An Efficient Algorithm for Terrain Simplification*

Pankaj K. Agarwal [†] Pavan K. Desikan[†]

Abstract

Given a set $\overline{S}$ of n points in $\Re^3$, sampled from an unknown bivariate function $f(x, y)$ (i.e., for each point $p \in \overline{S}$, $z_p = f(x_p, y_p)$), a piecewise-linear function $g(x, y)$ is called an *ε-approximation* of $f(x, y)$ if for every $p \in \overline{S}$, $|f(x, y) - g(x, y)| \leq \varepsilon$. The problem of computing an ε-approximation with the minimum number of vertices is *NP-Hard*. We present a randomized algorithm that computes an ε-approximation of size $O(c^2 \log^2 c)$ in $O(n^{2+\delta} + c^3 \log^2 c \log \frac{n}{c})$ expected time, where c is the size of the ε-approximation with the minimum number of vertices and δ is any arbitrarily small positive number. Under some reasonable assumptions, the size of the output is close to $O(c \log c)$ and the expected running time is $O(n^{2+\delta})$. We have implemented a variant of this algorithm and include some empirical results.

1 Introduction

Modeling and construction of surfaces representing objects is an important area in many scientific disciplines like Geographic Information Systems (GIS), interactive walk-through systems, and molecular and medical data visualization. Polygonal description of objects are currently the most frequently used descriptions, because computer rendering of polygons is widely supported by commercial graphics hardware and software, and many techniques are known for constructing polygonal models of real objects.

In order to ensure that all local features and the details of the complex geometry of the surface are captured in the model, often thousands or millions of the polygons are generated, but these models are not practical, especially in real time applications, since the speed of computation and the memory requirements are proportional to the number of polygons used to model the surface, and the current hardware cannot support such a large data set. Moreover, in interactive walk-through systems and other graphics applications, there is no need to display the objects that are far from the view point in all details. Hence, there is a need to simplify the model to a prescribed resolution. We require that the simplified polygonal models that fit the given surface well and that they are composed of a small number of polygons.

Although there has been a lot of work on surface simplification in computer graphics, GIS, and scientific computing, most of the work is based on heuristic approaches [4, 11, 18, 19, 21, 25, 28, 29, 35, 39]. It is therefore not surprising that these algorithms do not guarantee any bounds on the quality of the simplification.

In this paper, we study the surface-approximation problem for xy-monotone surfaces (i.e., for terrains). The surfaces represent graphs of bivariate functions $f(x, y)$ and arise naturally in GIS and scientific visualization. The input to the algorithm is a set $\overline{S}$ of n points sampled from the graph of a bivariate function f. This form of input is sometimes called a *height field*.

Let f be a bivariate function. The graph of f represents a xy-monotone surface in $\Re^3$. Let $\overline{S}$ be a set of n points sampled from f. A piecewise-linear function g is called an *ε-approximation* of f if

$$|g(x_p, y_p) - z_p| \leq \varepsilon$$

for every point (x_p, y_p, z_p) in $\overline{S}$. For a given ε, an approximation is called *optimal* if it contains the minimum number of vertices. The goal is to find an optimal ε-approximation, g. The vertices of g are not necessarily points of $\overline{S}$. In many applications, the function f is hypothesized to fit the observed data and the function g is the computationally efficient substitute for f. The parameter ε is used to achieve

* Both authors have been supported by National Science Foundation Grant CCR-93-01259, by an Army Research Office MURI grant DAAH04-96-1-0013, by a Sloan fellowship, by an NYI award, and by matching funds from Xerox Corp. The first author is also supported by a grant from the U.S.-Israeli Binational Science Foundation.

† Department of Computer Science, Duke University Durham, NC27708-0129.

a complexity-quality tradeoff — smaller the ε higher the fidelity of approximation (more number of vertices).

Even for terrain simplification, most of the known algorithms are heuristic based. They typically start with a single triangle (or tetrahedron for general surfaces) and refine it locally until the resulting surface becomes an ε-approximation, or they start with a fine triangulation and coarsen it locally (by removing a vertex and filling the hole) until one can no longer remove a vertex [12, 13, 14, 15, 17, 16, 27, 33, 34, 36, 38]. The former method is called *refinement*, and the latter is called *decimation*. Some other approaches that extend to arbitrary surfaces have also been proposed. They include an optimization method that formulates the problem as an energy optimization problem [28, 29], and an approach based on wavelets [21, 24]. For a more comprehensive summary of known results on terrain simplification, see the survey by Heckbert and Garland [27]. However none of the above algorithms give any quantitative bound on the size of the approximation.

Provable surface-approximation algorithms were recently developed for the case of convex surfaces. Given a convex polytope P with n vertices in $\Re^3$ and a parameter ε, Mitchell and Suri gave an $O(n^3)$-time algorithm that computes a convex polytope Q with $O(c \log n)$ vertices such that $(1-\varepsilon)P \leq Q \leq (1+\varepsilon)P$; c is the size of the smallest polytope that lies between $(1-\varepsilon)P$ and $(1+\varepsilon)P$ [31]. These bounds were subsequently improved by Clarkson [7] and Brönnimann and Goodrich [5]; the latter gave an algorithm, with $O(nc(c + \log n) \log(n/c))$ running time, to compute a polytope Q with $O(c)$ vertices. It is an open question whether an approximate convex polytope with the minimum number of vertices can be computed in polynomial time.

