Simplifying Terrain Models and Measuring Terrain Model Accuracy

David Scott Andrews

May 3, 1996

Abstract

We describe a set of TIN simplification methods that enable the use of the triangulation hierarchy introduced by Kirkpatrick [Kir83] and modified by de Berg and Dobrindt [dBD95a, dBD95b]. This triangulation hierarchy can be used to form a terrain model combining areas with varying levels of detail. One variant of the delete simplification method formed simplifications with accuracy close to the greedy method.

We also investigated different variables that can be used to measure the accuracy of our simplified terrain models. Although the use of derivative statistics did not significantly alter our evaluation of the performance of our simplification methods, we recommend that any future comparisons should be aware of these alternative variables of surface characterization.

Contents

List of Tables

List of Figures

Chapter 1

Introduction

There are many fields that work with models of terrain. As computers were incorporated into these fields, various Digital Terrain Models were proposed and developed. One popular terrain model, introduced in 1978, is the Triangular Irregular Network [PFLM78]. It is based on irregularly distributed surface-specific points. The selection of these points (as well as the triangulation criteria) has been well studied.

A single terrain model might not be appropriate for all users because of constraints on computing space and time. A solution is to simplify the model, providing a balance of size and detail to suit each individual. The user could select the best model from a set of precomputed simplifications or they could create a new simplified model that would meet their own requirements.

There exist many ways to simplify the model of a terrain. To compare the effectiveness of these methods we need to examine their speed and their accuracy. Previous comparisons have used only the elevation error of the models to grade their performance. But other fields such as geomorphometry, the study of landform, and geometry of surfaces indicate that there are other variables that can be used to characterize a terrain. The effect of a simplification on these variables should also be observed.

Chapter 2 describes and compares three Digital Terrain Models. The next chapter covers some of the work we did on simplifying polygonal lines and measuring the concavity of polygonal lines. Chapter 4 describes various methods for Triangular Irregular Network simplification and Chapter 5 describes our surface accuracy measures. The results of our experiments are in Chapter 6 and our conclusions are in Chapter 7.

Chapter 2

Digital Terrain Models (DTMs)

A **terrain** is the physical features of a tract of land. Its **topography** is its physical or natural features and their structural relationships. If the earth's surface was regular, it could be perfectly modelled by mathematical functions. Unfortunately, it is continuous but irregular, requiring the use of models based on samples of the surface [Kum94].

A **Digital Terrain Model (DTM)** is a representation of terrain and topography. A DTM can be stored on a computer as a data structure and a set of associated procedures. Computer applications containing DTMs are used by people who specialize in terrains, such as geomorphologists, surveyors, photogrammeters, cartographers and mathematicians [Mar78]. DTMs are also a vital component of Geographical Information Systems (GISs).

The first section of this chapter lists some of the terrain characteristics that can be obtained from a DTM. The next three sections describe three common DTMs, the Digital Line Graph (DLG), the Digital Elevation Model (DEM) and the Triangular Irregular Network (TIN). The final section compares these three DTMs.

2.1 Terrain Characteristics

The following paragraphs describe some of the problems that require DTMs to solve. Obtaining various terrain characteristics from a DTM help to form the solutions to these problems. The data structure and procedures of a DTM should support the computation of these terrain characteristics. Some sample characteristics are listed below.

- elevation
- slope
- aspect
- visibility
- drainage

A **slope map** is a map of the slope magnitude at each point of the terrain. Hammond [Ham64] notes that slope has a strong effect on landuse. One of the variables he uses to classify landform is the frequency of gentle slope. At his upper limit of eight percent, machine cultivation becomes difficult, the effects of erosion become troublesome and vehicle movement becomes impeded. A DTM should be able to compute the slope at a given point or generate a slope map for the complete surface.

De Berg and van Kreveld [dBvK93] define a height level map to be built from a DTM. This map can be used to answer path queries with height restrictions. Elevation has an effect on temperature. The effects of slope have been noted in the paragraph above. A path that satisfies elevation and slope constraints might be preferred to the Euclidean shortest distance path.

De Floriani, Marzano and Puppo [DFMP94] study line-of-sight visibility problems. Their example problem asks for the arrangement of a set of microwave transceiver stations in such a way that a signal transmitted from any station can be received by any other station. Since a signal propagates along a straight line and is interrupted by physical obstructions, the towers must be mutually visible. Other related line-of-sight problems include the location of radar, laser or sonar surveillance systems and the location of television transmitters. They use a discrete visibility map consisting of a set of candidate points with edges connecting points that are mutually visible.

The **viewshed** of a point on the terrain is the area of the terrain visible to that point. Fisher [Fis94] considers a variant of the binary viewshed. The smoke from a forest fire might be in the viewable area of an observation tower even if the fire is not in the line-of-sight. The same situation applies to the visual impact of a tall structure built beyond the horizon. His variant would encode the distance between the horizon and the surface. The height of the smoke of a forest fire or the height of a building could be compared to this distance to obtain a more accurate viewable area than a simple line-of-sight viewshed.

Mark [Mar78] proposes a typology for DTM data structures with an initial classification based on how elevation is computed. His typology has been reproduced in Figure 2.1. Tabular or discrete data

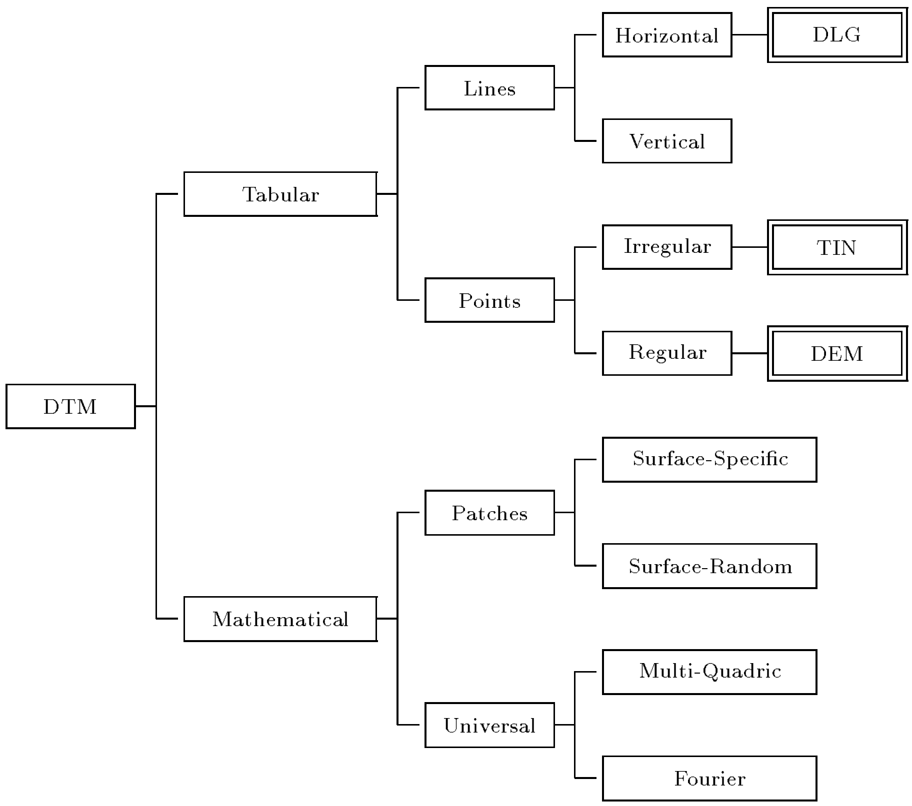

Figure 2.1: Mark's DTM typology

structures obtain the elevation by interpolating from the surrounding data. Mathematical or continuous data structures obtain the elevation by evaluating a modelling function. The difference between the two branches of his initial division is minor since interpolation can be considered as the evaluation of a function. He also notes that the digital data structure of a continuous function is represented by a discrete set of coefficients. We only use a subdivision of his typology as the data structures of the three DTMs discussed below are all part of the tabular or discrete data structure division.

A discrete data structure requires an interpolation method. The interpolation method depends on the data structure and the desired level of continuity. Different interpolation methods can affect the continuity of the surface and its derivatives. For example, the TIN model discussed in Section 2.4 commonly uses linear interpolation. The surface of this model is continuous but its slope is not differentiable.

2.2 Digital Line Graphs (DLGs)

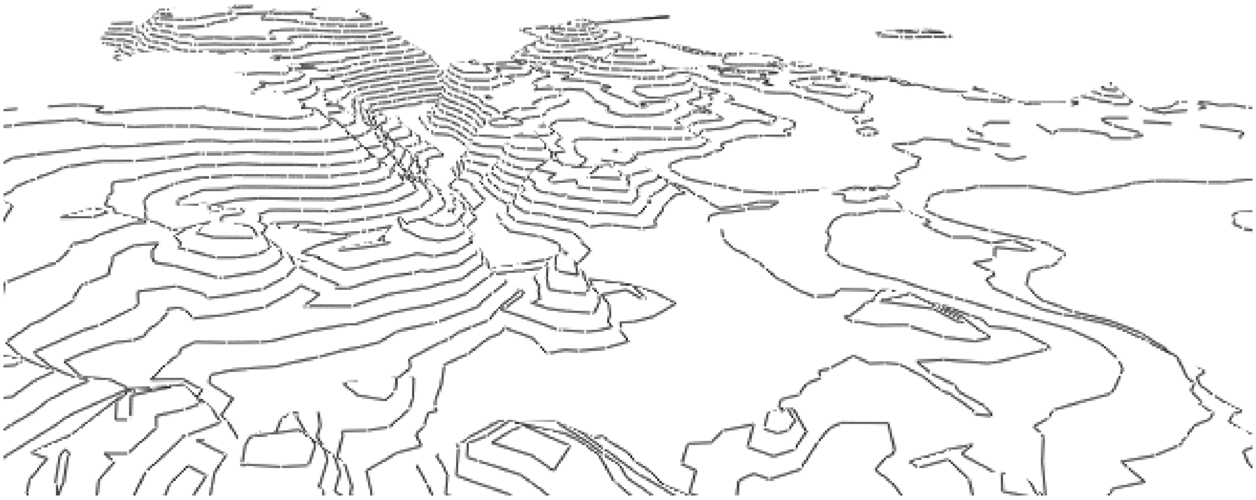

Figure 2.2: Digital Line Graph (DLG)

A **Digital Line Graph (DLG)** is a set of contour lines. Figure 2.2 shows a perspective view of a DLG. Each contour line is a list of x and y coordinate vertices with a common z coordinate altitude. Each vertex requires the storage of two coordinates. DLGs are part of Mark's [Mar78] Tabular - Lines - Horizontal DTM subdivision.

Terrain and topography information is commonly stored in non-automated contour maps [Eva72]. Data files of contours are widely available because digitizing contour lines from existing maps is a popular input method. Other DTMs are often constructed from digitized contour lines because of their availability. Elevation contours can be formed from other DTMs but this is only done because contour lines are a familiar visual representation of an elevation surface [Kum94].

Terrain characteristics can be quickly computed if they only depend on a single contour line. Unfortunately most terrain characteristics, like the sample characteristics listed in Section 2.1, are much more difficult to compute because they require identifying nearby contours [Eva72].

2.3 Digital Elevation Models (DEMs)

A **Digital Elevation Model (DEM)** or **regular grid** is a matrix of z coordinate altitudes with x and y coordinates implicit in the grid. Figure 2.3 shows a perspective view of a DEM with edges connecting neighbouring vertices. Each vertex requires the storage of only one coordinate. DEMs are part of Mark's [Mar78] Tabular - Points - Regular - Constant Density DTM subdivision.

DEMs are widely available. The United States Geological Survey produces DEM data files corresponding to its published maps. These are created by manually profiling stereo models or by using an automatic image correlator [Kum94].

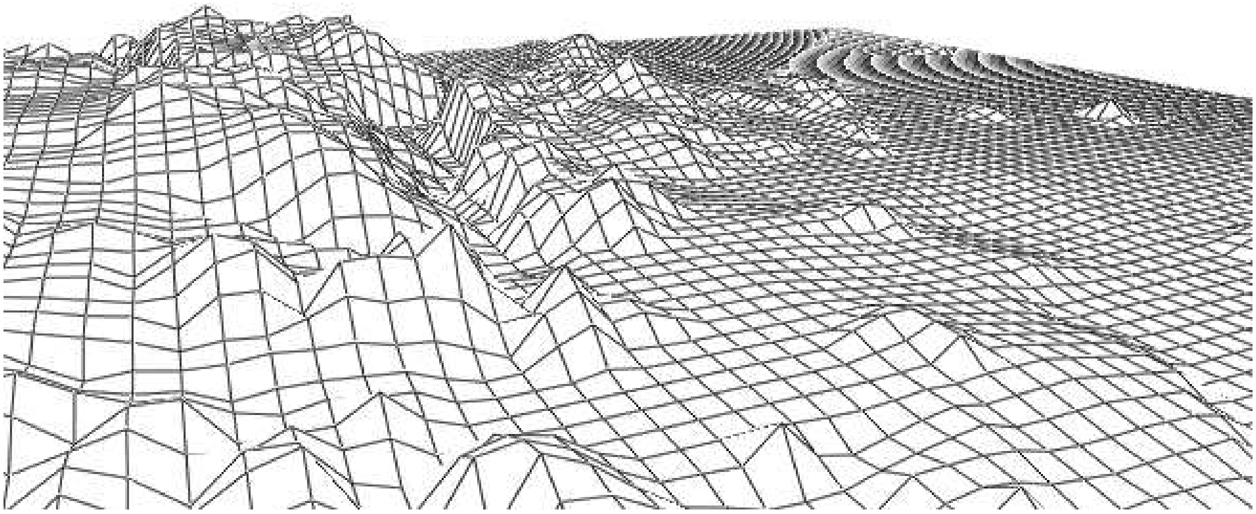

Figure 2.3: Digital Elevation Model (DEM)

DEMs are widely used because they are convenient for programming and for machine storage [Mar78]. Many problems requiring a DTM benefit from the fixed data resolution of a DEM. Information concerning a restricted area, a **neighbourhood**, can be easily selected since the sequence of storage is known and regular [Eva72]. But a fixed resolution is also a disadvantage. The resolution of the regular grid may not adequately model rough areas of the surface while being redundant in smooth areas.

Kumler [Kum94] notes that many early favorable comparisons of DEMs and other DTMs concerned the trade-off between the speed of processing and the cost of storing the structure. Later comparisons considered accuracy and required storage space.

2.4 Triangular Irregular Networks (TINs)

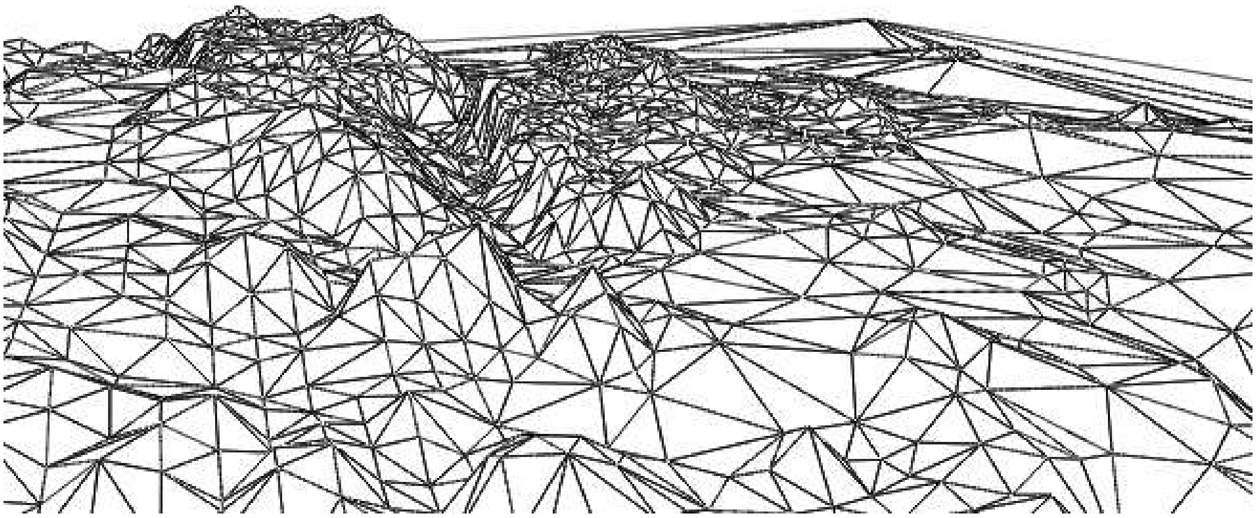

Figure 2.4: Triangular Irregular Network (TIN)

Peucker, Fowler, Little and Mark [PFLM78] were motivated to develop an alternative model because of problems with the Digital Elevation Model (DEM). A DEM must use a fine regular grid to accurately model rough terrain; this grid can be highly redundant in smooth terrain. Also, the DEM data structure doesn't correspond with the "structure of the phenomenon" [Mar78]. The design of a DTM data structure should reflect the terrain it will represent, unlike the DEM which was designed with the problems it solves and the machines it uses in mind. But the neighborhood of a point in a DEM can be reconstructed without searching the entire model. Any successful alternative must store this useful information explicitly.

A **Triangular Irregular Network (TIN)** is a set of irregularly-distributed vertices linked together by straight lines to form a continuous, non-overlapping set of triangular elements [Mar78]. Figure 2.4 shows a perspective view of a TIN. Each vertex requires the storage of all three coordinates. TINs are part of Mark's [Mar78] Tabular - Points - Irregular DTM subdivision.

Peucker, Fowler, Little and Mark [PFLM78] base their TIN on surface-specific vertices. Ridges and channels connecting the peaks, pits and passes of the surface form the skeleton of the TIN. Additional vertices are added to improve spot height accuracy and the set is then connected in a Delaunay triangulation.

