A STUDY OF DATA-DEPENDENT TRIANGULATIONS FOR TERRAINS

Weifei Wu

A thesis
in
The Department
of
Computer Science

Presented in Partial Fulfillment of the Requirements
For the Degree of Master of Computer Science
Concordia University
Montréal, Québec, Canada

August 2004

Your file Votre référence
ISBN: 0-612-94755-6
Our file Notre référence
ISBN: 0-612-94755-6

Abstract

A Study of Data-Dependent Triangulations for Terrains

Weifei Wu

When simplifying and approximating surfaces with triangulations, we must determine what is a good triangulation. Thus, we need to define the criterion to measure the quality of a triangulation. There exist many useful criteria in the literature, such as Max-Min angle, Min-Max angle, least squares fit, Gaussian curvature, mean curvature criteria and so on. We introduce two further triangulations based on a minimum area criterion. This thesis compares these different criteria for terrain simplification for a variety of types of terrains.

Acknowledgments

I would like to thank my supervisor Dr. Thomas Fevens for his inspiring guidance, support and magnanimousness.

Contents

List of Figures

List of Tables

Chapter 1

Introduction

1.1 Background

Terrains are among the simplest kinds of surfaces. The z coordinate of every point (x, y, z) on a terrain can be denoted as $z = H(x, y)$. Any lines perpendicular to the xy-plane cross the surface of the terrain at most once.

Terrain data find application in many fields, such as flight simulators, ground vehicle simulators, digital topographic map, virtual reality, computer animation, computer graphics, landscape planning and so on.

Nowadays, terrain models have become more and more precise. However, too detailed terrain models are not always helpful. For example, in real-time computer graphics applications, it is not feasible to display a highly detailed terrain with too many polygons. Moreover, sometimes, some parts of a terrain are unnecessary and they become a burden for rendering. Thus, we must simplify the terrain so as to speed up the process of rendering and computation and reduce the amount of the storage needed. Terrain simplification aims to represent a surface with fewer points but still approximating the original surface well enough.

In general, there are six kinds of terrain simplification methods. They are uniform grid methods, hierarchical subdivision methods, one-pass feature methods, multi-pass refinement methods, multi-pass decimation methods and optimal methods [17]. Uniform grid methods use regular grids to sample terrain data. Hierarchical subdivision methods divide a terrain into subdivisions recursively and use a tree structure to represent the terrain. Feature methods use a set of significant feature points in the original data set as vertices for triangulation. Refinement

methods start with a least approximate surface and keep inserting points or edges into the triangulation until a certain condition is met. For example, the refinement process can be stopped when the error is below an error threshold or the number of points inserted is greater than certain number. On the other hand, decimation methods are the reversion of refinement methods. Decimation methods start with a maximal approximation and keep deleting points from the triangulation until certain condition meets. The termination conditions can be when the error is above an error threshold or the number of points deleted is greater than certain number. Optimal methods calculate an optimal approximation for a surface using an optimal partition, such as a triangulation, of a subset of the input points.

Most terrain simplification methods use a triangulation to approximate terrains. A *triangulation* is a set of triangles that meet two conditions. The first condition is that any two triangles intersect with each other only at common vertex or along a common edge. The second is that the union of the triangles is a connected set. We denote a triangulation as $\triangle$ in this thesis.

For a finite number of input points, although the number of possible triangulations is tremendous, most of them lead to a poor approximation. Thus, we must define what kind of triangulations are the best approximations. A criterion gives the rule that defines the quality of a triangulation. It maps a triangulation to a real number, called the quality of the triangulation. We denote the quality of a triangulation as $Quality(\triangle)$ in this thesis. An optimal triangulation is such a triangulation $\triangle$ that $Quality(\triangle) >= Quality(\triangle')$ for all other possible triangulations $\triangle'$ of the same vertex set.

The most popular method to calculate an optimal triangulation is the Delaunay triangulation. It is based on the Max-Min angle criterion. It defines an optimal triangulation as

one whose minimal angle is the maximum among all possible triangulations of the same vertex set. In addition to the Max-Min angle, it also optimizes several more properties at the same time, such as minimizing the maximal circumcircle and minimizing the maximal min-containment circle. It is a purely two-dimensional method, because it does not use the z coordinate of a vertex when calculating the optimal triangulation, whereas, the triangulation that considers the z coordinate of a vertex as well is called data-dependent triangulation. A Delaunay triangulation approximates terrains so well that Garland and Heckbert [17] conjecture that data-dependent triangulation cannot work better [18]. Besides Delaunay triangulation, many other 2-dimensional triangulation methods also yield good approximation, such as Min-Max angle triangulation. Our experiments show that sometimes Min-Max angle triangulation works even better than Delaunay triangulation.

However, because the z coordinates of points on a surface also contain geometric information of the surface, many data-dependent triangulation methods have been proposed in recent years, such as tight triangulation, the triangulation minimizing absolute mean curvature and L_2 measure and so on [3] [25]. They have been used in many applications. These include smoothing surfaces, estimating the geometric properties of the actual surfaces such as curvatures, areas and volumes from the triangulation, extracting, describing the shapes of surfaces and reconstructing the surfaces. Surface smoothing helps to eliminate tiny noises caused by measure error. Data-dependent triangulations based on smoothness criteria, such as mean curvature criterion, work quite well on this purpose.

Data dependent triangulations find application in many fields. Precise reconstruction is an essential operation in image processing. There are many methods to reconstruct the images, such as nearest neighbor interpolation, bilinear interpolation, bi-cubic interpolation and data-

dependent triangulation [27]. Among them, data-dependent triangulations generate the most esthetically pleasing results [27]. Furthermore, data-dependent triangulation can yield lower error approximations than two-dimensional triangulations when approximating certain surfaces [18]. Moreover, some data-dependent triangulations, such as tight triangulation, preserve the convexity of the surface whereas Delaunay triangulation cannot.

Because data-dependent triangulations perform quite well in the various applications, we want to apply them in terrain simplification and check whether they give a better approximation than two-dimensional triangulations for terrains or not. In this thesis, we will compare triangulation methods based on both the two-dimensional criteria and data-dependent criteria and determine under different types of terrains each produces the best results in terms of terrain simplification.

1.2 Thesis outline

The outline of this thesis is as follows:

In Chapter 2, we will give a literature review. We will review the literature in 5 aspects. Firstly, we will discuss the algorithms for surface simplification. Secondly, we will study different criteria for optimal triangulation. Thirdly, we will state the models used to represent terrain data. Fourthly, we will present some popular data structures to store the resulting triangulation. Last, we will discuss ways to measure the accuracy of a triangulation. In Chapter 3, we will give detailed description of all the algorithms we implement in order to make the comparison. In Chapter 4, we will give the simulation results. In Chapter 5, we will discuss the results we obtain in Chapter 4. In the last chapter, we will draw some conclusions from our simulation results.

Chapter 2

Literature review

Formulating the problems:

When we simplify a surface, we must solve the following problems:

a) What is the goal? When we approximate a surface by a set of triangles, we must determine what kind of triangular meshes are the best approximations. Thus, we must define the criteria that measure the quality of a triangulation and compare different triangulations according to these criteria. There exist many useful criteria. The most commonly used ones are Max-Min angle, Min-Max angle, a least squares fit criterion, minimization of total length, and so on.

b) How to find an optimal triangulation with respect to certain criterion? We must compute an optimal triangulation with respect to certain criteria. For different criteria, the algorithmic paradigms suitable to find the best triangulation are different. For example, a local swap algorithm will find the global optimum with respect to Max-Min angle criterion [26], whereas it will only get the local optimal triangulation with respect to Min-Max angle criterion. However, edge insertion paradigm will calculate the global optimal triangulation with respect to Min-Max angle criterion [26].

c) Input. In order to approximate a surface, we must find a model to represent and store the actual surface. For terrains, we must define DTM, the digital terrain models. Three common kinds of DTM are DLG (Digital Line Graph), DEM (Digital Elevation Model) and TIN (Triangulation Irregular Network) [20].

d) Output. Similarly, we must find a data structure to represent and store the resulting triangulation we create. There are two popular kinds of data structures to represent the triangulations: Quad-Edge and DCEL [18] [7].

e) Result analysis: When we get the optimal triangulation with respect to certain criteria, we must compare the original surface with the approximation to find out how different they are. The error of the approximation is usually measured with respect to the differences of the elevations between the points on the actual surface and those on the approximation. The typical error measures are L_2 or L_∞ errors. Besides, we can also use curvature as an accuracy measure, because curvature is a significant feature of a surface.

We will review the literature with respect to the above five aspects.

2.1 Methods for surface simplification

We will review three kinds of methods for terrain simplification: refinement methods, optimal methods and decimation methods. Refinement methods repeatedly insert vertices into the triangular mesh until a certain condition is met. Optimal methods calculate an optimal approximation for a surface using triangulations of a subset of the input points. Optimal methods maintain the number of simplexes in the triangulation while optimizing the surface. Decimation algorithms repeatedly delete unimportant simplexes from the finest triangulation to simplify the surface. Refinement algorithms can achieve higher quality triangulation with fewer points than decimation algorithms, since they make decisions on a global base when choosing a point to add, but they are slower than decimation methods [18].

2.1.1 Refinement methods

Refinement methods approximate a surface by repeatedly inserting vertices into the triangulation until a certain condition is met. The termination condition can be the number of

points inserted is greater than certain amount or the error of the triangulation is lower than a certain threshold. A refinement method first creates an initial triangulation with minimal approximation that usually has only one triangle or two triangles containing all the points of the input data set in their interiors. Then it will keep inserting the most important points, as determined by some significance criterion, into the triangulation, adding spokes from the inserted points to their neighbors (which define the face or faces containing the newly inserted point) and optimizing the resulting triangulation after each insertion to maintain a certain criterion. Refinement methods differ in how to define the importance of a vertex, how many points will be inserted each time, how often to recalculate the importance of a vertex and how to optimize the new triangulation after a point insertion.

One-pass vs. Multi-pass: One-pass methods calculate the importance the vertices only once, whereas multi-pass methods recalculate the importance of the vertices from phase to phase. Multi-pass methods work better than One-pass methods because multi-pass methods always use the updated importance to choose a point, but it costs more time.

Sequential vs. Parallel: Sequential methods add one point at a time, whereas parallel methods insert a number of points before doing the triangulation. Sequential methods perform better than parallel insertion [18].

Vertex importance measures: We can choose a set of points with significant features, such as peaks, pits, valleys and ridges as the most important points. Besides, we can use some criteria to define the importance. The criteria for vertex importance are crucial to the refinement process because it can determine how much the approximated triangulation is similar to the actual surface.

a) Local Error Measure

Importance (v) = $|H(v) - H'(v)|$.

Here $H(v)$ is the actual elevation of point v and $H'(v)$ is the interpolated elevation of v on the simplified surface. The vertex-insertion algorithm that inserts the point with the highest local error each time generally produces high quality approximations. Moreover, it is simple and runs fast.

b) Global Error Measure

$$Importance\ (v) = \sum_{i=1}^{m} | H(v_i) - H'(v_i) |.$$

Here $H(v_i)$ is the actual elevation of v_i, whereas, $H'(v_i)$ is the interpolated elevation of v_i if v is inserted into the mesh. We will choose the point that leads to the lowest global error. The insertion algorithm with this criterion is very expensive. Moreover, the approximation with such criterion has poor quality, because it is too short sighted. In fact, introducing a significant feature into the approximation often requires some bad moves at first, which will increase the error a lot temporarily, but will be soon corrected by inserting several more points [18]. Local Error Measures often work better than global error measures.

c) Curvature Measure

Because the spots with significant features on a surface typically have high curvatures, we can choose the curvature as an important measure. We will choose the points with the highest curvatures to insert. The sum of the squares of the principal curvatures works better than Laplacian curvature $\partial^2 f / \partial x^2 + \partial^2 f / \partial y^2$ as an important measure [18]. However, curvature measures do not give high quality approximation [18].

All the above importance measures are pure importance measures because they lack the topographic knowledge of the surface. Not all of them do perform well, since they only make independent and local decisions [18].

Optimizing method:

After a point is inserted, the resulting triangulation will not go on satisfying a certain criterion. The triangulation must be adjusted to meet it again. With respect to some criteria, such as Max-Min angle criterion, we only need to adjust the neighborhood of the inserted point, because only this region does not conform the criterion. For example, in order to maintain Delaunay triangulation after a point v inserted, we inspect the edge e not adjacent to v of every triangle adjacent to v to check whether e still meets the empty circumcircle condition. The *circumcircle* of a triangle is a circle passing through all its three vertices. If so, e is unchanged. Otherwise, it will be replaced with the other diagonal of the quadrilateral Q around e. Then two more edges of Q not incident to v become candidates for inspection. The process will terminate when no candidates exist. The most number of edges that may have to be inspected is a small constant number [18]. However, with respect to all other criteria, we must adjust the whole mesh. We will use local swap algorithm to adjust it in order to meet the criterion again.

2.1.2 Optimal methods

Optimal methods calculate an optimal approximation for a surface. There are several common approaches to finding an optimal approximation, such as dynamic programming, edge-flipping algorithm, edge insertion paradigm and so on.

Dynamic programming can directly find the optimal triangulation for simple polygons with respect to some criteria, such as Min-Max angle criterion. Edge-flipping and edge-insertion algorithms aim to find an optimal triangulation by locally improving the triangulation. Research shows that the upper bound of the number of possible triangulations is 10^{13n} of a set of n input points [27]. Thus, it is impossible to enumerate all possible triangulations to find an optimum. Edge-flipping and edge-insertion algorithms were developed to limit the number of triangulations searched but still achieve a high chance of finding a global optimum or the

one close to it. An edge-flipping algorithm calculates the optimal triangulation by repeating swapping diagonals until reaching an optimum. *Swap* is the operation that replaces a diagonal of a triangulated quadrilateral with the other diagonal of it.

Before swapping e After swapping e

Figure 1: Making a swap

An edge insertion algorithm improves the quality of the current triangulation by inserting an edge into the triangulation, deleting intersected edges and optimally re-triangulating the two resulting polygonal holes.

2.1.2.1 Dynamic Programming

Dynamic programming is a commonly used problem solving technique. It is often used to find the best solution of a problem. It iteratively subdivides a problem into smaller sub-problems. At last, we end up with a finite number of small enough sub-problems that can be solved directly. Then it will solve smaller problems first and store the solution of smaller problems to avoid recalculation of their solutions. Later it will use the solutions of smaller problems to solve increasingly larger problems again storing their solutions. Dynamic programming can be used to find an optimal triangulation for simple polygons [26] [8]. It finds an optimal triangulation of a polygon with n vertices by combining the solutions of sub-polygons with fewer vertices. It is a bottom-up method. It first calculates the quality of all possible triangles (with *3* edges) of the polygon P. Then it calculates the quality of the optimal triangulation of all possible sub-polygons of P with *4* edges, with *5* edges and so on

until we reach the polygon with n edges. Then the last one is the solution. For a sub-polygon $\{v_i \ldots v_j\}$ P_{ij}, we will use all the vertices v_k between v_i and v_j to split P_{ij} into three sub-polygons $\{v_i \ldots v_k\}$, $\{v_k \ldots v_j\}$ and triangle $\{v_i v_k v_j\}$. We will calculate the quality of the triangulation of P_{ij} by combining the quality of the optimal triangulation of $\{v_i \ldots v_k\}$, $\{v_k \ldots v_j\}$ and triangle $\{v_i v_k v_j\}$. We will choose the best one among the qualities of P_{ij} resulting from all possible splits of $\{v_i \ldots v_j\}$ as the quality of optimal triangulation of P_{ij}. For a sub-polygon $\{v_i \ldots v_j\}$ P_{ij}, we will store the quality of the optimal triangulation of P_{ij} to avoid recalculation. We will also store the vertex v_k that the quality of the combination of $\{v_i \ldots v_k\}$, $\{v_k \ldots v_j\}$ and triangle $\{v_i v_k v_j\}$ is optimal. We will use this information to construct the optimal triangulation of the whole polygon P.

Figure 2: Dynamic programming

Dynamic programming can be used to optimize a triangulation locally. For example, it can be used to re-triangulate the polygonal hole formed after a point is deleted in the mesh.

2.1.2.2 Edge Insertion paradigm

Edge insertion paradigm computes the optimal triangulation by improving the initial triangulation step by step until no improvement can be done. Then the last triangulation is the optimum. Edge insertion paradigm requires establishing an initial triangulation of the input data set first. Then it repeats inserting a vertex pair that is not an edge into the current mesh, deleting the edges intersecting with it and re-triangulating the two holes that are

divided by it. It only inserts the vertex pairs whose insertion will improve the quality of the current triangulation. It ends when no such vertex pairs are available.

Figure 3: Edge insertion paradigm: ab is inserted

We can use any methods, such as dynamic programming, to re-triangulate the two polygons as long as their re-triangulations satisfy the same criterion.

The naive implement of edge insertion paradigm tests all possible vertex pairs and select one whose insertion will make an improvement to insert each time. It's time complexity is $O(n^8)$ [10].

Edge insertion paradigm can be applied to min-max, max-min criteria. Suppose measure μ is a function mapping a triangle T to a real value $\mu(T)$. Max-min μ criterion defines an optimal triangulation as one whose minimal $\mu(T)$ over all its triangles is maximum among all possible triangulations [26]. With respect to max-min μ criterion, a triangle T^- is the worst triangle in $\triangle$ if $\mu(T^-) \leq \mu(T)$ for every triangle T in $\triangle$. A vertex y is called an *anchor* of a triangle xyz if for any triangulation $\triangle$ of the actual surface with $\mu(\triangle) \geq \mu(xyz)$, $\triangle$ will contain xyz or break xyz at y. $\triangle$ is called breaking xyz at y, if there exists an edge yt in $\triangle$ such that $yt \cap xz \neq 0$. The anchor of a triangle is also called the worst vertex of the triangle. With respect to Min-Max angle criterion, the worst vertex is the one incident to the maximum angle and the worst triangle is the one owning the maximum angle.

Edge insertion paradigm can be improved to $O(n^3)$ if we only insert the vertex pair incident to the worst vertex of the worst triangle each time and if we re-triangulate the two resulting holes by repeating removing ears, not by dynamic programming [26]. Thus, edge-insertion paradigm improves a triangulation by inserting an edge e incident to the worst vertex of the worst triangle into the triangulation, deleting the edges intersecting with e and keeping cutting the ears whose qualities are better than the worst triangle of the two holes.

A criterion μ can be implemented by edge insertion paradigm, provided that for any triangulation $\triangle$ and any triangle xyz of $\triangle$ with $\mu(xyz) = \mu(\triangle)$, there exists an anchor of xyz or for every triangulation $\triangle$ and every triangle xyz, there exists an anchor of xyz. They are called weak anchor condition and strong anchor condition respectively, because if a criterion meets the second condition, it will certainly meets the first condition too. But it is not true on the contrary.

Edge insertion paradigm is more general than edge flipping algorithm, because it allows replacing more than one edge, whereas edge-flipping method only permits replacing a diagonal with the other diagonal of a quadrilateral. Thus, edge insertion paradigm will have lower probability to get stuck in a local optimum [26]. Therefore it has more chance to find a global optimum, but it costs more time. For example, local swap algorithm cannot find the global optimum with respect to Min-Max angle criterion in general, whereas edge insertion paradigm can find the global one.

A triangulation $\triangle$ is global optimal if $Quality(\triangle) >= Quality(\triangle')$ for all other possible triangulations $\triangle'$, whereas a triangulation is a local optimal if it cannot be improved locally, but there still exists a triangulation better than it. Edge insertion paradigm can find out the global optimum with respect to the following criteria: Min-Max angle criterion, maximizing the

minimum triangle height, minimizing the maximum eccentricity and minimizing the maximum slope. The slope of a vertex *(x, y, f(x, y))* can be denoted as $\sqrt{(\frac{\partial f}{\partial x})^2 + (\frac{\partial f}{\partial y})^2}$ [26]. The *eccentricity* of a triangle T is the smallest value among all distances between the center of T's circumscribed circles to points inside T.

2.1.2.3 Edge-flipping Algorithm

Because every triangulation with n vertices can be transformed to any other triangulations of the same point set by swapping a finite number of edges [5], we can find the global optimal triangulation by repeating swapping edges in an initial triangulation. Edge-flipping algorithm keeps swapping edges in the triangulation until a certain condition is met. An initial triangulation must have existed before the edge-flipping algorithm is applied. We often use Delaunay triangulation to build it, because Delaunay triangulation has quite good quality in general. There exist two common types of edge-flipping algorithm. One is local swap algorithm that only makes a swap when it will improve the triangulation. The other is a simulated annealing algorithm that not only makes a good swap, but also makes a bad swap according to some probability [25]. The simulated annealing algorithm was developed to find a globally optimal triangulation when the local swap often cannot find one.

In order to determine whether an edge should be swapped or not, we will assign a real number for every edge, called its swap value. A *swap value* is the difference between the quality of a triangulation $\triangle$ and the quality of a triangulation $\triangle$' after swapping an edge e. We denote it as $SwapValue(e) = Quality(\triangle') - Quality(\triangle)$. Because $\triangle$' differs $\triangle$ in only the neighborhood of e, we need not calculate the qualities of $\triangle$ and $\triangle$' to determine whether to swap an edge or

not. Instead, we only need to measure the difference of the quality of e's neighborhood before and after swapping e.

The local swap algorithm only swaps edges with positive swap values, whereas, simulated annealing not only swaps edges with positive swap values, but also swaps edges with negative swap values according to some probability.

2.1.2.3.1 Local Swap Algorithm

The local swap algorithm works quite well for optimizing many criteria such as minimizing the maximum angle. Moreover, it will find a globally optimal triangulation for Max-Min angle criterion. However, for all other commonly used criteria, it only reaches local optimum because it can get stuck in a local optimum where the triangulation cannot be improved by swapping any edges in the triangulation, but there still exists a better triangulation. And sometimes, the local optimum differs greatly from the global one. Moreover, it is almost impossible to determine whether the resulting triangulation is a global optimum or just a local one. An edge e is *swappable* if the quadrilateral Q associated with e is convex and there are no 3 points of Q lying on the same line. The local swap algorithm only swaps swappable edges. It repeatedly selects a swappable edge whose swap can improve the quality of the triangulation until no such edges are available.

The strategy about how to select an edge to swap has the greatest impact on finding the final optimum. Experience shows that selecting the edge whose swap leads to greatest improvement each time performs very well in most cases [14]. This maximal reduction strategy also finds a local optimum with fewest swaps. Thus, it runs fastest. In order to speed up the process of picking up the edge with greatest improvement, usually it is helpful to sort the edges in the mesh according to their swap values in an ordered heap Q. Then, we will repeat

selecting the top edge from Q to swap and update the swap values of its neighborhood until the swap value of the top edge in Q is equal to or less than zero.

However, sometimes, the maximal reduction strategy stops in a poor local optimum, because lots of edges become interior to non-convex quadrilaterals and thus cannot be swapped anymore.

The local swap algorithm can be modified slightly to find an equal or better triangulation [13]. It calculates the swap value $SwapValue(e)$ of each edge e in the same way as before in the beginning. However, when $SwapValue(e)$ of an edge e is less than 0, it will replace its swap value with another swap value $SwapValue^{+}(e)$. The new swap value $SwapValue^{+}(e)=SwapValue(e)+Max\{SwapValue'(e_i)\}$ where e_i is one of the 4 edges of the quadrilateral around e, $SwapValue'(e_i)$ is the new swap value of e_i after swapping edge e. Then, the updated swap values of all edges are stored into a priority queues Q. Every time an edge with the highest updated swap value will be selected from Q. If the $SwapValue(e) > 0$, only e will be swapped. Otherwise, both e and the edge making $SwapValue^{+}(e)$ maximal will be swapped. The above procedure continues until there is no edge with a positive updated swap value in Q. Because there are more edges with positive swap values with respect to this algorithm and it allows swapping two edges in the same time when one swap cannot improve a triangulation, it will have less chance to get stuck in a local optimum. Thus, it will find a better optimum.

2.1.2.3.2 Simulated Annealing

Simulated annealing is designed to search as many triangulations as possible in order to find an optimal triangulation. It tries to avoid getting stuck in a local optimal triangulation by making some swaps that will decrease the quality of the triangulation temporarily. Thus, it can go on to find the global optimum instead of terminating too quickly. Simulated annealing still

cannot guarantee that it will certainly find a global optimum. However, it will increase the chance of finding a global optimum if parameters are set reasonably.

Simulated annealing is based on probabilistic method [25]. That is, it will swap a bad edge according to some probability. In the beginning, there are more chances to make a bad swap. As time goes by, the probability of swapping bad edges decreases gradually. If parameters are reasonably set, in the end, there are no bad swaps made at all. Thus it will reach an optimum, and most probably a global optimum. Then, the simulated annealing process will converge in the end.

The process of swapping is divided into a number of stages. We try to make a finite number of swaps at each stage. At every stage, there is a specific value t_k called the temperature that controls the probability of a bad swap. For each t_k, it meets the following condition: $t_1 > t_2 > \ldots > t_{NoOfStages} > 0$. For every stage, the probability of making a bad swap of an edge e is equal to $exp(d/t_k)$, where $d=SwapValue(e)$. We will repeat choosing an edge e in the mesh randomly. When the swap of e improves the quality of the triangulation, we make the swap definitely. Otherwise, we still make the swap with the probability of $exp(d/t_k)$, where $d=SwapValue(e)$. In general, we will choose $t_k = r^k t_0$ where $0<r<1$ is a constant and t_0 is called the initial temperature. For a bad swap, $d < 0$. Thus, the probability of making a bad swap will decreases from stages to stages. Moreover, in order to decrease the chance of getting stuck in a local optimum, we limit the number of good swaps that can be made at each stage to avoid making too many good swaps too soon.

When the parameters are set properly, the optimal triangulation that it calculates will be better than that with local swap algorithm. However, a disadvantage of simulated annealing is

that it is very sensitive to the number of stage, the initial temperature, the number of swaps at each stage and the number of good steps allowed at each stage. If the parameters are set improperly, it cannot find an optimum, even a local one. This happens when there is still possible to improve the quality of the triangulation by swapping some edges, but the algorithm has already terminated. Thus, the parameters must be carefully selected. Because in most cases, we cannot tell whether we have reach a global optimum or not, we should try parameters with different values and compare the results to find the suitable ones. Schumaker [25] suggests that a reasonable value for the initial temperature is about twice of the largest swap value of the edges in the triangulation. The reasonable values for the number of swap at each stage and the number of good swaps allowed at each stage are *5* to *10* times of the number of edges in the triangulation [25]. The lower the initial temperature, the longer the algorithm runs and thus the more chances to find a global optimum. It works best when converging process is very slow [25]. Simulated annealing algorithm usually runs much slower than local swap algorithm. In order to shorter the executing time, we can modify the algorithm a little [21]. We can add some code to terminate the program when there are no swaps made at a temperature. Furthermore, we can modify the calculation of the swap values of edges in the same way as we did for the local swap algorithm above to increase the chance to find the global optimum.

2.1.3 Decimation algorithms

Because we need to store, transmit, analyze, edit and display the surface efficiently, a full detailed model will slow down the operations on it and add unnecessary space overhead. Thus, we need to simplify it to a coarser model. Decimation algorithms aim to approximate the surface by fewer triangles, but maintain the topographic feature of the surface as accurately as possible. A good decimation method must preserve the basic shape and geometric characteristics of the model. It must preserve sharp or pointed edges. There are three types of

decimation algorithms: vertex-decimation algorithms, edge-decimation algorithms and triangle-decimation algorithms. Vertex-decimation algorithms repeatedly delete a number of points from the mesh and re-triangulate the resulting polygonal holes. Edge-decimation algorithms and triangle-decimation algorithms iteratively contract an edge and a triangle with lowest cost into a single vertex respectively. They both do not need to re-triangulate the holes after deleting a simplex.

2.1.3.1 Vertex-decimation algorithms

Vertex-decimation algorithm is the reversion of vertex refinement method. It first creates a finest possible approximation for the input data set first. Then it keeps selecting the least importance points from the mesh to delete, re-triangulating the resulting holes and optimizing the resulting triangulation to maintain a certain criterion until a certain condition is met. The termination condition can be that the error is above a certain threshold or the number of points deleted is greater than a certain number. We can speed up the deletion by sorting the vertices by the importance.

Vertex-decimation algorithms in general produce good approximations and preserve the topology of the actual surface [16]. We usually select an independent set of vertices, i.e. no two of them sharing an edge, to delete. It has several advantages. Firstly, we can delete them and re-triangulate the resulting holes in parallel. Thus, it improves efficiency. Secondly, it can reduce drift, because the neighbors of the deleting vertices will remain in the mesh. *Drift* is an occurrence that tiny errors resulted from deleting vertices can accumulate into big errors.

Like vertex-refinement methods, vertex-decimation approaches also differ in how to define the importance of a vertex, how many points to delete each time, how often to recalculate the importance of vertices and how to re-triangulate the holes and optimize the resulting triangulation after vertex decimation.

Sequential vs. Parallel: In contrast to vertex refinement methods, vertex decimation methods yield better approximation when deleting points in parallel, because it helps reduce drift [20].

Vertex importance measures: Like vertex-refinement methods, the measure of vertex importance affects the quality of the approximation. In general, the importance of a vertex should relate to how much the shape of the model will be changed after this point is deleted. Thus, a point in a flat region should own an importance close to zero, whereas, a point at a peak or a pit should have the highest importance in its neighborhood. Many heuristics for vertex importance measures define the importance of vertex based on its neighborhood [20]. These kinds of vertex importance measures are called local importance measures. However, vertex importance measures can also be based on global information. For example, we can choose the point producing the lowest sum of errors of all vertices if it is deleted to delete each time. The global error measures approximate the surface more accurately in a decimation method than a refinement method [18].

a) Degree Method

Importance (v) = Degree (v).

The point with lowest degree will be deleted from the triangulation every time. This will produce a large number of independent vertices. Because degree method does not consider any characteristics of the terrain, it can be viewed as a random deletion. However, experience shows that it works much better than simple random methods, although it performs worse than other common heuristics. It is based on the fact that the sum of the degree $\Sigma d'$ of the new triangulation is equal to $\Sigma d+(d-6)$ if we delete a vertex with degree d, where Σd is the degree of the former triangulation. Thus, deleting a vertex with high degree will create badly shaped triangles and may change the topographic features [4].

b) Height Difference Heuristic (HDH)

$$Importance\ (v) = |\sum_{i}(Height(v) - Height(v_i))|$$

Here v_i is a neighbor of v. *Height(v)* and *Height* (v_i) is the elevations of v and v_i respectively. The point whose elevation is nearest to the average of the elevations of its neighbors will be deleted from the triangulation [20].

c) Absolute Binary Heuristic (ABH)

ABH uses the difference between the number of neighbors with lower elevation and the number of neighbors with higher elevation as a vertex importance measure.

d) The volume heuristic:

The volume heuristic is a look-ahead method. It uses the difference in the volume of the neighborhood of a vertex v before and after v is deleted as important measure [20]. The *volume* of the neighborhood of a vertex v is the number of cubic units required to fill the space enclosed by the triangles adjacent to v.

e) The Drop Heuristic

$$Importance\ (v) = Height\ (v) - Height'\ (v)$$

Here *Height(v)* is the actual height of v and *Height'(v)* is the linearly interpolated elevation of v if v is deleted.

Among all the criteria for vertex selection, the Drop heuristic works the best. The Drop heuristic is a look-ahead approach, because we estimate the importance of a point by deleting the point and re-triangulating the hole. Usually, Delaunay triangulation is used to re-triangulate the hole. The Drop heuristic is an exact method, because it calculates the exact error produced after deleting a point. However, the Drop heuristic is vulnerable to drift [5]. Except the Drop heuristic, the other methods only calculate an approximate error and thus are computed faster.

Re-triangulation method:

After a vertex is deleted, the polygonal hole left must be re-triangulated. A local re-triangulation is one that does not change the edges outside the hole. Delaunay triangulation can be maintained by local re-triangulation [5]. However, local re-triangulation may not satisfy other criteria. If so, we first form an arbitrary re-triangulation. Later, we will use an edge-flipping algorithm on the whole triangulation to calculate an optimal triangulation. The re-triangulation methods to form an arbitrary triangulation include ear cutting solution and minimum angle method [12]. Minimum radius of circumcircle method is used to locally re-triangulate the hole of Delaunay triangulation [5].

a) Ear cutting solution (O' Rourke; Hinker and Hansen) [12]

This method re-triangulates the hole of n vertices by repeatedly connecting two adjacent edges that form an angle smaller than π until $n-2$ new triangles have been created. However, this sometimes yields edges that intersect with others. We can avoid such occurrences by testing whether the new triangle formed by adding a candidate edge e will include any origin points of all other edges in the polygon. If so, e will not be added into the mesh.

b) Minimum angle method [12]:

Minimum angle method sorts the inner angles of the hole and always adds the edge connecting the minimum angle. This works faster than the ear cutting solution.

c) Minimum radius of circumcircle [5]:

This method re-triangulates the hole by repeating adding the edge incident to a potential triangle with the smallest radius of its circumcircle.

2.1.3.2 Edge collapsing method

Edge collapsing methods preserve the geometric features of the real surface better than vertex-deletion methods. We can generalize edge-collapsing methods by allowing the

contraction of vertex pairs that do not share edges. Vertex-pair contraction methods have become popular recently, but they fail to preserve the topology of the surface [22]. Edge collapsing methods or, in more general, vertex-pair contraction methods differ in how they select an edge to collapse. For example, Heckbert and Garland use quadric error metrics to measure the importance of vertex pairs [19].

2.1.3.3 Triangle collapsing method

Triangle collapsing method can be used to construct progressive meshes that have different levels of detail (LOD). It is simple and fast because it does not need to re-triangulate holes and it can quickly decrease the number of simplexes in the triangulation.