None of the algorithms for convex surfaces extend to terrains. In fact, Agarwal and Suri have shown that for a given S, ε, and an integer k, the problem of determining whether an ε-approximation of S with at most k vertices exists, is NP-Hard. Hence, we can only hope to compute an ε-approximation whose size is near optimal. Observe that we are dealing with two approximations measures here: (i) ε, which measures the maximum difference between f and the simplified surface, and which is given as the part of the input; and (ii) the difference between the size of the optimal ε-approximation and the size of the ε-approximation computed by the algorithm (for a fixed ε). Agarwal and Suri give a polynomial-time algorithm that computes an ε-approximation of S of size $O(c \log c)$, where c is the size of an optimal ε-approximation [3]. But the running time of their algorithm is $O(n^8)$. This raises the question whether there is a simpler and faster algorithm for computing a small size ε-approximation of S.

In this paper, we present a randomized algorithm that computes an ε-approximation of size $O(c^2 \log^2 c)$ in expected time $O(n^{2+\delta} + c^3 \log^2 c \log \frac{n}{c})$, where δ is any arbitrarily small positive number; the constant of proportionality depends on δ. As we will see below, the algorithm produces an ε-approximation of size roughly $O(c \log c)$ in time $O(n^{2+\delta})$, for some cases.

We will be combining the ideas from the algorithms of Agarwal and Suri [3] and Clarkson [7] along with some new ideas to obtain the faster algorithm. As in [5, 7] we will formulate the problem as a two-dimensional *hitting set* problem and use a variant of Clarkson's randomized algorithm to compute a small hitting set. In order to expedite the running time, we do not construct the underlying set system explicitly. Instead, we run the Clarkson's algorithm on an implicit representation of the set system.

Our technique is quite general and can be applied to many other problems. For example, we can apply it to the *planar bichromatic-partition* problem, which arises in machine learning: Given a set R of m 'red' points and another set B of n 'blue' points in the plane, find a minimum number of pairwise disjoint triangles so that each blue point lies in a triangle and no red point lies in any of the triangles. Using our technique we can obtain in time $O((m+n)^{2+\delta} + c^3 \log^2 c \log \frac{n}{c})$ a solution with $O(c^2 \log^2 c)$ triangles, where c is the size of an optimal solution. Our algorithm can also be used to construct a linear decision tree of size $O(c^2 \log^2 c)$ that is consistent with respect to R and B, where c is the size of an optimal linear decision tree.

The paper is organized as follows. In Section 2, the problem is formulated as an instance of a hitting set problem on the plane. Since the set system is infinite, we describe in Section 3 how to compute a "good" finite approximation of this set system. We then present in Section 4 an algorithm for computing a hitting set for this set system. Section 5 discusses the implementation of a variant of this algorithm. We conclude by mentioning a few open problems.

2 Abstract Formulation of the Problem

Let $\overline{S}$ be the given set of sampled points in $\Re^3$. For a point $\overline{p} \in \overline{S}$, let p denote the projection of the point $\overline{p}$ onto the xy-plane; and for subset $\overline{A} \subseteq \overline{S}$, let $A = \{p \mid p \in \overline{A}\}$. A triangle $\overline{\Delta} \in \Re^3$ is called a *lifting* of a triangle Δ on the plane if Δ is the xy-projection of $\overline{\Delta}$. A triangle Δ in the xy-plane is called *valid* if it can be lifted to a triangle $\overline{\Delta}$ in $\Re^3$, i.e., Δ is the xy-projection of $\overline{\Delta}$ and, for all $p \in \Delta \cap S$, the vertical distance between $\overline{\Delta}$ and the corresponding point $\overline{p} \in \overline{S}$ is at most ε. See Figure 1. Observe that if a triangle Δ is valid, then any triangle $\Delta' \subseteq \Delta$ is also valid.

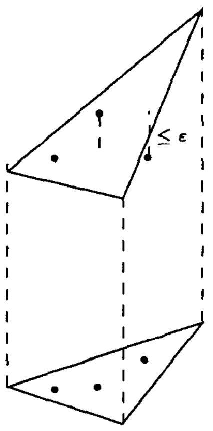

Figure 1: A triangle being lifted

A set of disjoint valid triangles in $\Re^2$ is called a *simplicial partition* of S if their union contains all the points of S. The *size* of a simplicial partition is the number of disjoint triangles in the partition. The following simple observation is by Agarwal and Suri [3].