Although DEMs have better elevation detail and DEM files are widely available, TINs have the advantage of efficient storage and better handling of intervisibility analysis and extraction of hydrological terrain features [Lee91].

Variations of TINs exist with different vertex distributions and different triangulation criteria. The following section reviews triangulations in general and the Delaunay triangulation in particular.

2.4.1 Triangulations

A **triangulation** of a set of vertices V is a set of closed triangles T formed by edges connecting pairs of vertices in V that satisfy the following conditions [DLR90].

- The set of all vertices of triangles in T is V.
- Each edge of a triangle in T contains only two vertices (its endpoints) in V.
- The union of all triangles in T is equal to the convex hull of V.
- The intersection of the interiors of any two distinct triangles in T is empty.

Lee and Schachter [LS80] present an iterative algorithm for constructing a triangulation based on two primitive operations, adding a vertex and swapping a diagonal. In Figure 2.5, a new vertex is connected to each of the three vertices of its enclosing triangle in the existing triangulation. If the quadrilateral formed

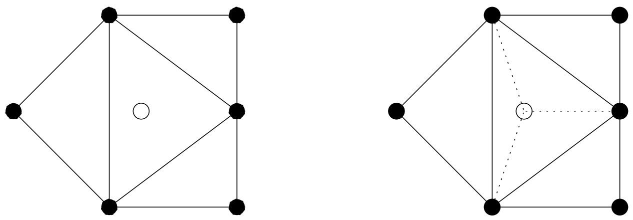

Figure 2.5: Adding a vertex to a triangulation

by the union of a new triangle and its neighbour is convex, it will have an alternate diagonal. The diagonal should be swapped with the alternate if required by a local optimization procedure. Figure 2.6 shows how a diagonal is swapped.

Different triangulations can be defined by different local optimization procedures. If a diagonal is swapped, two new triangles are formed with quadrilaterals that have to be similarly optimized. Some of the many possible local optimization procedures are listed below.

- minimize the maximum or sum of $perimeter^2/area$ of the two triangles [Law72]

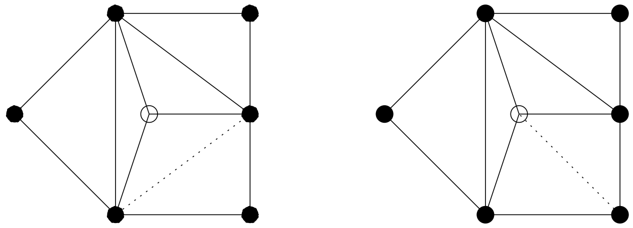

Figure 2.6: Swapping a diagonal in a triangulation

- maximize the minimum angle in the two triangles [LS80]
- minimize the angle between the normals of the two triangles [DLR90]

A local cost function on each pair of neighbouring triangles can be used as an optimization procedure. A global cost function for the triangulation can be formed by using either the maximum or the sum of the local cost functions. Lawson proved that any triangulation of a finite vertex set can be transformed into another triangulation of the same set with a finite number of diagonal swaps [Law72]. This theorem shows that any triangulation can be transformed into a triangulation that minimizes a given global cost function. Unfortunately, minimizing the local cost function for each diagonal does not guarantee that the global cost function will be minimized.

There are triangulations that do not have a local optimization procedure. One example is the Minimum-Weight Triangulation [PS85]. This triangulation is formed by minimizing the total length of the triangulation edges. Locally minimizing the length of the edges of each pair of triangles does not guarantee that the total length will also be minimized.

Delaunay Triangulation

A **Delaunay triangle** is a triangle whose circumcircle does not contain any other vertex in V. This property is called the **empty-circle property**. A **Delaunay triangulation** is a triangulation in which every triangle

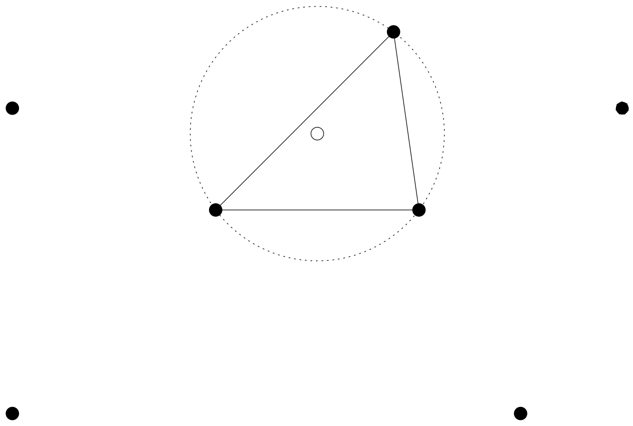

Figure 2.7: Delaunay triangle

in T is a Delaunay triangle.

The Delaunay triangulation is related to the Voronoi diagram. The **Voronoi diagram** for a set of vertices V is a set of Voronoi cells (convex polygons), one about each vertex. Each **Voronoi cell** contains all points closer to its vertex than any other vertex in V. By definition, an edge between two Voronoi cells is equidistant from two vertices. The point where three edges meet is equidistant from three vertices.

The Delaunay triangulation is the graph dual of the Voronoi diagram. An edge in the Delaunay

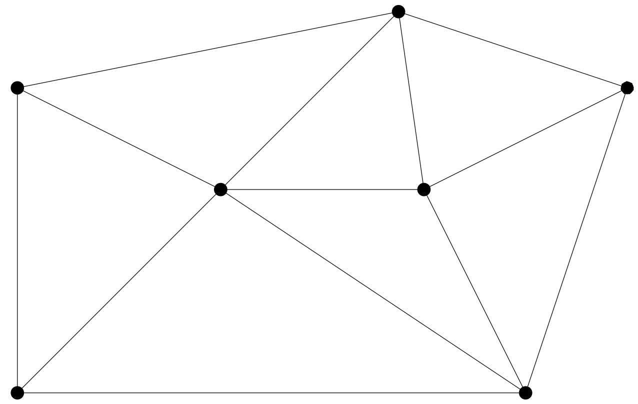

Figure 2.8: Delaunay triangulation

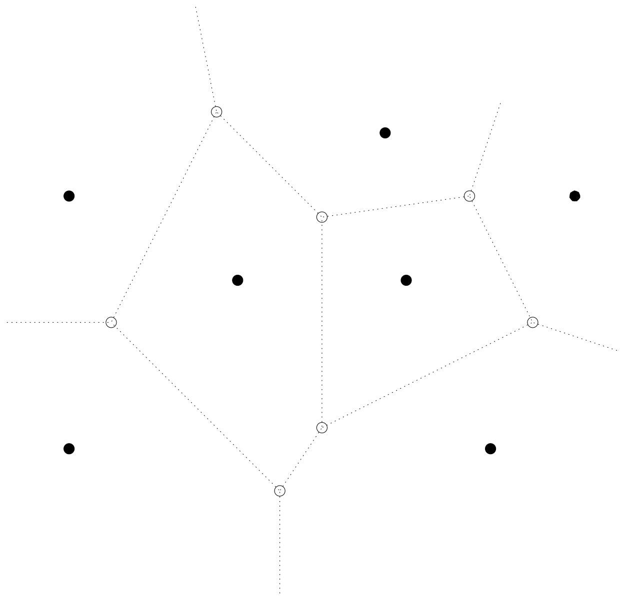

Figure 2.9: Voronoi diagram

triangulation connects vertices with adjacent Voronoi cells. A triangle in the Delaunay triangulation has a circumcircle centered at a point where three Voronoi edges meet. By definition, the Voronoi diagram for a set of vertices is unique. Since the Delaunay triangulation is the dual of the Voronoi diagram, the Delaunay triangulation is also unique under general position assumptions. The Delaunay triangulation can be constructed by using the empty-circle property of Delaunay triangles. It can also be constructed by using the local optimization procedure of maximizing the minimum angle in each pair of triangles. This triangulation is used for fitting triangular faceted surfaces to digital terrain data by Lee and Schachter because it minimizes computation time and produces a good visual display [LS80]. The resulting TIN is used for flight simulators.

2.5 Comparison of DTMs

Availability, usability and storage space should be considered when comparing DTMs. DTM data files should be available or algorithms should exist to convert other DTMs with available data files to the desired DTM.

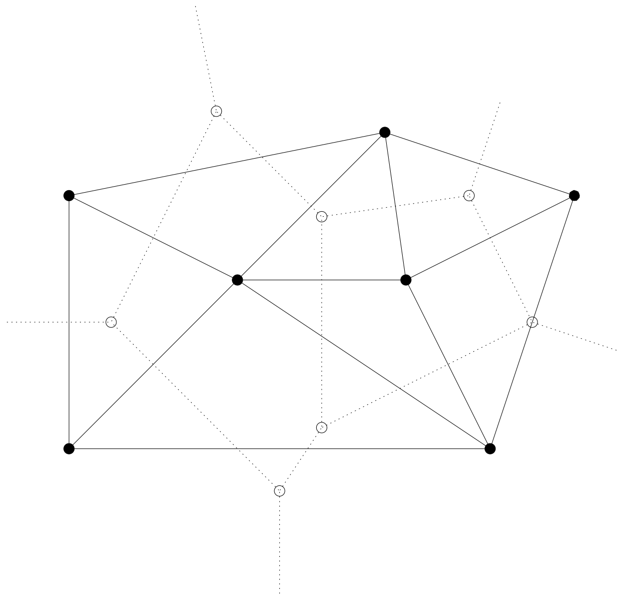

Figure 2.10: Delaunay triangulation and Voronoi dual

DTMs should be able to quickly and accurately compute the terrain characteristics needed by the user. DTMs should minimize the amount of space needed to store and use the model while keeping an acceptable level of accuracy.

Availability

DLG and DEM data files are widely available. TIN data files are not widely available but many algorithms exist for constructing a TIN from a DLG or a DEM. Four methods for building TINs from DEMs are reviewed in Section 4.2.1.

Usability

DLGs are not easy to use. Single contour operations are straightforward but tasks requiring multiple contours are much more difficult because moving from contour to contour is very awkward. DEMs are very easy to use. They are in a matrix form that is easy to access and manipulate. TINs are more complex than DEMs but they are also better for solving some of the more complex DTM problems.

Storage

DEMs only need to store one coordinate per point while TINs need to store all three coordinates as well as adjacency information. But the number of points required to accurately describe a surface with a TIN is usually smaller than the number of points required by a DEM. Kumler [Kum94] studies this tradeoff by comparing the accuracy of a “Super TIN” with 50000 points to DEM requiring equal data file storage space with 170000 points.

Chapter 3

Work with Polygonal Lines

We started working with both DLGs and TINs but we were unable to transfer ideas useful in forming a simplification of a surface and measuring the accuracy of a simplified surface from one model to the other. The main obstacle was the number of dimensions of the primitives in each model; each polyline in a DLG is 2–D while a triangular facet and its neighbours in a TIN are 3–D. We present some of our work on simplifying and measuring the concavity of the polylines in a DLG in the first two sections of this chapter. The third section discusses the ideas that were transferred to our work with TINs.

Mokhtarian [Mok90] proposes a multi-scale shape representation technique for planar curves based on curvature. This work can be related to both polyline simplification discussed in Section 3.1 and shape measurement discussed in Section 3.2. He uses an multi-scale approach to find a quick, approximate match at a coarse, simplified scale and then uses the finer scales to obtain more accurate matches. He also uses a technique based on curvature to represent the shape of the curve that is invariant under rotation, uniform scaling and translation.

3.1 Simplification of Polygonal Lines

A **polyline** is a sequence of vertices $v_0, v_1, \ldots, v_n$ connected by a sequence of straight line segments $v_0v_1, v_1v_2, \ldots, v_{n-1}v_n$. We will refer to the section of the polyline from v_i to v_j as $v_i \ldots v_j$. We wanted a method for simplifying a polyline. The method should produce a polyline with a reduced number of vertices but still a good caricature of the original polyline. The algorithm presented by Douglas and Peucker [DP73] performs well.

3.1.1 Douglas-Peucker Algorithm for Line Simplification

Douglas and Peucker [DP73] present a method for line simplification. The method computes a simplification value for each vertex of the original polyline. A vertex is included in a simplification if its simplification value is more than a given threshold. We implemented an algorithm that computes the simplification values of the n vertices of a polyline in $O(n^2)$ time and we also implemented an algorithm that forms a simplified polyline in $O(n)$ time after the simplification values have been computed.

The same simplification values can be used with different thresholds to form polylines with different levels of detail. The vertices that are not included in the simplification are all within a small perpendicular distance to the line segments connecting the vertices that are included.

Algorithm notes

This algorithm recursively sets the simplification value of the vertex furthest from the line segment connecting the endpoints of the polyline. It is described below.

1. Initialize the simplification value of the two endpoints of the polyline to infinity.
2. Form a line segment v_0v_n with the two endpoints of the polyline $v_0 \ldots v_n$.
3. Find the vertex v_i in the polyline $v_0 \ldots v_n$ with the largest perpendicular distance to the line segment v_0v_n.
4. Set the simplification value for v_i to the minimum of the perpendicular distance from v_i to the line segment v_0v_n and the simplification values of the two endpoints.
5. Repeat steps 2 to 5 with the polyline $v_0 \ldots v_i$ and the polyline $v_i \ldots v_n$ until every vertex has a simplification value.

If the method is applied to a set of contour lines, the simplified contour lines may intersect. Also, the method retains the points where a contour crosses a ridge or a valley almost too well, resulting in too many vertices concentrated on those features and not enough for the other parts of the surface [Kum94].

3.2 Concavity Measures for Polygonal Lines

The simplification of a polyline should preserve key properties of the original polyline. One property we examined was concavity. To observe the effect of the simplification of a polyline on concavity, we needed a concavity measure for points on a polyline. A good measure should not depend on the level of detail in the polyline. In addition, it should not be affected by translation or rotation of the polyline and it should be scalable.

The **window** is the section of the polyline around a given point that is used to determine the concavity of that point. The measure should have a parameter to control the size of the window. The size of the window can be set so that local changes do not dominate the concavity measure.

We developed two measures, the secant measure and the disk measure.

3.2.1 Secant Concavity Measure

The **secant concavity measure** of a point on a polyline is based on the ratio of the area between the polyline and a secant and the square of the length of the secant.

The area between the polyline and the secant is calculated by finding the difference between the area covered by the polyline and the area covered by the secant. The area covered by a directed line segment is equal to the area of a triangle formed by the line segment and a point. The area is positive if the point lies on the left hand side of the segment and negative if the point lies on the right hand side. The area covered by a polyline is equal to the sum of the areas covered by the directed line segments of the polyline for the same point. The area can be positive or negative. These areas are dependent on the location of the point but the difference of these areas is not.

Algorithm notes

Given a point p on the polyline and a window distance d.

1. Find the two points that are d along the polyline in both directions from p and compute the area covered by the section of the polyline between these two points. The measure is undefined if the distance along the polyline from p to an endpoint is less than d.
2. Form a secant between these two points and compute the area covered by the secant.

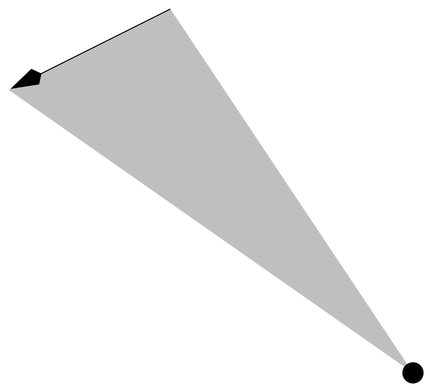

Figure 3.1: Area covered by a line segment

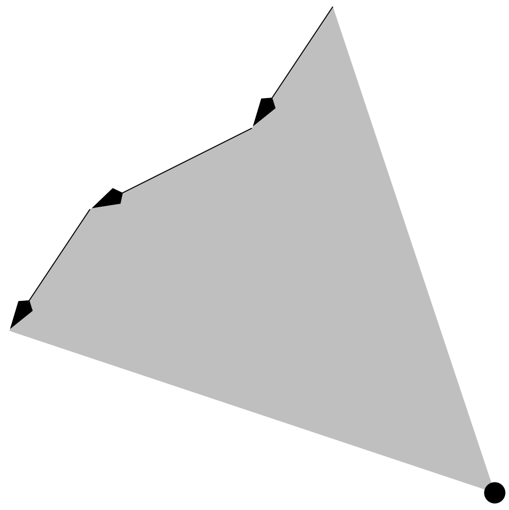

Figure 3.2: Area covered by a polyline

3. Calculate the difference of the area covered by the section of the polyline and the area covered by the secant. This difference represents the area between the polyline and the secant. It can be positive or negative.

4. Form the ratio of this area and the square of the length of the secant. Using the square of the length of the secant makes the measure scalable. It also makes the ratio an area:area ratio. A large positive ratio indicates a convex point relative to the left side of the polyline near p. A large negative ratio indicates a concave point. The ratio has no finite lower or upper bound.

As a polyline is simplified, the region specified by the window parameter gets larger. One solution to this problem is to keep the ratio of the window distance and the length of the polyline constant. If the secant concavity measure with a window distance of d is used on a polyline of length l, then a simplified polyline of length s should use the measure with a scaled window distance of $\frac{s}{l}d$.

Implementation notes

This implementation can classify every point (not just the vertices) on a polyline with n vertices in $O(n)$ time. It maintains the secant concavity measure for the "window" point and it maintains the "tail" point a distance d behind the "window" point and the "head" point d ahead of the "window" point.

- Initialize the points and the concavity measure.
  - Initialize the "tail" point to one endpoint of the polyline.
  - Initialize the "window" point d along the polyline.
  - Initialize the "head" point $2d$ along the polyline.
  - Initialize the concavity measure using the polyline section and the secant between the "tail" point and the "head" point.
- Maintain the concavity measure while moving the three points along the polyline. The concavity measure is a ratio of two areas. As the "window", "tail" and "head" points move along the polyline, this ratio forms a quadratic function.
  - If the "tail" point or the "head" point reaches a polyline vertex, the rate of change of the ratio changes.
  - If the ratio reaches a given positive or negative threshold, note the change of classification.