2.2 Geometric criteria for optimal triangulations

Criteria for optimal triangulations define the quality of a triangulation and define what is an optimal triangulation. In order to measure a triangulation more exactly, a criterion often maps a triangulation into a real number and uses it as the quality of a triangulation. We denote the quality of a triangulation as $Quality(\triangle)$. Optimal triangulation is the one that has the highest quality among all possible triangulations. We can define the quality of a triangulation in different ways. Firstly, because the shape of triangles is a significant feature of a triangulation, many criteria often relate the quality of a triangulation to the angles, edge lengths, height and area of a triangle. Secondly, there are also many criteria based on the curvature of the surface, because curvature is another significant feature of the surface. In addition, some criteria relate the quality of a triangulation to the elevation difference between the approximation and the real surface. The quality of a triangulation is the maximum, minimum or sum of the above measures of all triangles. An optimal triangulation is the one minimizing or maximizing a certain measure μ. We can denote many of the common criteria as Max-Min (maximizing the

minimum of certain measure), such as Max-Min angle, or Min-Max (minimizing the maximum of certain measure), such as Min-Max angle. Max-Min, Min-Max criteria often have two forms, classic form and vector form, also called lexicographical form. With respect to classic Max-Min μ criterion, such as classic Max-Min angle criterion, the optimal triangulation is the one whose smallest μ is maximal, whereas, with respect to lexicographical Max-Min μ criterion, we will sort the set $\{\mu\}_{\Delta}$ in $\triangle$ in increasing order $\{\mu_1, \mu_2, \ldots \mu_t\}$. Then the optimal triangulation is the one who has the largest vector $\{\mu_1, \mu_2, \ldots \mu_t\}$ in the lexicographical order. Vector form is more discriminating than the classic form.

The criteria can also be divided into two-dimensional criteria and data-dependent criteria. Two-dimensional criteria only use the x and y coordinates of points, whereas data-dependent criteria also consider the z coordinates.

2.2.1 Two-Dimensional Criteria

Many two-dimensional criteria are related to the angles of the triangles. The most popular two-dimensional criteria are Min-Max angle and Max-Min angle criteria. They prevail because they produce good shape triangles. Study has shown that the angle is a good indicator to decide what kind of triangulation produce better result. Angles that are too sharp or too flat are not good [9]. The error of a triangulation is related to the minimal and maximal angle of the triangulation [8]. Min-Max angle and Max-Min angle criteria both try to reduce the occurrence of thin and long triangles [10]. Besides, some criteria place a limit on the range of the angles. However, these criteria sometimes cannot be implemented successfully. For example, it is not always feasible to use acute angles only [26]. Thus, a criterion can only limit the occurrence of obtuse angles.

Besides, length is also a good choice, because the length of the longest edge is inversely proportional to the sharpness of certain error bounds [26]. The criteria with respect to the lengths of edges of the triangle are Min-Max length criterion, Max-Min length and one minimizing the sum of edges length.

Criteria relative to angles and edges are popular since the error bounds can be denoted in terms of them. Thus, some criteria are suggested which optimize both. For example, there exists a criterion relative to the aspect ratio, the ratio of the longest edge to the altitude from this edge [26]. Another common criterion maximizes the minimal area of the triangles in the triangulation.

In addition, some criteria are based on the heights of triangles in the triangulation, because the minimal height of all triangles in the triangulation relates to how good the approximation is [8].

Moreover, some other two-dimensional criteria also exist. For example, there is one criterion relevant to the degree of a vertex [26], because the degree tells the importance of this vertex. Another one is Min-Max eccentricity criterion [26].

Below we give more detail on the specific criteria that will be implemented for our comparisons.

a) Min-Max angle

Min-Max angle criterion defines that the quality of a triangulation $\triangle$ is the maximum angle among all the triangles in the triangulation. The optimal triangulation is the one whose maximum angle is the smallest among all possible triangulations.

$Quality(\triangle) = Max_{\Delta}\ (Angle)$.

Min-Max angle criterion is based on the fact that too flat angles are not preferred [9]. The angle used here is a two-dimensional angle. It can be denoted as $arccos\frac{(x_1-x_0)(x_2-x_0)+(y_1-y_0)(y_2-y_0)}{\sqrt{(x_2-x_0)^2+(y_2-y_0)^2}\sqrt{(x_1-x_0)^2+(y_1-y_0)^2}}$ where (x_0, y_0) is the coordinates of the origin of the angle, (x_1, y_1), (x_2, y_2) are the coordinates of the two end points.

Edge insertion paradigm will find an global optimum with respect to Min-Max angle criterion [8]. Incremental insertion algorithm and local swap algorithm can also find a quite good optimal triangulation with respect to Min-Max angle criterion, although they cannot always find out the global optimum.

b) Max-Min angle

Max-Min angle criterion defines that the quality of a triangulation $\triangle$ is the minimum angle among all triangles in $\triangle$. The optimal triangulation is the one whose minimum angle is the largest among all possible triangulations.

$Quality(\triangle) = Min_{\Delta}(Angle)$.

The optimal triangulation with respect to max-min angle criterion is called Delaunay triangulation. It is the most popular triangulation. This is in part due to the fact that a Delaunay triangulation optimizes several criteria simultaneously: Max-Min angle, Min-Max circumcircle and Min-Max min-containment circle [26]. The circumcircles of all triangles in Delaunay triangulation enclose no vertices of other triangles. Delaunay triangulation usually produces good shaped triangles. It decreases the occurrence of thin and long triangles, because slivers are undesirable for applications such as Finite Element Modelling and graphical rendering. Slivers may lead to numerical inaccuracies in the calculations of finite element modeling and produce visual discontinuities in smoothly shaded surfaces [11].

Delaunay triangulation can be implemented by edge swapping algorithm, dynamic programming, divide-and-conquer algorithm, sweep-line algorithm, randomized incremental algorithm and so on. Many of these algorithms can calculate a Delaunay triangulation in *O(nlogn)* time [26]. Studies show that the divide-and-conquer algorithm runs fastest, the sweep-line algorithm is the second fastest. The incremental algorithm works slower than the above two, because it spends lots of time in locating points. Re-triangulating a convex polygon by Delaunay triangulation can be done in *O(n)* time.

Local swap algorithm based on the circle test will also find the global optimum. Local swap algorithm will find the global optimum with respect to lexicographic Max-Min angle criterion [25], but it only finds a local optimal one if based on other criteria of Delaunay triangulation [18].

Delaunay triangulation minimizes the roughness of the approximations of terrains. It preserves the intrinsic properties of the real surface, but it does not preserve the extrinsic properties of the surface, such as the convexity. For instance, Delaunay triangulation may produce a concave edge when approximating a sphere [2]. Sometime, it will cause artificial break lines that do not exist in the original surface if swapping an edge inside a concave quadrilateral.

2.2.2 Data-dependent Criteria

Because the z-coordinates of points contain the topographic information of a surface, they affect the quality of an approximation. Data dependent triangulations indicate the gradual changes of the z coordinates of points of the surface and therefore, produce a smoother approximation. It can lead to lower error approximations than two-dimensional triangulations when approximating certain surfaces, such as ruled surfaces [18]. Although Delaunay triangulation gives good approximations for lots of surfaces, the triangles of the optimal

triangulation may not simply have edges with similar lengths. Instead, they may have long edges in the direction of minimum curvature and short edges in the direction of maximum curvature [26]. Long and thin triangles are necessary when approximating certain kind of surfaces. They in turn may reduce the approximation error.

Moreover, the optimal triangulation of a surface should preserve the intrinsic and extrinsic geometric properties of the actual surface [2]. Delaunay triangulation in general does not preserve some extrinsic properties of the actual surfaces, such as the convexity of the actual surface. At this aspect, tight triangulation, one of data-dependent triangulations, performs better than Delaunay triangulation, because it preserves convexity.

Many data-dependent criteria are based on different kinds of curvatures, because for a smooth surface, its curvatures are an essential characteristic. The primary kinds of curvatures for a surface are Gaussian curvature K, mean curvature H and principal curvatures k_1, k_2. It is enough to determine the shape of a surface simply with its mean curvature and Gaussian curvature. The Gaussian curvature of a point indicates whether its neighborhood is elliptic, hyperbolic or parabolic. Mean curvature of a point tells whether its neighborhood is convex or concave. If $H>0$, its neighborhood is convex, otherwise, concave. The signs of Gaussian curvature and mean curvature distinguish eight basic shapes. $H>0$, $K>0$ indicates a peak region. $H=0$, $K=0$ indicate a flat region. $H<0$, $K>0$ is a pit region. $H=0$, $K<0$ is a minimal region. $H>0$, $K=0$ a ridge region. $H>0$, $K<0$, a saddle ridge region. $H<0$, $K=0$ a valley region. $H<0$, $K<0$ is a saddle valley region. K and H are both defined in terms of k_1 and k_2. $K=k_1 k_2$, $H=(k_1+k_2)/2$. If the elevation of a point (x, y) on surface is denoted as $f(x, y)$, K and H of this point are $K=\frac{f_{xx}f_{yy}-f_{xy}^2}{(1+f_x^2+f_y^2)^2}$, $H=\frac{(1+f_x^2)f_{yy}-2f_xf_yf_{xy}+(1+f_y^2)f_{xx}}{2(1+f_x^2+f_y^2)^{3/2}}$ [23]

respectively. However, when we use a set of triangles to approximate the actual surface, the triangle mesh have no curvature at all because all triangles are flat and the curvature is not C^2 differentiable at the vertices and along edges of the mesh. Thus, the above formulae are not suitable for polyhedral surfaces. So the analogues of curvatures for polyhedral surfaces are required when criteria for optimal triangulations are based on these curvatures. K and H for polyhedral can be deducted by approximating the polyhedral with a smooth surface. If we replace every edge in the mesh with a tiny cylinder that joins the adjacent triangle tangentially and mix the cylinders smoothly at the vertices, K and H are integrable in a C^2 way. Then K and H can be calculated [13]. In fact, the analogue of integral Gaussian curvature is based on the angles around the vertex, because the concept of Gaussian curvature is close related to that of angle [3], whereas, the integral mean curvature is related to the angles of the normals of two neighboring triangles. Then, K and H of a vertex v can be inferred by dividing the integral Gaussian curvature and mean curvature by the area around the vertex respectively.

The optimal triangulations with respect to criteria related to certain curvatures often are those with the minimum of sum of the curvatures. For example, tight triangulation minimizes the sum of absolute Gaussian curvature.

In addition to criteria related to curvatures, several data-dependent criteria are based on different kinds of error metrics, because these criteria can reduce the error of the approximation. For example, least squares fit criterion minimizes L_2 error, the sum of squares of elevation differences over all input points. L_∞- optimal triangulation minimizes L_∞, the maximum error of the triangulation [18].

Some data-dependent criteria are based on certain functionals, such as the energy of a bending plate [3]. The optimal triangulation minimizes the functional. These functionals often

measure some geometrical properties of the surface. However, these criteria can only deal with functional data.

Besides, some other criteria also exist. These include minimizing the total area, minimizing the volume of the surface, Min-Max slope and so on. Slope measures the steepness of the surface where the slope of a flat surface is *0*.

Moreover, we can combine several criteria to find an optimal triangulation: Rippa [14] combined the Max-Min angle criterion and the one minimizing the sum of angles between normals (ABN) to determine an optimal triangulation. He compared the global error producing by Delaunay triangulation and the data-dependent triangulation on every pass and always used the one with lower error to update the triangle mesh. He found that this hybrid criterion produces better approximations than pure criteria in most cases [17]. However, hybrid criteria are slower than the pure criteria.

The amount of data-dependent criteria is much less than that of *2*-dimensional criteria. In addition, all of them have weaknesses. Data-dependent triangulations often create more slivers. The points inside a thin triangle are further to its vertices than points on a fat triangle. Thus, slivers can lead to inaccurate approximation because they interpolate the points on them over a long distance. Moreover, slivers give an artificial visual effect because of aliasing. In general, as well, data-dependent triangulations are more expensive than two-dimensional triangulations. The general algorithm calculating the optimal data-dependent triangulation is edge-flipping algorithm. It cannot guarantee to find the global optimum in most cases [26]. Furthermore, data dependent triangulations tend to ignore some geometric features, such as ridges and faults [28].

Some data-dependent triangulations perform better than Delaunay triangulations when approximating smooth surfaces, but they produce much worse result when dealing with non-smooth surfaces.

Next, we will give some more detail of the data-dependent triangulations that we will implement for our comparison experiments.

a) Minimizing Absolute Gaussian Curvature

The quality of a triangulation $\triangle$ is defined as the sum of the absolute Gaussian curvature of all the vertices in $\triangle$. An optimal triangulation has the minimal total absolute Gaussian Curvature.

$$\textit{Quality}(\triangle) = \sum_{v} |K(v)| = \sum_{v \in convex} |K(v)| + \sum_{v \in saddle} |K(v)| + \sum_{v \in mixed} |K(v)|.$$

Here $K(v)$ is the integral Gaussian curvature around vertex v.

Before calculating the curvatures of vertices, let us give some definitions. [3]. A *supporting plane* is a plane through a vertex v in the triangulation provided that all the neighboring vertices of v lie on the same side of or on the plane. *The star of a vertex* v is formed by all triangles incident to v. A *proper convex vertex* is such a vertex v that all triangles incident to v are supporting planes. A *proper saddle vertex* is such a vertex v that there does not exist any supporting plane through v at all. A *mixed vertex* is a vertex v provided that there exists a supporting plane through v, but there also exists a plane formed by two edges on same triangle that is incident to v such that the neighboring vertices of v lie on both side of the plane. A vertex that is neither proper convex vertex nor proper saddle vertex is a mixed vertex.

The calculation of the absolute integral Gaussian curvature $|K(v)|$ around a vertex can be divided into three steps:

I. Determining the type of a vertex:

The vertices in the triangulation can be divided into three classes: proper convex vertices, proper saddle vertices and mixed vertices. In order to find out a vertex v belongs to which kind, we must find out whether there exists a supporting plane and whether all triangles incident to v are supporting planes. We use the following theorem: Edges e_i, e_j incidental to v form a supporting plane, if and only if for all other edges e_k incidental to v ($k != i$ && $k != j$), $e_i \times e_j \bullet e_k >= 0$ [13]. If every two adjacent edges incidental to v form a supporting plane, then v is a proper convex. If no two edges incidental to v form a supporting plane, then v is a proper saddle vertex. Otherwise, v is a mixed vertex.

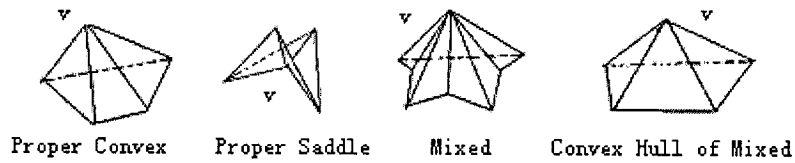

Figure 4: Vertex Type (based on [3])

II. Calculating the convex hull of edges incidental to *v*, if *v* is a mixed vertex:

We can calculate the convex hull in two ways. One tries to find out a plane that can intersect with every edge around v. Then it calculates all the intersection points with the plane. After that it will calculate the convex hull of these points. The other approach is to test every two edges e_i, e_j incidental to v to see if the plane formed by e_i, e_j is a supporting plane or not. If so, then e_i and e_j are on the convex hull. This algorithm tries to determine all edges on the convex hull.

The first algorithm works faster than the second one, but it finds no solution when two neighbors of v and v are on the same line, because there will be no plane intersect with all the edges around v.

III. Calculating the absolute integral Gaussian curvature around a vertex:

The absolute integral Gaussian curvature $|K(v)|$ around a convex vertex v : $|K(v)| = 2\pi - \sum_{i=1}^{d} \theta_i$, where θ_i is an angle formed by two adjacent edges incident to v, d is the degree of v. For a convex vertex, $\sum_{i=1}^{d} \theta_i < 2\pi$. $|K(v)|$ around a proper saddle vertex v : $|K(v)| = |2\pi - \sum_{i=1}^{d} \theta_i| = \sum_{i=1}^{d} \theta_i - 2\pi$. For a proper saddle vertex, $\sum_{i=1}^{d} \theta_i > 2\pi$. $|K(v)|$ around a mixed vertex v: $|K(v)| = 2\pi - 2*\sum_{j=1}^{d^+} \theta_j^+ + \sum_{i=1}^{d} \theta_i$, where θ_j^+ is an angle of two adjacent edges around v of the convex hull of the star of v, d^+ is the degree of the convex hull. For a mixed vertex, $d^+ < d$, $\sum_{i=1}^{d} \theta_i > 2\pi$ and $\sum_{j=1}^{d^+} \theta_j^+ < 2\pi$. $|K(v)|$ of a vertex v on the boundary of the mesh: $|K(v)| = \pi - \sum_{i=1}^{d} \theta_i$. No matter what type a vertex is, the flatter the neighborhood around v is, the smaller of $|K(v)|$ is.

A 2-dimensional manifold M is called tight if all points of M lie on the same side of each hyper-plane that passes a point of M, but not its neighborhood [2]. A tight polyhedron has the minimum total Gaussian curvature [2]. Thus, the triangulation minimizing absolute Gaussian curvature is call tight triangulation. Tightness generalizes the concept of convex [2]. Thus, tight triangulation maintained convexity automatically [3]. It is the smoothest one also [2].

It has been proved that local swap algorithm can actually find out the global optimum with this criterion if the surface is convex [3].

Because Gaussian criterion tries to minimize the total absolute Gaussian curvature, it will work well for the surface whose Gaussian curvature is zero everywhere, such as cylinders, cones, or surfaces whose points (x, y, z) can be expressed as $z=f(x)+ay$ or $z=f(y)+ax$. Since $K=k_1 k_2$, one of the two principal curvatures k_1, k_2 will be equal to 0 if K is 0. Thus, the optimal case for Gaussian criterion is the surfaces with zero curvature in one of the principal directions and nonzero one in the other principal direction.

However, tight triangulation tends to consist of many long thin triangles [3]. Thus, some edges around a vertex are too close to each other. Then it is easy to cause numerical errors and inconsistency when we test whether a plane is a supporting plane or not and when we calculate the convex hull if it exists. Moreover, tight triangulation works poorly for general surface, such as terrains [18]. It is also a very expensive triangulation method due to its heavy calculation.

Swap value: We can use edge-flipping algorithm to calculate the optimal triangulation. Because $Quality(\triangle)$ only differs from $Quality(\triangle')$ in the sum of the integral absolute Gaussian curvatures $\sum_{i=1}^{4} K(v_i)$ of the four vertices v_i of the quadrilateral Q surrounding edge e and the new sum of the integral absolute Gaussian curvatures $\sum_{i=1}^{4} K'(v_i)$ of the same four vertices after swapping e, $SwapValue(e)=Quality(\triangle')-Quality(\triangle)= \sum_{i=1}^{4} K(v_i) - \sum_{i=1}^{4} K'(v_i)$ where v_i is a vertex on the Q.

b) Minimizing Absolute Mean Curvature

The quality of a triangulation $\triangle$ is the sum of the absolute integral mean curvature $|H(e)|$ along every edge e in $\triangle$. The optimal triangulation minimizes $\sum_{e} |H(e)|$ [3].

$Quality(\Delta) = \sum_e |H(e)| = \sum_e a(e)\,|e| = \sum_e \arccos\left(\frac{n_1 \bullet n_2}{|n_1| \bullet |n_2|}\right)|e|.$

If e is an inner edge, $|H(e)| = |e| * \alpha(e)$, where $\alpha(e)$ is the angle between the normals $\{n_1, n_2\}$ of the two triangles shared e and $|e|$ is the length of e. Here $\alpha(e) = \arccos\left(\frac{n_1 \bullet n_2}{|n_1| \bullet |n_2|}\right)$. If e is on the boundary, $H(e)=0$.

Mean curvature measures how bent a surface is [1]. This criterion minimizes the roughness of the approximation of a surface.

Other relevant criteria:

Because we use cylinders to approximate the edges when calculating the integral mean curvature of edges and only one of the two principal curvatures of cylinders are nonzero, the points along edges will have only one principal curvature with nonzero value too. Moreover, because the mean curvature of vertices and the points inside triangles are all 0, the criterion above also minimizes the following functionals: $\int_\Omega (|k_1| + |k_2|)dS$, $\int_\Omega |k_1 + k_2|\,dS$, $\int_\Omega \max(|k_1|, |k_2|)dS$, $\int_\Omega \sqrt{k_1^2 + k_2^2}\,dS$ [3].

There are other criteria relative to mean curvature also. One is the jump in normal derivatives (JND). Experiences show that the mean curvature criterion and JND produce almost the same results [3]. JND also minimizes the roughness of the approximation [3]. Another relevant criterion minimizes the thin plate energy functional [3]. For a spline s defined on a triangulation Δ, the thin energy of s is $\sum_{i=1}^{N} \int_{T_i} (s_{xx}^2 + 2s_{xy}^2 + s_{yy}^2)dxdy$, where T_i is a triangle of Δ. A *spline* is a mathematical function that gives an interpolation or approximation of a finite number of input points. Because s_x and s_y of a smooth interpolating surface are

near zero, $\sum_{i=1}^{N} \int_{T_i} (s_{xx}^2 + 2s_{xy}^2 + s_{yy}^2) dxdy$ is almost equal to $\int_{\Omega} (k_1^2 + k_2^2) dS$. It is also equal to minimizing $\int_{\Omega} H^2 dS$ [2]. We often use this functional to smooth approximating surfaces. Experiences show that minimizing $\sum_{e} |H(e)|$ gives a similar result as minimizing $\int_{\Omega} H^2 dS$. It is hard to tell which of them are better [13]. The optimal triangulations with these two criteria improve the visual appearance of the initial triangle mesh significantly and enforce the sharp edges in the mesh if the actual surfaces are smooth [13]. The minimizations of any of them also decrease the error of the initial triangle mesh significantly [13].

However, it is not known whether the optimal triangulations generated with respect to mean curvature criterion maintain convexity or not [3], although in most cases, minimizing the two criteria above yield convex triangulations, no matter whether the initial one is convex or not [13].

c) Minimizing Area of the Surface

The quality of a triangulation $\triangle$ is the sum of the area *Area(T)* of every triangle T in $\triangle$. The optimal triangulation minimizes $\sum_{T} Area(T)$.

$Quality(\triangle) = \sum_{T} Area(T)$.

A surface is minimal if its mean curvature is close to *0* [2]. The triangulation minimizing area of the surface will lead to a saddle surface, because only the region with negative Gaussian curvature will have zero mean curvature [2]. Thus, the optimal triangulation with respect to this criterion does not preserve convexity and it sometimes results in a poor approximation, even if the original surface is quite simple. For example, the approximation of a cylinder becomes skewed [3].

Other relevant criteria:

This criterion is a data-dependent one, because we not only use x, y coordinates, but also use z coordinate when calculating the area of a triangle. However, there are also other criteria that only use the x and y coordinates to calculate the area of a triangle. One is Max-Min area criterion.

d) Minimizing Angle Between Normals(ABN)

For an edge e in a triangulation Δ, let $\alpha(e)$ is the angle between the normals $\{n_1, n_2\}$ of the two triangles shared e. $\alpha(e) = \arccos\left(\frac{n_1 \bullet n_2}{|n_1| \bullet |n_2|}\right)$. If we arrange all such angles in Δ in increasing order $\{\alpha(e_1), \alpha(e_2), \ldots\}$, the quality of Δ is the corresponding vector $\alpha(\Delta)$ of $\{\alpha(e_1), \alpha(e_2), \ldots\}$. The optimal triangulation has the minimal vector in the lexicographical order.

$Quality(\Delta) = \alpha(\Delta) = (\alpha(e_1), \alpha(e_2), \ldots)$.

The larger the angle between the normals of two triangles that share an edge e, the more bent the neighborhood of e is. Thus, the triangulation minimizing ABN minimizes the roughness [3]. Therefore, when approximating smooth, functions, ABN produces more accurate approximations than Delaunay triangulation. However, the triangulation minimizing ABN does not preserve the discontinuities of the original surface. Discontinuities often indicate significant characteristics of the surface, such as ridges lines and creases.

Other relevant criteria:

Besides the criterion above, there exist several criteria relative to the angle between normals. One is mean curvature criterion [3]. Another one minimizes the sum of the absolute values of the angles between normals of two triangles shared an edge. This criterion works well

in most cases when conducting the experiences on several smooth, synthetic functions [17]. It performs better than Delaunay triangulation on such functions. A third one minimizes the sum of the squares of ABN.

e) Minimizing L_2 error (Least squares fit criterion)

Let $H(x_k, y_k)$ be the elevation of a vertex (x_k, y_k) on the actual surface and $H'(x_k, y_k)$ be its interpolated approximation. The sum of the squares of elevation errors over all input points is called L_2 measure. It can be denoted as $\sum_{k=1}^{m}(H(x_k, y_k) - H'(x_k, y_k))^2$. The optimal triangulation is the one with minimal L_2 error. It is also called least squares fitting criterion.

$$Quality(\triangle) = \sum_{k=1}^{m}(H(x_k, y_k) - H'(x_k, y_k))^2 .$$

The triangulation with this criterion creates very good approximation. It can lower the error of the approximation. If the number of the points added $m \to \infty$, the error of the L_2-optimal triangulation converges as m^{-1} [18]. However, L_2 criterion sometimes produces very long and thin angles that cannot fit the surface well [17]. The algorithm cannot eliminate those slivers because they are too thin to contain any points and therefore add no error to L_2 measure.

Other relevant criteria:

Min-Max error criterion is also based on the elevation differences between input points and their approximations. It minimizes the maximum of the local approximation errors. However, Min-Max error criterion is vulnerable to outliers and thus performs worse than L_2 error measure.

2.3 Terrain Modeling

There are three common ways to represent and store the terrain. They are DLG, DEM and TIN.

a) DLG (Digital Line Graph)

DLG stores polygonal lines. Every polygonal line is a set of (x, y) coordinates that have a common elevation. There are many DLG files available, but DLG files are difficult to handle [5].

b) DEM (Digital Elevation Model)

DEM stores the elevations of points on the surface at regular intervals. The x and y coordinates of a vertex are implicit in the grid. DEM is popular because it is ease to use. The neighborhood of a point can be located fast. There exist lots of example DEM files. A drawback of DEM is that it cannot adapt to the irregularity of the terrain. Thus, redundant data will be stored when the part of surface of the terrain is flat.

c) TIN (Triangulation Irregular Network)

TIN stores a set of triangles that approximate the surface. TIN requires more storage for a vertex than DEM, because it must store the three coordinates of a vertex. However, because TIN adapts to the irregularity of the terrain, for the part of surface where topographic information is little, the number of vertices to approximate it is much less than DEM. Moreover, it can also express the features of a terrain. For instance, peaks can be represented by the vertices of TIN, whereas, ridges can be represented by the edges of TIN. However, there exist fewer TIN files than DEM and DLG files, but many algorithms are available to convert DEM or DLG files to TIN files.

2.4 Data structures

There are several popular data structures to hold the information of the resulting triangulation, such as Quad-Edge, DCEL, winged-edge data structure and so on.

a) Quad-Edge

Quad-Edge represents the subdivision of a manifold and its dual in the same time. It stores every undirected edge in the subdivision four times, two directed edges for itself and two for its dual. Every edge in the mesh stores a pointer to its origin, a pointer to its dual edge and a pointer to the counterclockwise next edge around its origin. The dual edge of a directed edge is the one that points from the face on it right to the one on its left. All other topological operators are defined in terms of the above pointers. Quad-Edge is preferred because it is elegant, it stores a graph and its dual in the same time and there is implementation of Quad-Edge structure available. For example, it will store Delaunay triangulation and its dual, i.e. Voronoi diagram simultaneously.

b) DCEL

DCEL stores a record for every face, edge and vertex of the subdivision. The vertex record of each vertex v stores its coordinates and a pointer to a half-edge originated from v. The face record of each face stores a pointer to a half-edge on its outer and inner boundary respectively. The half-edge record of a half edge e stores a pointer to its origin, a pointer to its twin half-edge, a pointer to the face bounded by it, a pointer to its next and previous half-edge on the boundary of the face respectively.

2.5 Accuracy Measures

Firstly, we can evaluate the fidelity of a triangulation by visual similarity. A good triangulation should avoid visual artifacts and enhance the geometric features of a surface.

Since slivers may distort the appearance of the approximation because of aliasing, a good triangulation should limit the amount of slivers. Both Min-Max angle and Max-Min angle criteria try to avoid long and thin triangles, whereas, most data-dependent triangulations produce many slivers. However, visual similarity is quite subjective and vague as an accuracy measure.

Secondly, we can use the difference in elevation between the actual surface and its approximation as accuracy measure. Geometric error measures evaluate the accuracy of an approximation more objectively and precisely. Commonly used elevation error measures include L_2 error, L_∞, RMS error, total error and average error.

Thirdly, we can use certain derivative statistics, such as mean curvatures, as accuracy measures. Experience shows that mean curvature as an accuracy measure gives similar results to the elevation error measure to some extent [5].

Furthermore, we can check whether the triangulation with respect to certain criteria preserve the geometry of the actual surface. An optimal triangulation should express both the intrinsic and extrinsic geometry of the real surface so that we can extract the properties of a surface from its approximation. Intrinsic properties of a surface are those determined only by the surface itself, whereas, extrinsic properties are those relevant to the way the surface is embedded in the space. Intrinsic properties include the length of a curve on the surface, angle between two curves on surface, area, Gaussian curvature and Gaussian map. The extrinsic properties include mean curvature, convexity and concavity, extrinsic curvature and so on. Delaunay triangulation almost preserves intrinsic geometry of surfaces, but it does not preserve extrinsic properties, such as convexity, in many cases [2]. Minimal area criterion does not

preserve both intrinsic and extrinsic geometry [2]. As mentioned previously, the Gaussian curvature criterion preserves the convexity of the surface.

Chapter 3

Algorithms

In this section we discuss all algorithms we implement in order to compare the following criteria: Max-Min angle criterion, Min-Max angle criterion, minimization of total Gaussian curvature, minimization of mean curvature, minimization of surface area, minimization of normals between angles, and minimization of L_2 error.

We implement both the refinement and the decimation of surfaces with respect to above criteria. The algorithmic paradigms we implement include greedy insertion, dynamic programming, local swap, simulated annealing and vertex decimation. We use greedy insertion to implement Delaunay triangulation and Min-Max angle triangulation for surface refinement. Although greedy insertion algorithm cannot guarantee to find the global optimum with respect to Min-Max angle criterion, it still works quite well for it. Moreover, we also discuss edge insertion paradigm, because it can find the global optimum with respect to the Min-Max angle criterion. We use local swap algorithm or simulated annealing to calculate the optimal triangulation with respect to ABN criterion, L_2 measure criterion, Gaussian curvature criterion and mean curvature criterion after points are inserted or deleted. We use a modified greedy insertion algorithm combined with dynamic programming to calculate the optimal triangulation with respect to Min area criterion for surface refinement. Furthermore, we implement a vertex-decimation algorithm for surface decimation.

In order to compare the results of different criteria, we also generate some terrains automatically. We use the fault algorithm [15] to create random terrains and Diamond-Square algorithm [24] to yield terrains with significant features.

3.1 Triangulation Algorithms

Here we will give more detail on the triangulation algorithms that we have implemented.

3.1.1 Greedy insertion algorithm [18]

Greedy insertion algorithm:

Create an initial triangulation $\triangle$ with two triangles containing all points

for (every triangle T in $\triangle$) {

Scan T to find out the unused point v with highest local error

Insert (T, v) into an order heap Q ordered by the importance of vertices

}

while (error>error threshold **and** number of points inserted<certain amount) {

Pop the top pair (T, v) from Q

Labeled v as used

Insert v into T

if (v lies on an edge e)

Delete e

Add spokes from v to the vertices of the surrounding polygon P

for (every edge e of P) {

if (e does not conform certain criterion C) {

Swap e

Add two edges of the quadrilateral around e not adjacent to v into P

}

}

for (every triangle T adjacent to v) {

Scan T to find out an unused point v with highest local error

if (*T* is found in $\mathcal{Q}$)

Update the old pair *(T, ...)* with *(T, v)* in $\mathcal{Q}$

else

Insert *(T, v)* into $\mathcal{Q}$

}

}

end

Variation: The test whether an edge *e* conforms to the Max-Min angle criterion is based on whether *e* meets the empty circumcircle condition. With respect to other criteria *C*, such as Min-Max angle criterion, an edge *e* is considered to meet *C* if the swap value of *e* with respect to *C* is smaller than or equal to *0*.

The above algorithm can also be modified to calculate the optimal triangulation with respect to Min area criterion. After a point *v* is inserted, we will not add spokes from *v* to its neighbors. Instead, we will delete all the edges of the containing polygon of *v*. Then we will use dynamic programming to find a minimum area re-triangulation of the resulting polygonal hole.

Complexity: Greedy insertion algorithm can be implemented in time $O((m+n)\log n)$ and space $O(m+n)$, where *m* is the number of input points and *n* is the number of points in the mesh [18].

3.1.2 Edge insertion algorithm [26]

Edge insertion algorithm:

Establish an initial triangulation $\triangle$ of the surface

do {

$\triangle_{last} = \triangle$

Find an unused worst triangle $\triangle xyz$ in $\triangle$

Mark $\triangle xyz$ as used

Let y be the anchor of $\triangle xyz$

for (every vertex pair *(y, t)* in $\triangle$) {

Let $\triangle' = \triangle$

Add edge yt to $\triangle'$

Delete all edges in $\triangle'$ that intersect yt

Retriangulate the holes by cutting ears whose qualities > $\triangle xyz$

if (the retriangulation of the two holes succeed) {

Let $\triangle = \triangle'$

break

}

}

} **while** ($\triangle_{last}$!= $\triangle$ **and** there exists an unused worst triangle in $\triangle$)

end

Complexity: Edge insertion paradigm can find the optimal triangulation with respect to Min-Max angle criterion or Max-Min height criterion in time $O(n^2 \log n)$ and space $O(n)$, with respect to Min-Max slope criterion or Min-Max eccentricity criterion in time $O(n^3)$ and space $O(n^2)$ [26].