Proposition 2.1 *Given a set $\overline{S}$ of n points in $\Re^3$ and a parameter $\varepsilon > 0$, if a simplicial partition of S of size c can be computed in time $T(n)$, then an ε-approximation of $\overline{S}$ of size $O(c)$ can also be computed in time $T(n) + O(c \log c + n)$.*

Proof: Let $\Pi = \{\Delta_1, \Delta_2, \ldots, \Delta_c\}$ be a simplicial partition of S. For each triangle Δ_i, let $S_i = S \cap \Delta_i$. Since Δ_i is a valid triangle, Δ_i can be lifted to a triangle $\overline{\Delta}_i$ so that the vertical distance between any point in $\overline{\Delta}_i$ and S_i is at most ε. Let $h_i : z_i = a_i x + b_i y + c$ be the equation of the plane containing $\overline{\Delta}_i$. For every point $\overline{p} = (x_p, y_p, z_p) \in \overline{S}$ whose xy-projection lies in Δ_i, h_i has to satisfy the following inequality

$$z_i - \varepsilon \leq a_i x_p + b_i y_p + c_i \leq z_i + \varepsilon$$

Since this is an instance of linear programming in $\Re^3$, h_i and $\overline{\Delta}_i$ can be computed in $O(n)$ time, using Megiddo's linear-programming algorithm [30]. Since Π may not cover the entire xy-plane, stitch together $\overline{\Delta}_1, \overline{\Delta}_2, \ldots, \overline{\Delta}_c$ by adding $O(c)$ new triangles as described in [3]. □

We have thus reduced the terrain-approximation problem to computing a simplicial partition, which we formulate as a hitting-set problem for a certain set system.

Let X be the set of all valid triangles in the plane, with respect to the given set of points $\overline{S}$; there are infinitely many such triangles. We define the set system $(X, \mathcal{R})$ where $\mathcal{R} = \{\{\Delta \mid \Delta \in X \text{ and } p \in \Delta\} \mid p \in S\}$, that is, for each point $p \in S$ we include all valid triangles that contain p. A *hitting set* of $(X, \mathcal{R})$ is a subset $\mathbf{\Delta} \subseteq X$, so that each point of S lies in at least one triangle of $\mathbf{\Delta}$. If the triangles of $\mathbf{\Delta}$ are disjoint, then $\mathbf{\Delta}$ is a simplicial partition of S, otherwise we compute the arrangement of $\mathbf{\Delta}$ and triangulate each face. Hence, we obtain

Lemma 2.2 *Given a set $\overline{S}$ of n points in $\Re^3$ and a parameter $\varepsilon > 0$, if the set system $(X, \mathcal{R})$ has a hitting set of size k (not necessarily disjoint), then there exists an ε-approximation of S of size $O(k^2)$.*

Remark 2.3 Notice that the size of the simplicial partition is proportional to the number of vertices in the arrangement of $\mathbf{\Delta}$. In many practical applications the number of vertices will be much smaller than k^2.

We now introduce the notion of *basis*, which is a subset of valid triangles. A set of valid triangles $\mathcal{B} \subseteq X$ is called a *k-basis* if for any triangle $\Delta \in X$, there exist $j \leq k$ triangles $\Delta_1, \Delta_2, \ldots, \Delta_j$ (not necessarily disjoint) in $\mathcal{B}$ so that for each $1 \leq i \leq j$, $\Delta_i \subseteq \Delta$ and $\Delta \cap S = \bigcup_{1 \leq i \leq j} \Delta_i \cap S$. The set $\{\Delta_1, \Delta_2, \ldots, \Delta_j\}$ is called the *generator* for Δ. For the present we assume that the basis is given to us. Later we show how to compute a k-basis for the set system $(X, \mathcal{R})$, for some constant k.

We define the set system induced by the basis $\mathcal{B}$ of $(X, \mathcal{R})$ to be $(\mathcal{B}, \mathcal{R}_{\mathcal{B}})$, where $\mathcal{R}_{\mathcal{B}} = \{r \cap \mathcal{B} \mid r \in \mathcal{R}\}$, i.e., for each point $p \in S$ there is an element in $\mathcal{R}_{\mathcal{B}}$ consisting of all the basis triangles that contain p. An immediate observation is the following lemma.

Lemma 2.4 *Let $(X, \mathcal{R})$ be the set system as defined above and $\mathcal{B}$ be a k-basis of $(X, \mathcal{R})$. If $(X, \mathcal{R})$ has a hitting set of size c, then $(\mathcal{B}, \mathcal{R}_{\mathcal{B}})$ has a hitting set of size at most kc.*

Putting Proposition 2.1 and Lemmas 2.2 and 2.4, we obtain the following result:

Theorem 2.5 *Given a set $\overline{S}$ of n points in $\Re^3$, a parameter $\varepsilon > 0$; an algorithm A that computes in time $T_A(n)$ a k-basis for the set system $(X, \mathcal{R})$ as defined above, and an algorithm B that computes in time $T_B(n, c)$ a hitting set of size $h(c)$ for the set system induced by the basis (c is the size of the minimum hitting set for the set system induced by the basis), we can compute an ε-approximation of size $O(h^2(kc))$ in time $O(T_A(n) + T_B(n, c) + h^2(kc) \log h(kc))$.*

In particular, if $h(c) = c\log c$ and $k = O(1)$, the theorem implies that we can compute an ε-approximation of size $O(c^2 \log^2 c)$, where c is the size of the optimal ε-approximation. In the next two sections we describe the algorithms A and B.

3 Computing a Small Basis

In this section we describe an algorithm for computing an $O(1)$-basis for the set system $(X, \mathcal{R})$ defined above. The algorithm works in three stages. In the first stage we compute a family of *canonical subsets* of S using a simplex range-searching data structure [9], in the second stage we compute a set of *canonical triangles* from these canonical sets, and in the third stage we discard those canonical triangles that are not valid. The canonical triangles generated in the second stage have the property that for any triangle Δ in the plane, there are at most $k = O(1)$ canonical triangles $\Delta_1, \Delta_2, \ldots, \Delta_k$, each contained in Δ, such that $\Delta \cap S = \bigcup_{1 \le i \le k} \Delta_i \cap S$. We will argue that the canonical triangles not discarded in the third stage form a k-basis for $(X, \mathcal{R})$.