3.2.2 Disk Concavity Measure

The **disk concavity measure** of a point on a polyline is based on the ratio of the area on the left side of a directed polyline within a circle of given radius to the area of that circle.

Algorithm notes

Given a point p on the polyline and a window radius r.

1. Form a disk with a radius r around p.
2. Find the two points where the polyline crosses the boundary of the disk.
3. Calculate the area of the intersection of the disc and the left side of the polyline section between these two boundary points.
4. Compute the difference of this area and the area of half the circle.
5. Form the ratio from this difference and the area of half the circle. A large positive ratio indicates a convex point. A large negative ratio indicates a concave point. The ratio has a lower bound of -1.0 and an upper bound of 1.0.

There is no need to scale the window radius as the polyline is simplified.

Implementation notes

Our implementation only classified the vertices of the polyline. It calculates the disk concavity of each vertex independently. The upper time bound is $O(n^2)$ because calculating the disk concavity measure for a single point is an $O(n)$ operation and this must repeated for all n vertices.

3.2.3 Comparison of Concavity Measures

The implementation of our disk measure only computes the concavity for polyline vertices while the implementation of our secant measure algorithm computes concavity for all points on the polyline. Also, our secant measure implementation has a better time bound than our disk measure algorithm. The ratio bounds are different so ratios cannot be directly compared.

3.3 Transferring Ideas to TINs

3.3.1 Simplification

A useful feature of the Douglas-Peucker line simplification method is its use of recursion. The algorithm subdivides the polyline into two sections and simplifies these sections independently. This would also be a useful feature of a TIN simplification method.

The line simplification method uses the distance from a point to a line segment of the polyline to determine if the line should be subdivided. With a TIN, the distance from a point to the planar facet of a triangle can be used as a measure of the importance of that point.

3.3.2 Concavity Measures

The ideas behind the secant measure cannot be easily transferred to a polyhedral surface. The secant measure uses a secant, which can be defined for a polyline or a curve but not for a surface. The ideas behind the disk measure can be transferred but the computation is difficult. Finding the volume of the intersection of a ball and a set of planes is harder than computing the intersection of the disk and the area inside the polyline.

Chapter 4

Simplification of a Triangular Irregular Network (TIN)

This chapter describes the methods we used to simplify TINs. The first two sections review past work on hierarchical triangulations and hierarchical TINs. Section 4.3 describes our methods for selecting and removing vertices to form a hierarchical TIN that can combine different levels of detail. We used another TIN simplification method, described in Section 4.4, to compare the surface accuracy of our simplifications. The details of our implementation of this hierarchical method are described in Section 4.5.

4.1 Hierarchical Triangulations

Kirkpatrick [Kir83] presents an algorithm for optimal search in planar subdivisions that uses a triangular subdivision hierarchy. A **planar subdivision** is a finite collection of line segments inducing a partition of the plane into polygonal regions. A **triangular subdivision** is a finite collection of finite line segments inducing a partion of the plane into regions bounded by three line segments. This is equivalent to a triangulation. A planar subdivision can be reduced to a triangular subdivision in two steps. In the first step, the planar subdivision is intersected with a triangle chosen to contain all intersections of the planar subdivision. In the second step, each interior region of this triangle is triangulated.

The **neighbourhood** of a vertex v in a triangular subdivision is the star-shaped polygon formed by the union of the triangles incident to v. A triangular subdivision can be simplified by removing a vertex and retriangulating its neighbourhood. The **parents** of a triangle t in the retriangulated neighbourhood are the triangles in the original neighbourhood that intersect t. If a vertex v with degree $\deg(v)$ is removed, each triangle in the retriangulated neighbourhood will have at most $\deg(v)$ parents regardless of how the neighbourhood is retriangulated.

The simplification achieved by removing a single vertex is minimal. If a set of vertices is removed, the simplification can be more substantial. The neighbourhoods of non-adjacent vertices do not intersect except possibly along edges. The vertices in an independent set can be removed in parallel and their neighbourhoods can be retriangulated independently.

The triangulation hierarchy is a sequence of successively simplified triangular subdivisions. This hierarchy is used to find the region containing a given test point. At each level of the hierarchy, the search algorithm locates the region containing a given test point by searching the through a set of candidates. The parents of that region become the new candidates for the search at the next level. To optimize the search algorithm, the number of levels and the size of the set of search candidates or parents should be minimized. The degree of a vertex affects the size of the set of search candidates and the number of vertices removed affects the height of the hierarchy. The size of a set of independent vertices with degree at most 11 would

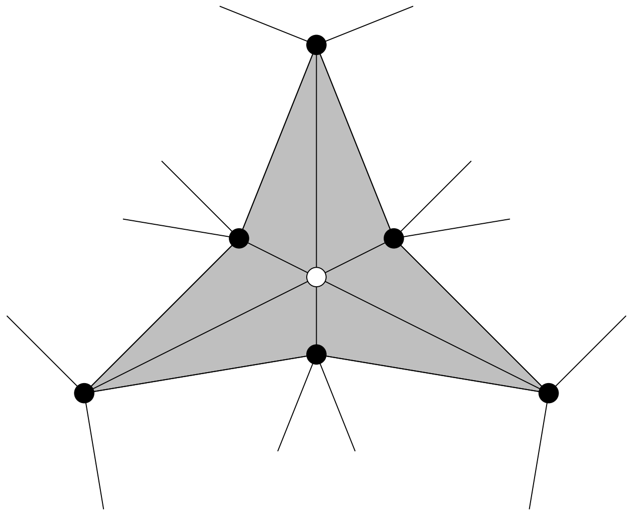

Figure 4.1: Neighbourhood of a vertex

be at least $n/18$ where n is the number of vertices in the current triangulation. The proof is below.

Notation

$$
\begin{aligned}
V &= \text{vertices} \\
V_{high} &= \{v_i \in V | \deg(v_i) > 11\} \\
V_{low} &= \{v_i \in V | \deg(v_i) \leq 11\} \\
n_v &= \text{number of vertices} \\
n_e &= \text{number of edges} \\
n_t &= \text{number of triangles} \\
n_h &= \text{number of edges in hull (minimum 3)}
\end{aligned}
$$

Proof

$$
\begin{aligned}
n_t - n_e + n_v - 1 &= 0 \text{ (Euler's formula)} \\
3n_t + n_h &= 2n_e \text{ (counting edges)} \\
n_t &= 2(n_e - n_h)/3 \\
2(n_e - n_h)/3 - n_e + n_v - 1 &= 0 \text{ (substitute } n_t \text{ in Euler's formula)} \\
2n_e - n_h - 3n_e + 3n_v - 3 &= 0 \\
n_e &= 3n_v - n_h - 3 \\
n_e &\leq 3n_v - 6 \text{ (minimum } n_h \text{ is 3)}
\end{aligned}
$$

$$
\begin{aligned}
\text{average degree} &= 2n_e/n_v \\
&\leq 2(3n_v - 6)/n_v \\
&< 6
\end{aligned}
$$

$$
\begin{aligned}
3|V_{low}| + 12|V_{high}| &< 6|V| \text{ (total degree)} \\
9|V_{high}| &< 3|V| \text{ (subtract } 3|V|) \\
|V_{high}| &< n_v/3 \\
|V| - |V_{high}| &\geq n_v - n_v/3 \\
|V_{low}| &\geq 2n_v/3 \\
|V_{low}|/12 &\geq n_v/18 \text{ (vertex and 11 neighbours)}
\end{aligned}
$$

Kirkpatrick's optimal search algorithm has a $O(n)$ preprocessing time, $O(\lg n)$ search time and $O(n)$ space.

4.2 Hierarchical TINs

A hierarchical TIN attempts to combine the useful features of a hierarchical data structure with a TIN. The result should be able to approximate the surface at different resolutions while keeping the benefits associated with TINs, such as irregularly-distributed vertices, efficient storage and better handling of visibility and drainage problems. This section reviews different approaches of forming a hierarchical TIN.

Gómez and Guzmán [GG79] present a model for three-dimensional surface representation that uses a tree of triangles. If the surface represented by a planar triangle does not satisfy a prespecified error tolerance it is subdivided. Figure 4.2 shows the four child triangles formed by adding new vertices to the three edges of the parent triangle. This hierarchical model uses more vertices to describe areas of rough terrain but the

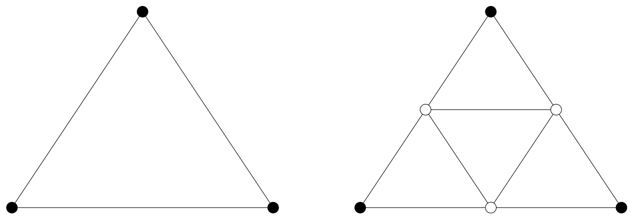

Figure 4.2: Subdividing a triangle into four children

surface of this model is not continuous. Discontinuities can appear along the edge shared by a triangle that is subdivided and a neighbouring triangle that is not subdivided.

De Floriani, Falcidieno, Nagy and Pienovi [DFFNP84] present a different hierarchical structure. A triangle is subdivided by inserting the vertex that will minimize the interpolation error of the remaining vertices within that triangle. Figure 4.3 shows the three child triangles formed by the edges connecting the new vertex to the three vertices of the parent triangle. This model includes more points in rough areas

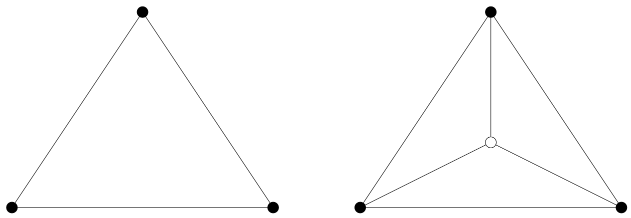

Figure 4.3: Subdividing a triangle into three children

and it forms a continuous surface at each level of the hierarchy. The major problem with this model is its tendency to form elongated triangles that can lead to inaccuracies in numerical interpolation [dBD95a]. In Section 4.2.1, this hierarchy is used as a method for building a TIN from a DEM.

These two structures are modified in an attempt to address the problems of surface discontinuities and elongated triangles.

Scarlatos and Pavlidis [SP90] aim to triangulate while looking more carefully at the terrain features that are being approximated. They also wish to avoid thin triangles that can appear in other hierarchical triangulations. The refinement technique they present inserts vertices and splits edges of a single triangle to approximate pit or peak points and ridge or channel lines. Scarlatos and Pavlidis compare their technique to the hierarchical structure presented by De Floriani, Falcidieno, Nagy and Pienovi. They find a substantial decrease in the number of thin triangles because their technique allows edge splitting. Their technique also results in a decrease in the number of levels in the hierarchy required to achieve a given degree of approximation because each triangle can be subdivided into more than three children.

De Floriani and Puppo [DFP92] refine each triangle by inserting new vertices along the edges of the parent triangle and then inserting vertices in the interior. These new vertices are connected in a local Delaunay triangulation. Elongated triangles are avoided but the union of the triangles in these local triangulations do not necessarily form a global Delaunay triangulation. The surface is continuous at each level of the hierarchy but discontinuities may appear if areas from different levels of the hierarchy are combined.

The triangulation hierarchy presented with Kirkpatrick's optimal search algorithm was modified by de Berg and Dobrindt [dBD95a, dBD95b] to be used for polyhedral terrains or TINs. They present a method for displaying terrains combining areas with different levels of detail. The details of our implementation of their method are described in Section 4.5.

The other hierarchical TINs presented above replace a single triangle with a set of triangles. These hierarchies have the advantage of a tree structure but the disadvantage of either producing skinny triangles or surface discontinuities. The hierarchical TIN presented by de Berg and Dobrindt replaces a group of triangles with another group of triangles. This hierarchy is not a tree but a directed acyclic graph which makes it more difficult to combine parts. But it has the advantage of allowing the use of a global Delaunay triangulation at every level.

An importance value is assigned to each vertex so that peaks, passes, pits and other points important to the shape of the terrain would not be deleted. Vertices are then selected and removed in order of increasing importance. Their method also allows the users to select a set of "fixed" vertices that are never removed.

4.2.1 Measuring Vertex Importance

De Berg and Dobrindt use a method for assigning an importance value to each vertex mentioned by Lee [Lee91]. Lee reviews four methods for building TINs from DEMs. These methods are summarized in this section.

The Skeleton method [FL79] uses peaks, pits, passes, ridges and valleys to form the skeleton of the TIN. A 3×3 point window is used to define peak and pit points. The center point of a 3×3 point window is defined to be a peak if it is a relative local maximum. Similarly, the center point is defined to be a pit if it is a relative local minimum. A 2×2 point window is used to define ridge and valley points. A point appears in four 2×2 windows. It is defined to be a ridge point if it is never the minimum point of these four windows. Similarly, a point is defined to be a valley point if it is never the maximum point. Ridge and valley lines are formed by connecting ridge and valley points. Next, the Douglas-Peucker algorithm for line simplification described in Section 3.1.1 is used to reduce the number of points needed to describe these lines. Support points are then added to improve differences in elevation. This complex method was excluded from Lee's comparison because it required too many tolerances.

The Filter method [CG89] discards points that can be closely interpolated by the points in its 3×3 grid neighbourhood. Since each point is visited only once this method is faster than the other methods in Lee's review. The disadvantage of this method is that only local information is used to select the points.

The Hierarchy method [DFFNP84] recursively subdivides each triangle by inserting the point with the largest difference between the original and interpolated elevations. This method does not use a Delaunay triangulation and tends to produce long and thin triangles but it has the advantage of a hierarchical data structure.

The Drop Heuristic method [Lee89] iteratively discards the point which will cause the least difference in elevation when dropped. The advantage of this method is that it uses the Delaunay neighbours rather than the grid neighbours of a vertex to find its importance in a global context. Unfortunately, this method is susceptible to **drift**; small errors introduced by discarding points can accumulate into large errors.

The Filter method, the Hierarchy method and the Drop Heuristic method each have two possible stopping conditions. Vertices can be inserted or deleted until either the TIN reaches a pre-set number of points or the TIN reaches a pre-set tolerance of elevation error.

The Drop Heuristic method can be considered a TIN simplification method instead of a method for building a TIN from a DEM because it begins by connecting the entire DEM into a TIN. The Hierarchy method can also be considered a TIN simplification method because it has no constraints on the distribution of its input data. The Skeleton and the Filter methods are not TIN simplification methods because they require gridded data as input but they are of interest to us because of their different definitions of important points.

4.2.2 Removing Vertices

Kirkpatrick simplifies the levels in his triangular subdivision hierarchy by removing vertices and retriangulating their neighbourhoods. In de Berg and Dobrindt's modification, these two tasks must be completed while retaining a Delaunay triangulation.

Midtbø [Mid94] studies three algorithms for removing vertices from a Delaunay triangulation. He describes a simple algorithm for retriangulating the neighbourhood of a deleted vertex. This algorithm uses pairs of edges along the hull of the neighbourhood to form triangles. The radius of the circumcircle of each potential triangle is calculated and an edge is added to complete the triangle with the smallest radius. This process is repeated until the neighbourhood is completely retriangulated. The simple algorithm retriangulates a neighbourhood with d vertices with a worst case running time of $O(d^2)$ while the other two algorithms he describes have worst case running times close to $O(d \log d)$. But the average d is small enough that the actual running times for the simple algorithm are faster than the running times for the other algorithms he tested.

4.3 Delete Simplification Methods

We wish to simplify a TIN so the hierarchy described by Kirkpatrick and modified by de Berg and Dobrindt can be used. We describe a set of **delete simplification methods** that meet this constraint. The simplification is formed in two steps. The first step selects a set of vertices to be deleted. The second step deletes these vertices and retriangulates their neighbourhoods. These two steps can be repeated for further simplification.

Since our method of deleting vertices places constraints on how these vertices are selected, we will present the two steps of the simplification in the reverse order. Section 4.3.1 provides the details on how selected vertices are deleted from the TIN and Section 4.3.2 provides the details on how vertices are selected.

4.3.1 Deleting Vertices and Retriangulating Neighbourhoods

If a vertex is deleted from a triangulation, its neighbourhood must be retriangulated. A **local retriangulation** is a retriangulation that does not affect the triangles outside of the neighbourhood. A local retriangulation is always possible, but there is no guarantee that there will be any local retriangulation that can satisfy a given triangulation criteria.

If a vertex is deleted from a Delaunay triangulation, the retriangulation required to form the Delaunay triangulation of the remaining vertices is a local retriangulation [Mid94]. The triangles outside of the neighbourhood are unaffected so the retriangulation must be local. The proof is below.

Notation

$$\begin{aligned} V &= \text{set of vertices including } v_{del} \\ t &= \text{any triangle on } V \\ v_{del} &= \text{deleted vertex} \end{aligned}$$

Proof

$$\begin{aligned} t \text{ is Delaunay on } V &\Leftrightarrow \forall v_i \in V | v_i \text{ outside circumcircle of } t \\ t \text{ is Delaunay on } V &\Leftrightarrow t \in \text{Delaunay triangulation of } V \\ & \\ & \quad \text{if } t \text{ does not have } v_{del} \text{ as a vertex then} \\ & \quad t \in \text{Delaunay triangulation of } V \\ &\Rightarrow t \text{ is Delaunay on } V \\ &\Rightarrow \forall v_i \in V | v_i \text{ outside circumcircle of } t \\ &\Rightarrow \forall v_i \in V - \{v_{del}\} | v_i \text{ outside circumcircle of } t \\ &\Rightarrow t \text{ is Delaunay on } V - \{v_{del}\} \\ &\Rightarrow t \in \text{Delaunay triangulation of } V - \{v_{del}\} \end{aligned}$$

We will be using the simple fast retriangulating algorithm described by Midtbø [Mid94].