3.1.3 Dynamic programming algorithm [26]

Definition: given triangulations $\triangle$ of simple polygons, a quality measure *(Quality:*$\triangle \rightarrow R$*)* is *decomposable,* provided that:

1. There exists a function G, for an arbitrary diagonal $\{v_i, v_j\}$ that splits an arbitrary

polygon P into two simple polygons P_1, P_2, $Quality(\triangle)=G(Quality(\triangle_1),Quality(\triangle_2),v_i,v_j)$, where $\triangle$, $\triangle_1$, $\triangle_2$ are arbitrary triangulations of P, P_1, P_2 respectively.

2. G can be compute in time $O(1)$.

3. G is monotonic in $Quality(\triangle_1), Quality(\triangle_2)$.

4. $Quality(\triangle)$ can be computed in time $O(1)$ if $\triangle$ is a triangle.

The following measures are decomposable: Maximal/minimal angle, maximal/minimal length of an edge, maximal/minimal area of a triangle and so on. If $Quality(\triangle)=Max(Angle\ in\ \triangle)$, then $G(Quality(\triangle_1), Quality(\triangle_2), v_i, v_j) = max\{Quality(\triangle_1), Quality(\triangle_2)\}$.

Given polygon $P = \{v_0, v_1, v_2 \ldots v_{n-1}\}$, let $Q(v_i, v_k, v_j)$ be the quality of the triangle formed by vertices $\{v_i, v_k, v_j\}$ and $Q(i,j)$ be the quality of polygon $\{v_i \ldots v_k \ldots v_j\}$. Array *Diagonal[i, j]* is used to recover the optimal triangulation of P.

Dynamic programming algorithm:

for (*i=0* to *n-1*)

Initialize $Q(i, (i+1)\%n)$

for (*size= 2* to *n-1*) {

for (*i = 0* to *n-1*) {

$j = (i + size) \% n$

if ($v_i v_j$ is not a diagonal)

$Q(i,j)=-\infty$

else {

$k = (i+1) \% n$

```
        do {
          if (Q(i,j)<G(G(Q(v_i,v_k,v_j),Q(i,k),v_i,v_k),Q(k,j), v_k, v_j)) {
              Q(i,j)=G(G(Q(v_i,v_j,v_k),Q(i,k), v_i,v_k),Q(k,j), v_k, v_j)
              Diagonal[i,j]=k
            }
            k=(k+1)%n
        } while (k != j)
      }
    }
}
Recover(0,n-1)
end
```

Function Recover() adds edges to *P*. We add edges to the polygon recursively:

```
Recover (i, j)
if ((j+n-i)%n<=2)
   return
k= Diagonal[i, j]
Add edge v_i v_k , edge v_k v_j
Recover(i, k)
Recover(k, j)
end
```

Variation: With respect to different criteria, dynamic programming differ in what value will be assigned to *Q(i, (i+1)%n)*, i.e., the quality of directed edges in counter-clockwise order, what

value will be assigned to $-\infty$, i.e., the negative infinity, how to compare two qualities and how to define $G(\ )$. For example, with respect to the Min area criterion, we will assign 0 to $Q(i,(i+1)\%n)$ and the largest value in the world to $-\infty$. If the area of $\triangle$is smaller than that of $\triangle'$, we will say that $Quality(\triangle)>Quality(\triangle')$. We will define

$$G(\triangle) = \sum_{T\in\Delta} Area(T)$$

$$= G\,(Quality\,(\triangle_1), Quality\,(\triangle_2), v_i, v_j)$$

$$= Quality\,(\triangle_1)+Quality\,(\triangle_2)$$

$$= \sum_{T\in\Delta_1} Area(T) + \sum_{T\in\Delta_2} Area(T)\,.$$

We use dynamic programming to re-triangulate the polygonal hole caused by deleting the edges adjacent to the inserted point with respect to the Min area criterion. Moreover, we can also use it to re-triangulate the polygonal hole caused by deleting a point with respect to Min-Max angle, Max-Min angle or Min area criteria.

Furthermore, we can also use dynamic programming to calculate the importance of a point with respect to the Drop heuristic. With respect to the Drop heuristic, the importance of a point v is defined as the difference between its real elevation and its linearly interpolated elevation if it is deleted. Thus we must find out what triangle v is located after v is deleted and the hole is retriangulated. We will use dynamic programming to retriangulate the hole. We will not use Recover() to actually retriangulate the hole. Instead, we will replace Recover() with the following function Locate() to locate the triangle that v is on. After that we can get the importance of v. Locate(v, i, j) locates the triangle T of the sub-polygon $\{v_i \ldots v_j\}$ that contain v. It will return the three vertices of T.

Locate (*v, i, j*)

k= *Diagonal[i, j]*

if (*v* is not on the right of $v_i v_k$ **and** *v* is not on the right of $v_k v_j$)

return (v_i, v_k, v_j)

if (*v* is on the right of $v_i v_k$)

Locate(*v, i, k*)

else

Locate(*v, k, j*)

end

Complexity: A triangulation of a simple polygon that optimizes an arbitrary decomposable function can be solved in time $O(n^3)$ [8]. The triangular mesh needs space $O(n)$. It uses space $O(n^2)$ to store the quality of every possible sub-polygon. The array *Diagonal[]* needs space $O(n^2)$. Thus, the space complexity is $O(n^2)$.

3.1.4 Local swap algorithm [14]

We will denote the current triangulation as $\triangle$ below.

Local swap algorithm:

Establish an initial triangulation of the input data set

for (each edge *e* in $\triangle$) {

if (*e* is swappable) {

Let $\triangle'$ = the triangulation after swapping *e* from $\triangle$

SwapValue(e) = *Quality* ($\triangle'$) – *Quality* ($\triangle$)

}

else

SwapValue(e)= $-\infty$

Insert *(e,SwapValue(e))* into an ordered heap Q ordered by swap values

}

while (Q is not empty **and** the swap value of the top pair of $Q > 0$) {

Let e = the edge of the top pair of Q

Swap e

for (every edge e' in the neighbors of e) {

if (e' is swappable) {

Let $\triangle'$ = the triangulation after swapping e' from $\triangle$

$SwapValue(e') = Quality(\triangle') - Quality(\triangle)$

}

else

$SwapValue(e') = -\infty$

Update the old pair $(e', \ldots)$ with $(e', SwapValue(e'))$ in Q

}

}

end

Variation: With respect to different criteria, local swap algorithm differs in how to calculate the swap value of an edge and what is its neighborhood whose swap values need to be updated. Figure 5 shows the neighborhood of edge e with respect to Gaussian curvature criterion and mean curvature criterion.

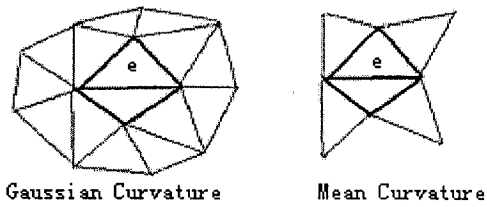

Figure 5: The neighborhood of edge e

Complexity:

The local swap algorithm can be divided into several parts:

1) Establish an initial triangulation. We use the greedy vertex-insertion algorithm to build a Delaunay triangulation of n points. This can be done in time $O(nlogn)$ [18].

2) Calculate the swap values of all edges in the initial triangulation. Suppose the time to calculate the swap value of an edge is X. Because there are $O(n)$ edges in the mesh, the total time to calculate the swap values of all edges is $O(n*X)$.

3) Sort the swap values of all edges. This can be done in $O(nlogn)$.

4) Recalculate swap values. After swapping an edge e, all the swap values of its neighborhood should be recalculated. Suppose the average number of its neighborhood is Y. Then, the time of recalculation after a swap is $O(X*Y)$. Since there exist $O(n^2)$ swaps, the total time of recalculation is $O(n^2*X*Y)$.

Therefore, the time complexity is $O(n^2*X*Y)$.

We will determine X and Y with respect to different criteria.

With respect to Gaussian curvature criterion, in order to find out the time we spend in calculating the swap value of an edge, we must first determine the time we spend in calculating the Gaussian curvature of a vertex v. The calculation can be divided into the following steps:

1. Find out the type of v. Supposed the degree of v is D. To find out whether every pair of edges forms a supporting plane or not, we must test all the other D-2 edges left. There are $D*(D-1)/2$ pairs. Thus, it can be done in time $O(D^3)$.

2. Calculate the convex hull of edges around v if v is a mixed vertex. After finding all edges forming the supporting planes, these edges form the convex hull. It takes time $O(1)$.

3. Sum the angles around v and the angles on the convex hull of the edges around v if v is a mixed vertex. They both take time $O(D)$.

Thus, the total time to calculate the Gaussian curvature of v is $O(D^3)$. Because the average degree of a vertex in the triangulation is less than 6, the total time of the calculation is $O(1)$.

For every edge e in the triangulation, in order to calculate the swap value of e, we will calculate the Gaussian curvature of the four vertices incident to the quadrilateral around e twice, one before swapping e and one after swapping e. Thus, it takes time $O(1)$ to get the swap value of e. Then X is $O(1)$.

The neighborhood of an edge e whose swap values should be recalculated is the edges on the triangles incident to the four vertices of the quadrilateral around e. The number is 8*D. Because the average of D is less than 6, the number of vertices in the neighborhood of e is $O(1)$. Then Y is $O(1)$.

Similarly, we can find out that X and Y with respect to other criteria are all $O(1)$ too. Thus, the time complexity of local swap algorithm is $O(n^2)$. The triangular mesh needs space $O(n)$. The ordered heap that stores the swap values of edges needs space $O(n)$. Thus, the space complexity is $O(n)$.

3.1.5 Simulated annealing algorithm [25]

We will denote the current triangulation as $\triangle$ below.

Simulated annealing algorithm:

Establish an initial triangulation of the surface

for (*k=1* to *NoOfStages*) {

NoOfGoodSwaps = 0

for (*j=1* to *NoOfSwaps*) {

if (*NoOfGoodSwaps* > *NoOfGoodSwapAllowed*)

exit to the outer loop

Select a random edge *e* in $\triangle$

if (swapping *e* will improve $\triangle$) {

Swap *e*

NoOfGoodSwaps= NoOfGoodSwaps+1

}

else {

Choose a random number *x* between *0* and *1*

Let $\triangle'$ = the triangulation after swapping *e* from $\triangle$

Let *d = Quality*($\triangle'$)-*Quality*($\triangle$)

if ($x <= exp(d/t_k)$)

Swap *e*

}

}

}

end

NoOfStages is the number of stages. *NoOfSwaps* is the number of swaps that can be made at each stage. *NoOfGoodSwaps* is the number of good swaps that can be made at each stage. t_k controls the probability of a bad swap.

Complexity: Simulated annealing algorithm can be implemented in time *O(NoOfStages*NoOfSwaps)*. If we let *NoOfStages* and *NoOfSwaps* both be *O(e)=O(n)*, the time complexity becomes $O(n^2)$, where e is the number of edges and n is the number of points in the mesh. The total memory cost is *O(n)*.

3.1.6 Vertex-decimation approach

We will delete an independent set of points each time.

Vertex-decimation approach:

Create a finest triangulation Δ of the input data points

if (One-pass method) {

  for (every vertex v in Δ) {

    Calculate the importance of v

    Push v into an ordered heap Q ordered by the importance of vertices

  }

}

while (error<error threshold **and** number of points deleted < certain amount) {

  if (Multi-pass method) {

    for (every vertex v in Δ) {

      Calculate the importance of v

      Push v into an ordered heap Q ordered by the importance of vertices

    }

}

while ($\mathcal{Q}$ is not empty **and** the number of points deleted < certain amount) {

Pop the top vertex v from $\mathcal{Q}$

if (v has not been used **and** v can not be ignored) {

Delete v from the $\triangle$

Marked v as used

Marked the neighbors of v as ignored

Retriangulate the resulting polygonal hole

}

}

Optimize the resulting $\triangle$ to meet certain criterion again

Reset the igored points to its former status

}

end

3.2 Sample Terrains

In the following section, we will introduce two algorithms that we will use to generate test terrains. The first algorithm is the fault algorithm. The second algorithm is diamond-square algorithm.

3.2.1 Fault algorithm [15]

Fault algorithm repeats subdividing a terrain by a random line and raising the part on one side of the line and lowering the part on the other side.

The fault algorithm:

$D = \sqrt{(width^2 + length^2)}$

```
for (step=1 to NoOfStep) {
    θ = Rand ()*2* π,  A = sin ( θ ), B = cos ( θ ), C= Rand ()*D - D/2
    for (i=0 to width) {
        for (j=0 to length) {
            Distance = A*i + B*j-C
            if (want to create a smooth terrain) {
                if (Distance > WaveSize or Distance < -WaveSize)
                    Height (i, j)= Height(i, j)- Displacement
                else
                    Height (i,j)=Height(i,j)+Displacement*cos(Distance/ WaveSize* π )
            }
            else {
                if (Distance >0)
                    Height (i, j) = Height (i, j) + Displacement
                else
                    Height (i, j) = Height (i, j) - Displacement
            }
        }
    }
}
end
```

Rand() creates a random fraction. The constants *width* and *length* are the width and length of the grid, respectively. *Displacement*, *WaveSize* and *NoOfStep* are all constants.

3.2.2 Diamond-Square algorithm [24]

An advantage of the terrains this algorithm generates is that the structure in a smaller scale resembles the structure of the whole, just like natural terrains.

Let us give some concepts first. The middle of a square is a corner of a *diamond* that has *3* or *4* corners. The middle of a diamond is a corner of a square that has *4* corners. The whole grid is a square.

The Diamond-Square algorithm can be divided into the following steps:

1. Create a $(2^n+1)*(2^n+1)$ grid.

2. Assign elevations to some spots of the grid. We can merely assign random elevations to its *4* corners. However, if we want to create terrains with significant features, we must seed the grid on purpose.

3. Diamond step: For every square in the grid, assign the average of the elevations of its corners plus a random fraction to its middle.

4. Square step: For every diamond in the grid, assign the average of the elevations of its corners plus a random fraction to its middle.

5. If the grid is not dense enough, go to step 3.

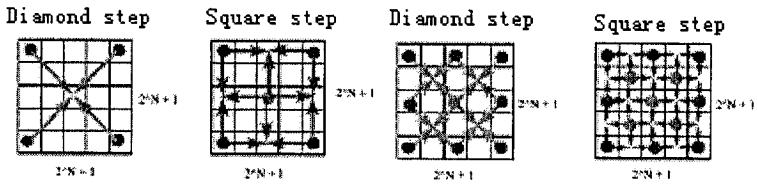

Figure 6: Diamond and Square step (is from [24])

Diamond-Square algorithm:

for $(i = 0$ to $n)$ {

$r = 2^{n-i}$, $s = r / 2$

for $(j = 0; j < 2^n; j = j + r)$

for $(k=0; k < 2^n; k = k + r)$

if $(r>=2)$

$H[j+r/2][k+r/2]=(H[j][k]+H[j][k+r]+H[j+r][k]+H[j+r][k+r])/4+\text{Rand}()*roughness^i$

if $(s >=1)$ {

for $(j = 0; j<= 2^n; j = j + s)$

for $(k=((j+s) \bmod r); k <= 2^n; k = k + r)$

$H[j][k]=\text{Average}(H[j-r/2][k], Height[j][k-r/2], H[j+r/2][k],$

$H[j][k+r/2])+\text{Rand}()*roughness^i$

}

}

end

Rand() creates a random fraction between *–1* and *1*. *Roughness* controls the roughness of the resulting terrain. Average() averages the heights of 3 or 4 valid points. A point is valid if it is inside the grid.

When we generate a terrain with a peak or a valley, we will assign our own values to *H[j][k]* and *H[j+r/2][k+r/2]* in the first several passes before running Diamond-Square algorithm. For example, when we generate a terrain with a peak, we can let *H[j][k]* be the function of *j* and *k* so that the closer the point *(j,k)* is to the center of the peak, the higher its elevation is. Later when we run the algorithm, we will skip the spots that have already have values.

Chapter 4

Simulations

We will compare the following criteria: Min-Max angle criterion, Max-Min angle criterion, Gaussian curvature criterion that minimizes the total absolute Gaussian curvature, Mean curvature criterion that minimizes the total absolute mean curvature, MAT criterion that minimizes the area of the surface, ABN criterion that minimizes the angle between normals of any two adjacent triangles and L_2 measure criterion that minimizes the sum of the square of elevation differences of all input points. We will evaluate their performances for both terrain refinement and decimation.

When dealing with terrain refinement, every time when we insert a point, we search all points of the input data set and determine the point that has the highest error, i.e. the difference between its actual elevation and the interpolated elevation is highest. Then we insert it. To calculate the optimal triangulations with respect to Min-Max angle criterion and Max-Min angle criterion, we use an incremental insertion algorithm (from section 3.1.1). The incremental insertion algorithm can find the global optimum with respect to Max-Min angle criterion. It also works quite well for Min-Max angle criterion. When calculating an optimal triangulation with respect to Min Area criterion, we choose the most important point v and delete the edges of the triangle that contains v. Then we use dynamic programming to optimally re-triangulate the resulting hole with respect to the minimum area criterion. With respect to other data-dependent criteria, we create a Delaunay triangulation first and then apply an edge-flipping algorithm to this initial triangulation to calculate the optimal triangulation. We use local swap algorithm to calculate the optimal triangulations. Although simulated annealing

algorithm has more chance to find the global optimal triangulation than local swap algorithm, in our experience, it is too slow and too sensitive to the parameters. Thus, we will not use simulated annealing algorithm in our final comparisons of triangulation.

Comparison between Local Swap and Simulated Annealing:

We try to use simulated annealing algorithm to find an optimal triangulation for many input data sets, but unfortunately, we found that in our experience, simulated annealing algorithm does not work better than local swap algorithm in most cases, and it is much slower than local swap algorithm, especially when the input data set is large. Simulated annealing algorithm is too sensitive to the parameters. Figure 7 demonstrates the sensitivity of the simulated annealing algorithm. The error measure we use in the plot is the average absolute error, i.e. $\frac{1}{n}\sum_{k=1}^{n}|H(x_k,y_k)-H'(x_k,y_k)|$, where $H(x_k,y_k)$ be the elevation of a vertex (x_k,y_k) on the actual surface, and $H'(x_k,y_k)$ be its interpolated approximation. When we generate this error plot, we set the parameters as L.L. Schumaker[25] recommended: the initial temperature=*2**(the highest swap value of edges in $\triangle$), *NoOfSwaps = 5 ** (no of edges), *NoOfGoodSwaps =5 ** (no of edges), *r = 0.95*. We set *NoOfStages=100*. From the plot, we can see that the approximation errors of the optimal triangulation generated by simulated annealing are higher than those by local swap algorithm and the error curve resulting from the simulated annealing algorithm fluctuates dramatically. The non-monotonic slopes of the error curves indicate that the parameters are not set properly in general, because the error becomes even higher than before when more points are inserted.

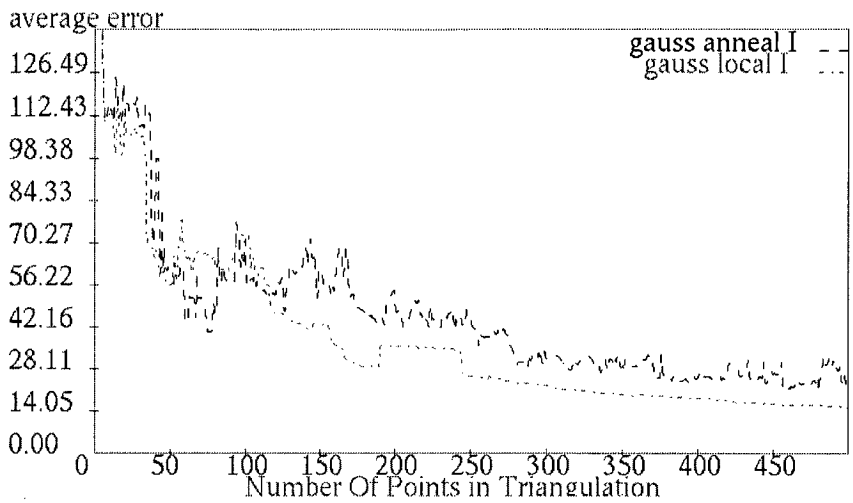

Figure 7: Simulated anneal vs. Local swap algorithm

The only case we found that simulated annealing is better than local swap algorithm is when we test on the function: $f(x,y)=exp(-81.0/4.0*((x-0.5)*(x-0.5)+(y-0.5)*(y-0.5)))$. We test this function on a *20*30* grid. Simulated annealing algorithm still does not work better than local swap algorithm when we set the parameters as L.L. Schumaker [25] recommended. Then, we change the parameters to the following values: the initial temperature = *0.01*, *NoOfSwaps=15**(no of edges), *NoOfGoodSwaps =15**(no of edges), $r=0.95$, *NoOfStages =500*. With these values, we can see that the simulated annealing sometimes works better than local swap algorithm for this function. Figure 8 indicates that. The error measure we use in the plot is the average absolute error. Because this data set is very small, it is not difficult for us to adjust the parameters to reach reasonable values. However, when the data set is large, it is very expensive for us to adjust the parameters to reach reasonable values. Thus, we will not use the simulated annealing to calculate the optimal triangulation in our experience.

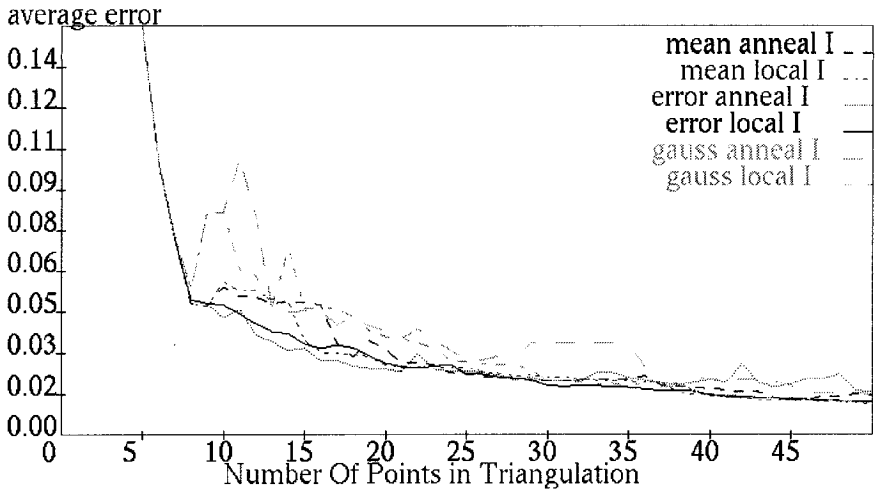

Figure 8: Simulated annealing vs. Local swap algorithm

When dealing with terrain decimation, we use Delaunay Greedy Insertion algorithm [18] to create a full-detailed approximation of the actual surface first. The full-detailed approximation we generate is the one whose error is zero, although it may not contain all input points. Then we repeatedly delete the least important points from the initial triangulation and re-triangulating the polygonal resulting holes by ear cutting solution [12] (from section 2.1.3.1) until the number of points deleted reaches a certain amount or the error is beyond a certain threshold. We choose an independent set of points to delete each time. We will use HDH (The Height Difference heuristic) criterion or the Drop heuristic to determine the importance of a point. As introduced earlier in section 2.1.3.1, the HDH sums the signed differences between the elevation of a vertex v and that of each of its neighboring vertex and uses the absolute value of the result as the importance of v. The Drop heuristic uses the elevation difference between a point and its approximation as the importance of the point. HDH is much faster than the Drop heuristic, but the Drop heuristic yields better results [20]. When generating the plots of all terrains and the tables of the crater terrain data set, we use HDH to calculate the importance of points, whereas, when generating the tables of other terrains, we

use the Drop heuristic to compute the importance of points. After that, we use the edge-flipping algorithm to calculate the optimal triangulation with respect to certain criterion.

After a point is deleted, we will re-triangulate the resulting polygonal hole. We will only allow convex triangulation, because if not so, it will conflict with some assumptions we make in the programs and cause errors. For example, when we try to locate a point v in the triangulation, the program will return an edge that shares a same triangle T with v. In our program, we assume that v is an endpoint of T. Non-convex triangulation will conflict with this assumption we make. Moreover, non-convex triangulation will cause artificial break line in the triangulation.

Because some criteria, such as Gaussian curvature criterion and L_2 measure criterion, often generate many very thin triangles, sometimes, it is extremely difficult to find a convex triangulation of the resulting hole after a point is deleted. Thus, we will not allow deleting a point if we cannot find a convex re-triangulation of the resulting hole after it is deleted, even if it has a very high importance.

We use DEM files to store the original surfaces and the Quad-Edge data structure to store the optimal triangulations we generate.

4.1 Error measures

We use the popular evaluation methods that compute the differences in elevations between the actual surface and its approximation to evaluate the quality of an approximation with respect to certain criterion. The commonly used error measures include L_∞ error, L_2 error, RMS error, average absolute error and total absolute error. Let $H(x_k, y_k)$ be the elevation of a vertex (x_k, y_k) on the actual surface, and $H'(x_k, y_k)$ be its interpolated approximation. Then these error measures can be defined as follows:

$$L_\infty \ error = max_{k=1}^{n} |H(x_k, y_k) - H'(x_k, y_k)|.$$

$$L_2 \ error = \sqrt{\sum_{k=1}^{n} (H(x_k, y_k) - H'(x_k - y_k))^2}.$$

$$RMS \ error = \frac{1}{\sqrt{n}} \sqrt{\sum_{k=1}^{n} (H(x_k, y_k) - H'(x_k - y_k))^2}.$$

$$Average \ absolute \ error = \frac{1}{n} \sum_{k=1}^{n} |H(x_k, y_k) - H'(x_k, y_k)|.$$

$$Total \ absolute \ error = \sum_{k=1}^{n} |H(x_k, y_k) - H'(x_k, y_k)|.$$

Although the order of the error curves with respect to different criteria is quite consistent over all these error measures, they have different focuses. L_∞ error determines the global error bound over the approximation. It makes sure that the difference between the approximation and the real surface will not go beyond the bound. However, L_∞ error is too sensitive to those points that have uncommon high individual errors. L_2 error is more robust to outliers than the L_∞ error. It gives fairly good evaluation for the overall fitting quality of the approximation. Average absolute error is more robust than L_2 error against outliers because it treats the elevation differences of all points equally, whereas L_2 error inclines to large errors that occur infrequently. Thus, in order to have a comprehensive knowledge of the qualities of different criteria, we will choose the following error measures: L_2 error, average absolute error and total absolute error. Because L_∞ error is vulnerable to the outlier, we will not use it. Figure 15 demonstrates the vulnerability of L_∞ error.

4.2 Input

Our experience is based on six classes: crater terrain, random terrains, terrains with a single significant feature, functions, natural terrains and hybrid terrains that combine two or more different significant features. The Crater Lake contains *154,224* points on a regular grid and is widely used as an input data set for many experiments on terrain simplification [18]. Random terrains are those generated by the fault algorithm (introduced in section 3.2.1) randomly. We will create both rough and smooth random terrains. The random terrains we generate have many small valleys and peaks. We will use Diamond-Square algorithm and the tool Gforge to create the terrains with significant features, such as peaks and valleys. A peak is a spot that has greatest elevation locally, whereas a valley is a spot that has lowest elevation locally. The tool Gforge is a height field generator and was developed by J. Beale [6]. We will test on terrains with a single feature and with two or more features. We will call the later ones hybrid terrains. With these programs, we can easily create the artificial terrains with certain features we want and we can adjust the parameters of the programs to create slightly different terrains and compare the qualities of the approximations with respect to different criteria for these slightly different terrains. For example, we can adjust the parameter *Displacement* in the fault algorithm to create more curved or more flat surfaces. Then we can test on these slightly different terrains to find out whether a criterion is suitable for all these terrains with little difference. Although programs provide the flexibility to create terrains, the terrains they generate may be different from those in nature. Thus, we also test on the real natural terrains. Moreover, we also make a comparison of criteria on some functions that the corresponding surfaces generated are not common in the real world. These surfaces we test include spheres and cylinders, because they are quite common geometrical objects.

Figure 9 shows examples of some terrains we are going to test the triangulation algorithms on.

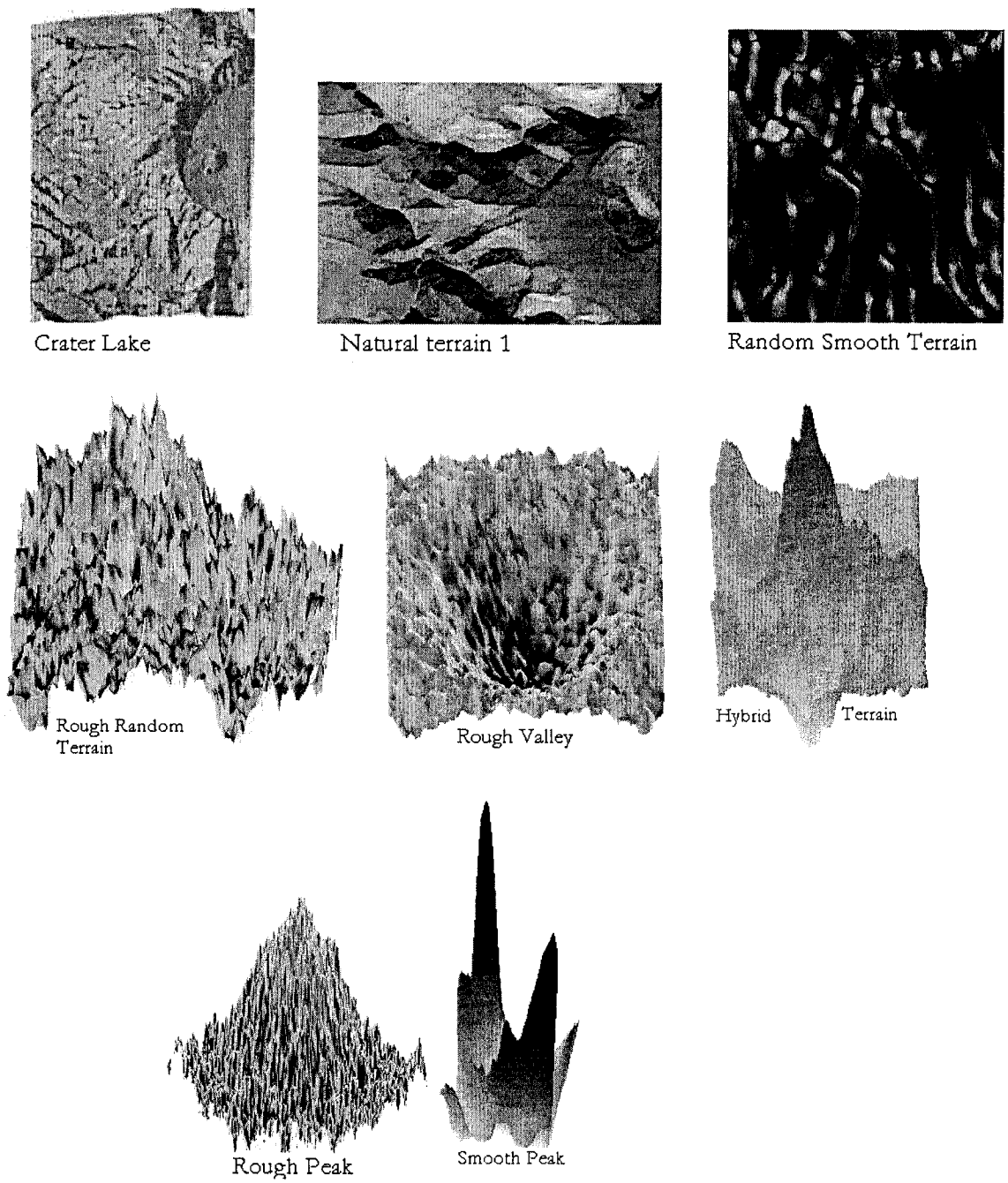

Figure 9: Surfaces are used for testing algorithm

4.3 Results

When comparing different criteria, we use tables to describe the results after certain percentage of points are inserted or deleted. Moreover, we also give plots to describe the error curves for the refinement process and the decimation process. In the refinement process, we insert a point one at a time, whereas in the decimation process, we delete an independent set of points each time. Thus, the error curves of refinement are continuous, whereas the error curves of decimation are piecewise continuous.

In the plots, the 1st, 2nd and 3rd columns in the ledge in the upper, right-hand corner of the plots show the criterion of the triangulation, the optimizing algorithm and refinement/ decimation method respectively. "Error", "mean" and "gauss" denote L_2 measure criterion, mean curvature criterion and Gaussian curvature criterion respectively. "Incremental" means that the optimal triangulation is generated by incremental insertion algorithm. "Local" means that the optimal triangulation is produced by local swap algorithm, whereas, "anneal" means that the optimal triangulation is calculated by simulated annealing algorithm. The third column with "I" means the simplification process is a refinement one that inserts points one by one, whereas, "D" denotes a decimation process that deletes an independent set of points at each step.

We will list the approximation errors for six kinds of terrains with respect to different criteria in sequence. For every kind of terrains, we will list the approximation errors for refinement process first and the approximation errors for decimation process later. Although for every kind of terrains, we only list the errors of one sample of it, we have tested on many other terrains of the same kind. Some terrains of a same kind share some common features.

4.3.1 Crater:

We will list the approximation errors with respect to all three error measures we use: L_2 error, average absolute error and total absolute error in order to get a full knowledge of the qualities of the approximations with respect to different criteria.

Refinement:

Figure 10 and Figure 11 show the approximations with respect to different criteria that use *500* points of the Crater terrain data set. As we can see, the approximation has already demonstrated its major features.