The First Stage: Let $r = n^\alpha$ be a parameter, for some $\alpha > 0$. For computing the family of canonical subsets, we use the triangle range-searching data structure by Chazelle et al [9]. Their algorithm constructs in time $O(n^{2+\delta})$ a multi-level partition tree on S for any $\delta \ge 6\alpha$. Roughly speaking, it stores a family of subsets of S, called *canonical subsets*, into a tree-like data structure. For any $j \ge 0$, there are at most $O((n/r^j)^{2+\delta})$ canonical subsets of size between r^{j-1} and r^j. For any given triangle Δ in the plane, the set of all points in $\Delta \cap S$ can be reported as the union of $u = O(1/\alpha^3) = O(1)$ pairwise-disjoint canonical subsets, and these subsets can be reported in time $O(\log^3 n)$. The total time spent in the first stage is $O(n^{2+\delta})$.

The Second Stage: The canonical subsets that we obtained are just some arbitrary subsets of S. In order to obtain canonical triangles from these canonical subsets, we process the canonical subsets as follows. Consider a canonical subset C. Compute the convex hull of C. Let $p_{\min}$ (resp. $p_{\max}$) be a point of C with the minimum (resp. maximum) y-coordinate. For any two points p_1, p_2 of the convex hull of C, let Π be the pentagon formed by the lines supporting the (at most four) edges of the convex hull of C incident to the points p_1 and p_2 and by the horizontal line containing the point $p_{\min}$; see Figure 2. Π may be degenerate, in the sense that it might consist of less than 5 edges, e.g, if p_1 and p_2 are adjacent; or it may be unbounded. If Π is unbounded, we discard it. Otherwise we triangulate Π into at most three triangles. We add these triangles to the set of canonical triangles. Similarly we triangulate the pentagon formed by p_1, p_2, and $p_{\max}$, and add the resulting triangles to the set of canonical triangles.

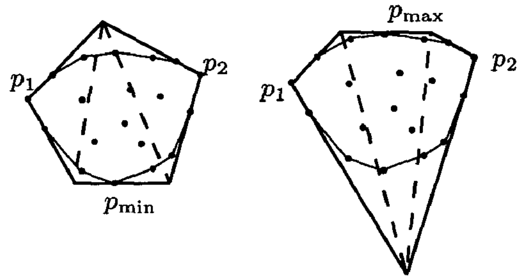

Figure 2: Constructing canonical triangles from a canonical subset

The second stage generates at most $O(n^{2+\delta})$ canonical triangles. This follows from the fact that for each canonical subset C, we create at most $6|C|^2$ canonical triangles. Since there are at most $O((n/r^i)^{2+\delta})$ canonical subsets of size between r^i and r^{i+1}, the number of canonical triangles generated is at most

$$\sum_{i=1}^{\log_r n} 6(r^{i+1})^2 O\left(\left(\frac{n}{r^i}\right)^{2+\delta}\right) = O(n^{2+\delta'})$$

for any $\delta' > 2\delta$. So we have a set of $O(n^{2+\delta})$ canonical triangles on the plane. A similar argument shows that the time spent in computing the set of canonical triangles is also $O(n^{2+\delta})$.

The Third Stage: In the final stage, we discard all the canonical triangles that are not valid. We preprocess S into the triangle range-searching data structure described above. Let S_i be a canonical subset. Determining whether there is a plane h in $\Re^3$ for which the vertical distance between h and all the points of $\overline{S_i}$ is at most ε can be formulated as an instance of linear programming in $\Re^3$, as described in the proof of Proposition 2.1. Let P_i be the set of points dual to such planes. P_i is a convex polyhedron with at most $2|S_i|$ faces, and it can be computed in time $O(|S_i| \log |S_i|)$. We preprocess P_i for point-location queries, using the Dobkin-Kirkpatrick hierarchy [20]. The total time spent in preprocessing S is once again $O(n^{2+\delta})$.

Let Δ be any triangle in the plane. We determine whether Δ is valid as follows. Let $S_\Delta = \Delta \cap S$ and $\overline{S}_\Delta = \{\overline{p} \mid p \in S_\Delta\}$. Δ is valid if and only if there is a plane h so that all the points in $\overline{S}_\Delta$ lie within a distance of ε from h. We first compute $\overline{S}_\Delta$ and then we determine whether a plane h exists. We query the range-searching data structure with Δ and report in $O(\log^3 n)$ time S_Δ as the union of $O(1)$ canonical subsets, say $S_1, S_2, \ldots S_u$. Δ can be lifted if and only if there is a plane h so that for each $1 \le i \le u$, the distance between any point in $\overline{S_i}$ and h is at most ε. Hence Δ is valid if and only if there is a point common to the polyhedra $P_1, \ldots, P_u$, i.e., $P_\Delta = \bigcap_{i=1}^{u} P_i$ is nonempty. The intersection of k convex polyhedra can be computed in $O(k^3 \log^3 n)$ using the algorithm by Eppstein [22]. Therefore, we can check in time $O(u^3 \log^3 n) = O((1/\alpha^3) \log^3 n)$ whether a given Δ can be lifted. If a triangle Δ is valid, the lifting $\overline{\Delta}$ can be obtained by intersecting the plane corresponding to a point in P_Δ with the vertical prism erected on Δ (i.e., with the prism $\{(x, z) \mid x \in \Delta, z \in \Re\}$). Hence, the total time spent in sifting out all the valid triangles is

$$O\left(n^{2+\delta} + \frac{1}{\alpha^9} n^{2+\delta} \log^3 n\right) = O(n^{2+\delta'})$$

for any $\delta' > \delta$. Let $\mathcal{B}$ be the set of all valid canonical triangles. We have the following result