4.3.2 Selecting Vertices

If the retriangulation is local and no two vertices in the set of vertices to be deleted are adjacent, each delete and retriangulate operation is independent. This allows us to delete a set of vertices without worrying about the interactions among pairs of delete and retriangulate operations. Our retriangulating algorithm may work on some intersecting neighbourhoods of adjacent vertices but will work on all neighbourhoods of independent vertices.

Each of the following methods uses a different importance measure to select vertices. All three of the measures are local; they only use the neighbourhood of the vertex to measure its importance.

The only criteria for selecting vertices in Kirkpatrick's [Kir83] original triangulation hierarchy is the degree of the vertex. We name this method the **degree delete simplification method**. We deleted a set of independent vertices with degree equal or less than 11. This simplification method only uses a binary measure for each vertex.

This method does not use elevation information to select the vertices. We do not expect this method will form accurate simplifications but it is included for comparison purposes.

The next two methods use a "flatness" measure to select vertices with low importance.

The **volume delete simplification method** uses the difference in volume as the "flatness" measure. The absolute difference in the volume under the neighbourhood of the vertex with and without the vertex is computed. If the absolute difference is small, deleting that vertex will have a relatively small effect on the model.

This method is similar to the algorithms evaluated by Lee [Lee91] and Kumler [Kum94]. Those algorithms built TINs from DEMs and used the absolute difference in elevation at grid locations. By calculating the difference in volume, this method considers the difference in elevation at all points and not just the differences at vertices of a TIN or DEM.

The **normal delete simplification method** uses the normal vectors of the triangular facets to compute the "flatness" measure. The unit normal of each triangle in the neighbourhood of the vertex is computed. Another unit normal is formed by the average of these normals. The mean of the dot products of each individual unit normal and the average unit normal forms the "flatness" measure. Equal unit normals have a dot product of one. If the mean of the dot products is near one, the terrain around that vertex is flat and deleting that vertex will have a relatively small effect on the model.

This "flatness" measure is harder to interpret than the measure used by the volume method. We know the best "flatness" value is one but we do not know what range of values to expect in our experiments.

4.4 Insert Simplification

We wanted to test the delete simplification methods against another type of simplification method.

Garland and Heckbert [GH95] examine the greedy insertion algorithm which we rename the insert simplification method. The **insert simplification method** builds a simplification of a TIN by repeatedly inserting the vertex with the largest associated error. This error is measured as the height between the current TIN and the vertex. The process is continued until the desired number of vertices is reached. The initial TIN is the flat plane defined by four vertices at elevation zero.

This simplification method does not guarantee that the hierarchy described by de Berg and Dobrindt can be formed.

4.5 Rendering Terrain with Varying Levels of Detail

We implemented a program to render a terrain with varying levels of detail based on de Berg and Dobrindt's [dBD95a, dBD95b] hierarchy for TINs. Each of the delete simplification methods in Section 4.3 generates a sequence of terrain models which can be used to form the hierarchy. This hierarchy can be used to form a terrain using less data, resulting in faster rendering. It can also be used to form a terrain with non-uniform levels of simplification. These models and the links between them are stored in the data structures explained below.

4.5.1 Data Structures

As the TIN is simplified, vertices are deleted and their star-shaped neighbourhoods are retriangulated. A simplified TIN can be refined by reversing this process. These refinements are stored by the hierarchy. Each triangle belongs to a group of triangles that form the retriangulated neighbourhood of a deleted vertex. These triangles are refined by inserting the deleted vertex and restoring the triangulation. In our implementation, each triangle has a link to its **group** polygon and each polygon has links to the its unrefined **reject triangles**, the **accept vertex** that is used to refine these triangles and its refined **accept triangles**. The triangle links are shown in Figure 4.4 and the polygon links are shown in Figure 4.5, Figure 4.6 and Figure 4.7.

We implemented a program to render terrain with varying levels of detail in C++. The header files for the four classes are in the sections below.

Vertex

```
class _vertex {
 public:
```

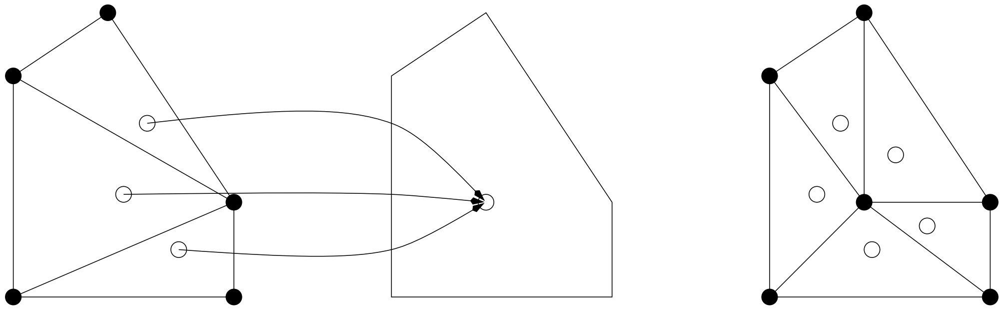

Figure 4.4: Links from triangles to their group polygon.

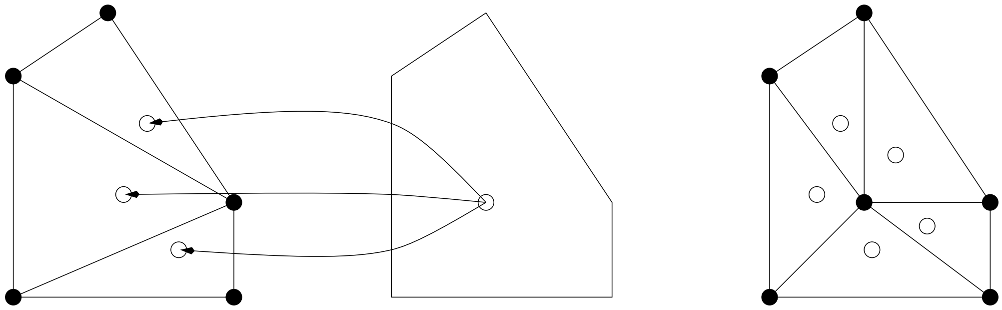

Figure 4.5: Links from the polygon to its reject triangles.

```
/* constructor */
  _vertex();

/* access functions */
  void set_v(float v0, float v1, float v2);
  void set_level(int level0);

/* returns true
   if distance from v0 to implicit vertex
   less than
   product of level of implicit vertex and d0 */
  int test_accept(const _vertex &v0, float d0) const;

/* render triangle specified by implicit vertex, v1 and v2 */
  void draw(const _vertex &v1, const _vertex &v2) const;
};
```

Triangle

```
class _triangle {
 public:
/* constructor */
  _triangle();

/* access function */
  void set(_vertex *v0, _vertex *v1, _vertex *v2,
```

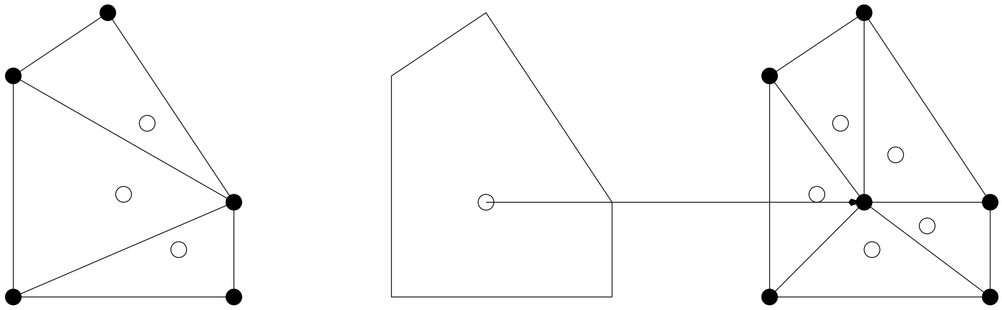

Figure 4.6: Link from the polygon to its accept vertex.

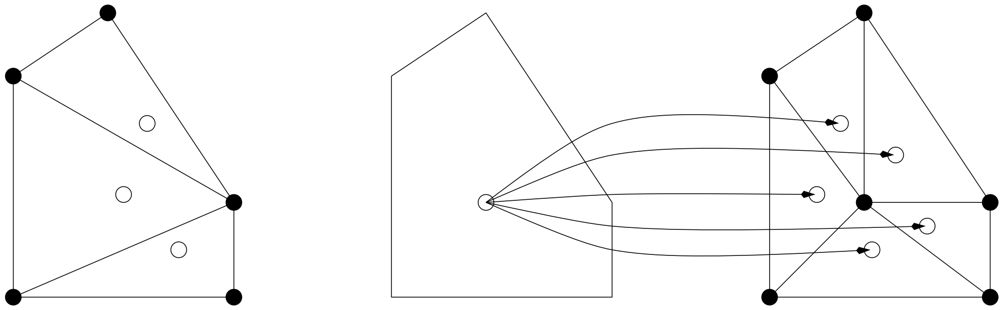

Figure 4.7: Links from the polygon to its accept triangles.

```
   _polygon *group0,
   int queue0);
  void set_queue(int queue0);
  int get_queue() const;
  _polygon *get_group() const;

/* render triangle using _vertex::render(_vertex, _vertex) */
  int draw() const;
};
```

Polygon

```
class _polygon {
 public:
/* constructor */
  _polygon();

/* access functions */
  void set_accept(_vertex *reject0);
  int add_reject(_triangle *tr0);
  int add_accept(_triangle *ta0);

/* if all reject triangles in queue and reject vertex accepted
   then remove all reject triangle and insert all accept triangles
   else remove and render any reject triangles in queue */
  int draw(const _vertex &v0, float d0) const;
```

```
};
```

Level

```
class _level {
 public:
/* constructor */
  _level();

/* access functions */
  int set_size(int nt0);
  int set_triangle(int i,
   _vertex *v0, _vertex *v1, _vertex *v2,
   _polygon *g0,
   int q0) const;
  void set_queue(int q0) const;

/* for each triangle in the queue
   if the triangle has no associated polygon
   then render the triangle
   else process the polygon using _polygon::draw(_vertex, float) */
  int draw(const _vertex &v0, float d0) const;
};
```

4.5.2 Rendering

Each time the TIN is simplified, another level in the hierarchy is formed. The original TIN forms the lowest level and each simplification forms a higher level. The level of a vertex is equal to the highest level of hierarchy that still includes that vertex.

Initially, each triangle in the highest level of the hierarchy is inserted in a queue. To render the terrain, each triangle in the queue is processed. A triangle is processed by either rendering the triangle or replacing the triangle and the other reject triangles in its polygon with the set of accept triangles. The union of the enqueued triangles and the rendered triangles will always cover the terrain.

If all the reject triangles of a polygon are in the queue, the accept vertex can be tested. In our implementation, the vertex is accepted if the distance between the accept vertex and a user-defined center of detail is less than the product of the level of the accept vertex and a user-defined detail dropoff distance. This test includes vertices from the lower levels of the hierarchy close to the centre of detail while allowing only vertices from higher levels further away. Other tests based on local polygon information can also be used.

Three of the four possible cases are illustrated in Figure 4.8, Figure 4.9 and Figure 4.10. The fourth case occurs when the enqueued triangle is in the lowest level of the hierarchy. No more refinement can be done so it is rendered. Enqueued triangles have a grey fill style and rendered triangles have a grid fill style. In every case, the area covered before and after the triangle's group polygon is processed is the same.

The results of this rendering method are displayed in three figures. Figure 4.11 shows the initial terrain model with 20002 triangular facets. It has a high level of detail, even in the background where it is not needed. This terrain model is repeatedly simplified to form the model shown in Figure 4.12. This simplified model has a low level of detail. The hierarchy is used to form the terrain model in Figure 4.13. It combines areas of high detail in the foreground with areas of low detail in the background.

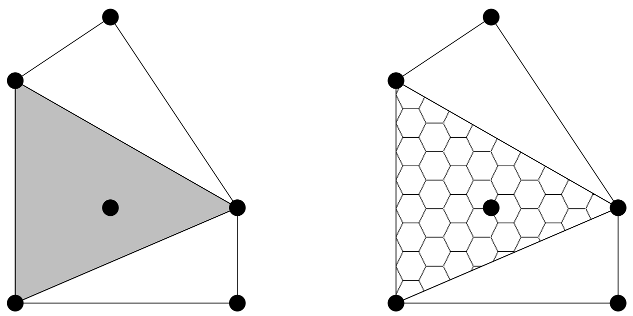

Figure 4.8: Only one of the three triangle in this polygon is in the queue. It is removed from the queue and rendered.

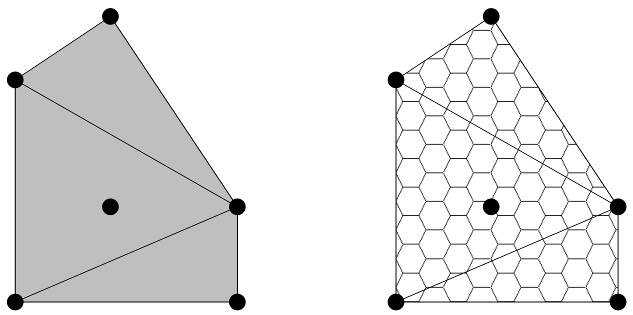

Figure 4.9: All three of the triangles in this polygon are in the queue but the reject vertex is not accepted. All three triangles are removed from the queue and rendered.

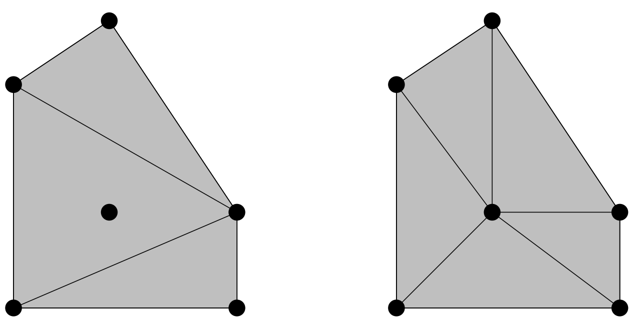

Figure 4.10: All three of the triangles in this polygon are in the queue and the reject vertex is accepted. All three triangles are removed from the queue and five new triangles are inserted into the queue.

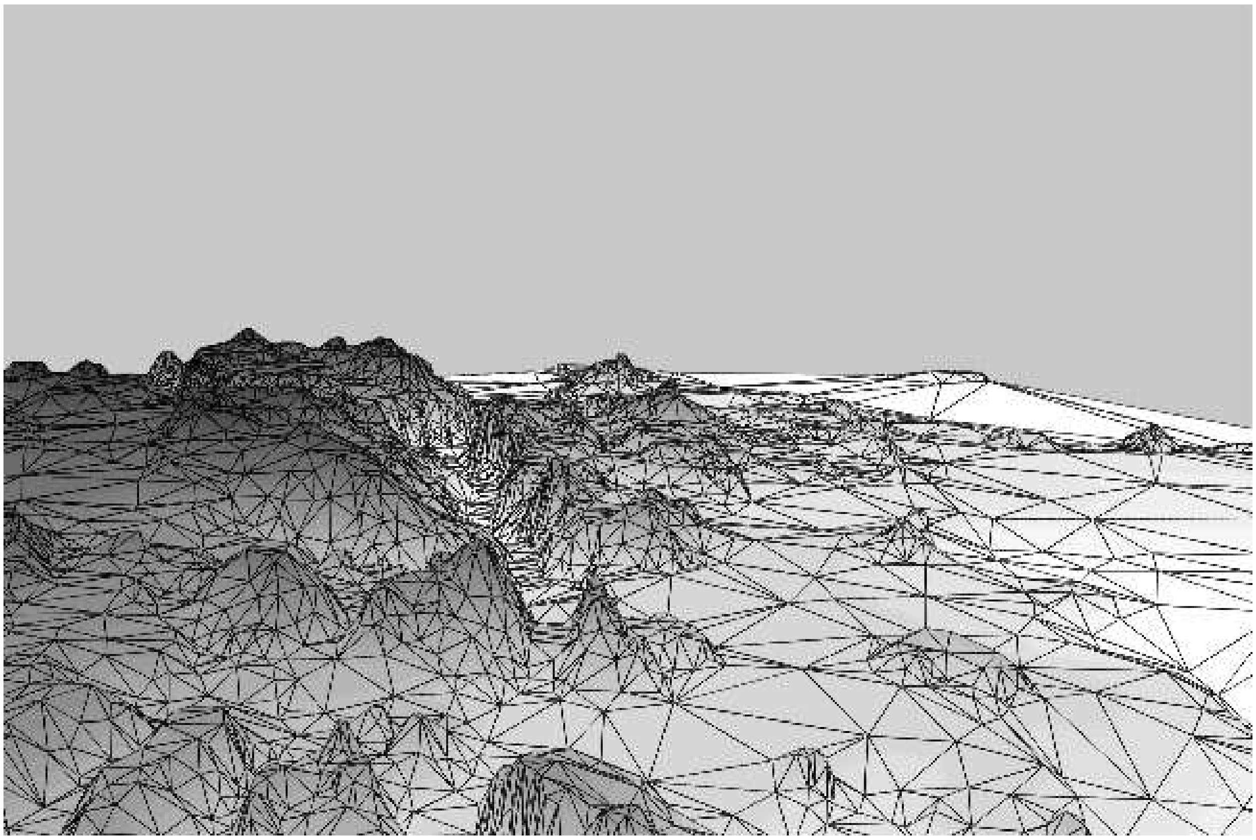

Figure 4.11: The initial terrain model with 20002 triangular facets.