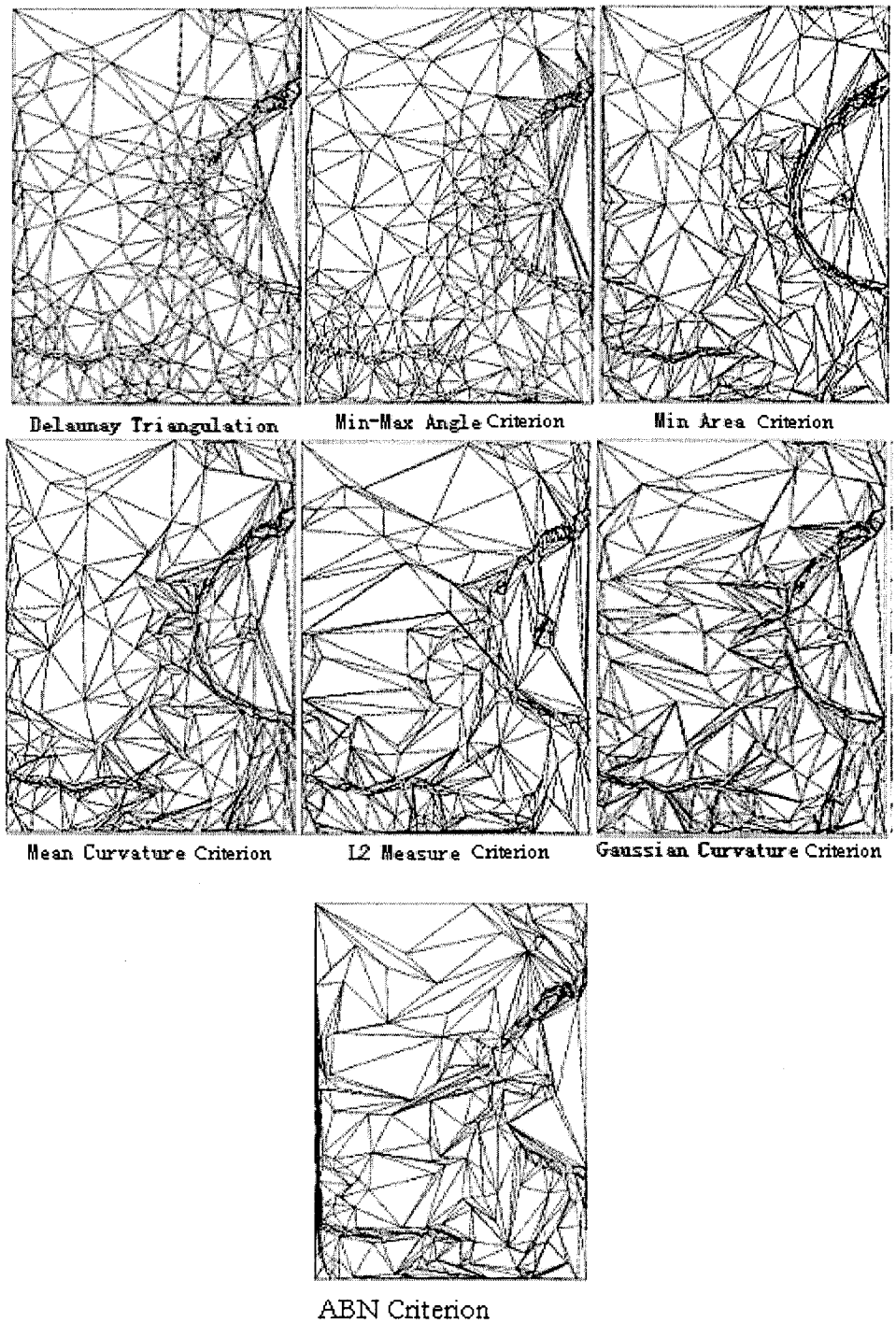

Figure 10: Insertion of 500 points of Crater

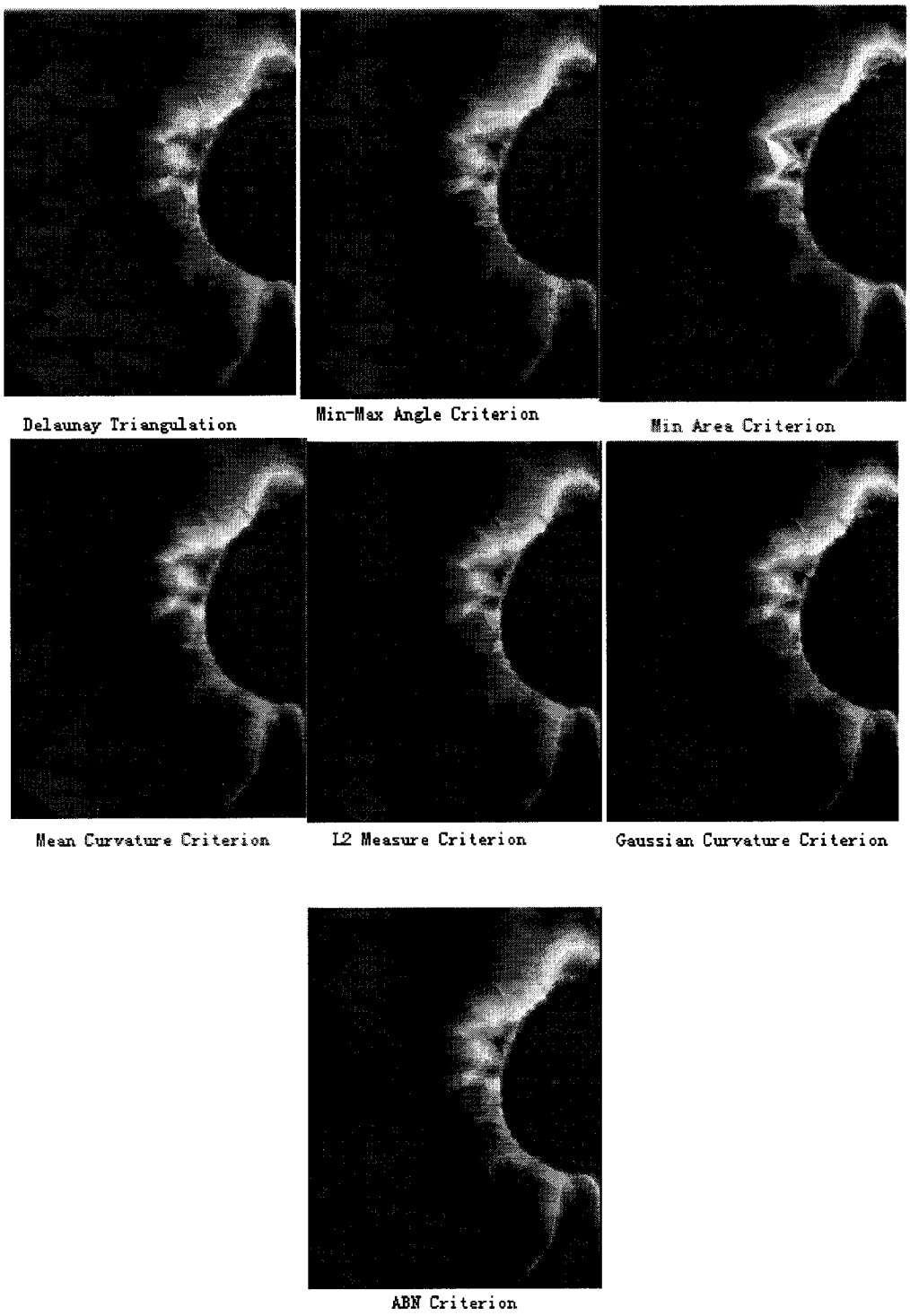

Figure 11: Insertion of 500 points of Crater

Figure 12, Figure 13, Figure 14 and Figure 15 show the error curves of the Crater with respect to average absolute error, total absolute error, L_2 error and L_∞ respectively. From Figure 15, we can see that the error plot of L_∞ error is too chaotic. Thus, we will not use it afterward. As we can see from the plots, the rank of the criteria with respect to their approximation errors is quite similar over all these four kinds of error measures. Thus, we will display only one kind of error measure afterwards in order to save space. The rest of the error plots can be found in Appendix A.

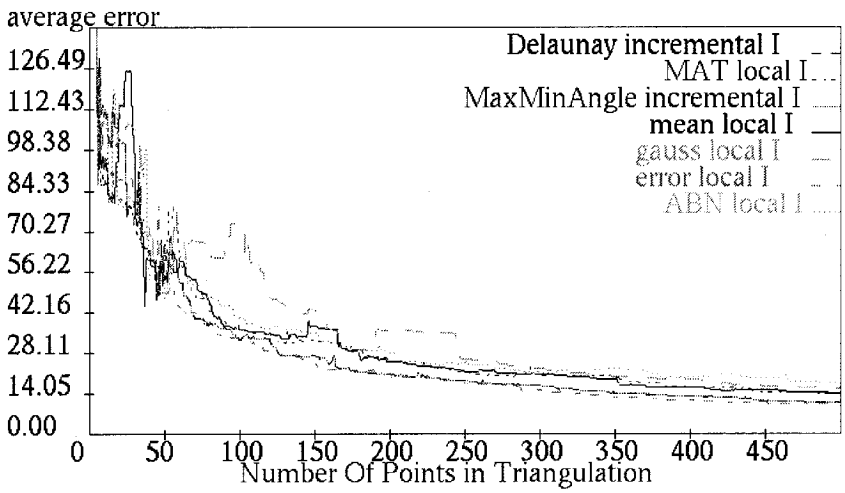

Figure 12: Average errors after inserting up to 500 points of Crater terrain data set sequentially

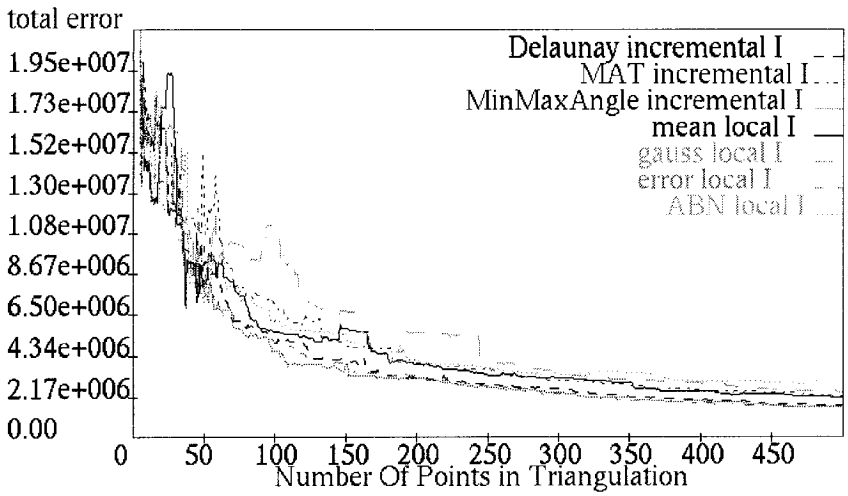

Figure 13: Total errors after inserting up to 500 points of Crater terrain data set sequentially

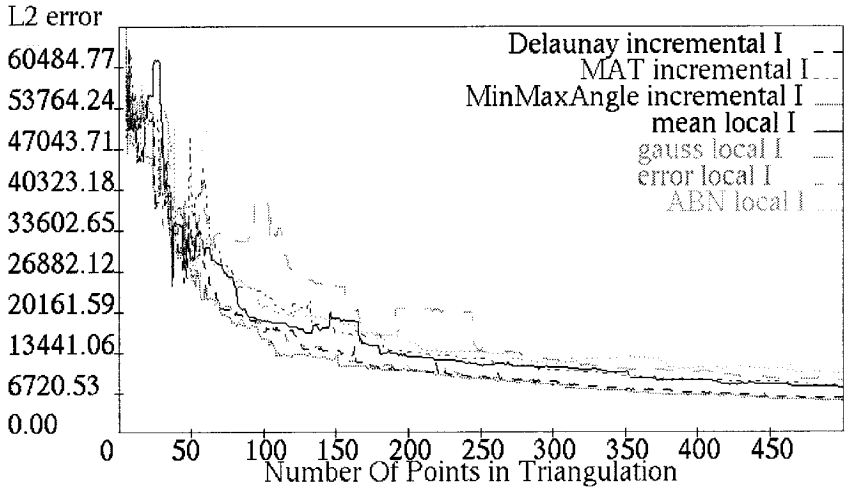

Figure 14: L2 errors after inserting up to 500 points of Crater terrain data set sequentially

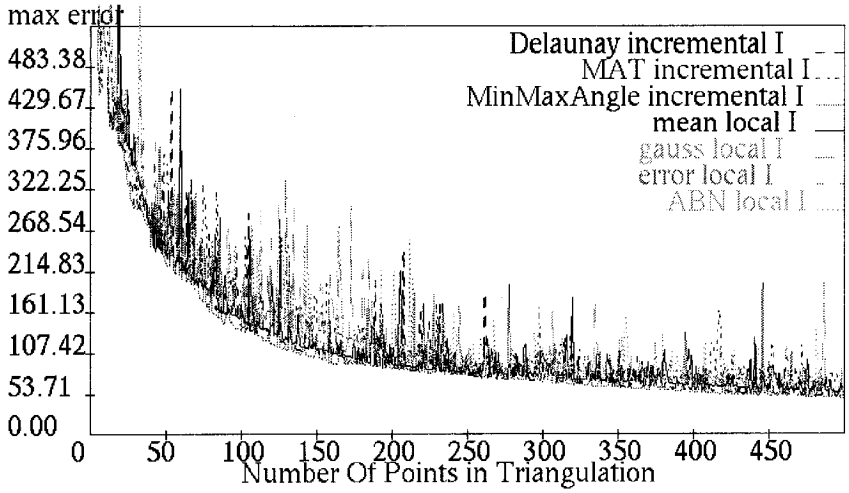

Figure 15: Max errors after inserting up to 500 points of Crater terrain data set sequentially

Since after *75%* points are inserted, the error is zero, we will not give the errors beyond this point for comparison. Also, we will only give the errors of the optimal triangulation after at least *1%* points are inserted, since the approximation with *1%* of the input points already looks quite similar to the original crater.

%	Points inserted	Delaunay	Min-Max angle	Min Area	Mean	Gaussian	ABN	L_2 measure
1	1542	5.26211	5.24758	7.52188	6.49782	6.86634	6.80974	4.24232
10	15422	0.809917	0.804217	1.46247	1.06522	1.08695	1.10184	0.671814
25	38556	0.315733	0.319663	0.580354	0.415044	0.418539	0.422042	0.262467
50	77112	0.0864432	0.0892028	0.198774	0.128764	0.126879	0.115222	0.0570321

Table 1: Average errors after inserting various percentages of points of the Crater terrain

Decimation:

As we can see from Figure 16, Figure 17 and Figure 18, the rank of the criteria with respect to their approximation errors in the decimation algorithm is also quite similar over all these three kinds of error measures. Thus, we will display only one kind of error measure

afterwards for the decimation algorithm too. The rest kinds of the error plots can be found in Appendix A.

Although crater has *154,224* input points, the error of the Delaunay triangulation of it has already reached zero after *100,807* points are inserted. Thus, we will generate the Delaunay triangulation of it with *100,000* points inserted before doing the decimation. Because the errors of the initial full-detailed triangulation with respect to different criteria are the same, we will not give errors of the initial triangulation. Instead, we will start the error curves from the triangulation after an independent set of points is deleted from the initial triangulation. At that time, the number of points in the mesh is about *73,000.*

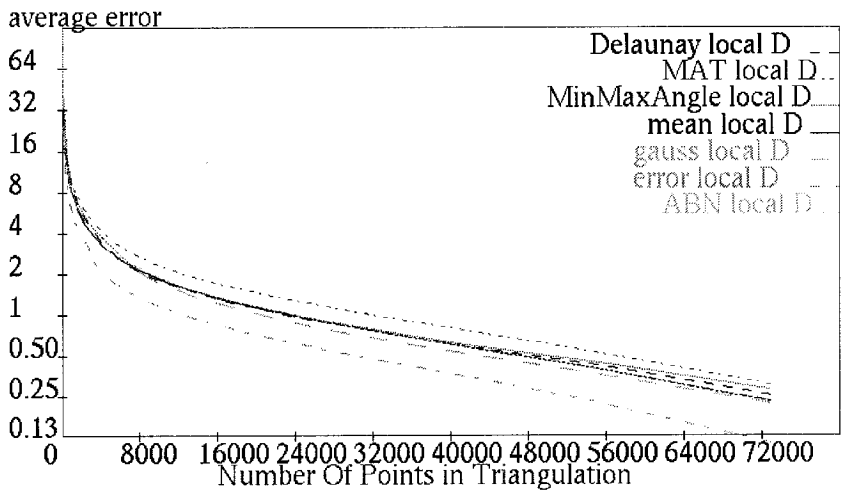

Figure 16: Average errors when deleting an independent set of points of the Crater terrain data set each time

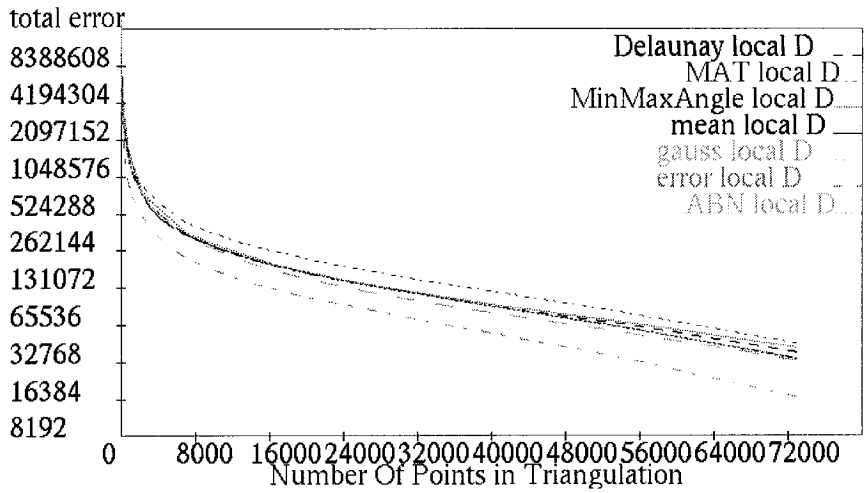

Figure 17: Total errors when deleting an independent set of points of the Crater terrain each time

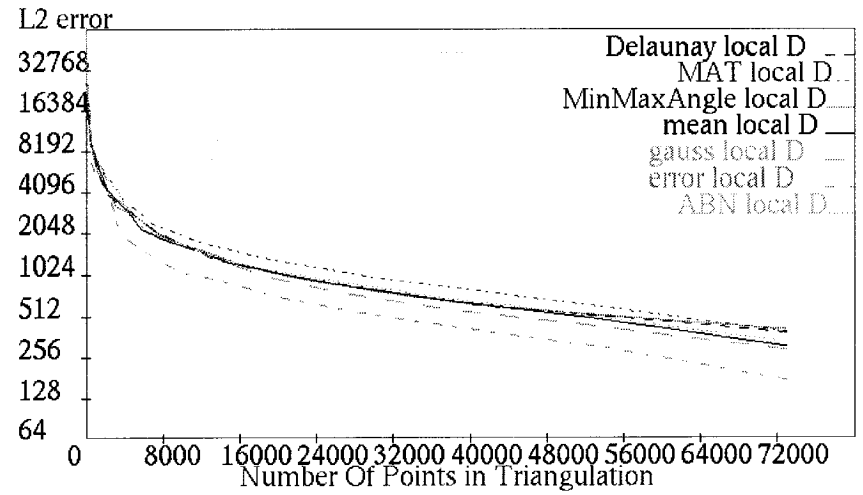

Figure 18: L2 errors when deleting an independent set of points of the Crater terrain each time

%	Points deleted	Delaunay	Min-Max angle	Min Area	Mean	Gaussian	ABN	L_2 measure
10	10000	0.0749882	0.101097	0.135328	0.0933276	0.0924291	0.079581	0.0346918
25	25000	0.192211	0.21639	0.253581	0.188033	0.184535	0.184385	0.0881985
50	50000	0.461684	0.484298	0.586801	0.452728	0.409	0.453085	0.257116
75	75000	0.886049	0.908521	1.17482	0.922817	0.828301	0.943499	0.57783

Table 2: Average errors when deleting a certain percent of points of the Crater terrain data set

4.3.2 Random terrain:

In our experience, we have adjusted the parameters, such as *Displacement* and *WaveSize*, of the fault algorithm to produce different random terrains. Some are more curved and some are more flat. Some are more smooth and some are more rough. From our experience, the error plots of the smooth random terrains often share some common features. So do that of rough random terrains. Moreover, because many criteria, such as ABN and mean curvature criteria, which we examine are smooth criteria, we will test whether they work well for rough terrains also, if they can work well for smooth terrains. We will distinguish between rough and smooth terrains.

4.3.2.1 Smooth Random Terrain:

The sample smooth random terrain we create is on a grid that has *80*80* points. When we generate this terrain by fault algorithm, the parameters *Displacement* and *WaveSize* are set to *20* and *5* respectively.

Refinement:

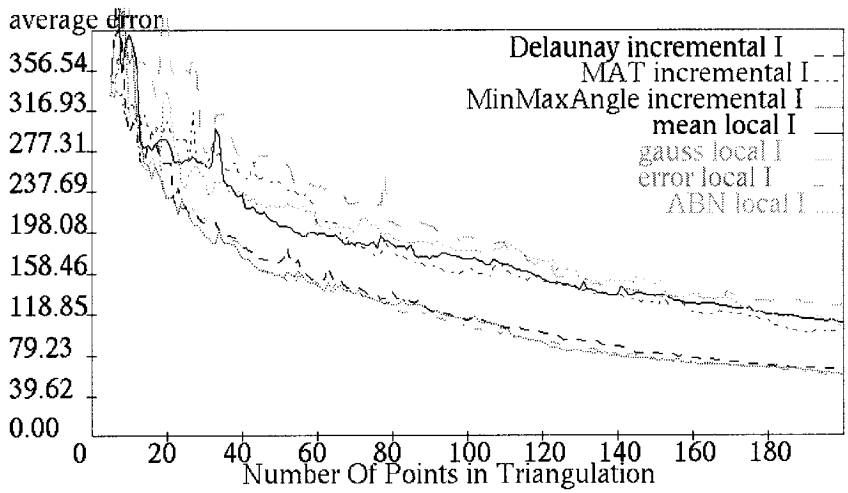

Figure 19: Average errors when inserting up to 200 points of a smooth random terrain sequentially

%	Points inserted	Delaunay	Min-Max angle	Min Area	Mean Curvature	Gaussian Curvature	ABN	L_2 measure
10	640	22.6548	22.7061	38.1835	33.5984	37.4308	33.2951	19.7869
25	1600	9.24025	9.08353	19.3872	12.0889	13.0941	12.9404	7.80284
50	3200	3.0535	2.94488	9.57973	4.53152	3.94519	4.59853	2.46956
75	4800	0.739063	0.706604	3.6522	1.43362	1.09366	1.31572	0.6406

Table 3: L2 errors when inserting a certain percent of points of smooth random terrain

Decimation:

The full-detailed approximation of it has *6,332* points. After an independent set of points is deleted from the initial triangulation, the number of points in the mesh is about *4,500*.

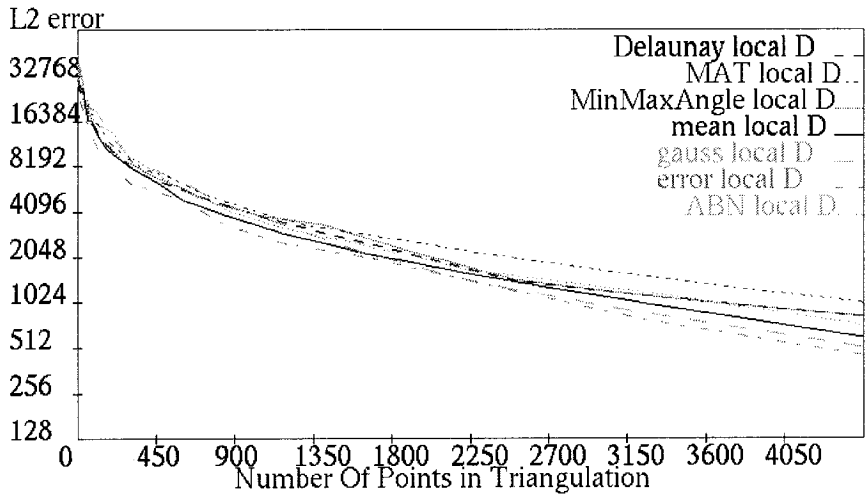

Figure 20: L2 errors when deleting an independent set of points of a smooth random terrain each time

%	Points deleted	Delaunay	Min-Max angle	Min Area	Mean	Gaussian	ABN	L_2 measure
10	633	1.02836	1.03242	0.862422	0.383227	0.486411	0.815086	0.18851
25	1583	3.27299	3.27581	3.2251	1.94238	1.91569	2.69302	1.35753
50	3166	6.34618	6.44216	7.81874	5.21429	5.18284	6.62613	3.79036
75	4749	14.6489	14.7836	18.3006	13.9677	12.5528	16.6008	10.0862

Table 4: Average errors when deleting a certain percent of points of a smooth random terrain

4.3.2.2 Rough Random Terrain:

The sample rough random terrain we create is on a grid that has *100*100* points. When we generate this terrain by fault algorithm, the parameters *Displacement* is set to *100.*

Refinement:

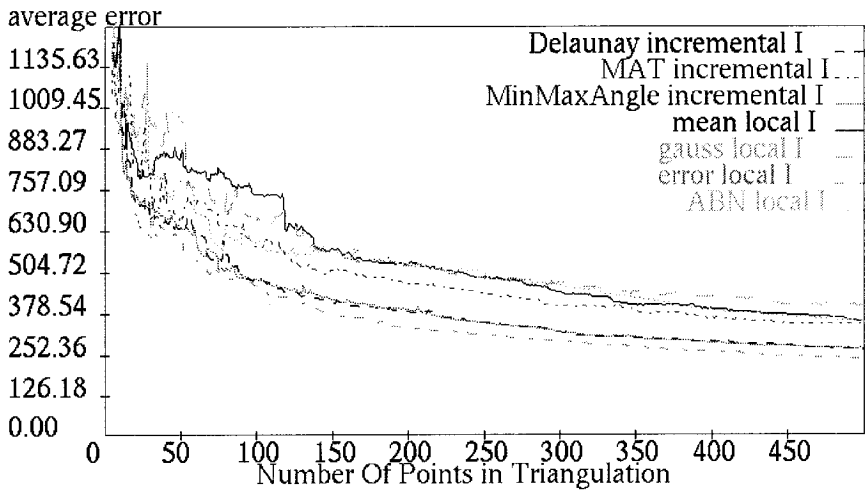

Figure 21: Average errors when inserting up to 200 points of a rough random terrain sequentially

Decimation:

The full-detailed approximation of it has *8,688* points. After an independent set of points is deleted from the initial triangulation, the number of points in the mesh is about *6,300*.

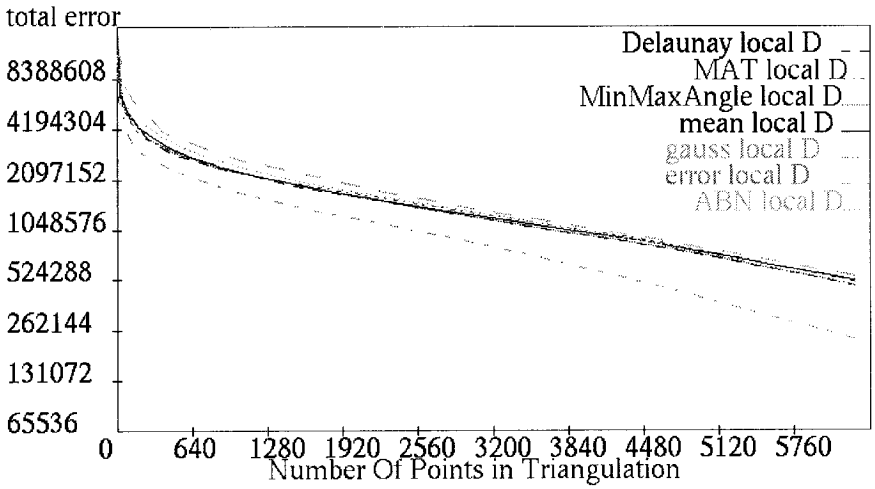

Figure 22: Total errors when deleting an independent set of points of a rough random terrain each time

4.3.3 Peak:

4.3.3.1 Smooth Peak:

We use the tool Gforge to create a smooth hill to compare different criteria. This input data set has *128*128* points in total. It has two very sharp peaks.

Refinement:

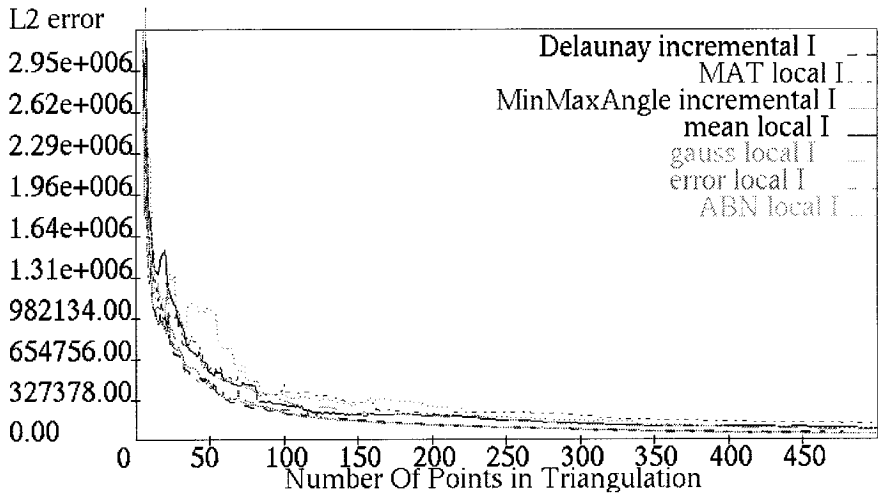

Figure 23: L2 errors when inserting up to 500 points of a smooth peak sequentially

Decimation:

The full-detailed approximation of it has *16,344* points. After an independent set of points is deleted from the initial triangulation, the number of points in the mesh is about *11,500*.

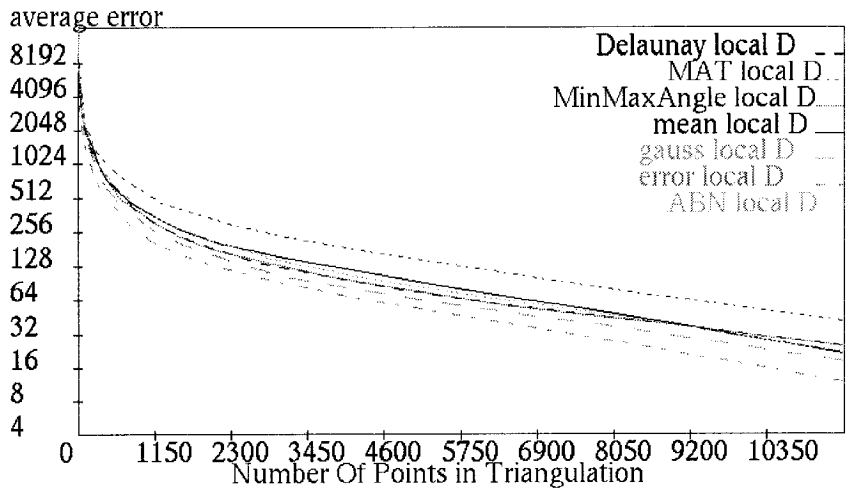

Figure 24: Average errors when deleting an independent set of points of a smooth peak each time

4.3.3.2 Rough Peak

The sample terrain with a peak we generate has *129*129*, i.e. *16,641* points totally. We use the Diamond-Square algorithm to generate it. It looks like a small mountain. Each small part of it has similar structure as the whole mountain. The parameter *roughness* of the algorithm is *0.7*. It controls the roughness of the generated terrain.