Lemma 3.1 *$\mathcal{B}$ is an $O(1)$-basis for the set system $(X, \mathcal{R})$, defined in the previous section.*

Proof: Let Δ be a valid triangle in the plane; assume that Δ does not have a horizontal edge. We partition Δ into two triangles Δ^+ and Δ^- by drawing a horizontal edge from the vertex with the median y-coordinate (see Figure 3(a)). We show that for each of Δ^+ and Δ^-, we can find a generator in $\mathcal{B}$ of size at most $3u$, where u, as defined above, is the maximum number of canonical subsets in a query output. First consider Δ^+. Using the range-searching data structure described above, we can find $\nu \le u$ disjoint canonical sets $S_1, S_2, \ldots S_\nu$ such that $\bigcup_{i=1}^{\nu} S_i = \Delta^+ \cap S$. For each $1 \le i \le \nu$, we do the following. Shrink the triangle Δ^+ (by translating each edge) until each edge touches a point of S_i; see Figure 3(b). The horizontal edge of the shrunk triangle $\breve{\Delta}$ touches the point of S_i with the minimum y-coordinate, and the other two edges touch two other points of S_i, say p_1 and p_2. We can now regard $\breve{\Delta}$ as a degenerate pentagon, with two pairs of collinear edges. We rotate the edges of $\breve{\Delta}$ incident to p_1 and p_2 inward until they touch another point of S_i, thereby obtaining a pentagon Π. Since the bottom edge of Π touches the lowest point of S_i and the edges incident to p_1 and p_2 are on the convex hull of S_i, Π was constructed and triangulated by the algorithm (see Figure 3(b)). Let τ_1, τ_2, τ_3 be the triangles into which Π was triangulated. Each τ_i is contained in Δ. Since Δ is valid, τ_i is also valid, and therefore belongs to $\mathcal{B}$. Repeating this for $1 \le i \le \nu$, we obtain at most $3u$ triangles of $\mathcal{B}$, each contained in Δ^+, that cover all the points of $S_i \cap \Delta^+$. We repeat the same process for Δ^- to obtain a generator of Δ of size at most $6u = O(1/\alpha^3) = O(1)$. Hence $\mathcal{B}$ contains a generator of $O(1)$ size for every valid triangle in the plane, as required. □

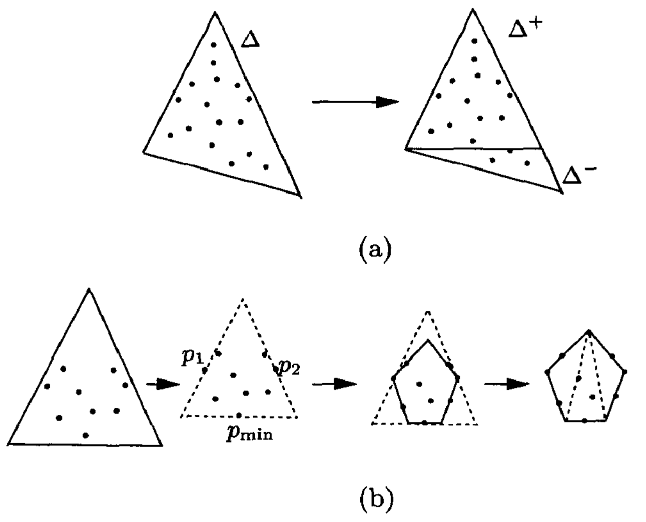

Figure 3: (a) Constructing the triangles Δ^+ and Δ^- (b) Covering each canonical subset by ≤ 3 triangles of $\mathcal{B}$

We thus obtain the following lemma

Lemma 3.2 *Given a set of n points in $\Re^3$, and $\varepsilon, \delta > 0$, we can compute in time $O(n^{2+\delta})$, an $O(1)$-basis for the set system $(X, \mathcal{R})$ as defined in Section 2.*

Remark 3.3 Although we have computed the basis, $\mathcal{B}$, we do not explicitly compute the ranges of the set system induced by the basis. Instead, for each triangle Δ in the basis, we store the three vertices (or the edges) of the lifted triangle $\overline{\Delta}$. Hence, we need only $O(n^{2+\delta})$ space to store $\mathcal{B}$. Had we stored the entire set system, we would have needed $O(n^{3+\delta})$ space.

4 Finding the Hitting Set

In this section we show how to compute a hitting set of size $O(c \log c)$ for the set system induced by the basis $\mathcal{B}$. We construct the hitting set using the algorithm by Clarkson [7] for polytope approximation, which was later generalized by Brönnimann and Goodrich [5] for computing set covers (or hitting sets) of set systems with a finite VC-dimension.