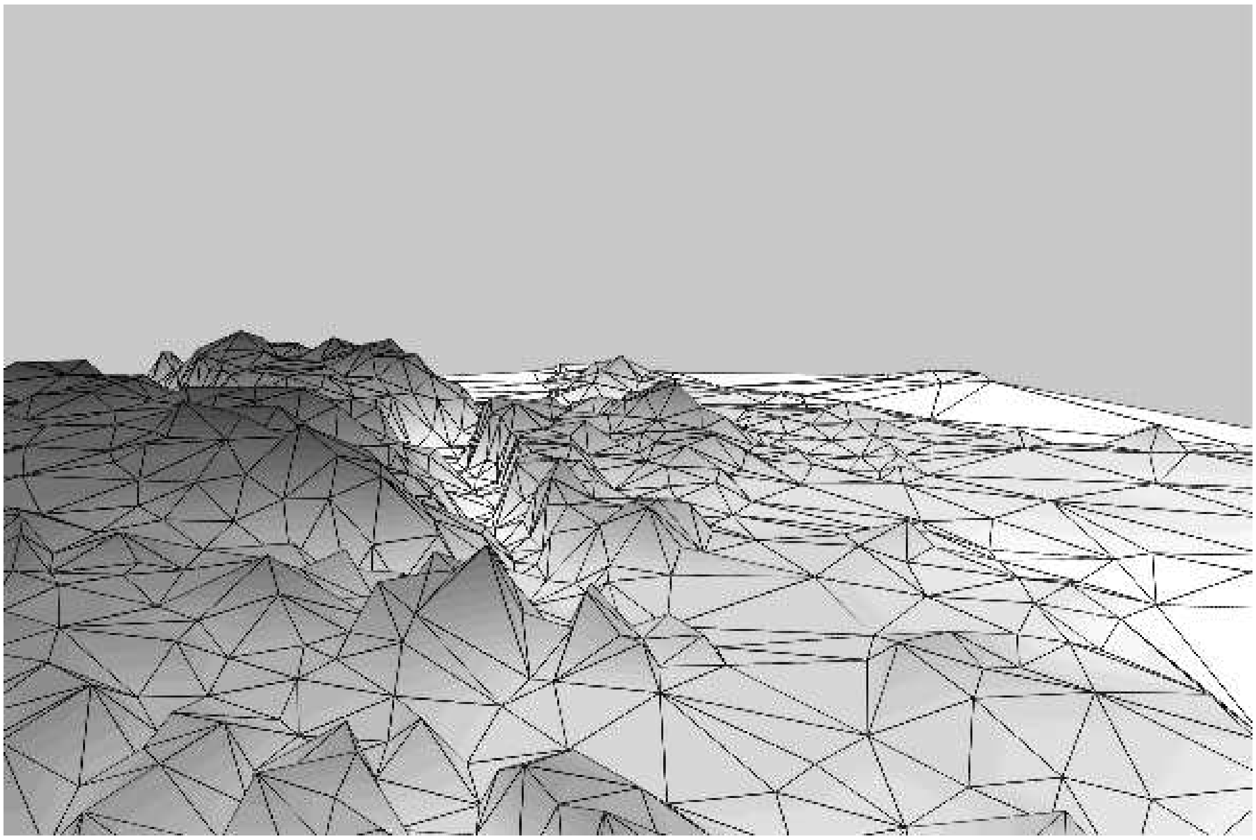

Figure 4.12: A simplified model with 3360 triangular facets.

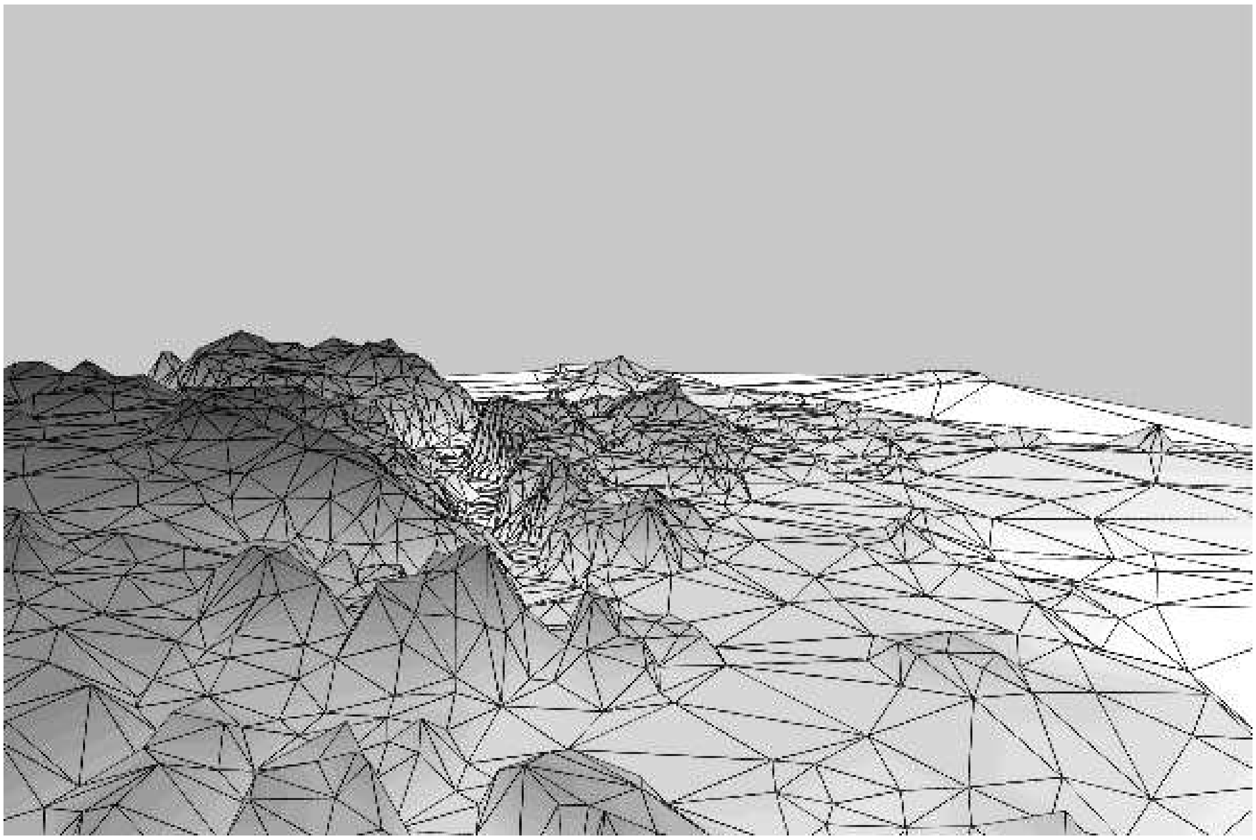

Figure 4.13: A terrain model combining regions of varying levels of detail from the hierarchy of the initial terrain model and its simplifications. It has 5280 triangular facets.

Chapter 5

Surface Accuracy

To evaluate the effectiveness of these simplification methods, we examined their accuracy and their speed. Lee [Lee91] and Kumler [Kum94] evaluate surface accuracy by computing the differences in elevation between the original surface and the approximate surface at a set of test points. In our surface evaluations, we used measures of curvature as well as elevation.

Lee [Lee91] reviews and evaluates four methods for building TINs from DEMs. Each approximate TIN surface is compared to the original DEM surface by computing the elevation difference at each DEM point. In addition to measuring the mean and standard deviation of the differences, Lee analyses the spatial pattern of the differences. He uses Moran's index, a spatial autocorrelation coefficient, to measure how clustered or how randomly the differences are distributed. Tests for the significance of randomization and normality are also used. He finds that none of the four methods he reviews has a clear overall advantage. It is interesting to note that the Drop Heuristic, a TIN building method that is closely related to the delete simplification methods, has the best Moran's index values.

Kumler [Kum94] compares eight different TIN and DEM construction methods. He uses DLGs as the source data. Each approximate DTM surface is compared to the original terrain by computing the elevation differences at points in three test sets. The first test set is the set of DLG spot heights included in each DLG data file. Kumler created the other two sets of test points for his comparison. The second set is created by using a set of coarse grid points "jiggled" so that each point falls on a DLG contour line. The third set is created by obtaining a set of dispersed, irregularly-distributed points and visually estimating the elevations with approximate interpolation between contour lines. Although he believes that TINs "look better" than DEMs, he finds that DEMs estimate spot heights better than TINs. He suspects that TINs may "look better" because TINs are better at modelling derivative statistics such as slope and aspect.

In attempting to find a basis for quantitative comparison of landscapes, Evans [Eva72] describes the field of general geomorphometry. Geomorphometry is the measurement and analysis of land-form characteristics applicable to any continuous rough surface. Evans calls for simple variables that are standardized for comparison, integrated and statistically stable. He also prefers to use point measures rather than measures defined in relation to an arbitrary area. He presents five basic variables for general geomorphometry.

- altitude, z
- gradient, z'_v
- aspect, z'_h
- vertical convexity, z''_v
- horizontal convexity, z''_h

Lee and Kumler only use altitude as a basis for comparing surfaces. The other four variables, all derivative statistics, are ignored.

We wanted to use both elevation and a derivative statistic to evaluate the accuracy of the simplification methods. We decided that mean curvature was a good derivative statistic because it is a meaningful characteristic value of the shape operator reviewed in Section 5.2. Elevation can be interpolated directly from a TIN surface but mean curvature is only defined at a point on a surface with continuous second partial derivatives. To overcome this obstacle, we fit a degree three B-spline surface to the surface of the TIN.

Evaluating the surface accuracy consisted of four steps. We first interpolated a set of gridded test points from the original TIN and the simplified TIN by locating the triangle that contains the xy point and using the three vertices of that triangle to linearly interpolate the z coordinate. We then fitted a B-spline surface to both these interpolated sets. At each test point, we calculated the elevation and mean curvature. Finally, we computed the correlation coefficients of the elevations and curvatures.

The following sections provide a background to these steps. Section 5.1 describes parametric curves and parametric surfaces. Section 5.2 reviews the definition and calculation of curvature and Section 5.3 reviews the use of the correlation coefficient. Kumler's comparison notes the importance of testing simplification methods over a wide range of study areas [Kum94]. Our approach to choosing study areas is outlined in Section 5.4.

5.1 Parametric Curves and Surfaces

This section describes a representation of curves and surfaces [FvDFH90]. We start by reviewing parametric curves and then extend the curve definition to parametric surfaces. We conclude by describing the properties of the specific surface we used, the B-spline surface.

5.1.1 Parametric Curves

Polylines are first-degree, piecewise linear approximations to curves. Large numbers of points may be needed to achieve reasonable accuracy. Parametric curves use higher-degree functions as a representation of curves. They are still only approximations, but they use less storage than linear functions.

Cubic polynomials are most often used because lower-degree polynomials give too little flexibility in controlling the shape of the curve and higher-degree polynomials can introduce unwanted wiggles and also require more computation. The cubic polynomials that define a curve segment $Q(t) = \begin{bmatrix} x(t) & y(t) & z(t) \end{bmatrix}$ are of the following form.

$$\begin{aligned} x(t) &= a_x t^3 + b_x t^2 + c_x t + d_x \\ y(t) &= a_y t^3 + b_y t^2 + c_y t + d_y \\ z(t) &= a_z t^3 + b_z t^2 + c_z t + d_z, \ 0 \le t \le 1 \end{aligned}$$

This is rewritten in matrix form.

$$\begin{aligned} T &= \begin{bmatrix} t^3 & t^2 & t & 1 \end{bmatrix} \\ C &= \begin{bmatrix} a_x & a_y & a_z \\ b_x & b_y & b_z \\ c_x & c_y & c_z \\ d_x & d_y & d_z \end{bmatrix} \\ Q(t) &= T \cdot C \end{aligned}$$

A curve segment is defined by constraints on endpoints, tangent vectors and continuity between adjoining curve segments. If two curve segments join together, the curve has G^0 geometric continuity. If the directions (but not necessarily the magnitudes) of the two segments' tangent vectors are equal at a join point, the curve has G^1 geometric continuity. If the tangent vectors of two curve segments are equal at the segments' join point, the curve has first-degree continuity and is said to be C^1 continuous. If the direction and magnitude of the $d^n/dt^n[Q(t)]$ through to the nth derivative are equal at the join point, the join is called C^n continuous.

The coefficient matrix C can be rewritten as $C = M \cdot G$ where M is the basis matrix and G is the geometry vector. The coefficients of geometry vector, G_1, G_2, G_3 and G_4, are also vectors.

$$
\begin{aligned}
Q(t) &= \begin{bmatrix} x(t) & y(t) & z(t) \end{bmatrix} \\
&= T \cdot C \\
&= T \cdot M \cdot G \\
&= \begin{bmatrix} t^3 & t^2 & t & 1 \end{bmatrix} \begin{bmatrix} m_{11} & m_{12} & m_{13} & m_{14} \\ m_{21} & m_{22} & m_{23} & m_{24} \\ m_{31} & m_{32} & m_{33} & m_{34} \\ m_{41} & m_{42} & m_{43} & m_{44} \end{bmatrix} \begin{bmatrix} G_1 \\ G_2 \\ G_3 \\ G_4 \end{bmatrix}
\end{aligned}
$$

B-spline Curves

Cubic B-splines are defined by four control points. They have C^1 and C^2 continuity and come close but generally do not interpolate their control points. The continuity conditions are achieved by sharing control points between segments. A spline with $m+1$ control points $P_0,\ P_1,\ \ldots,\ P_m$ has $m-2$ curve segments $Q_3,\ Q_4,\ \ldots,\ Q_m$. Curve segment $Q_i(t)$ with a curve parameter t is defined between $t_i = i-3$ and $t_{i+1} = i-2$ and depends on points $P_{i-3},\ P_{i-2},\ P_{i-1}$ and P_i. The index i ranges from 3 to m.

$$
\begin{aligned}
T_i &= \begin{bmatrix} (t-t_i)^3 & (t-t_i)^2 & (t-t_i) & 1 \end{bmatrix} \\
M_{Bs} &= \frac{1}{6} \begin{bmatrix} -1 & 3 & -3 & 1 \\ 3 & -6 & 3 & 0 \\ -3 & 0 & 3 & 0 \\ 1 & 4 & 1 & 0 \end{bmatrix} \\
G_{Bs_i} &= \begin{bmatrix} P_{i-3} \\ P_{i-2} \\ P_{i-1} \\ P_i \end{bmatrix} \\
Q_i(t) &= T_i \cdot M_{Bs} \cdot G_{Bs_i},\ t_i \le t \le t_{i+1}
\end{aligned}
$$

5.1.2 Parametric Surfaces

Parametric surfaces are a generalization of parametric curves. By replacing the constant vector coefficients of the geometry matrix G with parametric curves, a surface of two variables is formed.

$$
\begin{aligned}
S_i &= \begin{bmatrix} (s-s_i)^3 & (s-s_i)^2 & (s-s_i) & 1 \end{bmatrix} \\
G &= \begin{bmatrix} g_{11} & g_{12} & g_{13} & g_{14} \\ g_{21} & g_{22} & g_{23} & g_{24} \\ g_{31} & g_{32} & g_{33} & g_{34} \\ g_{41} & g_{42} & g_{43} & g_{44} \end{bmatrix} \\
T_i &= \begin{bmatrix} (t-t_i)^3 & (t-t_i)^2 & (t-t_i) & 1 \end{bmatrix} \\
Q(s,t) &= S \cdot M \cdot G \cdot M^T \cdot T^T
\end{aligned}
$$

B-spline Surfaces

Just as the degree 3 B-spline curve has C_1 and C_2 continuity, the B-spline surface has C_1 and C_2 continuity. This means its first and second partial derivatives will be continuous. Other parametric surfaces interpolate their control points but they cannot easily provide the required level of continuity.

The one variable in fitting the a B-spline to the grid of interpolated points is the grid spacing of the points. Using a small grid spacing and a large number of points will results in a surface that has many flat areas corresponding to the planar triangular facets. Using a large grid spacing will result in triangular facets in the TIN that are not represented by any point in the grid.

5.2 Computing Curvature

This section reviews the definition and calculation of curvature [O'N66]. We used mean curvature as our derivative statistic to measure surface accuracy.

The shape of a curve can be measured by its curvature and torsion functions. The **curvature** vector of a point on a curve represents the turning of the curve at that point. The **torsion** vector of a point on a curve represents the twisting of the curve at that point. The analogous measurement of a surface M is its shape operator S. The **shape operator** of a point on a surface represents the bending of the surface at that point. Surfaces with the same shape operator are congruent.

The shape operator $S_p(v)$ at a point p on the surface is defined for all vectors v tangent to the surface at that point. At each point p on the surface there are two unit normals U and $-U$ and two corresponding shape operators S_p and $-S_p$.

The normal curvature $k(u)$ in the u direction is the dot product of the shape operator and the vector u.

$$k(u) = S(u) \cdot u$$

If the normal curvature is positive the surface bends toward the normal. If the normal curvature is negative the surface bends away from the normal. The normal curvature of a sphere with a radius r and an outward oriented normal is $-1/r$ for any vector u at any point.

The minimum and maximum values of the normal curvature at a point are called the principal curvatures k_1 and k_2. The vectors in which these values occur are called the principal vectors. These principal vectors are orthogonal.

The Gaussian curvature K is the product of the two principal curvatures. The formula for Gaussian curvature is calculated from constants formed from the first and second partial derivatives of the surface M with parameters s and t at that point.

Constants

$$\begin{aligned}
E &= M_s \cdot M_s \\
F &= M_s \cdot M_t \\
G &= M_t \cdot M_t \\
U &= (M_s \times M_t)/W \\
W &= \|M_s \times M_t\| \\
l &= U \cdot M_{s^2} \\
m &= U \cdot M_{st} \\
n &= U \cdot M_{t^2}
\end{aligned}$$

$$K = \frac{ln - m^2}{EF - F^2}$$

The mean curvature is the average of the two principal curvatures. The formula for mean curvature H at a point is defined by the same constants that were used to compute the Gaussian curvature.

$$H = \frac{Gl + En - 2Fm}{2(EG - F^2)}$$

These two formulae do not require the calculation of the principal curvatures.

