.

[86] Robert B. McMaster and K. S. Shea. *Generalization in Digital Cartography*. Assoc. of American Geographers, Washington, D.C., 1992.

[87] Joseph S. B. Mitchell. Approximation algorithms for geometric separation problems. Technical report, Dept. of Applied Math. and Statistics, State U. of New York at Stony Brook, July 1993.

[88] Edmond Nadler. Piecewise linear best L_2 approximation on triangulations. In C. K. Chui et al., editors, *Approximation Theory V*, pages 499–502, Boston, 1986. Academic Press.

[89] Gregory M. Nielson. Tools for triangulations and tetrahedrizations and constructing functions defined over them. In Gregory M. Nielson, Hans Hagen, and Heinrich Mueller, editors, *Scientific Visualization: Overviews, Methodologies, and Techniques*. IEEE Comput. Soc. Press, 1997.

[90] Paul Ning and Jules Bloomenthal. An evaluation of implicit surface tilers. *Computer Graphics and Applications*, pages 33–41, Nov. 1993.

[91] Theodosios Pavlidis. *Structural Pattern Recognition.* Springer-Verlag, Berlin, 1977.

[92] Thomas K. Peucker and David H. Douglas. Detection of surface-specific points by local parallel processing of discrete terrain elevation data. *Computer Graphics and Image Processing*, 4:375–387, 1975.

[93] Michael F. Polis, Stephen J. Gifford, and David M. McKeown, Jr. Automating the construction of large-scale virtual worlds. *Computer*, pages 57–65, July 1995. http://www.cs.cmu.edu/∼MAPSLab.

[94] Michael F. Polis and David M. McKeown, Jr. Iterative TIN generation from digital elevation models. In *Conf. on Computer Vision and Pattern Recognition (CVPR '92)*, pages 787–790. IEEE Comput. Soc. Press, 1992. http://www.cs.cmu.edu/∼MAPSLab.

[95] Michael F. Polis and David M. McKeown, Jr. Issues in iterative TIN generation to support large scale simulations. In *Proc. of Auto-Carto 11 (Eleventh Intl. Symp. on Computer-Assisted Cartography)*, pages 267–277, November 1993. http://www.cs.cmu.edu/∼MAPSLab.

[96] Jean Ponce and Olivier Faugeras. An object centered hierarchical representation for 3D objects: The prism tree. *Computer Vision, Graphics, and Image Processing*, 38:1–28, 1987.

[97] Enrico Puppo, Larry Davis, Daniel DeMenthon, and Y. Ansel Teng. Parallel terrain triangulation using the

Connection Machine. Technical Report CAR-TR-561, CS-TR-2693, Center for Automation Research, University of Maryland, College Park, Maryland, June 1991.

[98] Enrico Puppo, Larry Davis, Daniel DeMenthon, and Y. Ansel Teng. Parallel terrain triangulation. *Intl. J. of Geographical Information Systems*, 8(2):105–128, 1994.

[99] Urs Ramer. An iterative procedure for the polygonal approximation of plane curves. *Computer Graphics and Image Processing*, 1:244–256, 1972.

[100] Kevin J. Renze and James H. Oliver. Generalized surface and volume decimation for unstructured tessellated domains. In *VRAIS '96 (IEEE Virtual Reality Annual Intl. Symp.)*, Mar. 1996. Submitted.

[101] Shmuel Rippa. Adaptive approximation by piecewise linear polynomials on triangulations of subsets of scattered data. *SIAM J. Sci. Stat. Comput.*, 13(5):1123–1141, Sept. 1992.

[102] Shmuel Rippa. Long and thin triangles can be good for linear interpolation. *SIAM J. Numer. Anal.*, 29(1):257–270, Feb. 1992.

[103] Rémi Ronfard and Jarek Rossignac. Full-range approximation of triangulated polyhedra. *Computer Graphics Forum*, 15(3), Aug. 1996. Proc. Eurographics '96.

[104] Jarek Rossignac and Paul Borrel. Multi-resolution 3D approximations for rendering complex scenes. In B. Falcidieno and T. Kunii, editors, *Modeling in Computer Graphics: Methods and Applications*, pages 455–465, Berlin, 1993. Springer-Verlag. Proc. of Conf., Genoa, Italy, June 1993. (Also available as IBM Research Report RC 17697, Feb. 1992, Yorktown Heights, NY 10598).

[105] Hanan Samet. *Applications of Spatial Data Structures.* Addison-Wesley, Reading, MA, 1990.

[106] Hanan Samet. *The Design and Analysis of Spatial Data Structures.* Addison-Wesley, Reading, MA, 1990.

[107] Lori Scarlatos. *Spatial Data Representations for Rapid Visualization and Analysis.* PhD thesis, CS Dept, State U. of New York at Stony Brook, 1993.