Assume that the size of the smallest hitting set, denoted by c, is known. This is no loss of generality because we can use the standard doubling trick to guess the value of c; see e.g., [7]. Let $(\mathcal{B}, \mathcal{R}_{\mathcal{B}})$ be the set system induced by the basis $\mathcal{B}$ on $(X, \mathcal{R})$, where X is the set of all valid triangles and $\mathcal{R} = \{\{\Delta \mid \Delta \in X \text{ and } p \in \Delta\} \mid p \in S\}$ as defined in Section Section 2. The outline of the algorithm for computing the hitting set of $(\mathcal{B}, \mathcal{R}_{\mathcal{B}})$ is described in Figure 4.

```
ALGORITHM FIND_HITTING_SET

(1)   Initialize the weights of all the triangles
      in $\mathcal{B}$ to be 1.
(2)   Repeat
(2.1)     Pick a random sample $R \subseteq \mathcal{B}$
          of $O(c \log c)$ triangles from $\mathcal{B}$;
          a triangle $\Delta \in \mathcal{B}$ is chosen with
          probability proportional to its weight;
(2.2)     If $S \subseteq \bigcup_{\Delta \in R} \Delta$
              output $R$ and exit;
                  /* $R$ is a hitting set */
(2.3)     Choose an arbitrary point $p$ not
          covered by $R$,
              let $\boldsymbol{\Delta}_p = \{\Delta \in \mathcal{B} \mid p \in \Delta\}$;
(2.4)     If $w(\boldsymbol{\Delta}_p) < \frac{1}{2c} w(\mathcal{B})$
              double the weights of the
              triangles in $\boldsymbol{\Delta}_p$;
      end repeat.
```

Figure 4: Algorithm for computing a hitting set.

We call an iteration *successful* if the weights of triangles in $\boldsymbol{\Delta}_p$ have been doubled (i.e., $w(\boldsymbol{\Delta}_p) < \frac{1}{2c} w(\mathcal{B})$). Clarkson has shown that the number of successful iterations is at most $4c \log(n/c)$ [7]. It can be shown that the set system $(\mathcal{B}, \mathcal{R}_{\mathcal{B}})$ has a finite VC-dimension. Therefore it follows from the ε-net theory, that the probability of an iteration being successful is at least 1/2 [8, 26]. Hence, the expected number of iterations is $O(c \log(n/c))$. Intuitively, if a point p is not covered by R, then the number of triangles containing p is small, and one of these triangles belongs to the optimal hitting set. Hence, doubling the weights of this small set of triangles improves the chances of choosing a triangle that belongs to the optimal cover.

Next, we show how to perform each of the above steps in detail and analyze the running time. The algorithm consists of the following two non-trivial operations:

(i) Check whether R covers all the points in S.

(ii) If there is a point p not covered by R, determine whether $w(\boldsymbol{\Delta}_p) < w(\mathcal{B})/2c$. If so, compute $\boldsymbol{\Delta}_p$ and double the weights of the triangles in $\boldsymbol{\Delta}_p$.

For checking whether R covers all the points, we compute the the union U of the triangles in R, and preprocess it for efficient point-location queries in time $O(c^2 \log^2 c)$, so that a point-location query can be answered in $O(\log c)$ time [20]. For every point $p \in S$, we perform a point-location query. If all the points lie in the union of R, then R is a hitting set. The total time spent in this step is $O(c^2 \log^2 c + n \log c)$.

In order to check whether a point p not covered by R has $w(\boldsymbol{\Delta}_p) < w(\mathcal{B})/2c$, we preprocess $\mathcal{B}$, in time $O(n^{2+\delta})$, into a data structure of size $O(n^{2+\delta})$, so that for any given point p, a set of $k \leq |\boldsymbol{\Delta}_p|$ triangles in $\mathcal{B}$ containing p can be returned in time $O(n^{1+\delta} + k)$. This preprocessing is done only once, before the first iteration. We query this data structure with p and report the triangles containing p one by one. As the triangles are being reported, we maintain the sum of the weights of the triangles reported so far. If the total weight exceeds $w(\mathcal{B})/2c$, we terminate the query process and declare the iteration unsuccessful. Otherwise we double the weights of all the triangles in $\boldsymbol{\Delta}_p$. Since the weight of each triangle is at least 1, the total query time is $O(n^{1+\delta} + n^{2+\delta}/c) = O(n^{2+\delta}/c)$ because $c \leq n$.

The total running time of one iteration is thus $O(n^{2+\delta}/c + c^2 \log^2 c)$. Summing over all the iterations, the expected running time of the algorithm for a fixed value of c is $O((\frac{n^{2+\delta}}{c} + c^2 \log^2 c) c \log \frac{n}{c})$. Hence, we have obtained the following result.

Theorem 4.1 *Given a set $\overline{S}$ of n points in $\Re^3$ sampled from a bivariate function and a parameter $\varepsilon > 0$, we can compute an ε-approximation of S of size $O(c^2 \log^2 c)$ in expected time $O(n^{2+\delta} + c^3 \log^2 c \log \frac{n}{c})$, for any $\delta > 0$, where c is the size of the optimal ε-approximation of $\overline{S}$.*

5 Implementation

We have a partial implementation of an algorithm for terrain approximation based on the algorithm described above. There are two steps in the above algorithm that are difficult to implement: (i) triangle range-searching data structures for computing the canonical subset and for determining the triangles not covered by a point, and (ii) computing the union of R and processing it for point-location queries.