It is infeasible to use the shape operator as a measure of surface accuracy because it is a operator, not a single number. But both the Gaussian curvature and the mean curvature are meaningful characteristic values of the shape operator. Gaussian curvature is independent of the orientation of the normal but mean curvature is dependent. Since we wanted a derivative statistic that was dependent on the direction of the normal and since we could easily orient the surface, we chose to use mean curvature.

5.2.1 Computing Curvature on a B-spline Surface

We can easily calculate the first and second partial derivatives at each interpolated test point on the B-spline surface. We can then substitute these values into the above formula to calculate the mean curvature. We calculate a first partial derivative as an example below.

$$\begin{aligned}
Q(s,t) &= S \cdot M_{Bs} \cdot G \cdot {M_{Bs}}^T \cdot T^T \\
&= \begin{bmatrix} s^3 & s^2 & s & 1 \end{bmatrix} \cdot M_{Bs} \cdot G \cdot {M_{Bs}}^T \cdot \begin{bmatrix} t^3 & t^2 & t & 1 \end{bmatrix}^T \\
Q_s(s,t) &= \begin{bmatrix} 3s^2 & 2s & 1 & 0 \end{bmatrix} \cdot M_{Bs} \cdot G \cdot {M_{Bs}}^T \cdot \begin{bmatrix} t^3 & t^2 & t & 1 \end{bmatrix}^T \\
Q_s(0,0) &= \begin{bmatrix} 0 & 0 & 1 & 0 \end{bmatrix} \cdot M_{Bs} \cdot G \cdot {M_{Bs}}^T \cdot \begin{bmatrix} 0 & 0 & 0 & 1 \end{bmatrix}^T \\
&= \frac{1}{6}\begin{bmatrix} -3 & 0 & 3 & 0 \end{bmatrix} \cdot G \cdot \frac{1}{6}\begin{bmatrix} 1 & 4 & 1 & 0 \end{bmatrix}^T
\end{aligned}$$

5.3 Computing Correlation

To evaluate surface accuracy, we computed the linear regression correlation coefficients of the elevations and the mean curvatures at test points on the B-spline surfaces fitted to the original and simplified TINs. The correlation coefficient r [MM89] measures the strength of the linear association between a set of n observations of two variables x and y with means $\overline{x}$ and $\overline{y}$ and standard deviations s_x and s_y.

$$r = \frac{1}{n-1} \sum_i \left(\frac{x_i - \overline{x}}{s_x}\right)\left(\frac{y_i - \overline{y}}{s_y}\right)$$

The coefficient is a unitless number between -1 and 1. A positive value indicates a positive association and a negative value indicates a negative association. A value of -1 or 1 indicates a perfect positive or negative linear association.

The square of the correlation coefficient r is the fraction of the variation of x that is explained by the least squares regression of y on x. Correlation is a symmetrical relationship so this statement is also true if x and y are interchanged. The slope b of the regression line of y on x can be expressed in terms of the two standard deviations and the correlation coefficient.

$$b = r\frac{s_y}{s_x}$$

5.3.1 Using Correlation to Evaluate Surface Accuracy

The correlation coefficient only measures linear association. Identical surfaces should not only have a perfect linear association but should also have a regression line passing through the origin with slope of one. By definition, the regression line passes through the point $(\overline{x}, \overline{y})$. The standard deviations s_x and s_y must be

Name		Hammond	Fenneman
ashby	Ashby Gap, VA	C5	5
belle	Belle Creek, MT-WY	C4	13
camas	Camas, WA-OR	B3 low	24
crater	Crater Lake West, OR	C6	23
eloise	Eloise, FL	A1	3
entriken	Entriken, PA	C5	6
hogans	Hogansburg, NY	B2	7
honolulu	Honolulu, HI	B6 low	
jabez	Jabez, KY	C3	11
jackson	Jackson, SC	A2	3
newbrit	New Britain, CO	B4 low/C4 low	9
tiefort	Tiefort Mountains, CA	B5 low	22

Table 5.1: Study Areas

equal for a regression line with perfect positive association to have a slope of one. The means $\overline{x}$ and $\overline{y}$ must be equal for a regression line with slope one to pass through the origin. While checking the correlation coefficients, we also checked the difference in means and standard deviations to ensure that these additional conditions were almost met.

5.4 Choosing Study Areas

We wanted to study the effectiveness of the delete simplification methods over a large range of terrain types.

Fenneman [Fen28] describes the physiographic divisions of the United States. He lists 25 homogeneous provinces, designated by the initial geological structure, the erosion process and the stage reached in the cycle of changes by erosion. This information captures the history of the terrain as well as the topography. These provinces extend into Canada and Mexico but are specific to North America.

Hammond [Ham64] prepares a map of the land form of the United States using indices of four properties. He chooses properties that describe the visual aspects and the land-use possibilities of the surface. The four properties are frequency of gentle slope, local relief, profile type and surface material. The combination of the indices of these four properties form a classification system that divides the United States into over 300 regions. These regions each fall into a class defined by the ninety-six combinations of the indices. Twenty-one of the ninety-six classes appear in significant quantity in the United States. This classification system can easily be extended to the land form of other countries with the possibility that new combinations of indices may appear.

Kumler [Kum94] uses these two terrain classification systems to select DLGs that represent the variety of terrain types found in the United States. His 25 study areas cover 18 of 25 Fenneman provinces and 20 of 21 Hammond classes. The 12 DEMs listed in Table 5.1 were created from a subset of these DLGs.

The 12 DEM grids were roughly 300×450 with a 30 meter grid spacing. A set of 15000 point TINs was formed by taking one random point from each 3×3 grid square. Each point was perturbed between 0.00 and 0.99 meters in both x and y coordinates to avoid degenerate cases in the Delaunay triangulation. We defined these TINs to be the original surfaces.

5.5 The Experiment

Each delete simplification method required two parameters. The **delete percentage** is the maximum percentage of vertices to delete at each iteration of a delete simplification. The **stopping percentage** is the percentage of vertices remaining in the TIN when the iterations stop. We tested the three delete simplification methods with delete percentages of 10% and 20% and a stopping percentage of 5% over the set of 12 TINs. We plotted the elevation and curvature correlation coefficients against the number of vertices for each TIN with a constant correlation range.

The insert simplification method was used for comparison purposes. It was implemented by Will Evans.

Chapter 6

Results

6.1 Speed

We examined the speed of the delete simplification methods by plotting the times required by the two steps in each method. The select time is the time required to select a percentage of vertices to be deleted. The delete time is the time required to delete these vertices and retriangulate their neighbourhoods. We plotted the mean select time and the mean delete time against the number of vertices for each delete simplification method. The mean times were used because we did not feel that the select and delete times depended on the type of terrain. The delete percentage was 20%.

6.1.1 Select Times

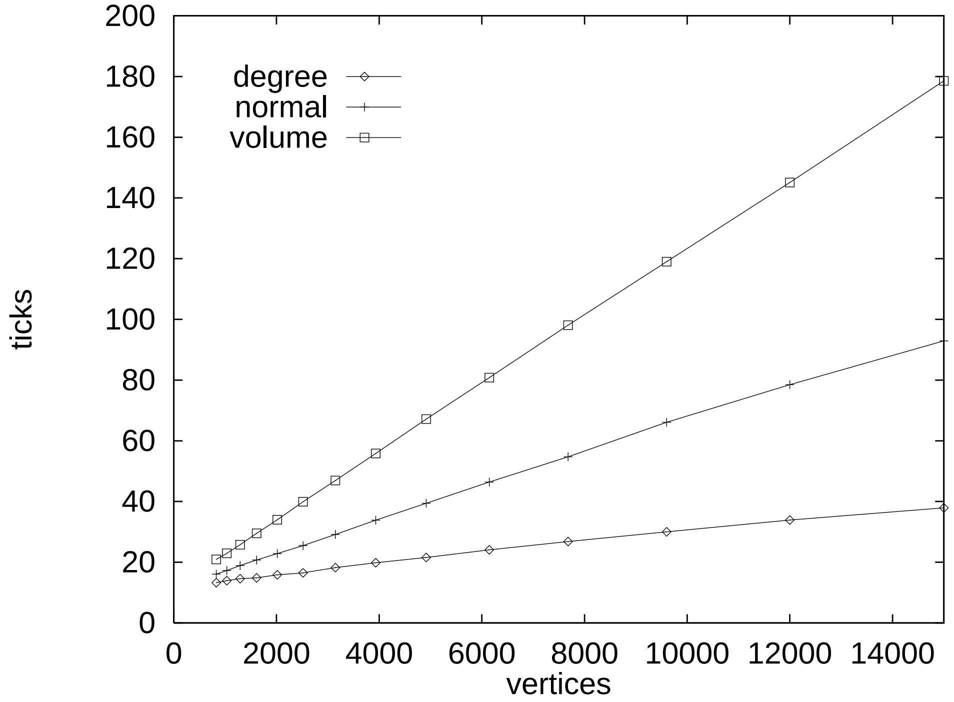

Figure 6.1: Time to select 20% of the vertices using the three delete methods

In Figure 6.1, the time to select 20% of n vertices appeared to be linear for all three methods.

The volume method and normal method select vertices in two steps. The first calculates a "flatness" value for each vertex. The second selects a set of non-adjacent vertices with the best "flatness" values. The size of the set is controlled either by specifying this size or by specifying a threshold "flatness" value.

If a threshold "flatness" value is used, each vertex is checked only once. If a vertex v is not adjacent to a selected vertex and the "flatness" value of v is better than the threshold, then v is selected. This method selects a set of vertices from a n point TIN in $O(n)$ running time, but the number of vertices with "flatness" values better than the threshold may vary with the type of terrain.

We decided to specify the size of the set of vertices to be selected so that the speed and accuracy of different delete simplification methods and different terrains could be easily compared. A fixed percentage of the current vertices was selected. A vertex is eligible if it has not been selected and it is not adjacent to a selected vertex. This method repeatedly selects the eligible vertex with the best "flatness" value until the specified number of vertices is selected.

To quickly find the vertex with the best "flatness" value, we used a heap. It takes $O(\log n)$ time to insert a single element into a heap of size n. Since we build the heap by repeatedly inserting elements, the total time to build the heap of n elements is be $O(n \log n)$. After building the heap, the vertex with the best "flatness" value can be found in $O(1)$ time. After finding the best vertex, it is deleted from the heap. The time to repair the heap after a single element is deleted is $O(\log n)$. Therefore, the total cost to find and delete a fixed percentage of vertices from the heap is $O(n \log n)$. Since both heap build and heap find and delete times are $O(n \log n)$, the select time must be $O(n \log n)$. But both the volume method and the normal method in Figure 6.1 appear to be $O(n)$ indicating that the constants associated with the $O(n)$ time to compute the "flatness" value of each vertex outweigh the constants associated with the $O(n \log n)$ time to order and select the vertices using the heap. Even though the time to select vertices appears linear, it is actually $O(n \log n)$.

In our experiments, the time to select vertices depends on the time to compute the "flatness" values rather than the time to order the "flatness" values. This means that the "flatness" value can affect both the speed and accuracy of the simplification methods. As the number of vertices is increased, the time to select vertices will eventually depend on the $O(n \log n)$ time to order the "flatness" values, not the $O(n)$ time to compute the "flatness" values. At this point, the "flatness" value will only affect the accuracy of the simplification method.

The degree method selected vertices by calculating the degree of each vertex and selecting vertices less than or equal to a fixed maximum degree. This algorithm is similar to the threshold method described above using the maximum degree as a threshold. Its time cost is $O(n)$.

6.1.2 Delete Times

The mean delete times appeared to be linear. The speed of the algorithm we used to retriangulate the neighbourhood of a deleted vertex depended on the degree of the vertex. The average degree of a vertex is less than 6 so the average time to delete a single vertex is $O(1)$ and the time to delete a fixed percentage of vertices is $O(n)$. This is supported by Figure 6.2.

The mean delete times were lower than the mean select times. This indicates that significant improvements to the select speed would result in significant improvements to the overall speed of the simplification method.

6.2 Accuracy

We examined the accuracy of the simplification methods by plotting the correlation coefficients for both elevation and mean curvature against the number of vertices.

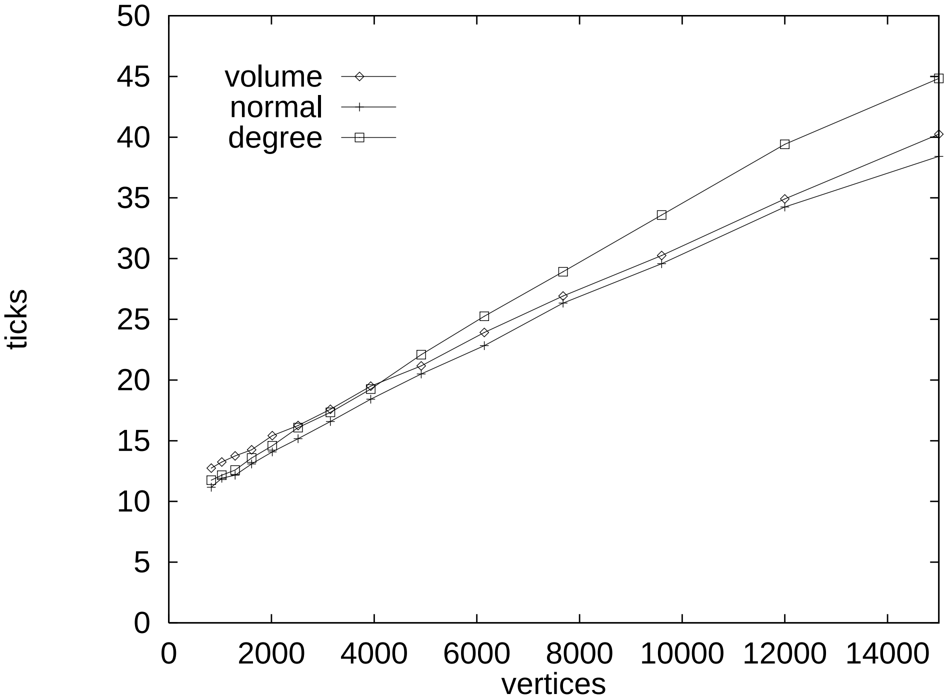

Figure 6.2: Time to delete 20% of the vertices using the three delete methods

6.2.1 Elevation

We plotted the elevation correlation coefficients against the number of vertices in the TIN. We used the logarithm of the difference between the correlation coefficient and one instead of the correlation coefficient. This transformation of the vertical axis provides more detail at correlation values close to one. Figure 6.3 shows a graph using a linear scale and Figure 6.4 shows the transformed scale. The vertical axis ranges from 0.99 to 0.9999, indicating that the correlation of the elevation of the simplified surface and the original surface is good.

With all three delete simplification methods, the correlation coefficients decrease as the number of vertices in the TIN decreases. This was expected, as a TIN with fewer vertices cannot represent all of the detail present in the original TIN. The correlation decreases slowly at first but tends to deteriorate faster when the TIN simplification has fewer than 30% of the original vertices.

The volume method tends to yield better correlation coefficients than the normal method but both are better than the degree method. This is not surprising since the degree method works independently of the z coordinate data. There is a quality versus speed tradeoff between the volume and the normal methods; the normal method works faster but with less accuracy.

Changing the delete percentage from 20% to 10% created some interesting effects. Figure 6.5 shows the correlation coefficients of the TINs simplified with the degree delete simplification method in 10% steps were slightly worse than the correlations coefficients of the TINs simplified in 20% steps. But the results using the normal method and the volume method are different. Figure 6.6 shows the normal method with 10% steps is initially better than the 20% steps but eventually becomes worse and Figure 6.7 shows that the volume method with 10% steps is better than the 20% steps at all levels. Selecting a smaller percentage of vertices would increase the accuracy because the selected vertices would have better importance values. But deleting a smaller percentage of vertices results of more simplifications. Errors due to drift can appear because the importance of a vertex is measured in relation to the current simplification of the terrain model instead of the original terrain model.

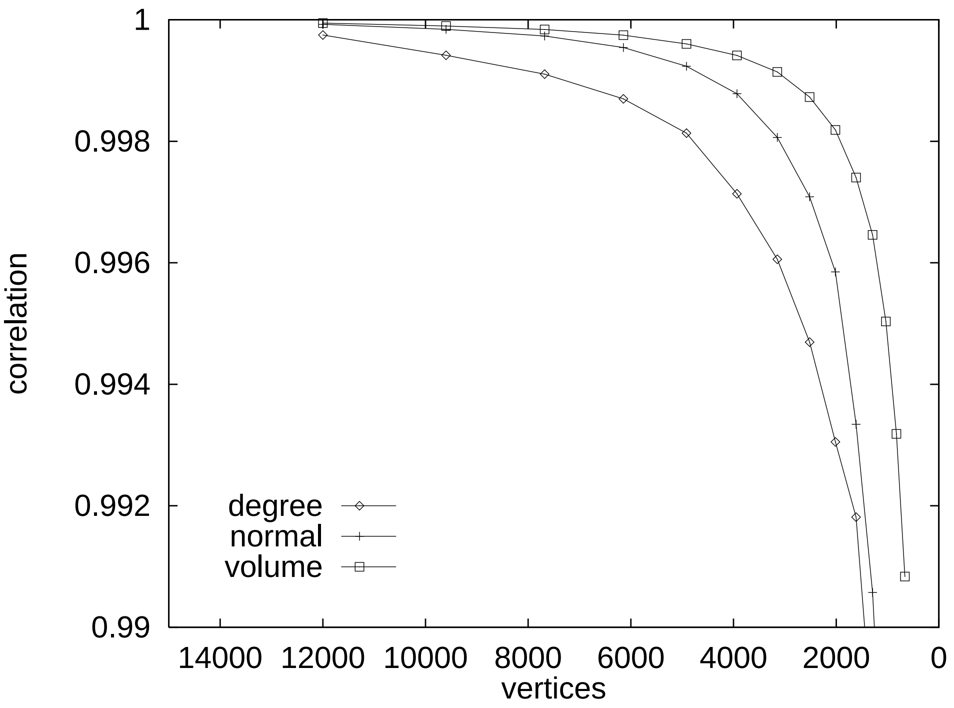

Figure 6.3: Linear scale on the elevation correlation axis

Two other studies [Lee91, Kum94] find that the insert simplification method is better than other tested methods at minimizing mean and maximum elevation error. But Figure 6.8 the performance of the volume delete simplification method is very close to, and sometimes better than, the performance of the insert simplification method. Both methods have $O(n \log n)$ running times but the volume method has the advantage of the simplification hierarchy.