[108] Lori Scarlatos and Theo Pavlidis. Hierarchical triangulation using cartographic coherence. *CVGIP: Graphical Models and Image Processing*, 54(2):147–161, March 1992.

[109] Lori L. Scarlatos and Theo Pavlidis. Optimizing triangulations by curvature equalization. In *Proc. Visualization '92*, pages 333–339. IEEE Comput. Soc. Press, 1992.

[110] Francis Schmitt and Xin Chen. Fast segmentation of range images into planar regions. In *Conf. on Computer Vision and Pattern Recognition (CVPR '91)*, pages 710–711. IEEE Comput. Soc. Press, June 1991.

[111] Francis Schmitt and Xin Chen. Geometric modeling from range image data. In *Eurographics '91*, pages 317–328, Amsterdam, 1991. North-Holland.

[112] Francis Schmitt and Behrouz Gholizadeh. Adaptative polyhedral approximation of digitized surfaces. In *Computer Vision for Robots*, volume 595, pages 101–108. SPIE, 1985.

[113] Francis J. M. Schmitt, Brian A. Barsky, and Wen-Hui Du. An adaptive subdivision method for surface-fitting from sampled data. *Computer Graphics (SIGGRAPH '86 Proc.)*, 20(4):179–188, Aug. 1986.

[114] Will Schroeder, Ken Martin, and Bill Lorensen. *The Visualization Toolkit, An Object-Oriented Approach To 3D Graphics.* Prentice Hall, 1996. Code at http://www.cs.rpi.edu:80/~martink/.

[115] William J. Schroeder and Boris Yamrom. A compact cell structure for scientific visualization. In *SIGGRAPH '94 Course Notes CD-ROM, Course 4: Advanced Techniques for Scientific Visualization*, pages 53–59. ACM SIGGRAPH, July 1994.

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

[117] Cláudio T. Silva, Joseph S. B. Mitchell, and Arie E. Kaufman. Automatic generation of triangular irregular networks using greedy cuts. In *Proc. Visualization '95*. IEEE Comput. Soc. Press, 1995. http://www.cs.sunysb.edu:80/~csilva/claudio-papers.html.

[118] Marc Soucy and Denis Laurendeau. Multi-resolution surface modeling from multiple range views. In *Conf. on Computer Vision and Pattern Recognition (CVPR '92)*, pages 348–353, June 1992.

[119] Marc Soucy and Denis Laurendeau. Multiresolution surface modeling based on hierarchical triangulation. *Computer Vision and Image Understanding*, 63(1):1–14, 1996.

[120] David A. Southard. Piecewise planar surface models from sampled data. In N. M. Patrikalakis, editor, *Scientific Visualization of Physical Phenomena*, pages 667–680, Tokyo, 1991. Springer-Verlag.

[121] Steven L. Tanimoto and Theo Pavlidis. A hierarchical data structure for picture processing. *Computer Graphics and Image Processing*, 4(2):104–119, June 1975.

[122] David C. Taylor and William A. Barrett. An algorithm for continuous resolution polygonalizations of a discrete surface. In *Proc. Graphics Interface '94*, pages 33–42, Banff, Canada, May 1994. Canadian Inf. Proc. Soc.

[123] Ivan Tomek. Two algorithms for piecewise-linear continuous approximation of functions of one variable. *IEEE Trans. Computers*, C-23:445–448, Apr. 1974.

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

[125] K. J. Turner. *Computer perception of curved objects using a television camera.* PhD thesis, U. of Edinburgh, Scotland, November 1974.

[126] Amitabh Varshney. *Hierarchical Geometric Approximations*. PhD thesis, Dept. of CS, U. of North Carolina, Chapel Hill, 1994. TR-050.

[127] Amitabh Varshney, Pankaj K. Agarwal, Frederick P. Brooks, Jr., William V. Wright, and Hans Weber. Generating levels of detail for large-scale polygonal models. Technical report, Dept. of CS, Duke U., Aug. 1995. CS-1995-20, http://www.cs.duke.edu/department.html#techrept.

[128] Brian Von Herzen and Alan H. Barr. Accurate triangulations of deformed, intersecting surfaces. *Computer Graphics (SIGGRAPH '87 Proceedings)*, 21(4):103–110, July 1987.

[129] Robert Weibel. Models and experiments for adaptive computer-assisted terrain generalization. *Cartography and Geographic Information Systems*, 19(3):133–153, 1992.

[130] Ellen R. White. Assessment of line-generalization algorithms using characteristic points. *The American Cartographer*, 12(1):17–27, 1985.

[131] Lance Williams. Pyramidal parametrics. *Computer Graphics (SIGGRAPH '83 Proc.)*, 17(3):1–11, July 1983.