Refinement:

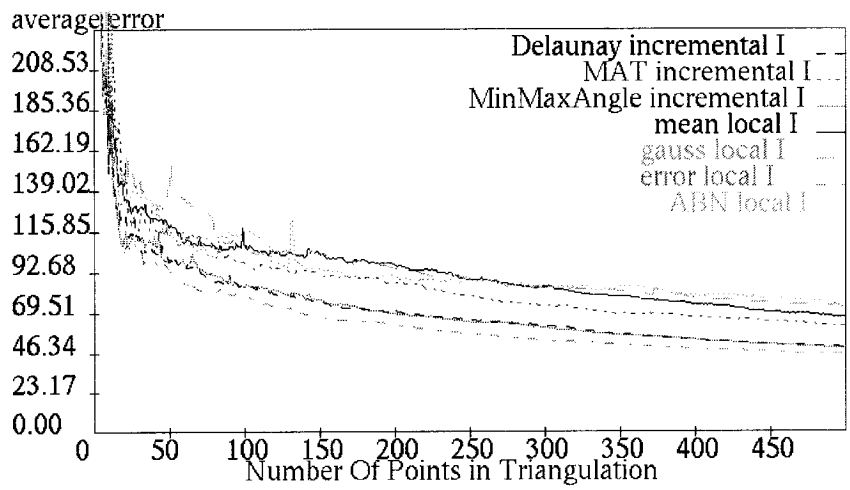

Figure 25: Average errors when inserting up to 500 points of a rough peak sequentially

%	Points inserted	Delaunay	Min-Max angle	Min Area	Mean	Gaussian	ABN	L_2 measure
10	1664	32.7462	32.5898	41.038	41.112	45.2291	39.5796	29.5456
25	4160	20.9325	20.8371	26.0177	26.0477	28.1473	25.5939	17.3786
50	8321	9.18912	9.24922	12.5166	12.3359	13.4154	11.8933	6.92758
75	12481	2.3925	2.37517	4.51597	4.09812	4.54727	3.75773	1.89635

Table 5: Average errors when inserting a certain percent of points of the rough peak

Decimation:

The full-detailed triangulation has *16,593* points. After an independent set of points is deleted from the initial triangulation, the number of points in the mesh is about *11,500.*

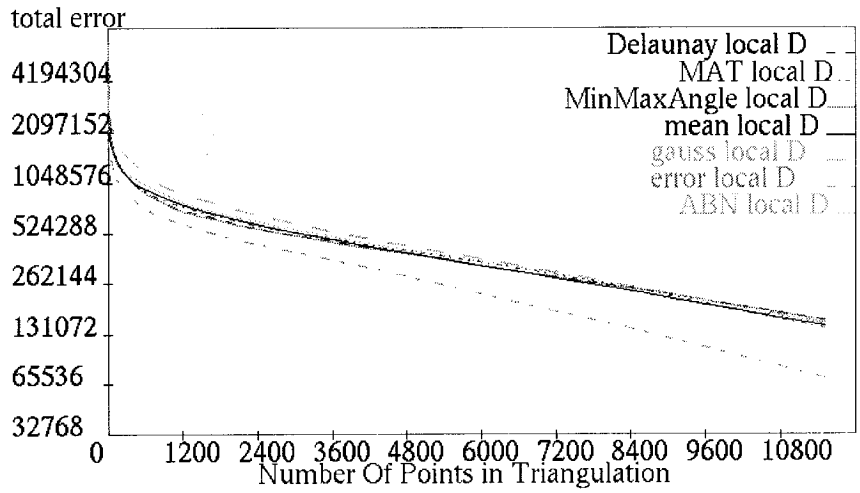

Figure 26: Total errors when deleting an independent set of points of rough peak each time

%	Points deleted	Delaunay	Min-Max angle	Min Area	Mean Curvature	Gaussian Curvature	ABN	L_2 measure
10	1659	2.09155	2.08789	0.896491	1.48896	2.09939	2.12119	0.418043
25	4148	6.54348	6.53855	4.3825	5.52314	6.53479	6.64843	2.66287
50	8297	13.8705	13.8632	13.0209	13.4327	14.4392	14.7925	8.11607

75	12445	25.1438	25.0054	25.1563	25.3353	27.6831	27.6408	18.3685

Table 6: Average errors when deleting a certain percent of points of rough peak

4.3.4 Valley:

The sample terrain with a valley we generate has *65*65*, i.e. *4,225* points totally. We use Diamond-Square algorithm to generate it. Each small part of it has similar structure as the whole mountain. The parameter *roughness* of the algorithm is *0.7*. It is a rough terrain.

Refinement:

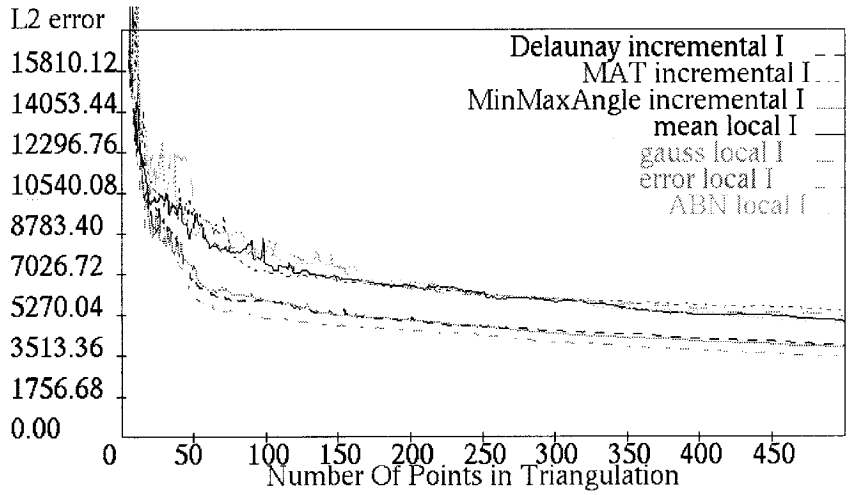

Figure 27: L2 errors when inserting up to 500 points of rough valley sequentially

%	Points inserted	Delaunay	Min-Max angle	Min Area	Mean Curvature	Gaussian Curvature	ABN	L_2 measure
10	423	49.1922	48.1616	57.9258	59.7968	64.327	59.0904	43.6821
25	1056	31.0728	31.4214	36.5262	37.3346	41.039	36.1181	25.5069
50	2113	13.5587	13.5165	18.0822	17.8452	19.5526	17.4554	10.3904
75	3169	3.51991	3.45761	6.43411	5.84483	6.64793	5.48321	2.70297

Table 7: Average errors when inserting a certain percent of points of rough valley

Decimation:

The full-detailed triangulation has *4,217* points. After an independent set of points is deleted from the initial triangulation, the number of points in the mesh is about *3,000.*

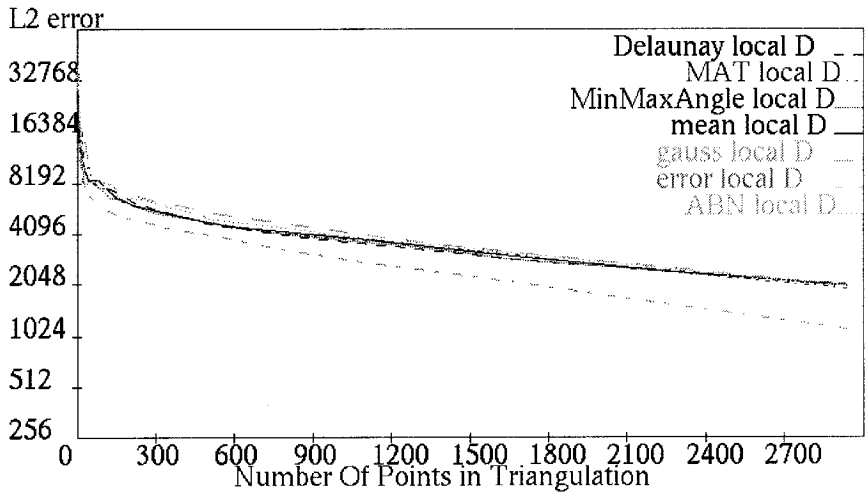

Figure 28: L2 errors when deleting an independent set of points of rough valley each time

%	Points deleted	Delaunay	Min-Max angle	Min Area	Mean Curvature	Gaussian Curvature	ABN	L_2 measure
10	421	3.09183	3.09183	1.43182	2.04361	3.20698	3.0183	0.595858
25	1054	9.2768	9.2768	6.36071	7.72152	9.6609	9.24178	3.84987
50	2109	20.6864	20.602	19.6713	19.0794	21.0774	21.3711	12.2889
75	3163	36.9557	37.3866	37.5351	37.1062	39.3103	39.4173	27.1727

Table 8: Average errors when deleting a certain percent of points of rough valley

4.3.5 Functions:

4.3.5.1 Sphere:

We will create a sphere with: $f(x, y) = 20*\sqrt{1-(x/20-1)^2-(y/20-1)^2}$ *if* $(x-20)^2 + (y-20)^2 <= 400$, else *0*. Its radius is *20*. Its center is *(20, 20)*. We will create a sphere with x in *[0,40]* and y in *[0,40]*.

Why we select a sphere to test is that a sphere has both constant Gaussian curvature $1/r^2$ and mean curvature $1/r$, where r is its radius and Gaussian curvature criterion and mean curvature criterion try to minimize the sum of mean curvature and Gaussian curvature respectively. Moreover, because a sphere is convex, the local swap algorithm can find the global optimum with respect to Gaussian curvature criterion.

Refinement:

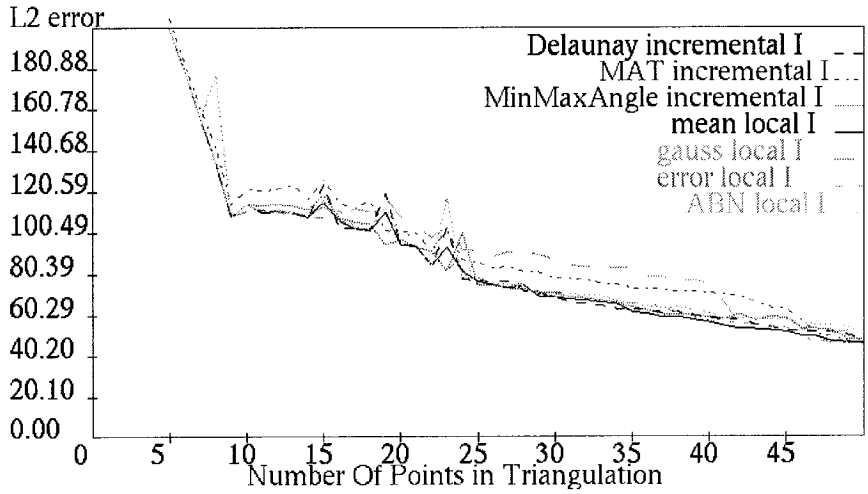

Figure 29: L2 errors when inserting up to 50 points of sphere sequentially

%	Points Inserted	Delaunay	Min-Max angle	Min Area	Mean Curvature	Gaussian Curvature	ABN	L_2 measure
1	8	4489.16	4489.16	5026.46	4483.72	4483.72	5677.04	4483.72
5	40	1822.52	1851.53	1708.53	1721.62	1768.22	2030.29	1708.08
10	80	643.681	728.638	591.827	588.431	656.341	742.811	582.554

25	400	133.58	145.848	130.507	126.171	155.432	117.969	94.3926

Table 9: Total errors when inserting a certain percent of points of sphere

Decimation:

The full-detailed triangulation has *724* points. After an independent set of points is deleted from the initial triangulation, the number of points in the mesh is about *540*.

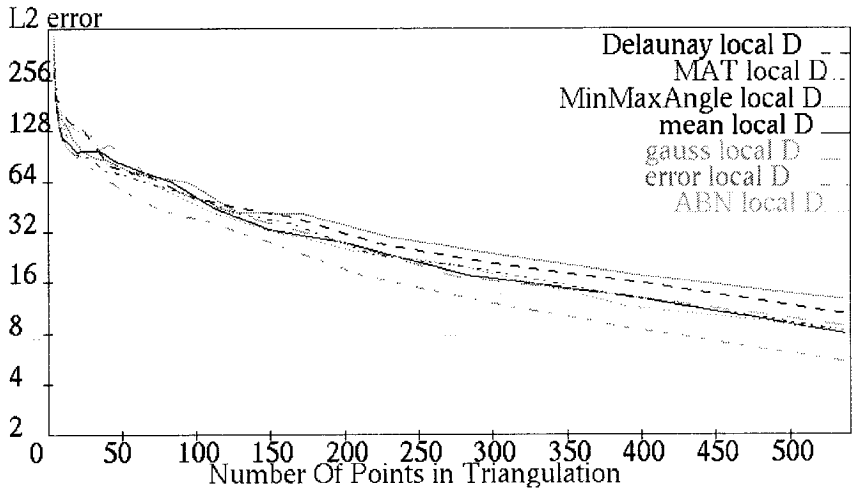

Figure 30: L2 errors when deleting an independent set of points of sphere each time

%	Points Deleted	Delaunay	Min-Max angle	Min Area	Mean Curvature	Gaussian Curvature	ABN	L_2 measure
10	72	7.57402	14.2959	3.21023	1.64992	5.55681	2.10819	1.10554
25	181	13.2452	17.9686	6.65552	6.18797	7.94799	5.66405	3.70103
50	362	20.2144	22.5932	12.446	9.95207	11.4595	9.15769	6.37404
75	543	26.8888	37.126	25.331	34.4812	23.4464	24.5936	18.361

Table 10: L2 errors when deleting a certain percent of points of sphere

4.3.5.2 Cylinder:

We will create a cylinder with: *f(x,y)=60 if ((x-20)*(x-20)+(y-20)*(y-20)<=400)* else *0*. Its radius and height are *20*, *60* respectively. Its center is *(20,20)*. We will create a cylinder with x in *[0,40]* and y in *[0,40]*. A cylinder is also a cliff. It has zero Gaussian curvature. So Gaussian curvature should work well for it. We want to check whether this is true or not. Moreover, we also want to check whether data-dependent triangulations perform better than two-dimensional triangulation when approximating a cliff.

Refinement:

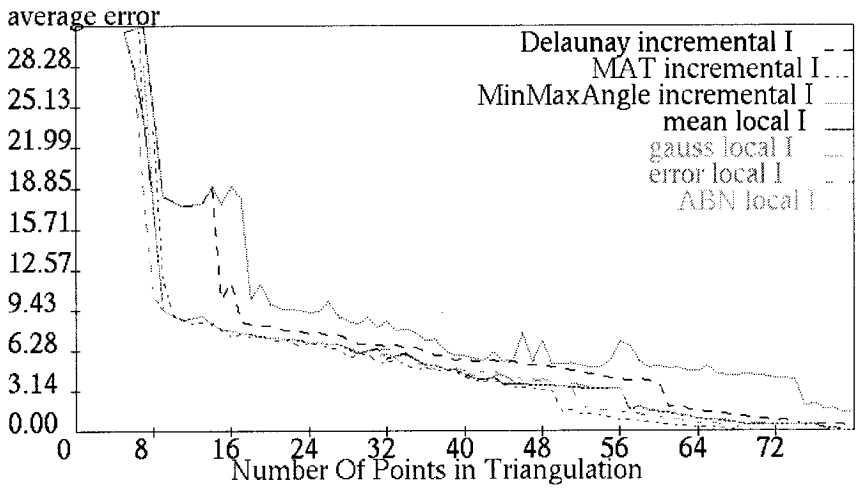

Figure 31: Average errors when inserting up to 80 points of cylinder sequentially

%	Points Inserted	Delaunay	Min-Max angle	Min Area	Mean Curvature	Gaussian Curvature	ABN	L_2 measure
0.5	8	39755.9	39755.9	27716.6	27716.6	27716.6	27716.6	27716.6
1	16	27783.1	30511.8	12459.1	12459.1	12459.1	12459.1	12459.1
2.5	40	5838.21	9373.2	6583.04	6490.39	7094.8	6578.04	6650.39

Table 11: Total errors when inserting a certain percent of points of cylinder

Decimation:

The full-detailed triangulation has *77* points. After an independent set of points is deleted from the initial triangulation, the number of points in the mesh is about *60*.

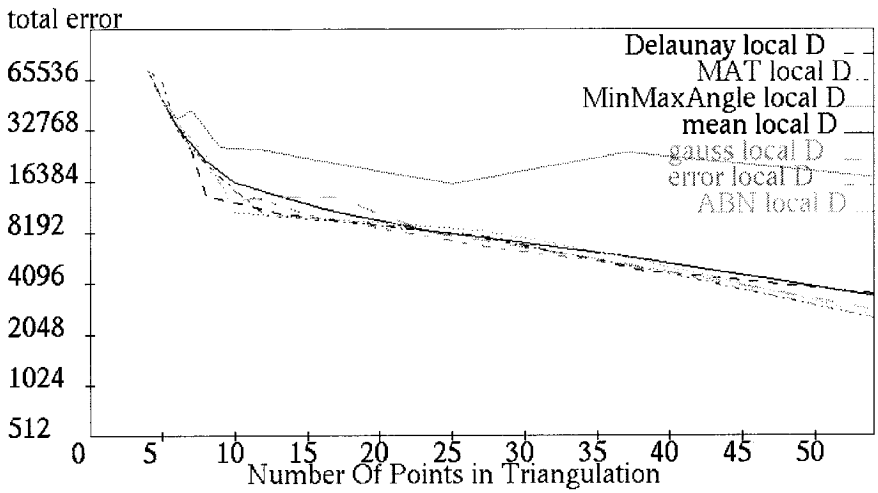

Figure 32: Total errors when deleting an independent set of points of cylinder each time

%	Points Deleted	Delaunay	Min-Max angle	Min Area	Mean Curvature	Gaussian Curvature	ABN	L_2 measure
10	8	709.091	16749.9	381.818	637.5	709.091	381.818	381.818
25	19	3133.09	24592.2	2031.82	2417.5	2389.48	2190.39	2031.82
50	39	6531.59	30401	3959.24	5714.37	4950.71	5355.33	5312.86
75	58	10375.7	33851.5	10961.2	21553.3	10134.6	11077.6	9580.57

Table 12: Total errors when deleting a certain percent of points of cylinder

4.3.6 Natural Terrain:

The sample natural terrain has *281*195* points in total. It is a small part of a real terrain. The elevation of the highest point in this data set is *255*. The elevations of the points in this terrain do not differ much from others.

Refinement:

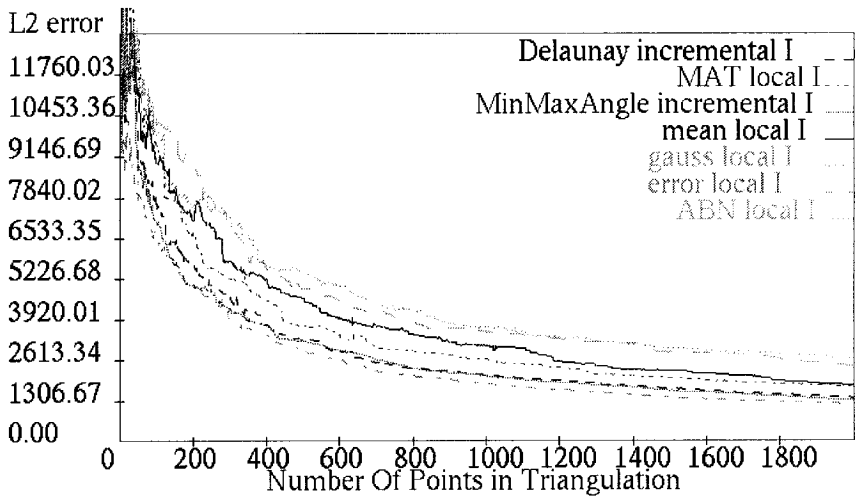

Figure 33: L2 errors when inserting up to 2000 points of a natural terrain sequentially

Decimation:

The full-detailed triangulation has *45,311* points. After an independent set of points is deleted from the initial triangulation, the number of points in the mesh is about *30,000.*

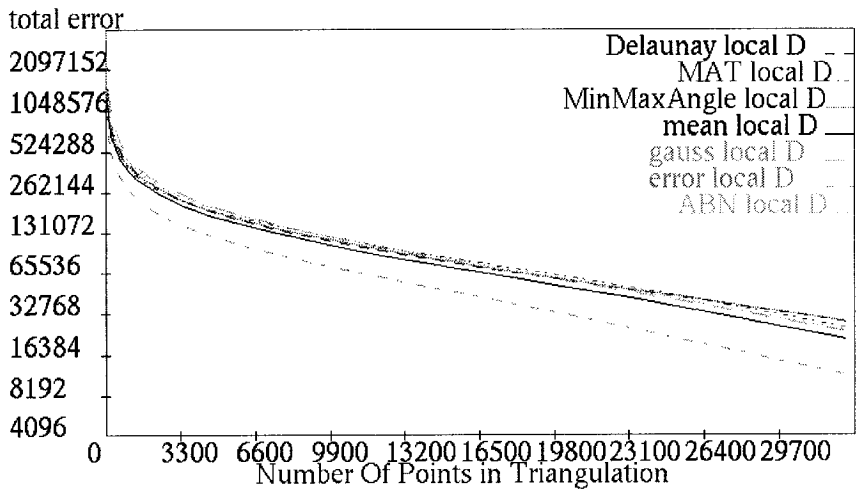

Figure 34: Total errors when deleting an independent set of points of a natural terrain

4.3.7 Hybrid:

The sample hybrid terrain has a sharp peak and a valley. It has *128*128* points. We use Gforge to create it.

Refinement:

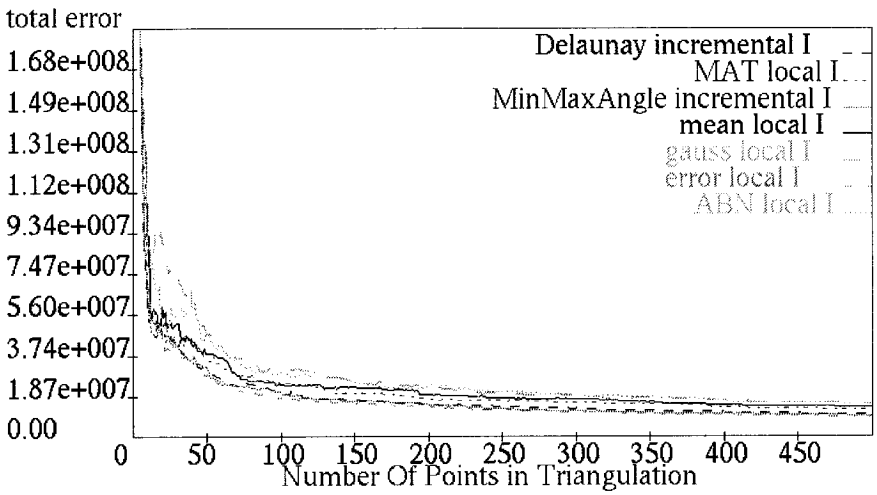

Figure 35: Total errors when inserting up to 500 points of a hybrid terrain sequentially

Decimation:

The full-detailed triangulation has *16,368* points. After an independent set of points is deleted from the initial triangulation, the number of points in the mesh is about *11,500*.

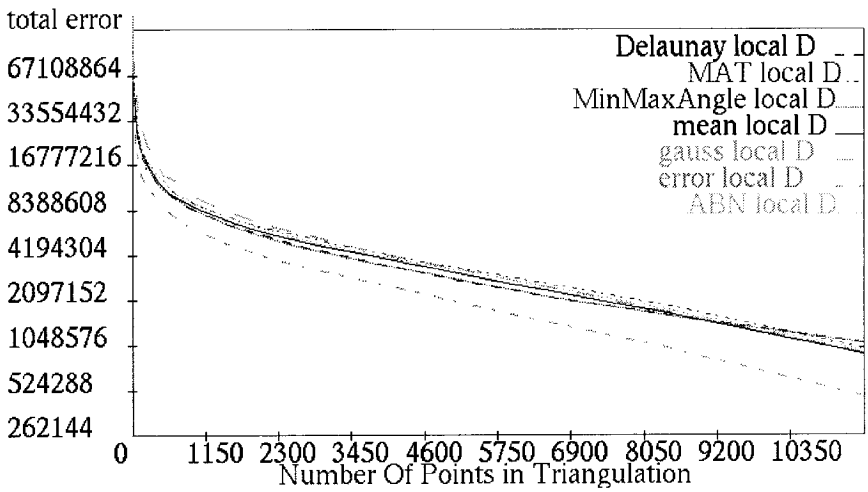

Figure 36: Total errors when deleting an independent set of points of a hybrid terrain each time

4.4 Time Complexity

We have tested the speed of simplification algorithm with respect to different criteria on Dell Inspiron *2500* laptop with an Intel Celeron *900* Processor and *192*MB of main memory. The data set we use to test the refinement algorithm is the Crater, whereas, the data set we use to test the decimation algorithm is the natural terrain. Table 13 shows the running time (seconds) of the refinement algorithm with respect to different criteria after *100,000* points are inserted. Figure 37 shows the running time (seconds) of the decimation algorithm with respect to different criteria. We will calculate the lasting time whenever an independent set of points is deleted and the resulting holes are re-triangulated.

Points Inserted	Delaunay	Min-Max angle	Min Area	Mean Curvature	Gaussian Curvature	ABN	L_2 measure
100000	9	16	14	38	1211	33	26

Table 13: Run time of refinement algorithm for different criteria on Crater terrain data set (Seconds)

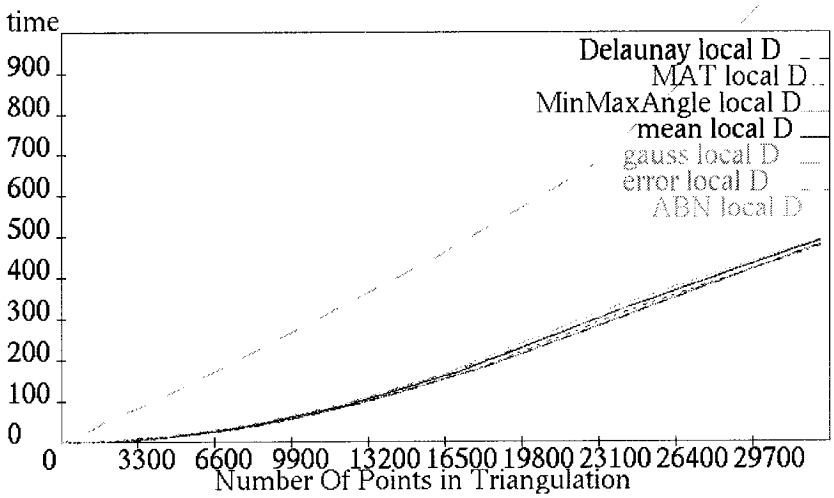

Figure 37: Run times of decimation algorithm for different criteria on a natural terrain data set (Seconds)

Chapter 5

Discussion

We will compare the performances of the refinement and decimation algorithm with respect to different criteria in the following aspects: speed, visual similarity to the original surface, slivers and analytical comparison based on error metrics.

5.1 Speed

Table 13 shows the time the refinement algorithm takes to insert *100,000* points of Crater into the triangle mesh with respect to different criteria. From section 3.1.2, we know that the time complexity of the greedy insertion algorithm that we use to find the optimal triangulation with respect to Delaunay triangulation and Min-Max angle criterion is $O((m+n)\log n)$, where m is the number of input points and n is the number of points in the mesh These two criteria have the same time complexity. Because when dealing with the data-dependent triangulations, we yield a Delaunay triangulation first and then apply the edge-flipping algorithm, which costs $O(n^2)$, on this initial triangulation, the data-dependent triangulations should run longer than Delaunay triangulation. Thus, in usual, the speed of the refinement algorithm with respect to Delaunay triangulation and Min-Max angle criterion is greater than all the other criteria, because the triangle mesh only needs to adjust locally after a point is inserted with respect to these two two-dimensional triangulations. However, sometimes some data-dependent triangulations may run faster than Min-Max angle triangulation if the number of swaps need to take is quite lower. Table 13 shows that Min Area criterion runs faster than Min-Max angle triangulation. It cannot tell which one runs faster among Min Area criterion, L_2 Measure

criterion, Mean curvature criterion and ABN criterion. It depends on how many swaps are needed to reach an optimal solution. The Gaussian curvature criterion is the slowest one among all criteria, although it has the same time complexity as L_2 Measure criterion, Mean curvature criterion and ABN criterion, because it takes much higher time to calculate the swap value of an edge.

Figure 37 shows the time of the decimation algorithm takes to delete an independent set of points from the full-detailed triangulation of the natural terrain data set step by step with respect to different criteria. Usually, Delaunay triangulation is faster than other criteria, because after a point is deleted from Delaunay triangulation, the resulting hole only needs to be retriangulated locally. Gaussian curvature criterion is the slowest one, because of the heavy calculation of the swap value of edges. It cannot tell which one runs faster among Min Area criterion, L_2 Measure criterion, Mean curvature criterion and ABN criterion. It depends on how many swaps will take to reach an optimal solution.

5.2 Visual Similarity

From Figure 10, we can see that the triangulations with respect to Min-Max angle criterion and Max-Min angle criterion look quite attractive and they enhance the significant features of Crater, whereas, the triangulations with respect to L_2 Measure criterion, ABN criterion and Gaussian curvature criterion look a little mess. They create visual artifact break line on the wall of the Crater Lake. Min Area criterion enhances the wall of the Crater Lake, but it arranges the wrinkle line on the wall in the horizontal direction, whereas, it should be in the vertical direction. Among them, Min-Max angle criterion looks best, because it displays more feature line of the Crater than other criteria.

5.3 Slivers

From Figure 10, we can see that Gaussian curvature criterion, L_2 measure criterion and ABN criterion seem to yield much more many slivers than other criteria. Thus, the triangulations with respect to them produce visual artifacts. Because Gaussian curvature criterion produces many slivers, it sometimes causes calculation error when defining the type and calculating the integral Gaussian curvature of a vertex. Delaunay triangulation and Min-Max angle criterion yield triangles with good shape. Because the data-dependent triangulations create many very thin slivers, sometimes it is extremely difficult to find a convex triangulation of the resulting hole when a point is deleted.

5.4 Analytical comparison

Because visual similarity is subjective and vague, we will compare the criteria based on different error measures that are more objective. We test the criteria based on Crater, random terrains, terrains with significant features, some other natural terrains and functions. We try different parameters when using fault algorithm to produce a random terrain. Some of the random terrains are more curved and some are more flat. Some are smooth and some are rough. Because we want to avoid the possibility that the surfaces we create by the fault algorithm have the same geometric features, we also use a tool, called Gforge, to generate other kinds of surfaces. Moreover, we note that the surfaces we generate by these programs are likely different from natural terrains in the world. We also use the natural terrains to test different criteria.

5.4.1 Refinement

For the refinement algorithm with respect to different criteria, we can see that L_2 measure criterion, Min-Max angle criterion and Delaunay triangulation work better than the

other four criteria in most cases. L_2 measure criterion usually produces the lowest error among all, because L_2 measure criterion and the error measures are based on the same criterion. The error curves of Min-Max angle criterion and Delaunay triangulation often mix together. Sometimes, Min-Max angle criterion works better than Delaunay triangulation. Sometimes, it is otherwise. Min-Max angle criterion and Delaunay triangulation often produce the second lowest error, no matter the original terrains are smooth or rough, curved or flat. With many terrains, there even exists an obvious gap between the group of {L_2 measure criterion, Delaunay triangulation, Min-Max angle criterion} and the group of {Min Area criterion, Mean curvature criterion, Gaussian curvature criterion, ABN criterion}. We can see such gap in Figure 19, Figure 21, Figure 25 and Figure 27. There is a gap in the refinement of Crater in Figure 12, Figure 13, Figure 14, Figure 23, Figure 33 and Figure 35 also, but it is less obvious. We try to use different parameters to create different random terrains by programs. We found that there always exists a gap. If the terrain we create is rougher, the gap is bigger. If the terrain we generate is smoother, the gap is smaller. We believe that the reason is probably due to the fact that ABN, mean curvature and Gaussian curvature criteria are all smooth criteria. Indeed, mean curvature criterion is often used to smooth surfaces. Thus, they are not suitable for approximating rough surfaces.

Mean curvature criterion works better than Gaussian curvature criterion in general, especially the approximated surfaces are rough and the number of points inserted are small. We can see this in Figure 12, Figure 19, Figure 21, Table 5, Figure 27, Table 7, Figure 33 and Figure 35.

Table 5 and Table 7 indicate that when the surface is rough in small-scale, Gaussian curvature criterion often performs worst in refinement algorithm. When the surface is smooth

in small-scale, Gaussian curvature criterion works poorly in the beginning. However, as more and more points inserted, Gaussian curvature criterion may become better than other data-dependent criteria, such as Mean curvature criterion sometimes. Table 1 and Table 3 demonstrate that when the number of points inserted is over *50%*, Gaussian curvature criterion has lower error than Mean curvature criterion. Gaussian curvature criterion approximates the surfaces that have zero curvature in one direction and non-zero curvature in the other direction well, because in such case, Gaussian curvature is zero. For example, Figure 31 and Table 11 show that Gaussian curvature criterion has the lower error than two-dimensional triangulation when approximating a cylinder.

Min area criterion usually does not work well, compared with other criteria, but it works quite well when approximating cylinders. Figure 31 and Table 11 show that Min Area criterion almost always has the lowest error.

5.4.2 Decimation

For decimation algorithms, L_2 measure criterion also produces the lowest errors among all criteria, as we see in the refinement algorithm, because L_2 measure criterion and error measures are based on the same criterion. We also find that there is a gap between the error curve of L_2 measure criterion and the error curves of all other criteria, whereas, we in general do not find such gap in the refinement algorithm. In many of the cases, the gap in the decimation algorithm is quite big. We can see this big gap in Figure 16, Figure 22, Figure 26, Figure 28, Figure 30 and Figure 34.

When approximating the rough surface, the Gaussian curvature criterion often produces higher error than other criteria. For example, we generate a smooth terrain and a rough terrain by the fault algorithm. Gaussian curvature criterion works the second best when

approximating the smooth one, but it works the worst when approximating the rough one. Figure 22, Figure 26, Table 6, Figure 28 and Table 8 show that Gaussian curvature criterion often has the highest error when approximating rough peak, valley and random terrain. So we tried changing the parameters, such as *Displacement* in the fault algorithm, to create other rough terrains. We found that Gaussian curvature criterion still often works the worst when approximating such rough terrains. When testing on the smooth random terrains, we found that Gaussian curvature criterion always works the second best to the L_2 measure criterion no matter the surface is more curved or more flat. Figure 16, Figure 20 and Figure 24 show that it often is the second best to L_2 measure criterion among all.

Unlike we see in the refinement algorithm, the two dimensional triangulations Min-Max angle triangulation and Delaunay triangulation are not better than data-dependent triangulations for the decimation algorithm in general. The error curves of Min-Max angle and Delaunay triangulation are often twisted together in decimation algorithm too, but Min-Max angle criterion is not as stable as Delaunay triangulation. Sometimes, it produces an abnormally high error. We can see this in Figure 30 and Figure 32. Maybe one of the reasons is that we cannot find the global optimum with respect to Min-Max angle criterion by local swap algorithm.

Actually, there exist some cases that Delaunay triangulation, Min-Max angle criterion, mean curvature criterion or Gaussian criterion works the second best to the L_2 measure criterion. Figure 36 gives the case when Delaunay triangulation works the second best. Figure 34 gives the case when mean curvature criterion works the second best. Although mean curvature criterion is a smooth criterion, it sometimes works well when the surface is a little rough. So do Delaunay triangulation and Min-Max angle criterion, because they can eliminate

the little noise on the surface. Figure 34 shows that mean curvature criterion produces very low error when approximating a slightly rough surface.