Since no fast-and-practical data structure is known for triangle range searching that returns the output as a family of canonical subsets, we use quad-trees to generate canonical subsets and the basis. We construct a quad-tree on S, the xy-projections of the

Terrain	ε	Our algorithm			Franklin	Silva-a	Silva-b
		SP-A	SP-B	ε-approx			
Buffalo	2.5	964	1399	2125	1994	1641	2279
Denver	2.5	1163	2049	2527	2688	2137	2849
Eagle	1.5	596	1332	1274	1564	1214	1578
Grand Canyon	15	1167	1928	2725	2822	2073	3115
Jackson	0.5	453	1150	1047	1297	859	1127
Moab	15	926	1827	1879	2561	1836	2430
Seattle	5	973	1948	2259	2671	2011	2763

Table 1: Number of triangles by various algorithms

input points $\overline{S}$. The points stored at each node of the quad-tree are used as canonical subsets, and for each canonical subset we generate only $O(1)$ canonical triangles in the plane. Our algorithm generates canonical triangles with the following two properties: (i) they do not overlap, and (ii) vertices are points of S. As a result the xy-projections of the vertices of the lifted triangles are points of S. For checking the validity of a canonical triangle, we use the linear-programming algorithm of Seidel [37]. We also consider the case in which the vertices of the lifted triangle are a subset of the input points. We use the greedy algorithm, instead of Clarkson's randomized algorithm, for computing a hitting set. We finally 'stitch' together the triangles by computing the *constrained Delaunay triangulation* of the edges of the simplicial partition, and thus obtain a valid triangulation of the plane.

Our implementation has about 2000 lines of C++ code. All the input data sets consisted of 120 $\times$ 120 elevation arrays, as in [38]. We ran our experiments on a Digital AlphaStation 500/266 workstation, and each of them took between 30 and 45 seconds. In the Table 1, we summarize our results and compare them with the results of Franklin [23] and Silva, et al. [38] on the same data sets. The column SP-A gives the number of triangles in the simplicial partition when we use the linear-programming algorithm and allow the vertices of the ε-approximation to be different from the input points, and SP-B gives the size of the simplicial partition when the vertices of the lifted triangles are restricted to be a subset of the input points. The column ε-approx gives the number of triangles after the constrained Delaunay triangulation of the triangles in SP-A.

Notice that the number of triangles in the simplicial partition is rather small if we do not restrict the vertices of the ε-approximation to be a subset of the input points. In this case, the number of triangles in the simplicial partition is the roughly half the number of triangles produced by the other algorithms. It is the *stitching* process that blows up the size by a factor of roughly 2. The stitching process adds triangles in those areas that do not contain any input point, and also ensures that we have a valid triangulation of the plane. We are currently working on better range-searching data structures and on a more clever approach for the stitching process.

6 Conclusion

We have presented an efficient algorithm for the terrain-simplification problem that computes an ε-approximate terrain of size $O(c^2 \log^2 c)$, where c is the size of the optimal ε-approximate terrain, and a preliminary implementation shows that our approach is promising. The performance of our current implementation is slightly inferior to that of the algorithms by Silva et al. [38]. However, a better range-searching data structure and a more clever stitching process should improve the performance significantly.

We conclude by mentioning two problems

(i) Can our approach be extended to a more general surface? That is, develop an efficient algorithm for simplifying arbitrary surfaces.

(ii) Design near-linear time algorithm for terrain simplification at the expense of the quality of the solution.

7 Acknowledgments

We wish to thank K. Clarkson for providing the code for computing the convex hull of a point set in two dimensions, to J. R. Shewchuk for making available the code for constrained Delaunay triangulation on the net, and to C. Silva for providing the terrain data.

References

[1] P. K. Agarwal, Geometric partitioning and its applications, *DIMACS Series in Discrete Mathematics and Theoretical Computer Science* (J. Goodman et al, eds), vol. 6, 1991, 1–37.

[2] P. K. Agarwal and M. Sharir, Applications of a new space-partitioning technique, *Discrete and Comput. Geom.*, 9 (1993), 11–38.

[3] P. K. Agarwal and S. Suri, Surface approximations and geometric partitions, *Proc. 5th Symp. on Discrete Algorithms* 1994, 24–33. (Also to appear in SIAM J. Comput.)

[4] J. Bloomenthal, Polygonalization of implicit surface, *Computer Aided Design*, 5 (1988), 341–355.

[5] H. Brönnimann and M. T. Goodrich, Almost optimal set covers in finite VC-dimension, *Discrete and Comput. Geom.,* 15 (1995), 463–479.

[6] K. L. Clarkson, New applications of random sampling in computational geometry, *Discrete and Comput. Geom.*, 2 (1987), 195–222.

[7] K. L. Clarkson, Algorithms for polytope covering and approximation, *Proc. 3rd Workshop on Algorithms and Data Structures*, Lecture Notes in Computer Science, vol. 709, Springer Verlag, Berlin, 1993, 246–252.

[8] K. L. Clarkson and P. W. Shor, Applications of random sampling in computational geometry, *Discrete and Comput. Geom.*, 4 (1989), 387–421.