The terrains with the higher correlations tended to be the terrains with high relief. An example is shown in Figure 6.9. But not all terrains with high relief had high correlations. This was the only difference observed when evaluating the simplification methods over the different terrains.

6.2.2 Mean Curvature

We plotted the mean curvature correlation coefficients against the number of vertices in the TIN. Figure 6.10 and Figure 6.11 show the same logarithmic transformation on the vertical axis for the mean curvature correlation coefficients. The scale of axis had to be changed because the correlation of mean curvature was much lower than the correlation of elevation. Almost every method dropped below 0.9 before the model is simplified to 40% of the original vertices.

The correlation coefficients again decreased as the number of vertices decreased but in this instance the decrease was smoother. Unlike the elevation correlations, there was no sudden deterioration at 30% of the original vertices.

The differences between volume method and the normal method were much smaller. On some graphs, including Figure 6.12, the normal method even outperformed the volume method. Both methods were much better than the degree method.

Changing the delete percentages had a different set of effects than those observed when plotting elevation correlations. The mean curvature correlations of the degree method in Figure 6.13 were slightly better when using a 20% delete percentage compared to a 10% delete percentage. But the 10% delete percentage was better when used with the normal method in Figure 6.14 and the volume method in Figure 6.15.

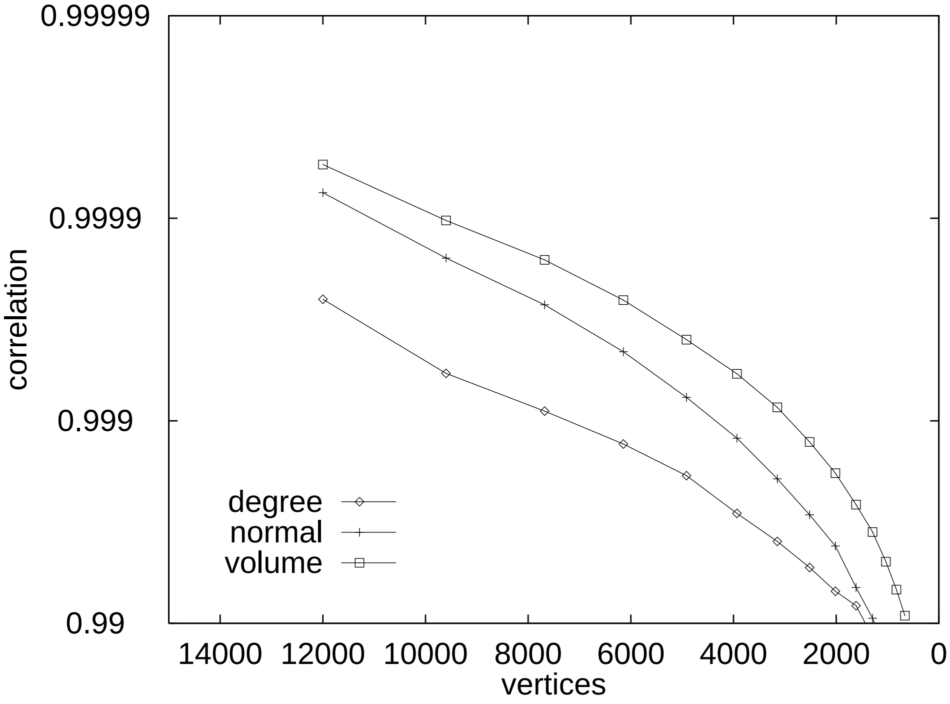

Figure 6.4: Logarithmic scale on the elevation correlation axis

The overall range of mean curvature correlations was lower than expected. This could be the result of our choice of grid spacing for the control points of the B-spline surface. An alternative approach would be to use natural neighbour interpolation to fit a surface through the vertices of the TIN. Implementing this method would be more complicated but we would avoid the problems of finding an appropriate grid spacing.

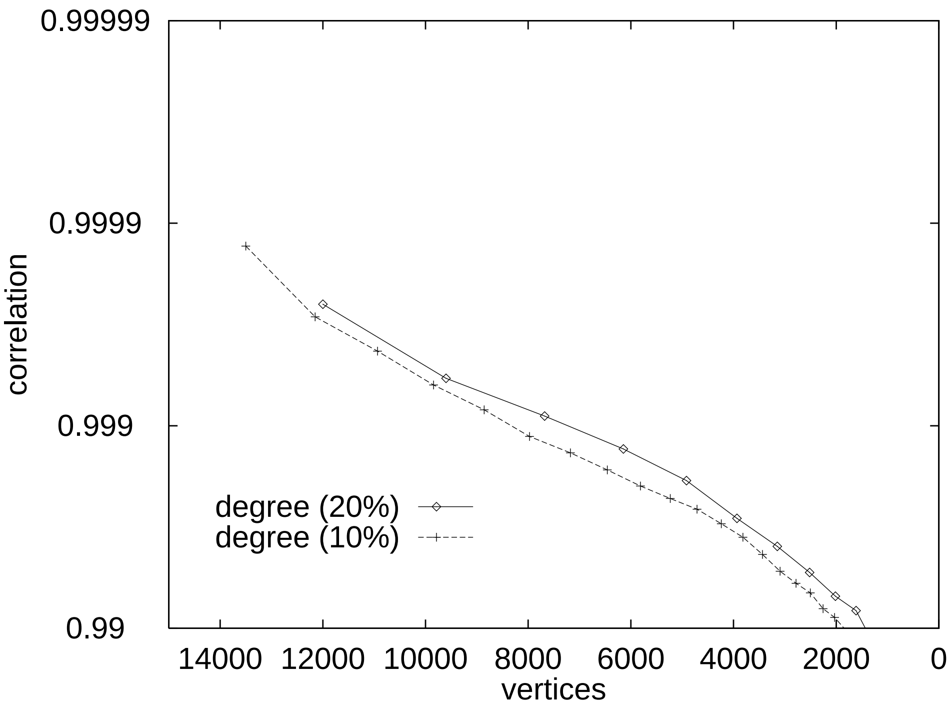

Figure 6.5: Elevation correlations using the degree method with two delete percentages

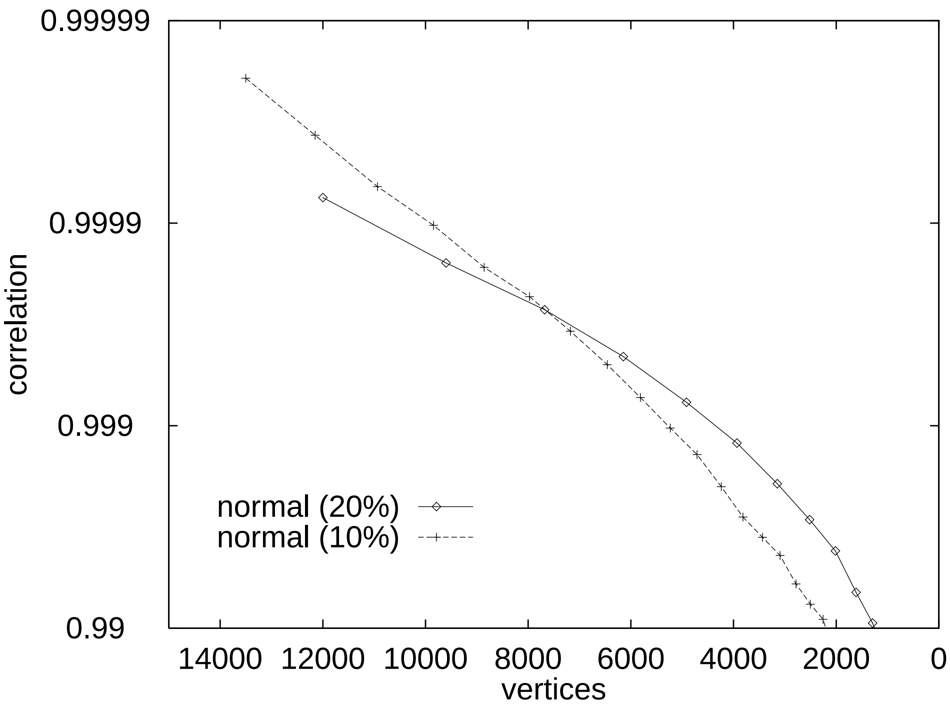

Figure 6.6: Elevation correlations using the normal method with two delete percentages

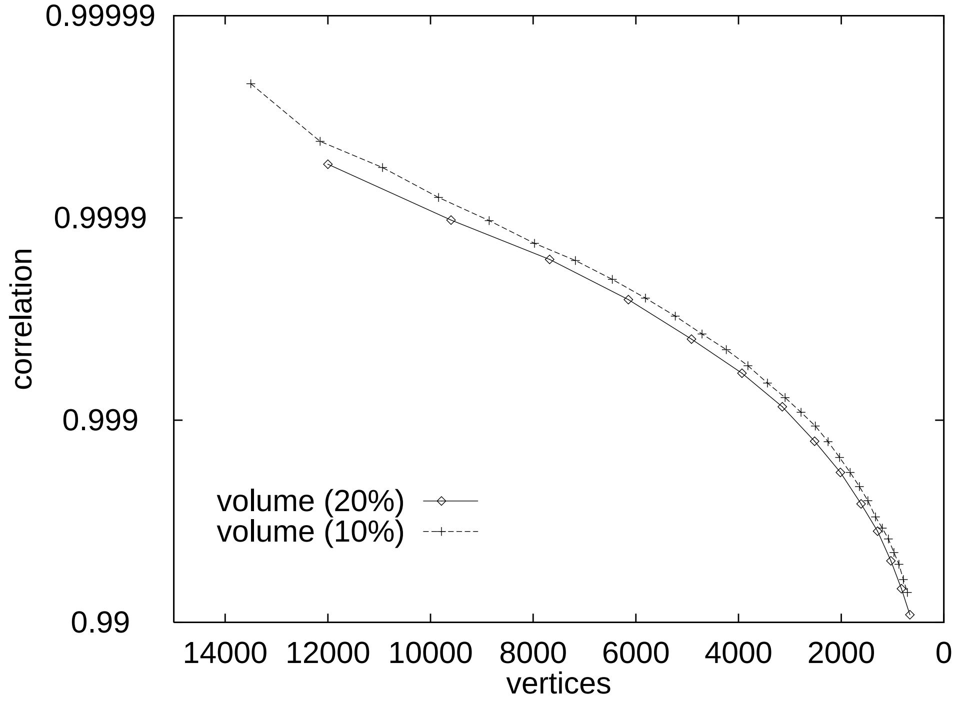

Figure 6.7: Elevation correlations using the volume method with two delete percentages

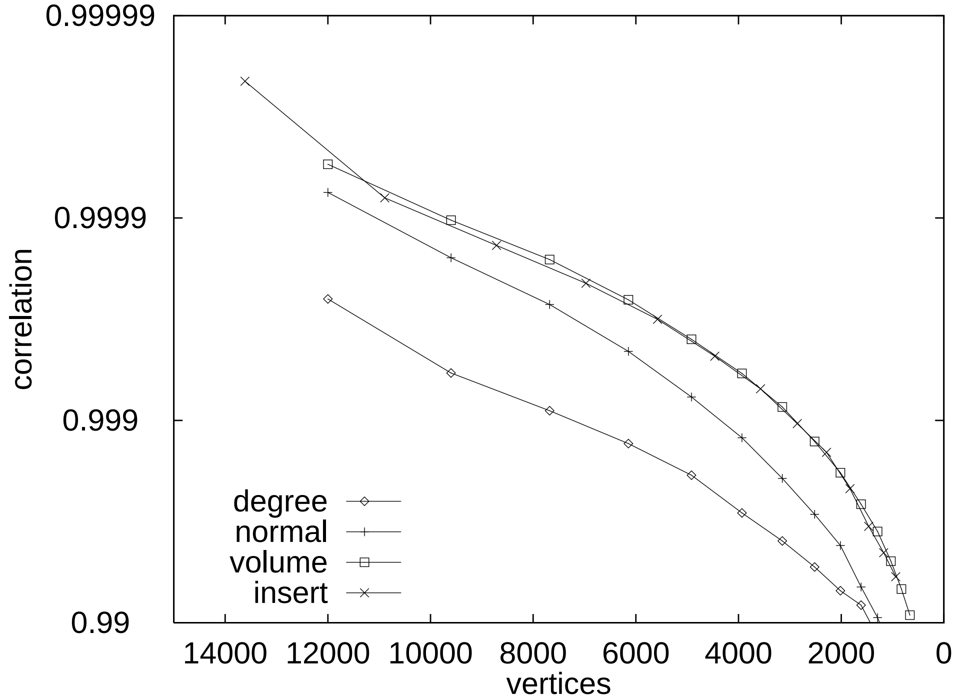

Figure 6.8: Comparing the elevation correlations of the insert method and the delete simplification methods