Min area criterion works worst in many cases. We can see this in Figure 16, Figure 20 and Figure 24. But it sometimes also works quite well. For example, when the number of points deleted is small, Min area criterion works much better than other criteria except L_2 measure criterion. We can see this in Table 6 and Table 8.

The reason that the two-dimensional triangulations do not work well for the decimation algorithm is probably because the decimation algorithm deletes an independent set of points each time, whereas, the refinement algorithm inserts one point at a time. The refinement algorithm will insert the most importance point into the triangle mesh each time. Although Delaunay triangulation does not contain the elevation information of a point, the latest inserted point contains such information. Thus, Delaunay triangulation can adjust the mesh in the neighborhood of the inserted point to produce an optimal triangulation. However, in the decimation algorithm, we delete an independent set of points. Thus, the triangle mesh changes a lot after a deletion. So the elevation information that data-dependent triangulation has can help find a better triangulation. Thus, data-dependent triangulations work better for decimation algorithm than two-dimensional triangulations.

When we approximate the surface of a cylinder, the four data-dependent triangulations work better than Delaunay triangulation and Min-Max angle criterion for both refinement and decimation algorithm. Actually, cylinders are cliff surfaces whose one side of the brim has large different height from the other side of the brim. Our experiences show that data-dependent triangulations perform better than two-dimensional triangulations when approximating cliffs. Figure 32 and Table 12 demonstrate this.

Chapter 6

Conclusions

Garland [18] asserted that data-dependent triangulations except the L_2 measure triangulation did not approximate terrains better than two-dimensional triangulations in *1995*. However, after that, more data-dependent triangulations are proposed. For example, L. Alboul [3] suggested that the mean curvature criterion and Gaussian triangulations were the best triangulations in *1999* after conducting experiments on a few smooth function. In this thesis, we want to find out whether data-dependent triangulations perform better than two-dimensional triangulations on terrains or not and which criteria are best. We test for both refinement and decimation algorithms.

From the comparison we conduct in this thesis, we can conclude that L_2 measure criterion, Delaunay triangulation, Min-Max angle criteria perform better than the four data-dependent triangulations, Gaussian curvature criterion, mean curvature criterion, ABN criterion, Min Area criterion in refinement algorithm. L_2 measure produces the lowest error in most cases, but it usually generates many thin and long angles. Delaunay triangulation and Min-Max angle triangulation work the second best among all the criteria on terrains in refinement algorithm. Our conclusion on refinement algorithm is consistent to the assertion Garland [18] made. Garland suspected that data-dependent triangulations in general do not work well on terrains in refinement algorithm, probably because of the isotropy of curvatures in the natural terrains [18] and data-dependent triangulation works better than Delaunay triangulation and Min-Max angle criterion in the refinement algorithm only on some specific

surfaces, such as ruled surfaces [18], which are rare in the world. Our experiments on cylinders, one kind of ruled surfaces, also confirm that.

However, in our experiments, we found that data-dependent triangulations perform better than two-dimensional triangulations for decimation algorithms on certain terrains. For decimation algorithms, L_2 measure criterion also produces lowest error. Gaussian curvature criterion often works the second best when approximating smooth terrains, but it in general performs worst when approximating rough surfaces. Min area criterion performs very poor in many cases. However, there are some cases when Mean curvature criterion, Min area criterion, Delaunay triangulation or Min-Max angle criterion works the second best to L_2 measure criterion.

Delaunay triangulation has similar performance as Min-Max angle triangulation in both refinement algorithm and decimation algorithm. Sometimes Delaunay triangulation creates better results than Min-Max angle triangulation. Sometimes Min-Max angle is better, but Delaunay triangulation is more stable than Min-Max angle triangulation.

The shapes of approximations with Delauany triangulation and Min-Max angle criterion are the best. They seldom generate thin and long triangles, whereas, Gaussian curvature criterion, ABN criterion and L_2 measure criterion often generate many thin and long triangles.

In the refinement algorithm, Min-Max angle criterion and Delaunay triangulation run faster than all data-dependent triangulations. In the decimation algorithm, Delaunay triangulation runs faster than all other criteria. Gaussian curvature criterion is the slowest one both in refinement algorithm and decimation algorithm.

Moreover, in this thesis, we propose a new method to calculate the optimal triangulation with respect to Min Area criterion. Instead of using local swap algorithm to find an optimal

triangulation, we combine the greedy insertion algorithm and dynamic programming algorithm to find the optimal solution. When calculating an optimal triangulation with respect to Min Area criterion, we choose the most important point v and delete the edges of the triangle that contains v. Then we use dynamic programming to optimally re-triangulate the resulting hole with respect to the minimum area criterion

Bibliography

[1] L. Alboul, "Curvature criteria in surface reconstruction," University of Twente, Netherlands.

[2] L. Alboul and R. van Damme, "Polyhedral metrics in surface reconstruction: tight triangulations," Faculty of Mathematical Sciences, University of Twente 1995.

[3] L. Alboul, G. Kloosterman, G. Traas, and R. van Damme, "Best data-dependent triangulations," *Journal of Computational and Applied Mathematics*, vol. 119, pp. 1-12, 2000.

[4] P. Alliez and M. Desbrun, "Progressive Compression for Lossless Transmission of Triangle Meshes," presented at the 28th annual conference on Computer graphics and interactive techniques, 2001.

[5] D. S. Andrews, "Simplifying Terrain Models and Measuring Terrain Model Accuracy," in *Computer Science*: UBC, 1996.

[6] J. Beale, "Gforge: height field generator," in *http://beale.best.vwh.net*, 1995.

[7] M. de Berg, M. van Kreveld, M. Overmars, and O. Schwarzkopf, "Line segment intersection," in *Computational Geometry: Algorithms and Applications*, Springer-Verlag, Ed. Heidelberg, 1997, pp. 31-33.

[8] M. Bern and D. Eppstein, "Mesh generation and optimal triangulation," Xerox Palo Alto Research Center, Tech report CSL-92-1, 1992.

[9] M. Bern and D. Eppstein, "Polynomial-size nonobtuse triangulation of polygons," *International Journal of Computational Geometry & Application*, vol. 2, pp. 241-255, 1992.

[10] M. Bertram, J. C. Barnes, B. Hamann, K. I. Joy, H. Pottmann, and D. Wushour, "Piecewise optimal triangulation for the approximation of scattered data in the plane," *Computer Aided Geometric Design*, vol. 17, pp. 767-787, 2000.

[11] P. Brown, "Selective Mesh Refinement for Rendering," University of Cambridge, 1998.

[12] A. Ciampalini, P. Cignoni, C. Montani, and R. Scopigno, "Multiresolution Decimation based on Global Error," *The Visual Computer*, vol. 13, pp. 228-246, 1997.

[13] N. Dyn, K. Hormann, S. J. Kim, and D. Levin, "Optimizing 3D Triangulations Using Discrete Curvature Analysis," in *Mathematical Methods for Curves and Surfaces*, Tom Lyche and Larry L. Schumaker, Eds.: Oslo 2000, 2001, pp. 135-146.

[14] N. Dyn, D. Levin, and S. Rippa, "Algorithms for the construction of data dependent triangulations," in *Algorithms for Approximation II*, J.C. Mason and M.G. Cox, Eds. London: Chapman & Hall, 1990, pp. 185-192.

[15] A. R. Fernandes, "Terrain Tutorial.," in *http://www.lighthouse3d.com/opengl/terrain/index.php3?fault*, Lighthouse 3D, Ed.

[16] M. Franc and V. Skala, "Parallel Triangular Mesh Reduction," presented at ALGORITMY 2000 Conference on Scientific Computing, 2000.

[17] P. S. Heckbert M. Garland., "Survey of Polygonal Surface Simplification Algorithms," *Multiresolution Surface Modeling Course*, 1997.

[18] P. S. Heckbert and M. Garland, "Fast Polygonal Approximation of Terrains and Height Fields," Carnegie Mellon University, Tech Report CMU-CS-95-181, 1995.

[19] P. S. Heckbert and M. Garland, "Surface simplification using quadric error metrics," in *Computer Graphics*, vol. 31, *Annual Conference Series*, 1997, pp. 209-216.

[20] B. Junger and J. Snoeyink, "Selecting independent vertices for terrain simplification," presented at WSCG'98, Pilsen University of West Bohemia, 1998.

[21] O. Kreylos and B. Hamann, "On simulated annealing and the construction of linear spline approximations for scattered data," presented at the Joint EUROGRAPHICS-IEEE TVCG Symposium on Visualization, Vienna, Austria, 1999.

[22] X. Li, "A Hybrid Algorithm for Terrain Simplification," in *Department of Computer Science*. Vancouveur: UBC, 2003.

[23] S. de Marchi, "On computing derivatives for C interpolating schemes: an optimization," *Computing*, vol. 60, pp. 29-53, 1998.

[24] J. Mulligan, "Assignment 3: Fractal Terrain," in *Computer Graphics. Course CSCI 4229. http://www.cs.colorado.edu/~janem/CS4229/asgn3.pdf*. Boulder: University of Colorado.

[25] L. L. Schumaker, "Computing optimal triangulations using simulated annealing," *Computer Aided Geometric Design*, vol. 10, pp. 329-345, 1993.

[26] T. S. Tan, "Optimal two-dimensional triangulations," in *Coordinated Science Lab*. Urbana-Champaign: Univ. of Illinois, 1993.

[27] Z. Toth, "Towards an Optimal Texture Reconstruction," presented at Central European Seminar on Computer Graphics, Budmerice, SR, 2004.

[28] E. Verbree and P. van Oosterom, "The STIN Method: 3D-Surface Reconstruction by Observation Lines and DELAUNAY TENS," presented at the ISPRS working group III/3 workshop '3-D reconstruction from airborne laserscanner and InSAR data', Dresden, Germany, 2003.

Appendix A

Below we will list the average error, total error and L2 error plots of all criteria among Max-Min angle criterion, Min-Max angle criterion, Min area criterion, mean curvature criterion, Gaussian curvature criterion, ABN criterion and L2 measure criterion for the following terrains: the Crater, a smooth random terrain, a rough random terrain, a smooth peak, a rough peak, a valley, a sphere, a cylinder, a natural terrain and a hybrid terrain. We will display the error plots for both refinement and decimation process. We will demonstrate the error plots according to each kind of terrains in sequence.

1. Crater:

Refinement:

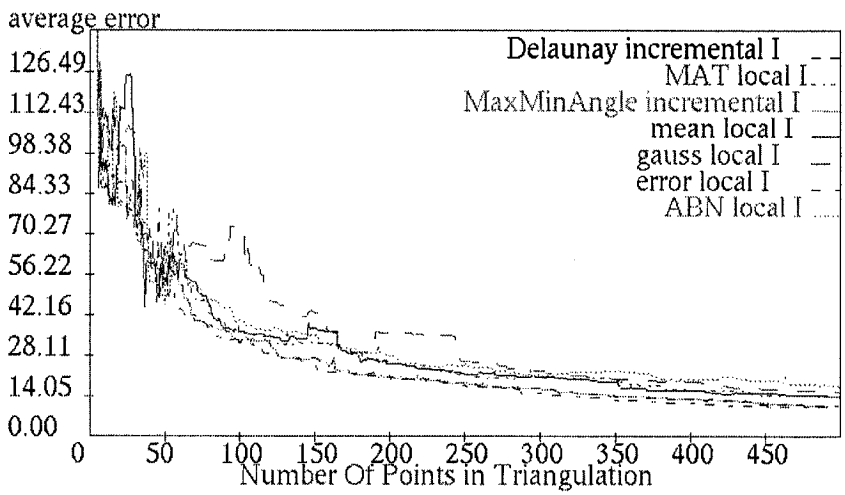

Figure A. 1: Average errors after inserting up to 500 points of Crater terrain data set sequentially

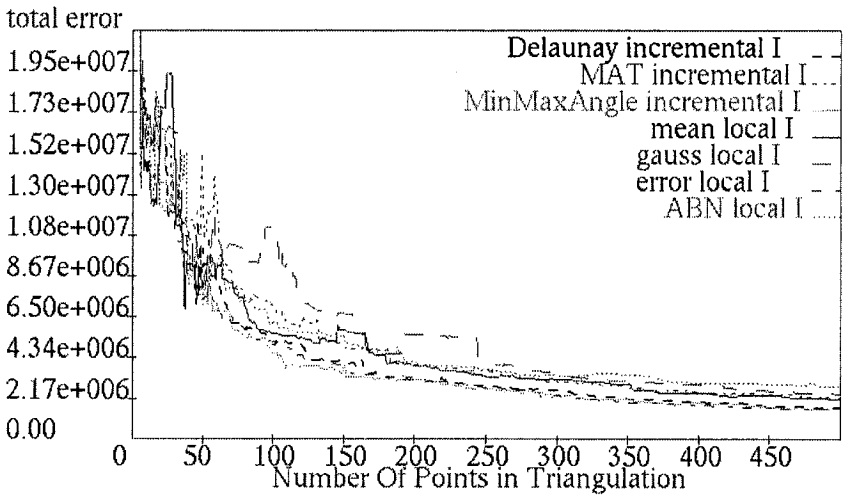

Figure A. 2: Total errors after inserting up to 500 points of Crater terrain data set sequentially

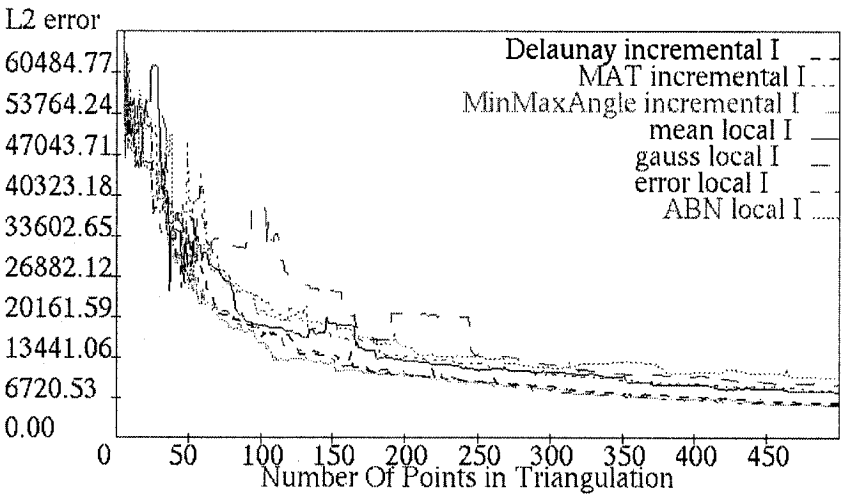

Figure A. 3: L2 errors after inserting up to 500 points of Crater terrain data set sequentially

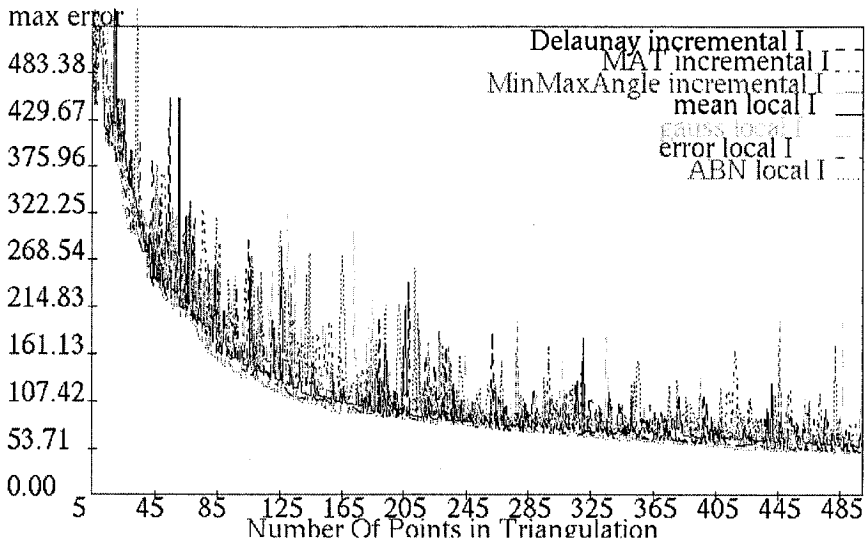

Figure A. 4: Max errors after inserting up to 500 points of Crater terrain data set sequentially

Decimation:

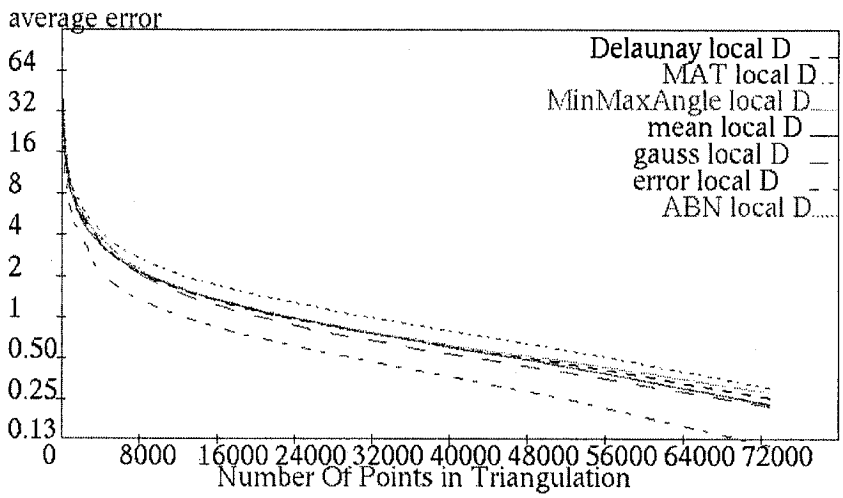

Figure A. 5: Average errors when deleting an independent set of points of the Crater terrain data set each time

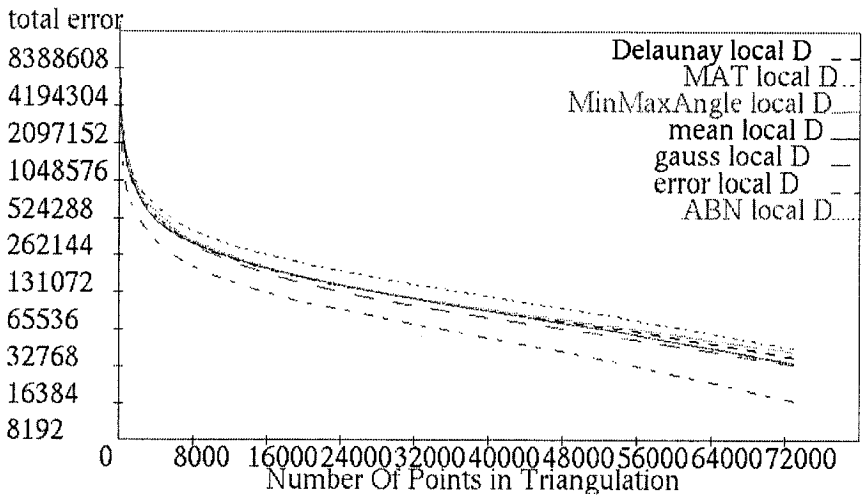

Figure A. 6: Total errors when deleting an independent set of points of the Crater terrain data set each time

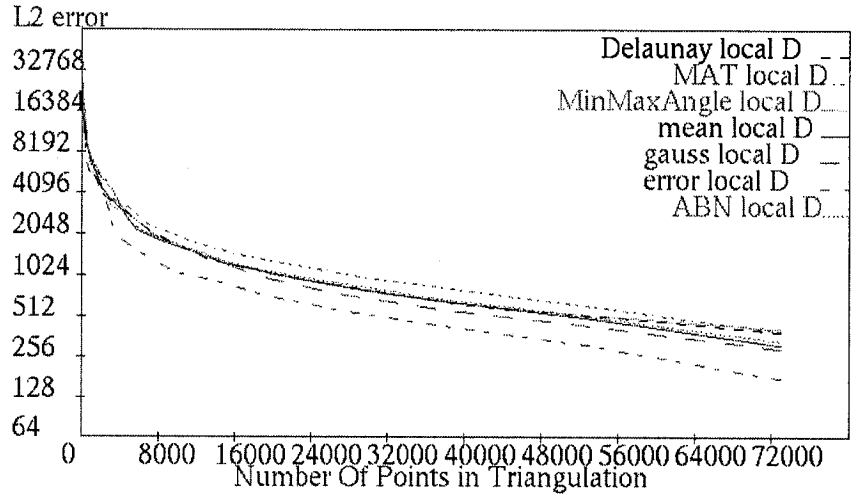

Figure A. 7: L2 errors when deleting an independent set of points of the Crater terrain data set each time

2. Smooth Random Terrain:

Refinement:

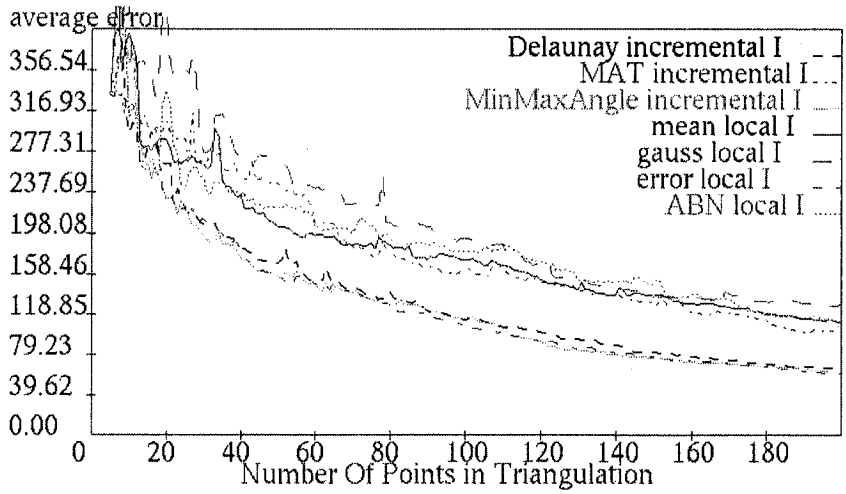

Figure A. 8: Average errors after inserting up to 200 points of smooth random terrain sequentially

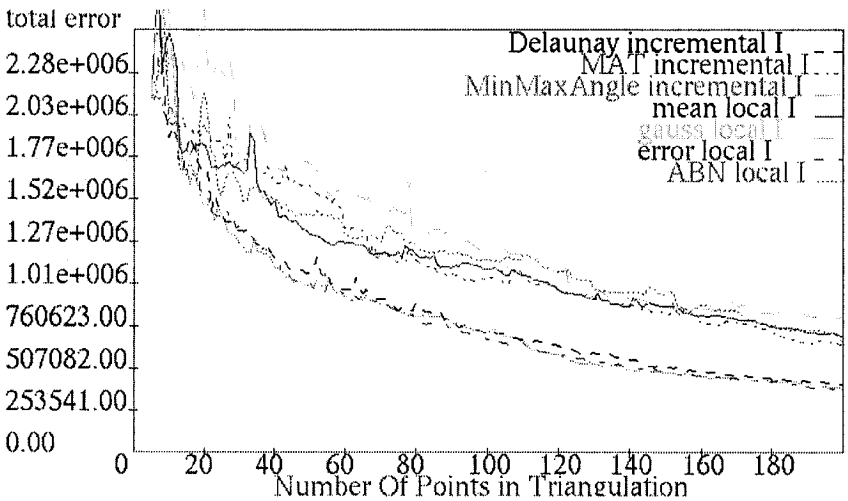

Figure A. 9: Total errors after inserting up to 200 points of smooth random terrain sequentially

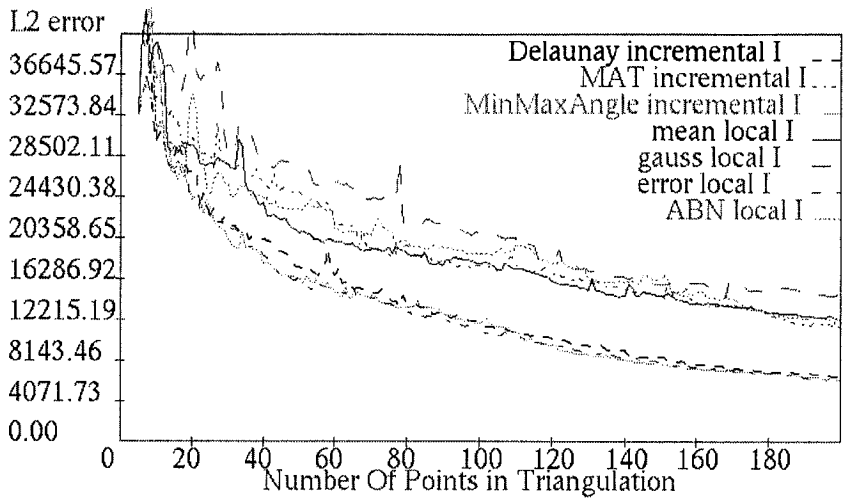

Figure A. 10: L2 errors after inserting up to 200 points of smooth random terrain sequentially

Decimation:

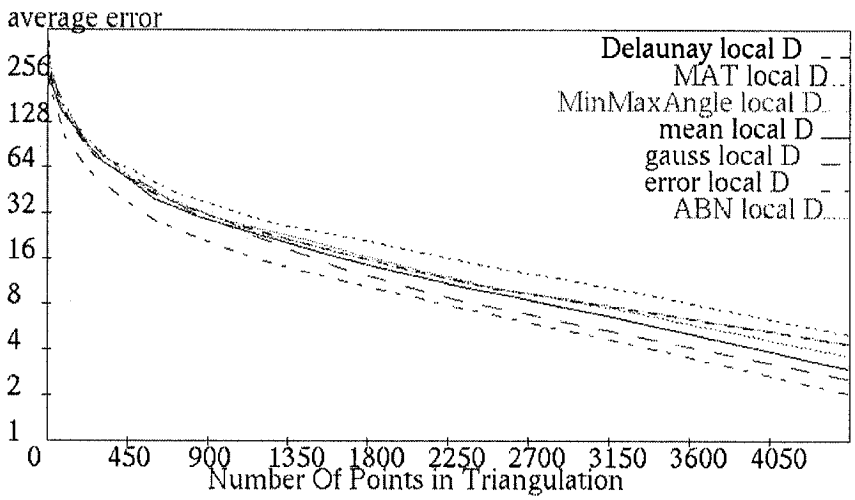

Figure A. 11: Average errors when deleting an independent set of points of the smooth random terrain each time

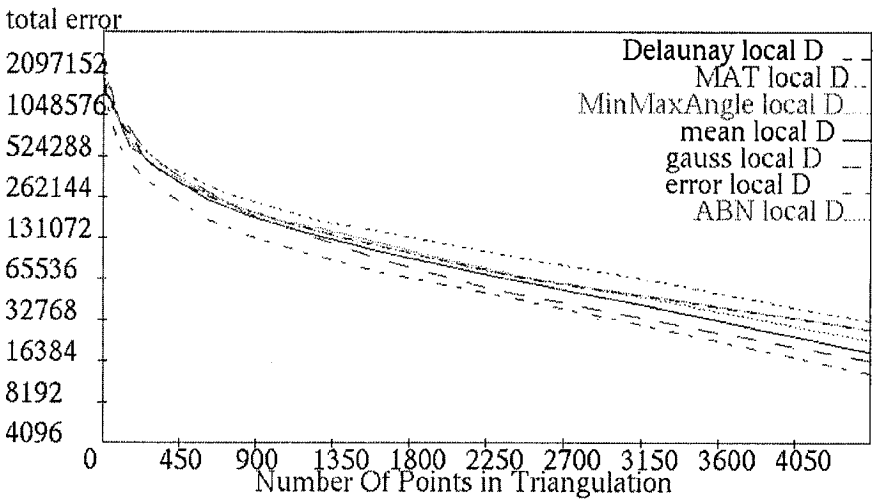

Figure A. 12: Total errors when deleting an independent set of points of the smooth random terrain each time

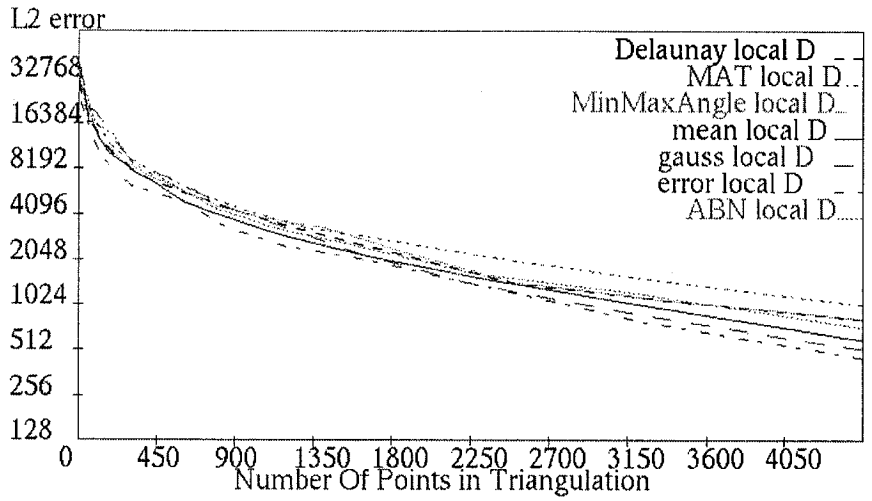

Figure A. 13: L2 errors when deleting an independent set of points of the smooth random terrain each time

3. Rough Random Terrain:

Refinement:

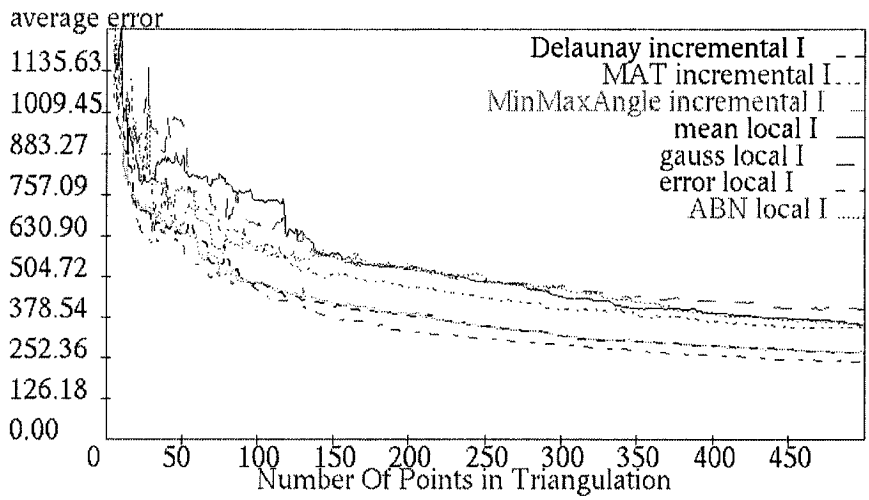

Figure A. 14: Average errors after inserting up to 500 points of rough random terrain sequentially

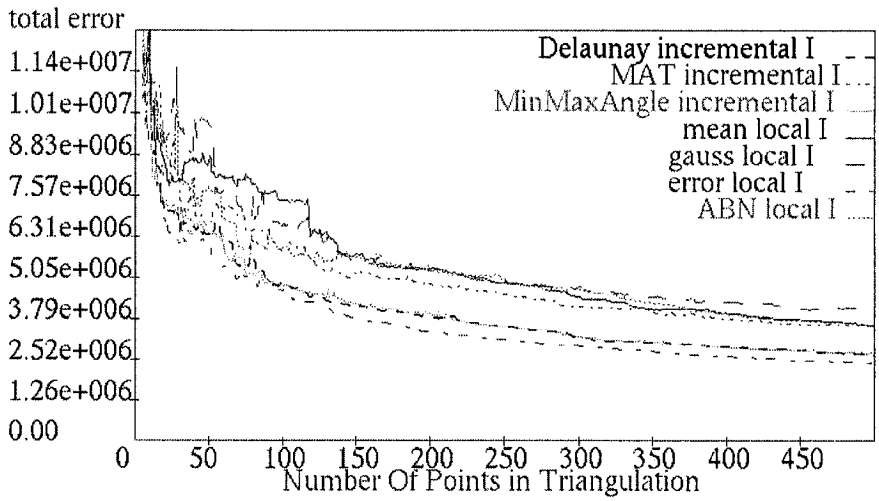

Figure A. 15: Total errors after inserting up to 500 points of rough random terrain sequentially

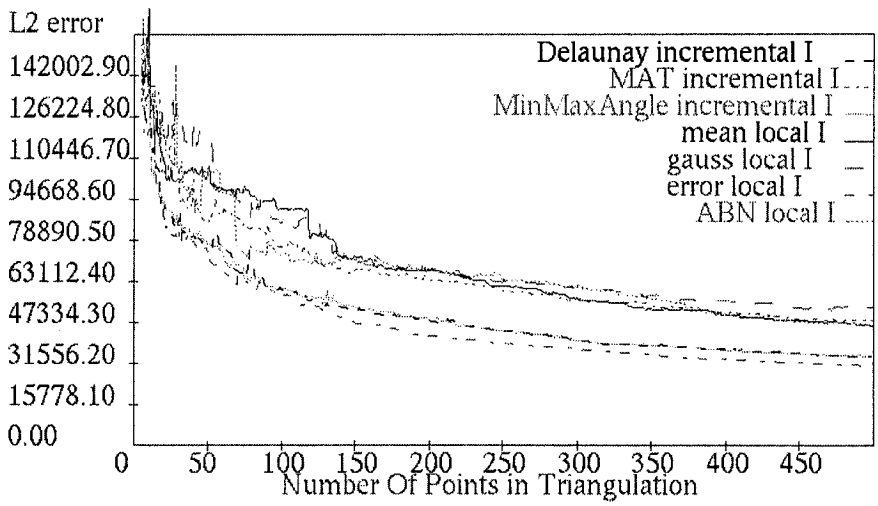

Figure A. 16: L2 errors after inserting up to 500 points of rough random terrain sequentially