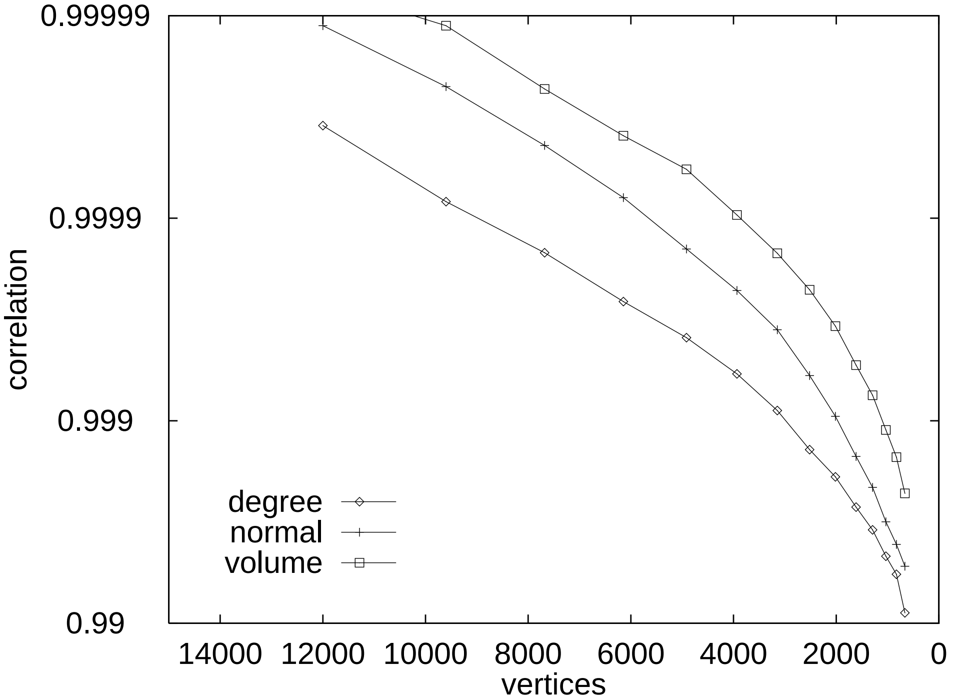

Figure 6.9: A terrain with higher elevation correlations and higher relief

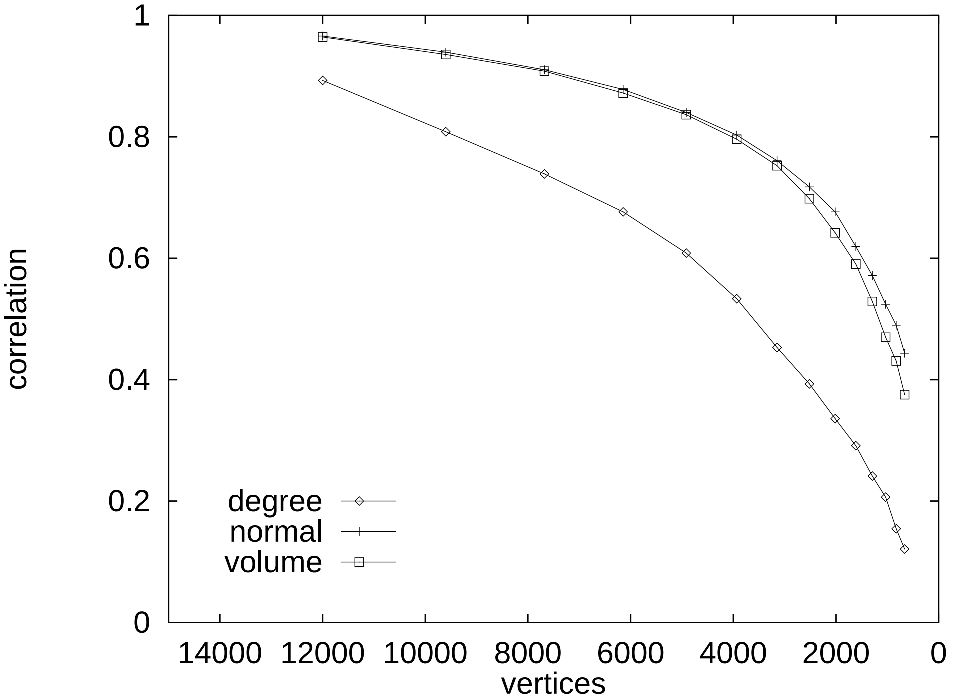

Figure 6.10: Linear scale on the mean curvature correlation axis

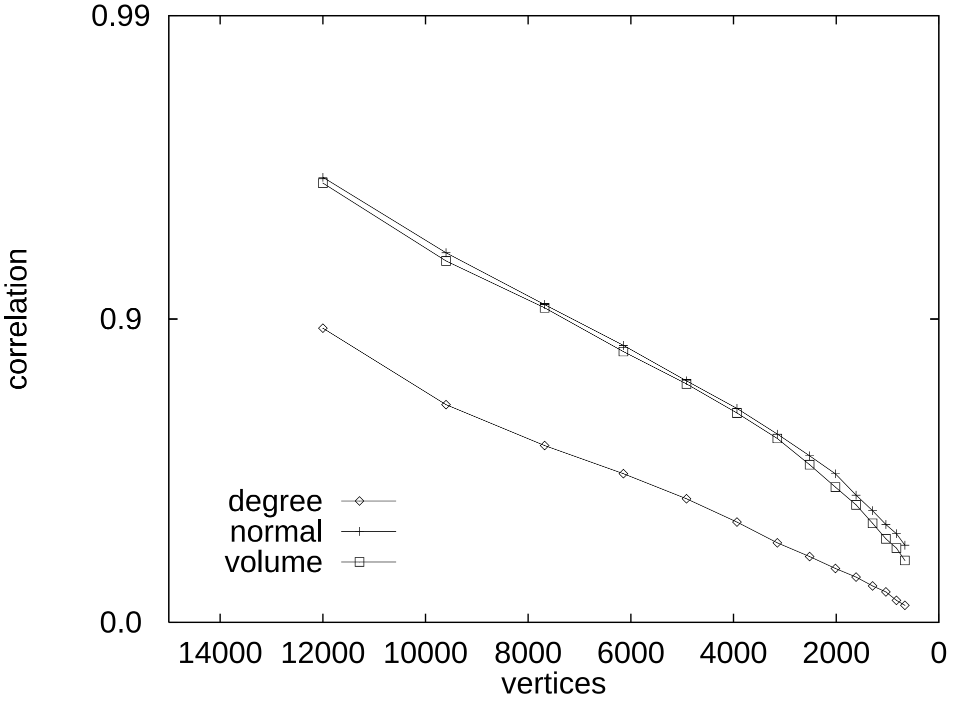

Figure 6.11: Logarithmic scale on the mean curvature correlation axis

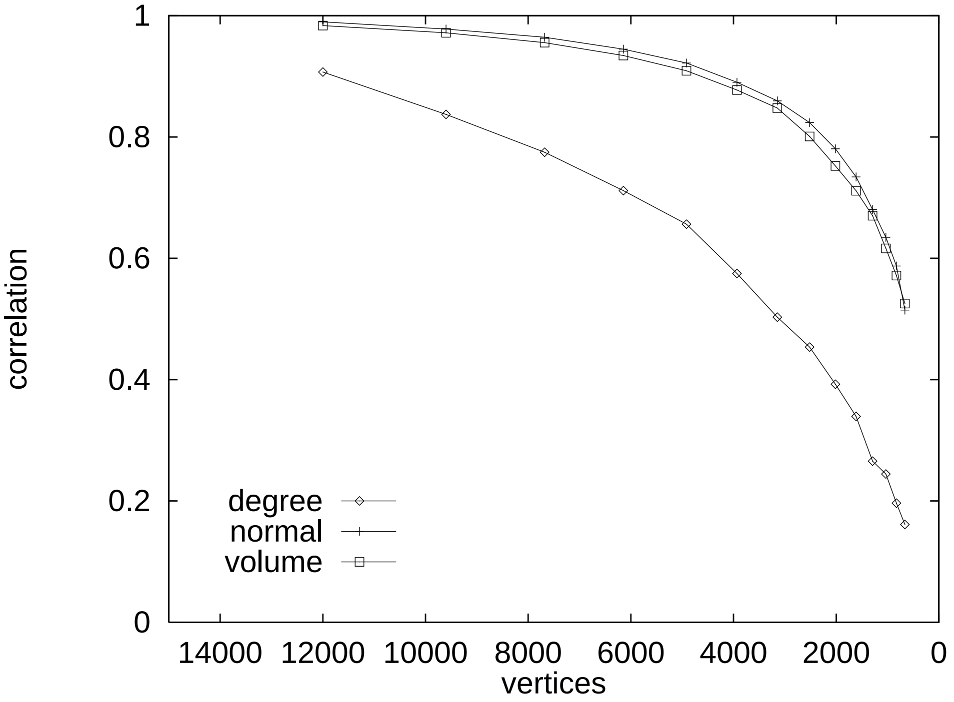

Figure 6.12: A terrain with better performance when using the normal method

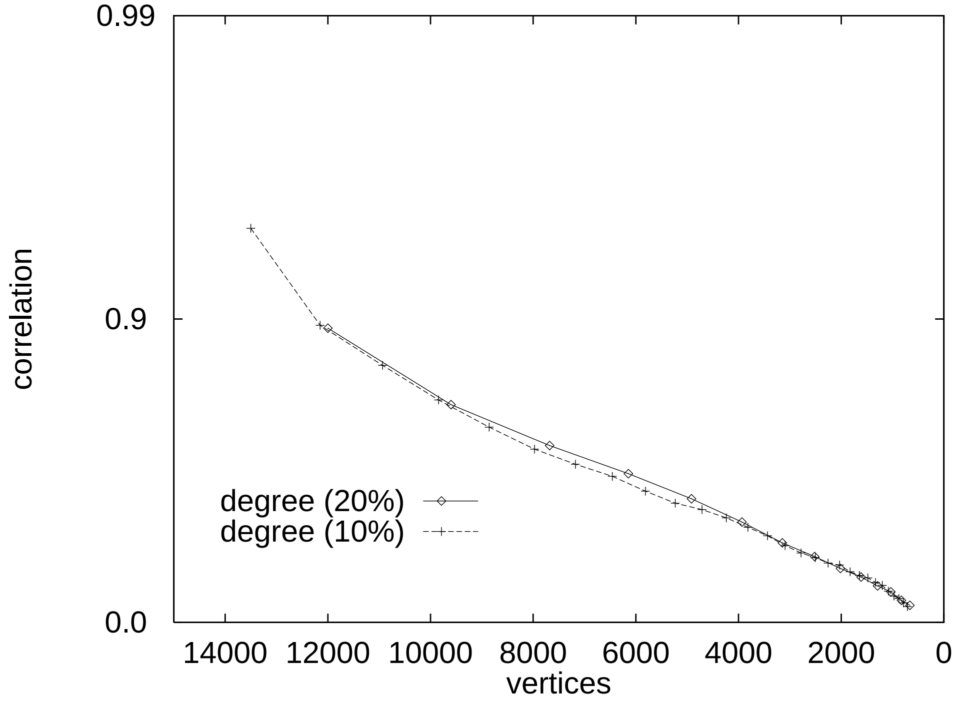

Figure 6.13: Mean curvature correlations using the degree method with two delete percentages

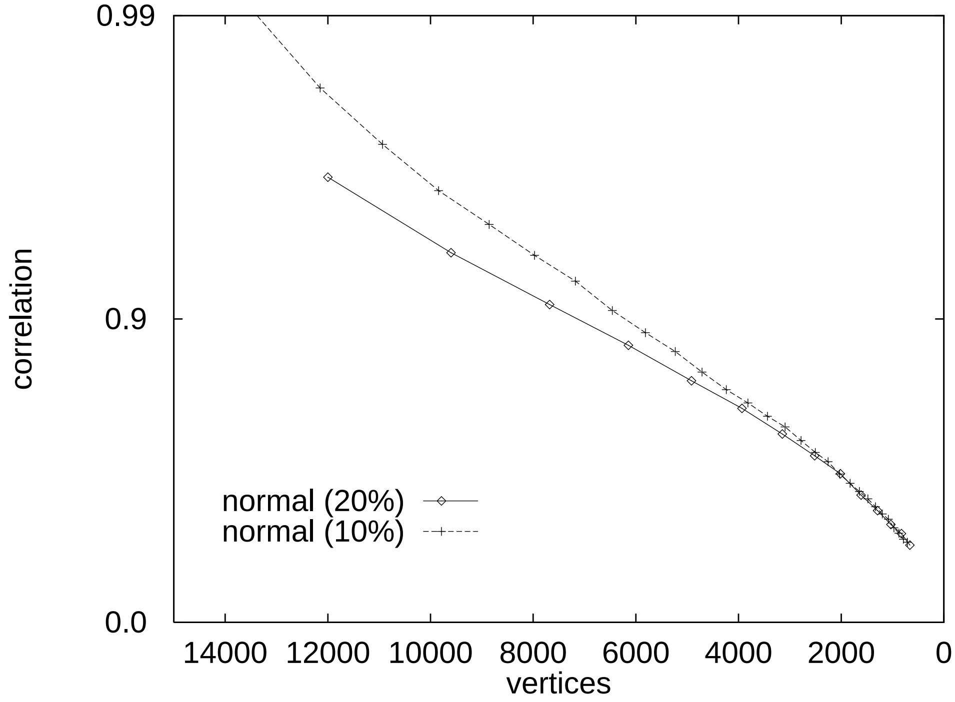

Figure 6.14: Mean curvature correlations using the normal method with two delete percentages

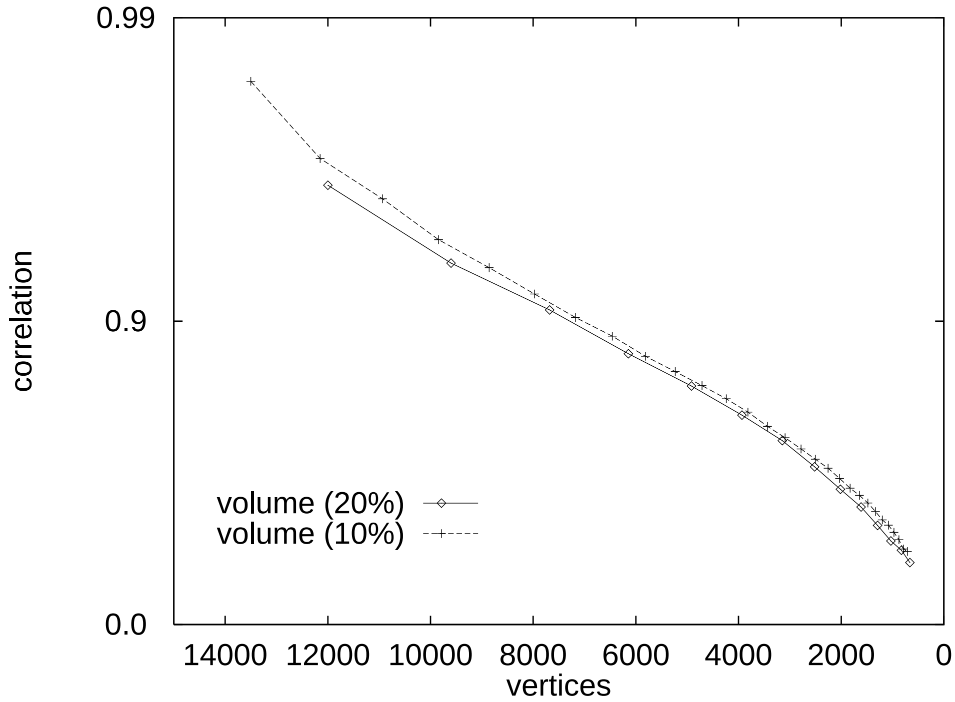

Figure 6.15: Mean curvature correlations using the volume method with two delete percentages

Chapter 7

Conclusion

We described a set of TIN simplification methods that we named delete simplification methods. They allow the use of the triangulation hierarchy introduced by Kirkpatrick and modified by de Berg and Dobrindt to form a TIN with varying levels of detail.

We also investigated different variables that can be used to measure surface accuracy. Acting on the recommendations of Evans [Eva72] and Kumler [Kum94], we chose to use a derivative statistic, mean curvature, in addition to elevation to grade the performance of our simplification methods. Since mean curvature is only defined at a point on a surface with continuous second partial derivatives, we were also required to fit a B-spline surface to a TIN.

One variant of the delete simplification methods, the volume method, formed simplifications with elevation and mean curvature correlations equal to or better than the greedy method. It has the added advantage of the hierarchical triangulation mentioned above.

The use of mean curvature as a variable to measure surface accuracy did not significantly alter our evaluation of the performance of these simplification methods. However, we recommend that any future comparisons of simplified terrains should be aware of these other surface characterization variables.

Bibliography

[CG89] Zi-Tan Chen and J. Armando Guevara. Systematic selection of very important points (VIP) from digital terrain models for constructing triangular irregular networks. In *AUTO-CARTO 8: Proceedings of the Eighth International Symposium on Computer-Assisted Cartography*, pages 50–56, 1989.

[dBD95a] Mark de Berg and Katrin Dobrindt. On levels of detail in terrains. Technical report, Utrecht University, 1995.

[dBD95b] Mark de Berg and Katrin Dobrindt. On levels of detail in terrains. In *Proceedings of the Eleventh Annual ACM Symposium on Computational Geometry*, pages C26–C27, 1995.

[dBvK93] Mark de Berg and Marc van Kreveld. Trekking in the Alps without freezing or getting tired. In *Algorithms - ESA '93*, Lecture Notes in Computer Science 726, pages 121–132, 1993.

[DFFNP84] Leila De Floriani, Bianca Falcidieno, George Nagy, and Caterina Pienovi. A hierarchical structure for surface approximation. *Computers and Graphics*, 8(2):183–193, 1984.

[DFMP94] Leila De Floriani, Paolo Marzano, and Enrico Puppo. Line-of-sight communication on terrain models. *International Journal of Geographical Information Systems*, 8(4):329–342, 1994.

[DFP92] Leila De Floriani and Enrico Puppo. A hierarchical triangle-based model for terrain description. In *Theories and Methods of Spatio-Temporal Reasoning in Geographic Space*, Lecture Notes in Computer Science 639, pages 236–251, 1992.

[DLR90] Nira Dyn, David Levin, and Samuel Rippa. Data dependent triangulations for piecewise linear interpolation. *IMA Journal of Numerical Analysis*, 10(1):137–154, 1990.

[DP73] David H. Douglas and Thomas K. Peucker. Algorithms for the reduction of the number of points require to represent a digitized line or its caricature. *The Canadian Cartographer*, 10(2):112–122, 1973.

[Eva72] Ian S. Evans. General geomorphometry, derivatives of altitude, and descriptive statistics. In *Spatial analysis in geomorphology*. Methuen & Co Ltd, 1972.

[Fen28] Nevin M. Fenneman. Physiographic divisions of the United States. *Annals of the Association of American Geographers*, 18(4):261–353, 1928.

[Fis94] Peter Fisher. Stretching the viewshed. In *Proceedings of the Sixth International Symposium on Spatial Data Handling*, volume 2, pages 725–738, 1994.

[FL79] Robert J. Fowler and James J. Little. Automatic extraction of irregular network digital terrain models. In *Computer Graphics: ACM SIGGRAPH '79 Proceedings*, pages 199–207, 1979.

[FvDFH90] James D. Foley, Andries van Dam, Steven K. Feiner, and John F. Hughes. *Computer Graphics: Principles and Practice*. Addison-Wesley Publishing Company, 1990.

[GG79] Dora Goméz and Adolfo Guzmán. Digital model for three-dimensional surface representation. *Geo-Processing*, 1(1):53–70, 1979.

[GH95] Michael Garland and Paul S. Heckbert. Fast polygonal approximation of terrains and height fields. Technical report, Carnegie Mellon University, 1995.

[Ham64] Edwin H. Hammond. Analysis of properties in land form geography: An application to broad-scale land form mapping. *Annals of the Association of American Geographers*, 54(1):11–19, 1964.

[Kir83] David Kirkpatrick. Optimal search in planar subdivisions. *SIAM Journal on Computing*, 12(1):28–35, 1983.

[Kum94] Mark Kumler. An intensive comparison of Triangulated Irregular Networks (TINs) and Digital Elevation Models (DEMs). *Cartographica*, 31(2), 1994.

[Law72] Charles L. Lawson. Transforming triangulations. *Discrete Mathematics*, 3:365–372, 1972.

[Lee89] Jay Lee. A drop heuristic conversion method for extracting irregular network for digital elevation models. In *GIS/LIS'89 Proceedings*, volume 1, pages 30–39, 1989.

[Lee91] Jay Lee. Comparison of existing methods for building triangular irregular network models of terrain from grid digital elevation models. *International Journal of Geographical Information Systems*, 5(3):267–285, 1991.

[LS80] D. T. Lee and B. J. Schachter. Two algorithms for constructing a Delaunay triangulation. *International Journal of Computer and Information Sciences*, 9:219–242, 1980.

[Mar78] David M. Mark. Concepts of "data structure" for digital terrain models. In *Proceedings of the Digital Terrain Models (DTM) Symposium*, pages 24–31, 1978.

[Mid94] Terje Midtbø. Removing points from a Delaunay triangulation. In *Proceedings of the Sixth International Symposium on Spatial Data Handling*, volume 2, pages 739–750, 1994.

[MM89] David S. Moore and George P. McCabe. *Introduction to the practice of statistics*. W. H. Freeman and Company, 1989.

[Mok90] Farzin Mokhtarian. *A Theory of Multi-Scale, Curvature and Torsion Based Shape Representation for Planar and Space Curves*. PhD thesis, University of British Columbia, 1990.

[O'N66] Barrett O'Neill. *Elementary Differential Geometry*. Academic Press, 1966.

[PFLM78] Thomas K. Peucker, Robert J. Fowler, James J. Little, and David M. Mark. The triangulated irregular network. In *Proceedings of the Digital Terrain Models (DTM) Symposium*, pages 516–532, 1978.

[PS85] Franco P. Preparata and Michael Ian Shamos. *Computational Geometry: An Introduction*. Springer-Verlag, 1985.

[SP90] Lori Scarlatos and Theo Pavlidis. Hierarchical triangulation using terrain features. In *Proceedings of the First IEEE Conference on Visualization*, pages 168–175, 1990.