Decimation:

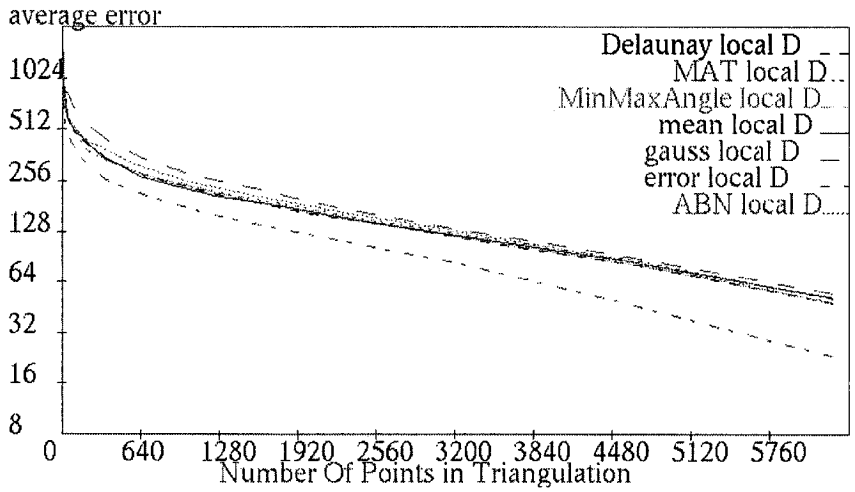

Figure A. 17: Average errors when deleting an independent set of points of the rough random terrain each time

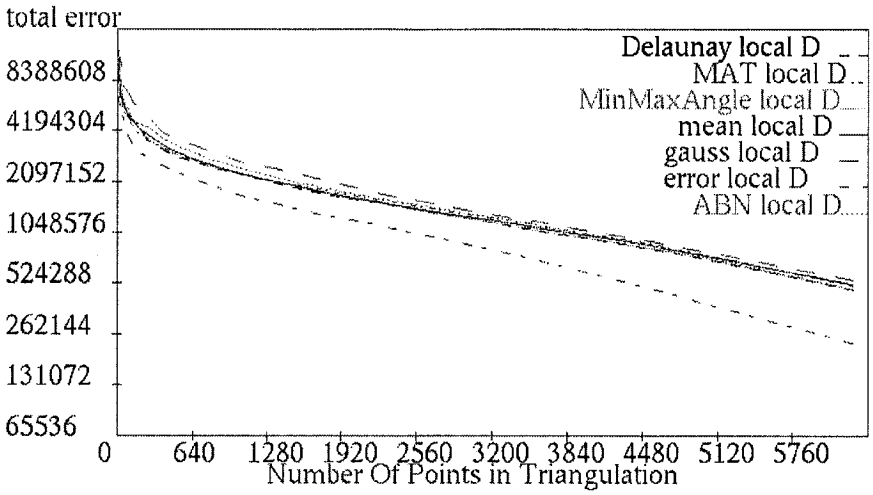

Figure A. 18: Total errors when deleting an independent set of points of the rough random terrain each time

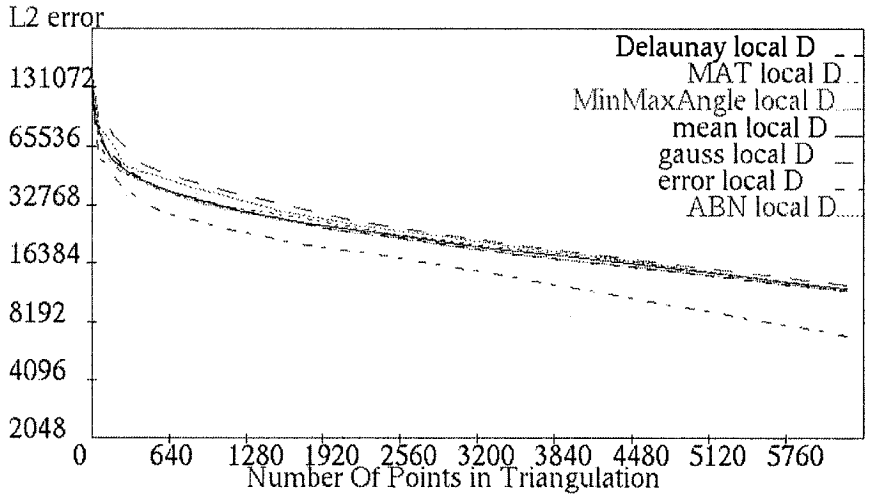

Figure A. 19: L2 errors when deleting an independent set of points of the rough random terrain each time

4. Smooth Peak:

Refinement:

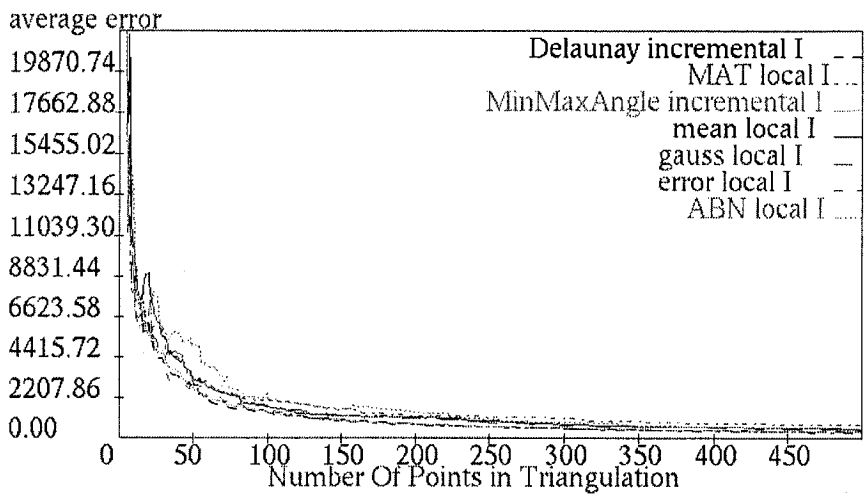

Figure A. 20: Average errors after inserting up to 500 points of smooth peak sequentially

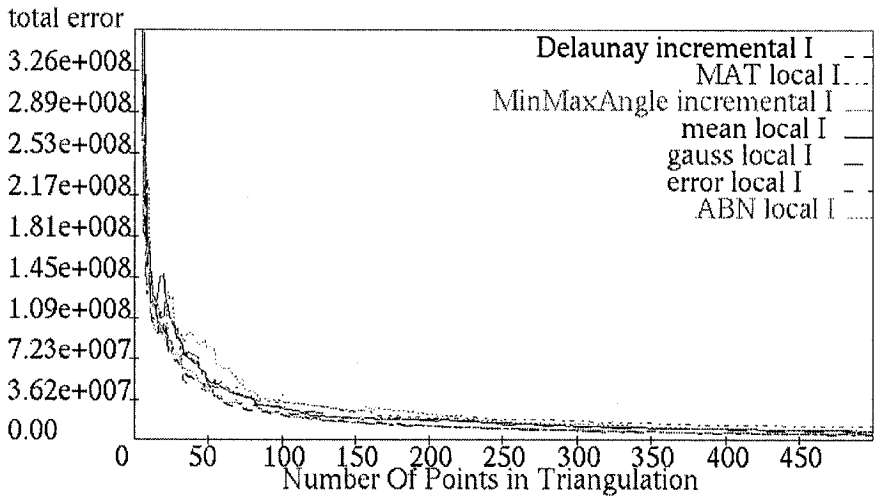

Figure A. 21: Total errors after inserting up to 500 points of smooth peak sequentially

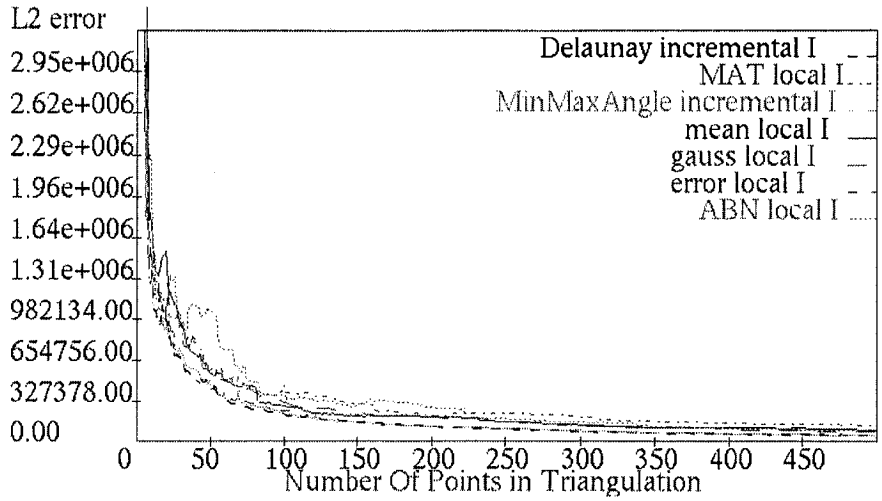

Figure A. 22: L2 errors after inserting up to 500 points of smooth peak sequentially

Decimation:

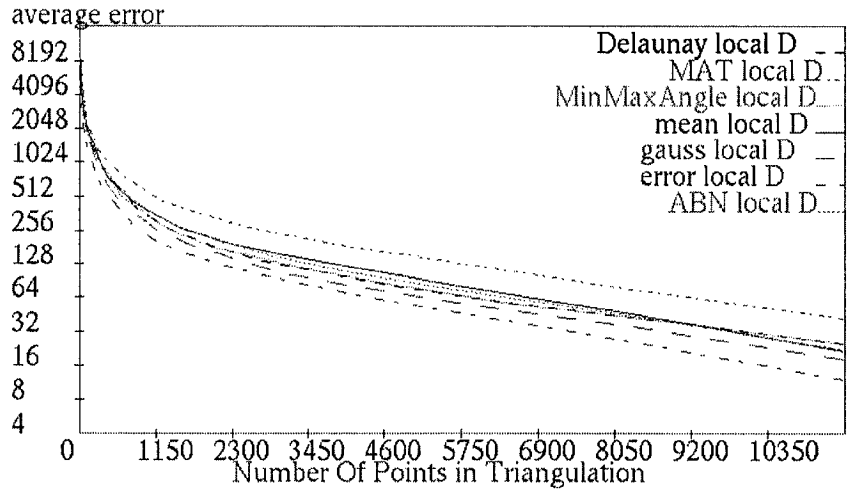

Figure A. 23: Average errors when deleting an independent set of points of the smooth peak each time

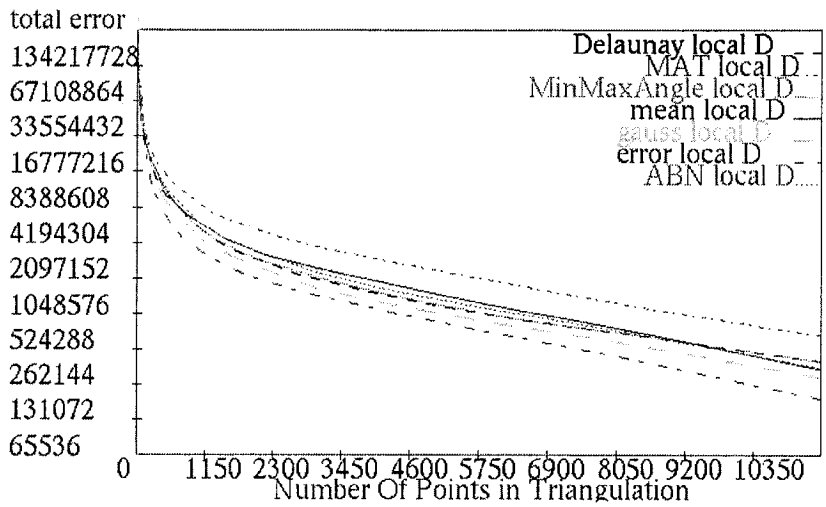

Figure A. 24: Total errors when deleting an independent set of points of the smooth peak each time

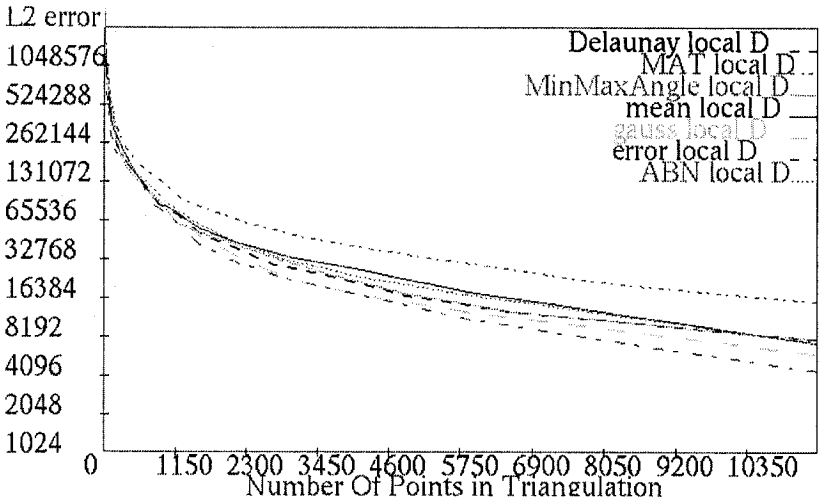

Figure A. 25: L2 errors when deleting an independent set of points of the smooth peak each time

5. Rough Peak:

Refinement:

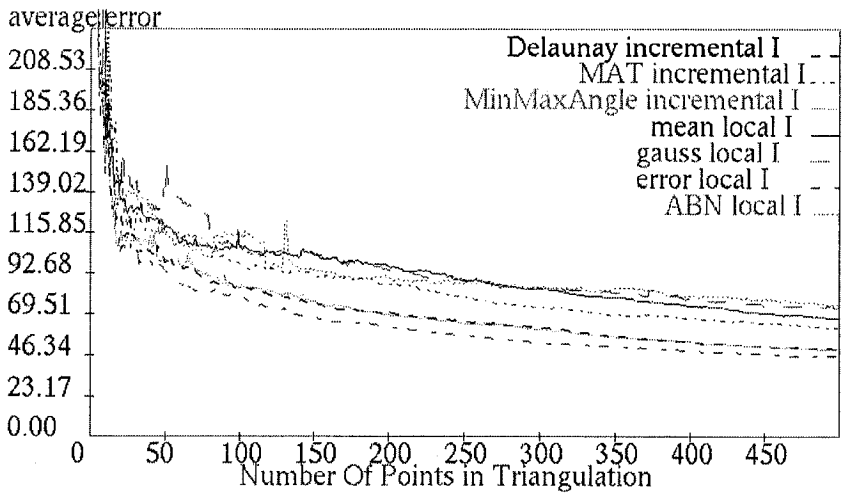

Figure A. 26: Average errors after inserting up to 500 points of rough peak sequentially

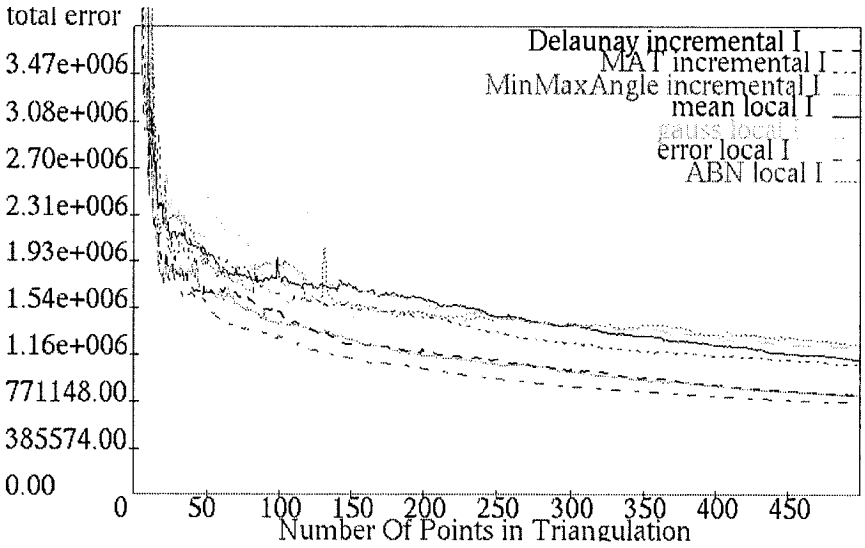

Figure A. 27: Total errors after inserting up to 500 points of rough peak sequentially

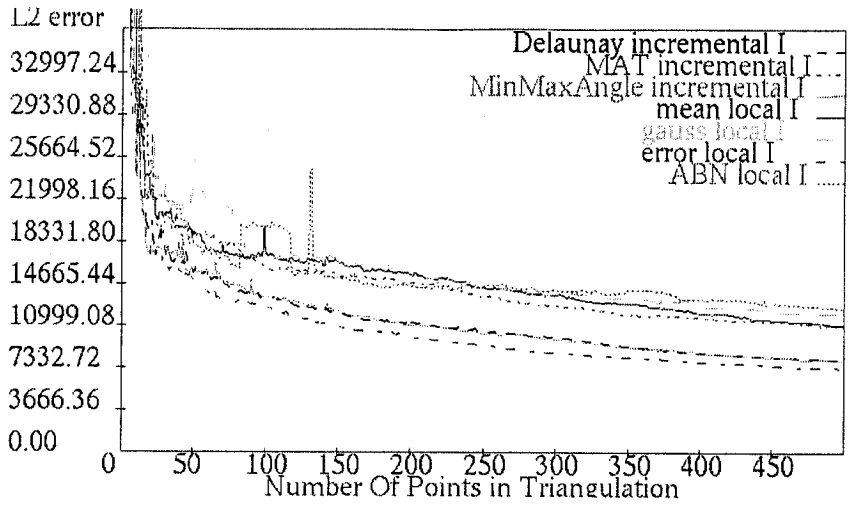

Figure A. 28: L2 errors after inserting up to 500 points of rough peak sequentially

Decimation:

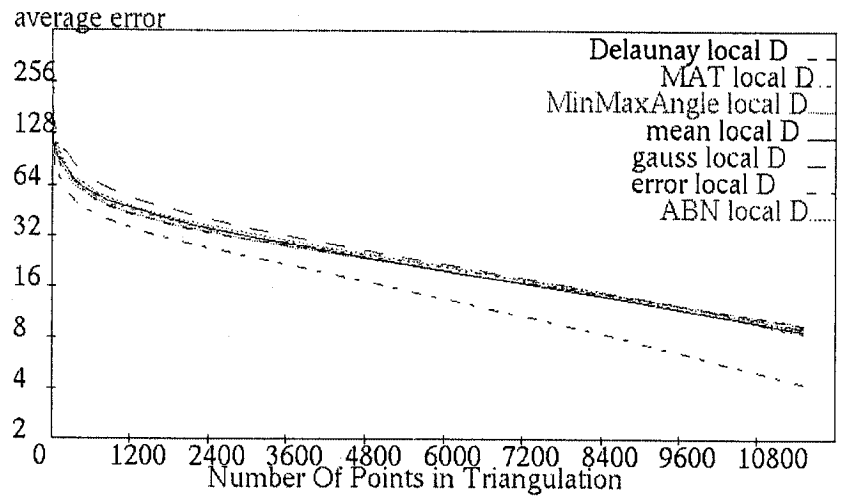

Figure A. 29: Average errors when deleting an independent set of points of the rough peak each time

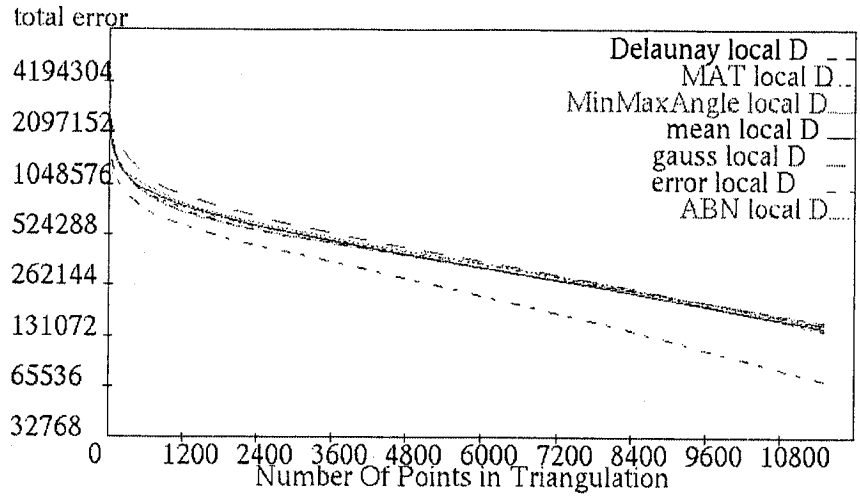

Figure A. 30: Total errors when deleting an independent set of points of the rough peak each time

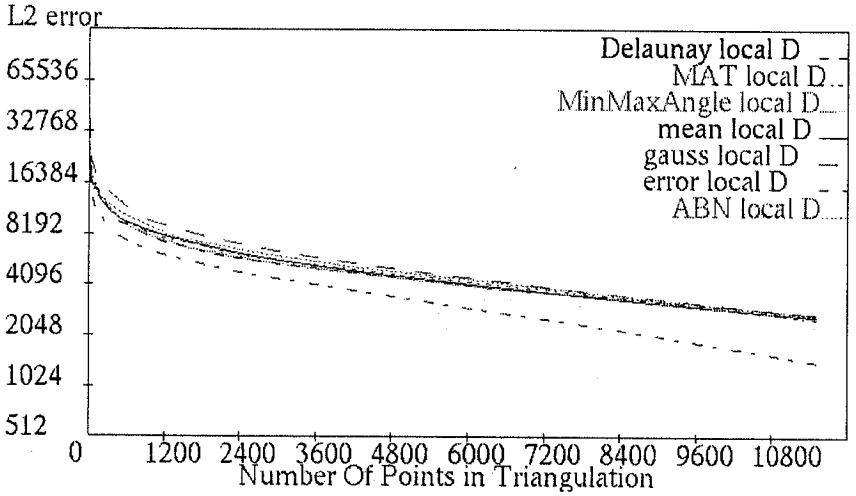

Figure A. 31: L2 errors when deleting an independent set of points of the rough peak each time

6. Valley:

Refinement:

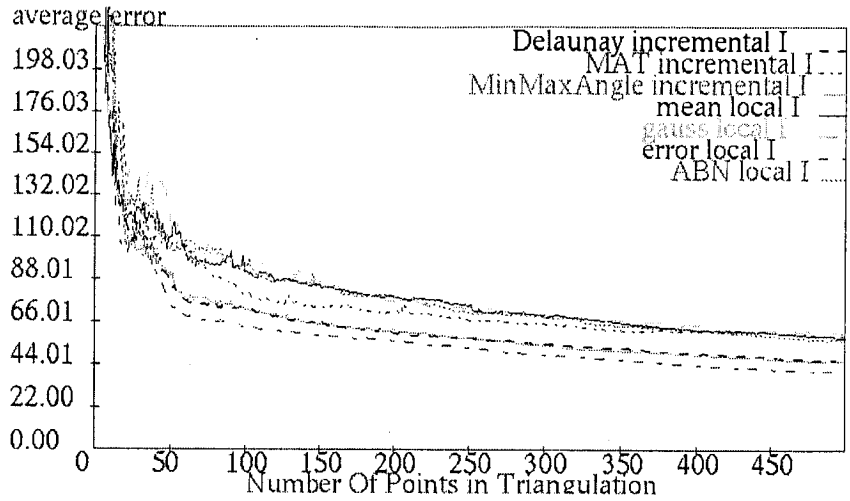

Figure A. 32: Average errors after inserting up to 500 points of rough valley sequentially

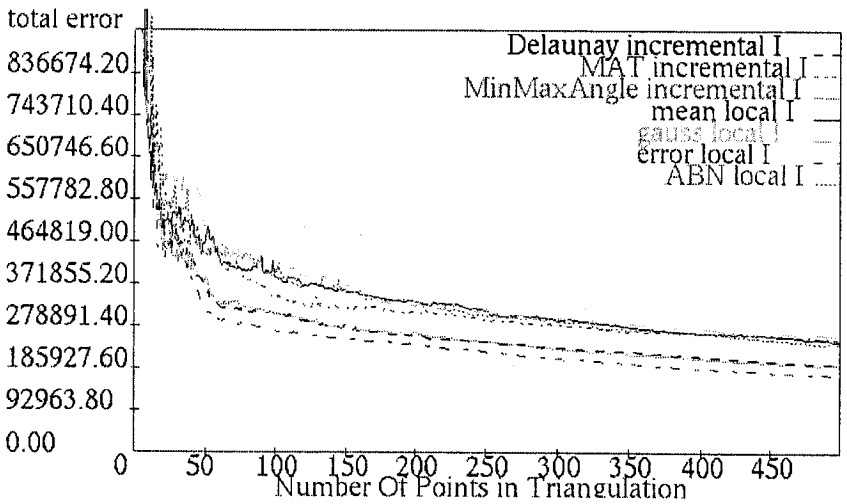

Figure A. 33: Total errors after inserting up to 500 points of rough valley sequentially

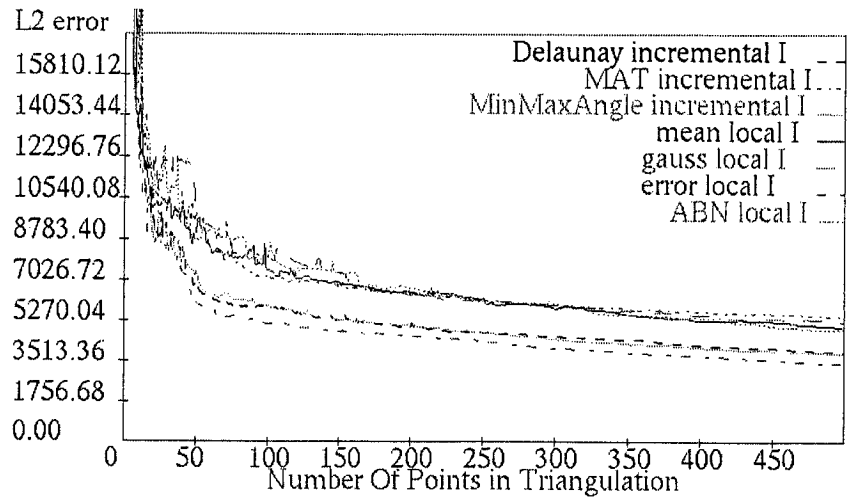

Figure A. 34: L2 errors after inserting up to 500 points of rough valley sequentially

Decimation:

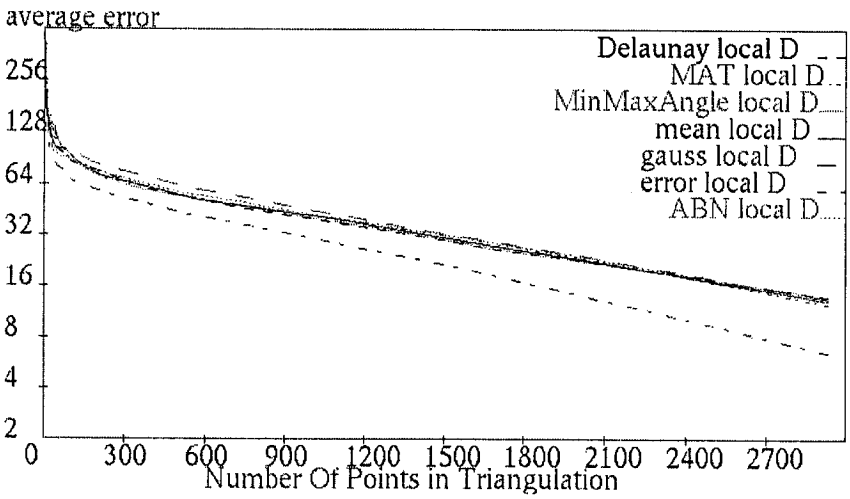

Figure A. 35: Average errors when deleting an independent set of points of the rough valley each time

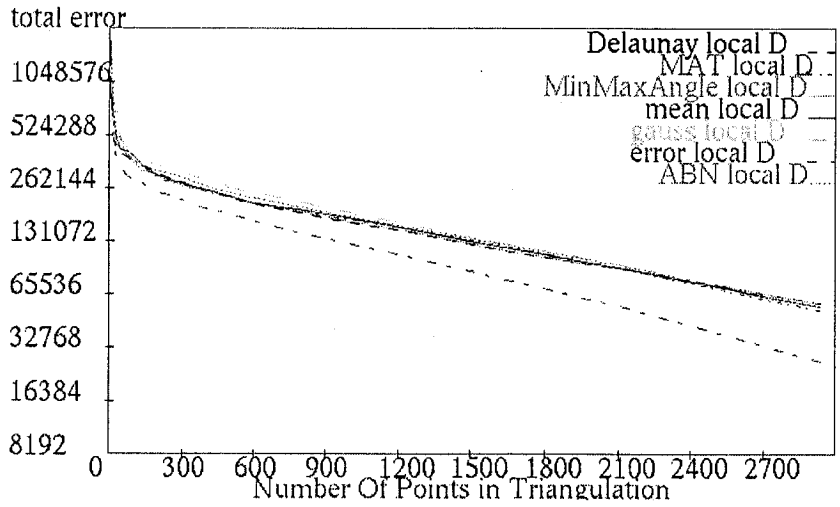

Figure A. 36: total errors when deleting an independent set of points of the rough valley each time

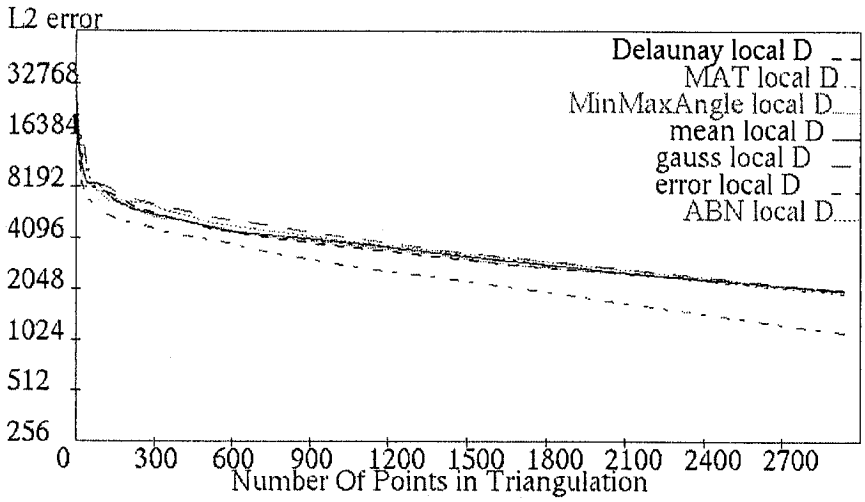

Figure A. 37: L2 errors when deleting an independent set of points of the rough valley each time

7. Sphere

Refinement:

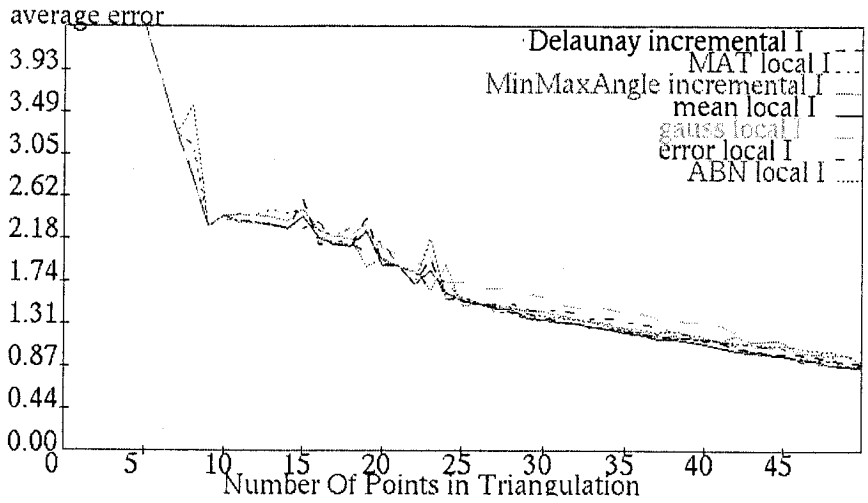

Figure A. 38: Average errors after inserting up to 50 points of a sphere sequentially

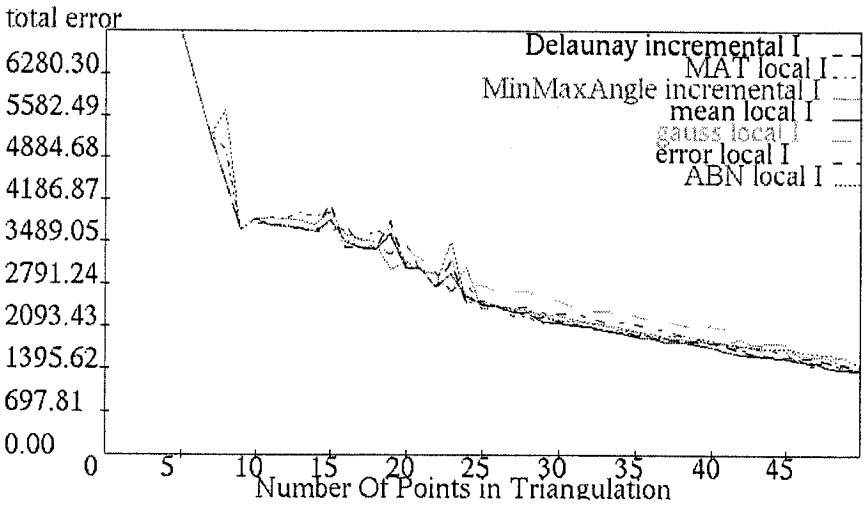

Figure A. 39: Total errors after inserting up to 50 points of a sphere sequentially

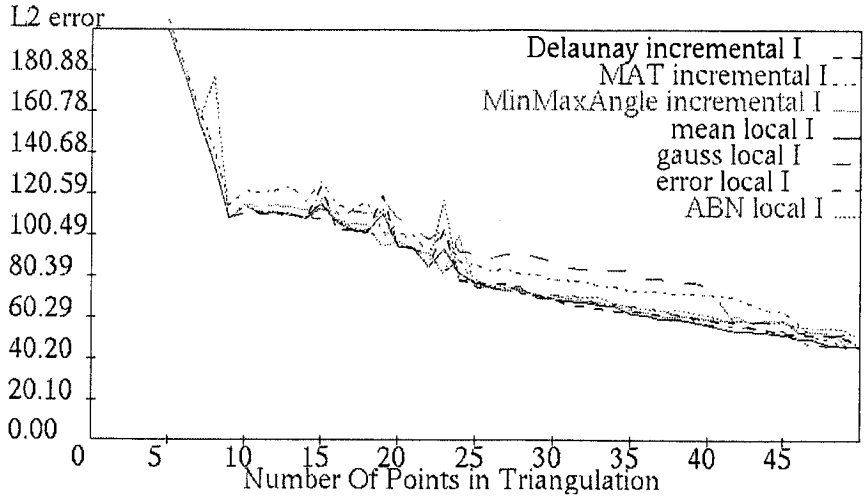

Figure A. 40: L2 errors after inserting up to 50 points of sphere sequentially

Decimation:

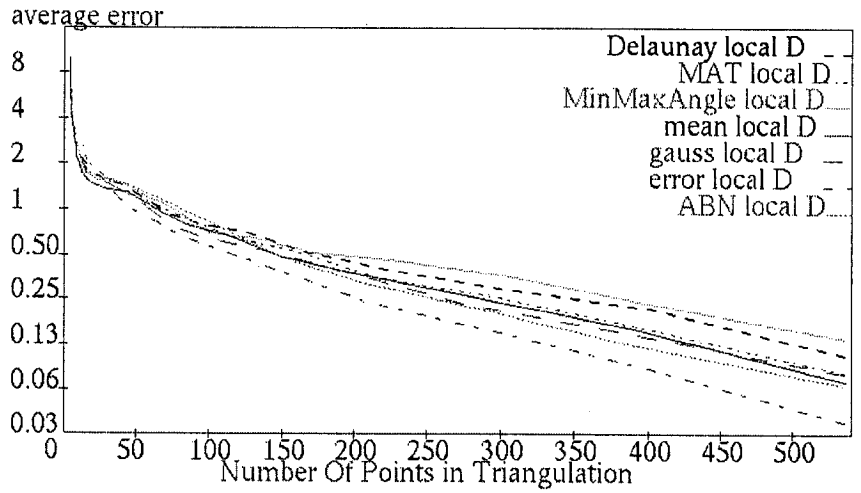

Figure A. 41: Average errors when deleting an independent set of points of a sphere each time

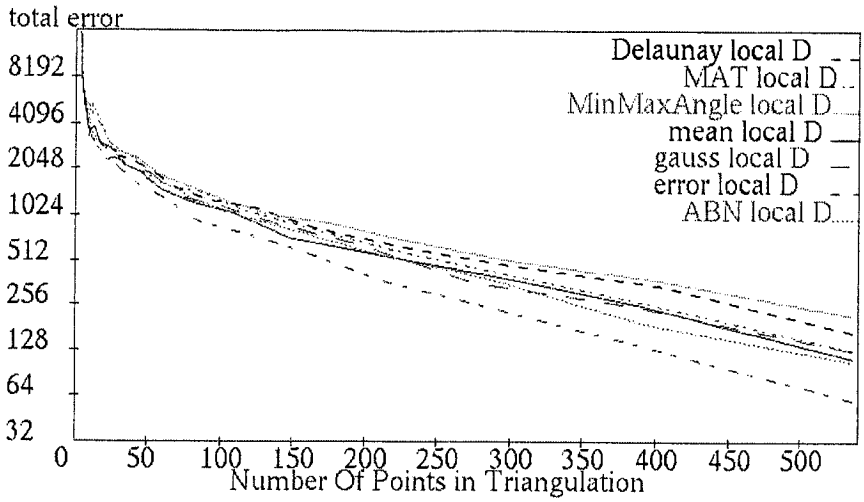

Figure A. 42: Total errors when deleting an independent set of points of a sphere each time

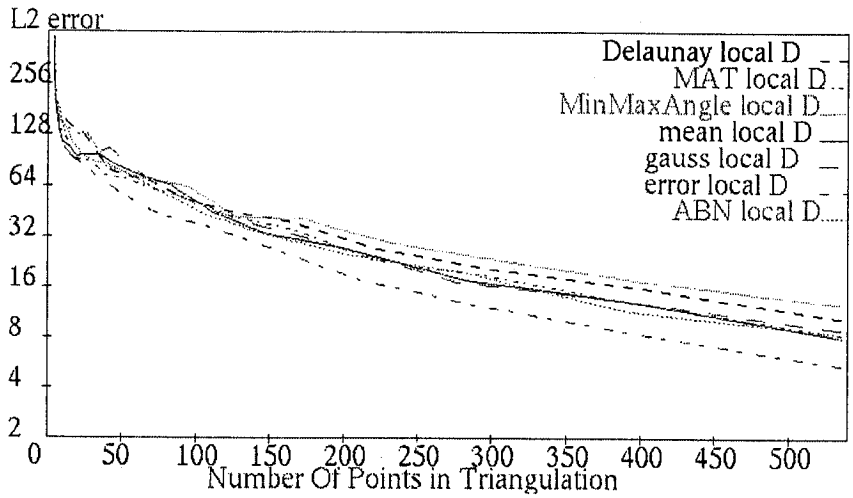

Figure A. 43: L2 errors when deleting an independent set of points of a sphere each time

8. Cylinder

Refinement:

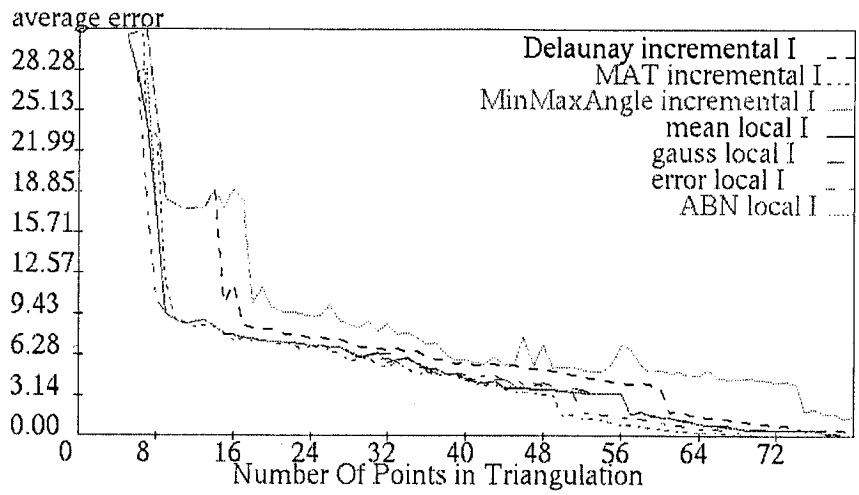

Figure A. 44: Average errors after inserting up to 80 points of a cylinder sequentially

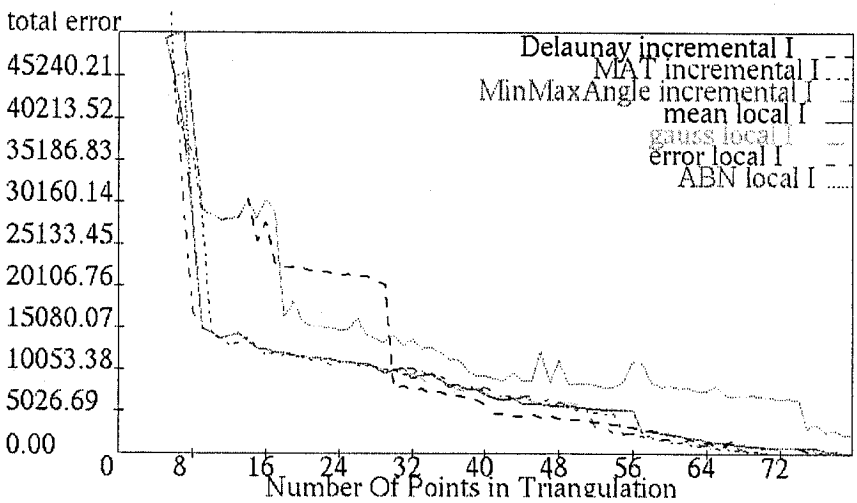

Figure A. 45: Total errors after inserting up to 80 points of a cylinder sequentially

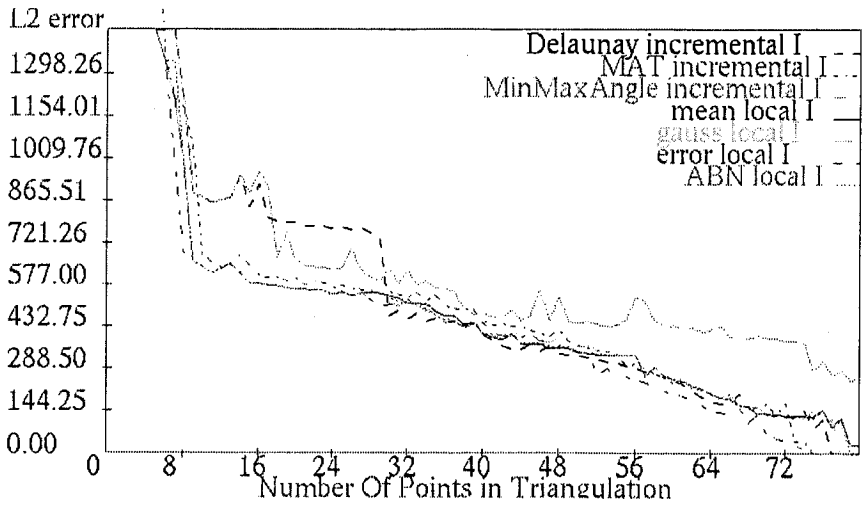

Figure A. 46: L2 errors after inserting up to 80 points of a cylinder sequentially

Decimation:

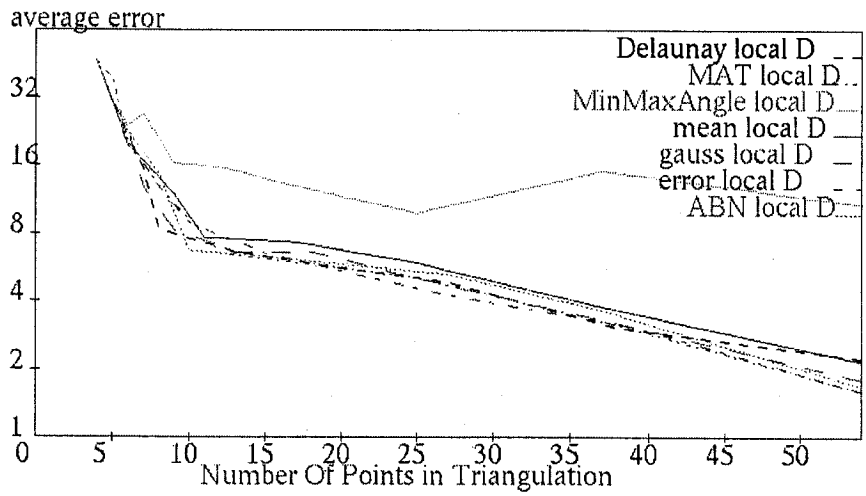

Figure A. 47: Average errors when deleting an independent set of points of a cylinder each time

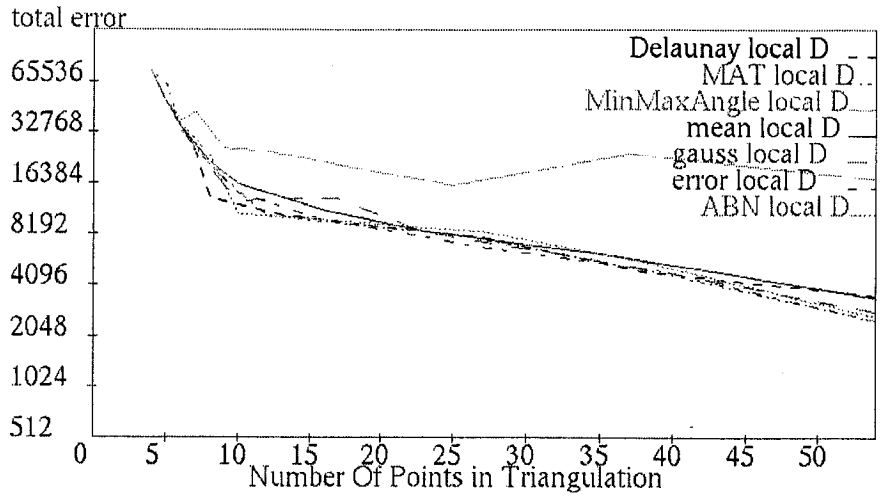

Figure A. 48: Total errors when deleting an independent set of points of a cylinder each time

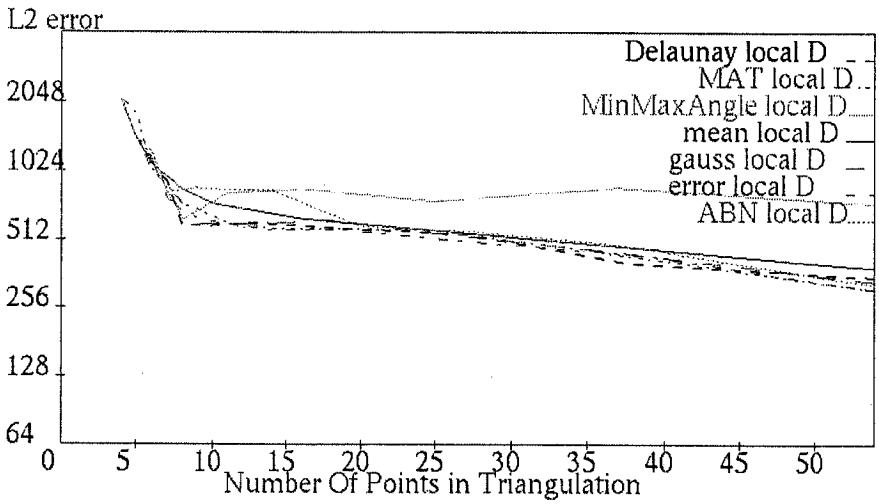

Figure A. 49: L2 errors when deleting an independent set of points of a cylinder each time

9. Natural Terrain:

Refinement:

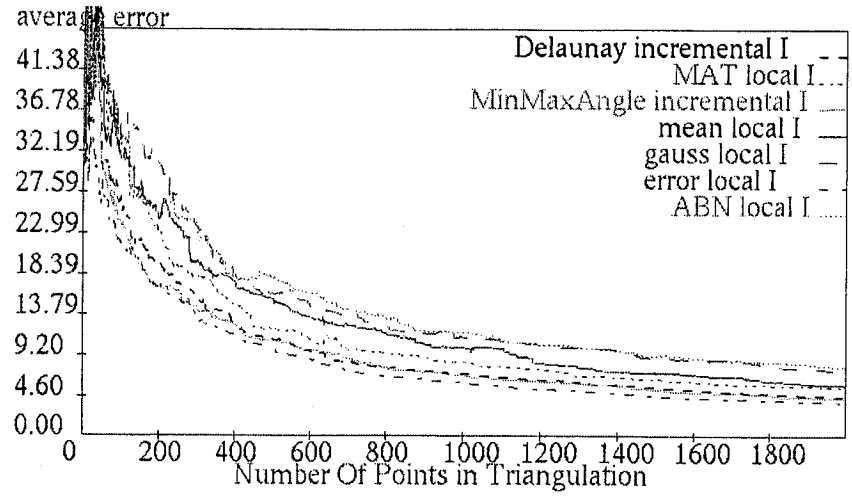

Figure A. 50: Average errors after inserting up to 2000 points of a natural terrain sequentially

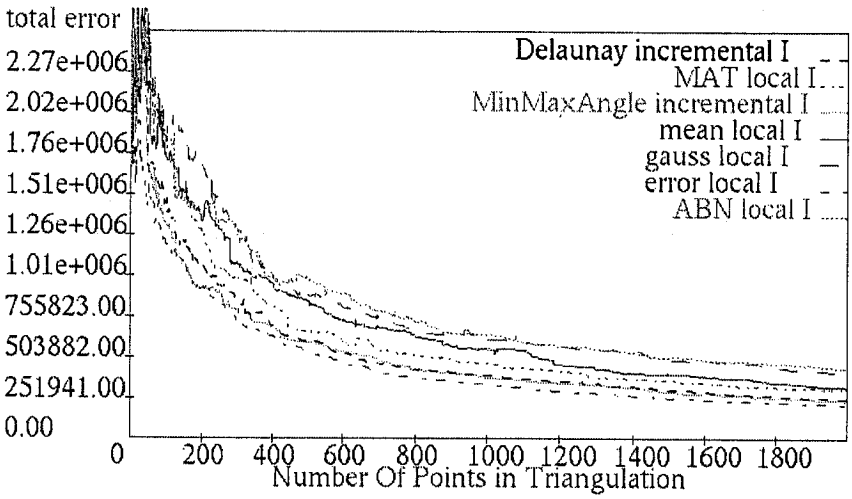

Figure A. 51: Total errors after inserting up to 2000 points of a natural terrain sequentially

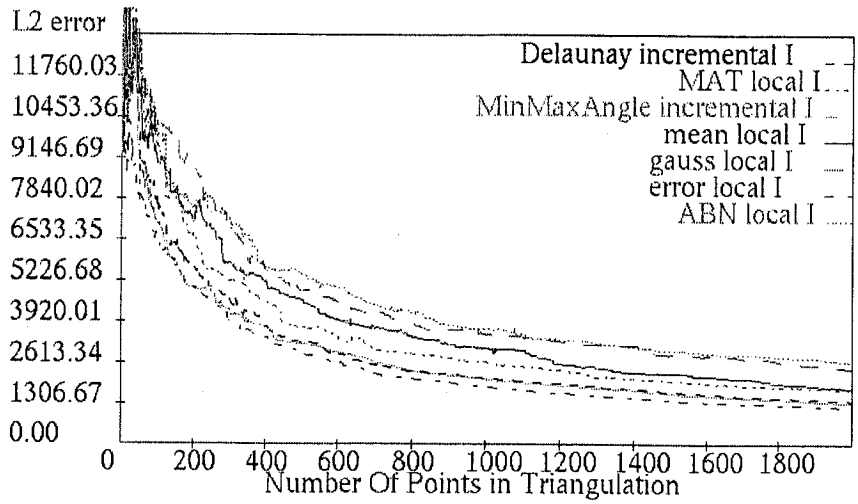

Figure A. 52: L2 errors after inserting up to 2000 points of a natural terrain sequentially

Decimation:

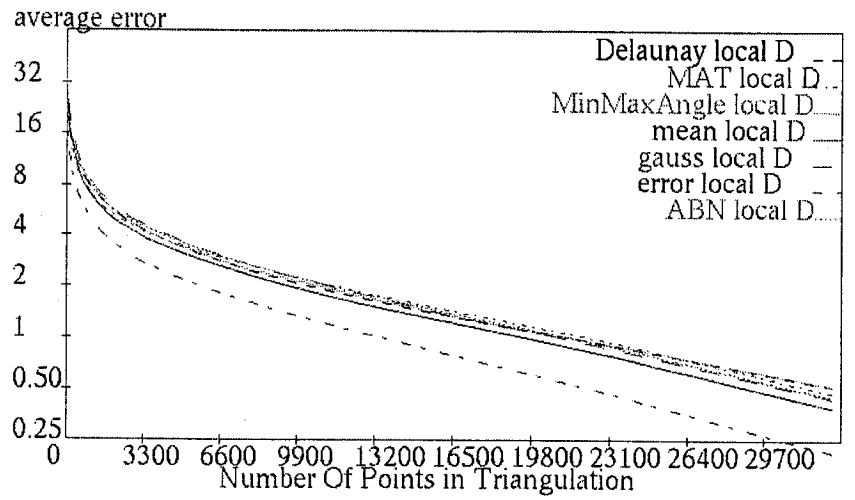

Figure A. 53: Average errors when deleting an independent set of points of a natural terrain each time

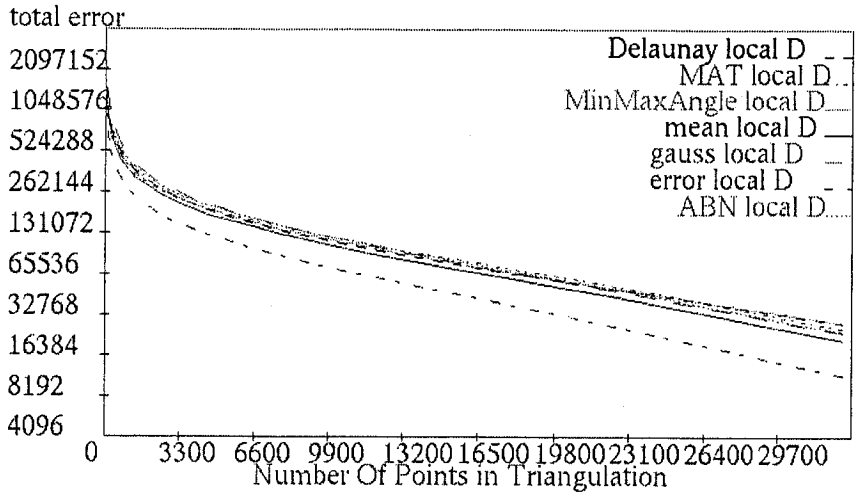

Figure A. 54: Total errors when deleting an independent set of points of a natural terrain each time

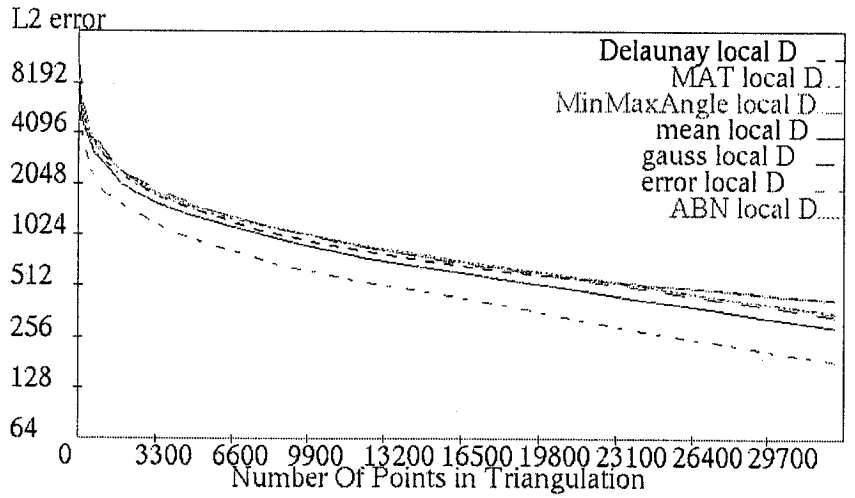

Figure A. 55: L2 errors when deleting an independent set of points of a natural terrain each time

6. Hybrid Terrain:

Refinement:

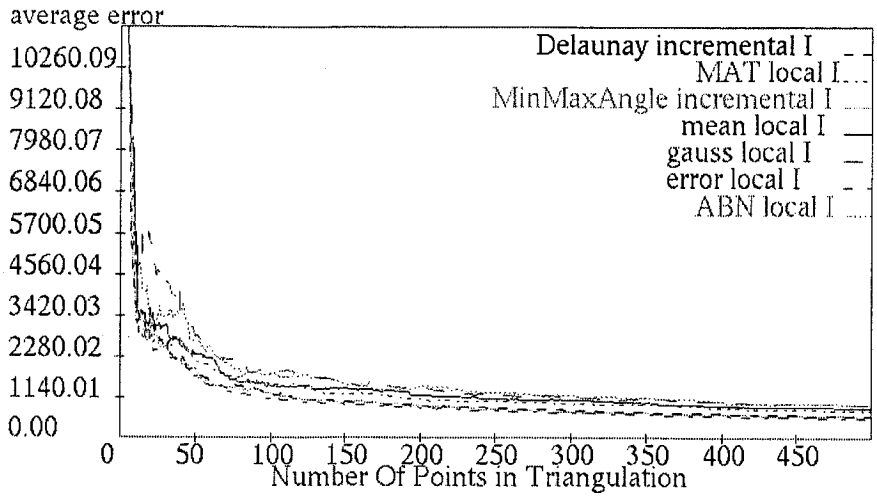

Figure A. 56: Average errors after inserting up to 500 points of a hybrid terrain sequentially

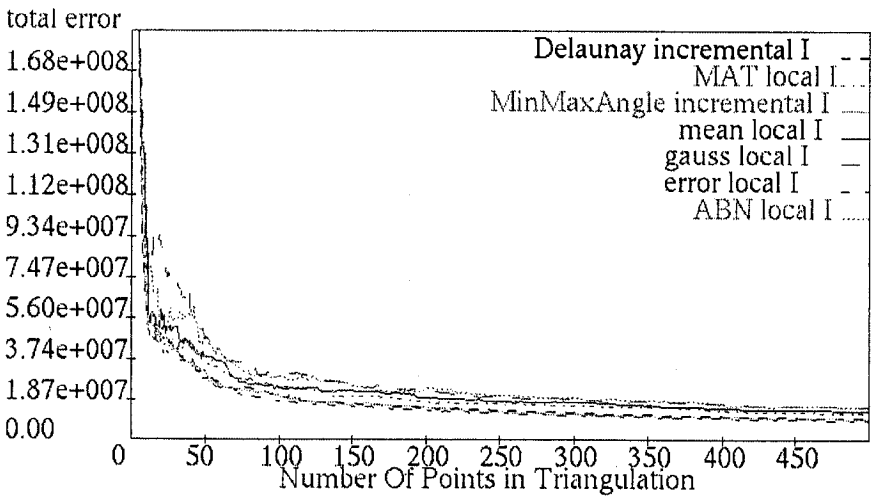

Figure A. 57: Total errors after inserting up to 500 points of a hybrid terrain sequentially

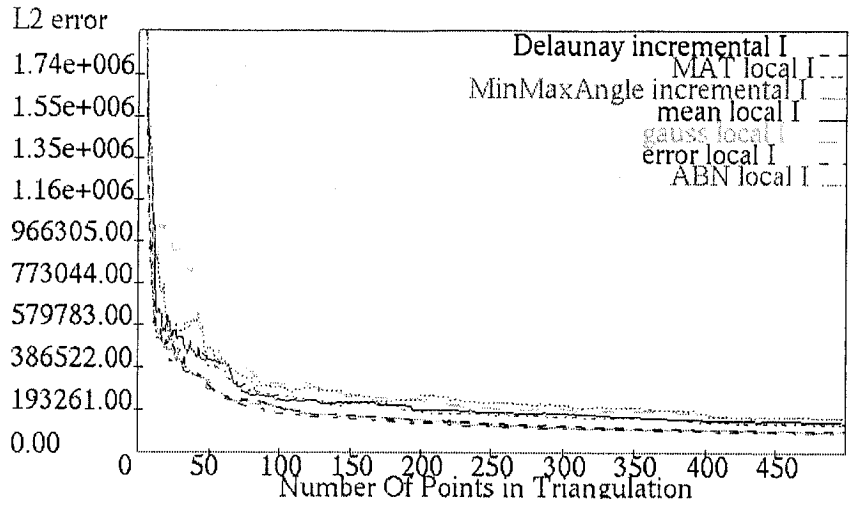

Figure A. 58: L2 errors after inserting up to 500 points of a hybrid terrain sequentially

Decimation:

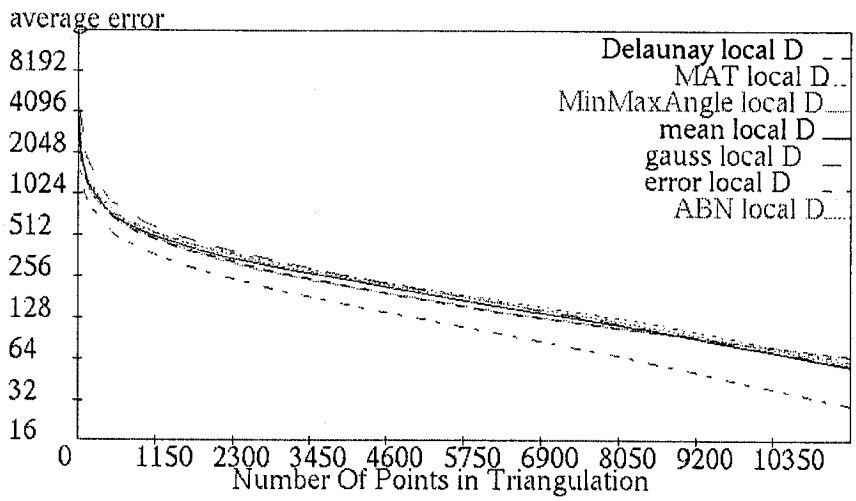

Figure A. 59: Average errors when deleting an independent set of points of a hybrid terrain each time

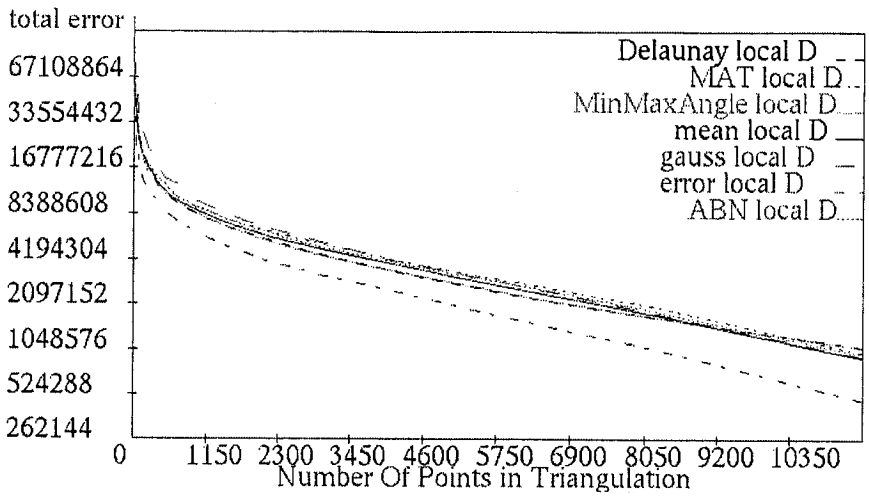

Figure A. 60: Total errors when deleting an independent set of points of a hybrid terrain each time

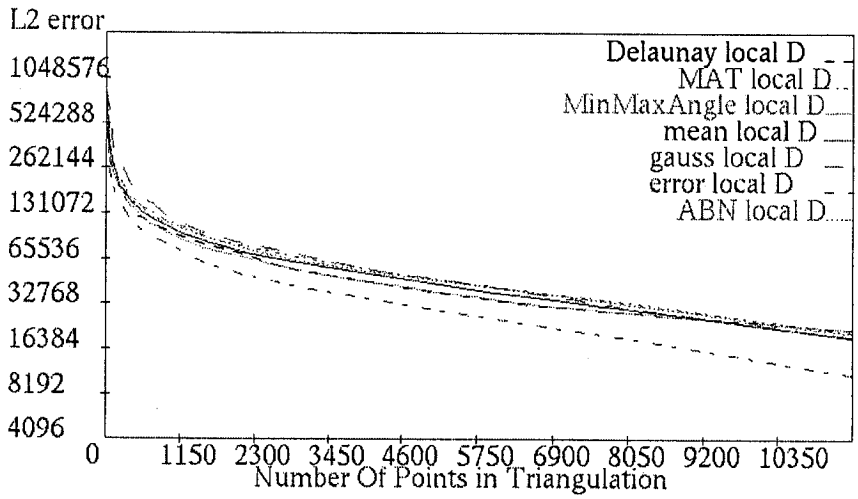

Figure A. 61: L2 errors when deleting an independent set of points of a hybrid terrain each time

Glossary

Aliasing. Jagged lines shown on curves or diagonal lines in the digital representation of an image.

Circumcircle. The circumcircle of a triangle is a circle passing through all its three vertices.

Drift. An occurrence that tiny errors resulted from deleting vertices can accumulate into big errors.

Eccentricity. The eccentricity of a triangle T is the smallest value among all distances between the center of T's circumscribed circles to points inside T.

Gaussian map. Suppose that a surface S has a unit normal vector field. Let S^2 be the unit sphere whose center is at point 0 in three-dimensional space. Then the Gaussian map g of S is the mapping $g: S \rightarrow S^2$ that maps every point p of S to the end point of the unit vector through 0 that has the same direction as the normal vector of S at p.

Spline. A mathematical function that gives an interpolation or approximation of a finite number of input points.

Swap. The operation that replaces a diagonal of a triangulated quadrilateral with the other diagonal of it.

Swap value. The difference between the quality of a triangulation $\triangle$ and the quality of a triangulation $\triangle$' after swapping an edge e.

Swappable. An edge e is *swappable* if the quadrilateral Q associated with e is convex and there are no 3 points of Q lying on the same line.

Terrain. A simplest kind of surfaces that the z coordinate of every point (x, y, z) on it can be denoted as $z = H(x, y)$.

Triangulation. A set of triangles that meet two conditions. The first condition is that any two triangles intersect with each other only at common vertex or along a common edge. The second is that the union of the triangles is a connected set.

Volume. The *volume* of the neighborhood of a vertex v is the number of cubic units required to fill the space enclosed by the triangles adjacent to v.