[9] B. Chazelle, M. Sharir, and E. Welzl, Quasi-optimal upper bounds for simplex range searching and new zone theorems *Algorithmica*, 8 (1992), 407–429.

[10] L. P. Chew, Constrained Delaunay triangulations, *Algorithmica*, 4 (1989), 97–108.

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

[12] M. de Berg and K. Dobrindt, On levels of detail in terrain, *Proc. 11th Annual ACM Symp. on Comput. Geom.* 1995, C26–C27.

[13] L. De Floriani, A pyramidal data structure for triangle based surface description, *IEEE Computer Graphics and Applications*, 9 (1989), 67–78.

[14] L. De Floriani, B. Falcidieno and C. Pienovi, A Delaunay-based method for surface approximation, *Proc. Eurographics'83*, Elsevier Science, 1983, 333–350.

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

[16] L. De Floriani, B. Falcidieno, and C. Pienovi, Delaunay based representation of surfaces defined over arbitrarily shaped domains, *Computer Vision, Graphics and Image Processing*, 32 (1985) 127–140.

[17] L. De Floriani and E. Puppo, A hierarchical triangle based model for terrain description, *Theories and Methods of Spatio-Temporal Reasoning in Geographic Space* (A. U. Frank et al. eds.), Springer Verlag, 1992, 236-251.

[18] M. DeHaemer and M. Zyda, Simplification of objects rendered by polygonal approximations, *Computers and Graphics* 15 (1992), 175–184.

[19] H. Delingette, Simplex meshes: A general representation for 3D shape reconstruction, *Proc. Int. Conf. on Computer Vision and Pattern Recognition*, 1994, 856–859.

[20] D. Dobkin and D. Kirkpatrick, Fast detection of polyhedral intersection, *Proc. 9th Int. Colloq. Automata, Languages and Programming*, Lecture Notes in Computer Science 140, Springer-Verlag, 1982, 154–165.

[21] M. Eck, T. DeRose, T. Duchamp, H. Hoppe, M. Lounsbery, and W. Stuetzle, Multiresolution analysis of arbitrary meshes, *Proc. SIGGRAPH'95,* 1995, 173–182.

[22] D. Eppstein, Dynamic three dimensional linear programming, *ORSA J. Computing*, 4 (1992), 360–368.

[23] W. R. Franklin, Triangulated irregular network to approximate digital terrain, Tech. Report, Electrical, Computer and Systems Enginnering Dept., Rensselaer Polytechnic Institute, Troy, NY, 1994.

[24] M. H. Gross, O. G. Staadt, and R. Gatti, Efficient triangular surface approximations using wavelets and quadtree data structures, *IEEE Trans. on Visualization and Computer Graphics* 2 (1996), 130–143.

[25] A. Gueziec and R. Hummel, Exploiting triangulated surface extraction using tetrahedral decomposition. *IEEE Trans. on Visualization and Computer Graphics* 1 (1995), 328–342.

[26] D. Haussler and E. Welzl, ε-nets and simplex range queries, *Discrete Comput. Geom.* 2 (1987), 127–151.

[27] P. S. Heckbert and M. Garland, Fast polygonal approximations of terrains and height fields, *Tech. Report* CMU–CS–95–181, Carnegie Mellon University, Pittsburgh, 1995.

[28] H. Hoppe, T. DeRose, T. Duchamp, J. McDonald, and W. Stuetzle, Mesh optimization, *Proc. SIGGRAPH 93*, 1993, 19–26.

[29] M. Margaliot and C. Gotsman, Piecewise linear surface approximation from noisy scattered samples, *Proc. Visualization'94*, 1994, 61–68.

[30] N. Megiddo, Linear programming in linear time when the dimension is fixed, *J. ACM*, 31 (1984), 114–127.

[31] J. Mitchell and S. Suri, Separation and approximation of polyhedral objects, *Computational Geometry — Theory and Applications*, 5 (1995), 95–114.

[32] F. Preparata and M. I. Shamos, *Computational Geometry: An Introduction*, Springer-Verlag, 1985.

[33] S. Rippa, Adaptive approximation by piecewise linear polynomials on triangulations of subsets of scattered data, *SIAM J. on Scientific and Stat. Computing*, 13 (1992), 1123–1141.

[34] S. Rippa, Long and thin triangles can be good for linear interpolation, *SIAM J. Numer. Anal*, 29 (1992), 257–270.

[35] J. Rossingac and P. Borrel, Multiresolution 3D approximations for rendering complex scenes, *Modeling in Computer Graphics: Methods and Applications* (B. Falcidieno and T. Kunii, eds), Springer Verlag, 1993, 455–465.

[36] L. Scarlatos, T. Pavlidis, Hierarchical triangulation using cartographic coherence, *CVGIP: Graphical models and image processing*, 54 (1992), 147–161.

[37] R. Seidel Linear programming and convex hulls made easy, *6th ACM Symp. on Comput. Geom.*, 1990, 211–215.

[38] C. T. Silva, J, S. B. Mitchell, and A. E. Kaufman, Automatic generation of triangular irregular networks using greedy cuts, *Proc. Visualization '95*, 1995, 201–208.

[39] G. Turk, Retiling of polygonal surfaces, *Computer Graphics* 26 (1992), 55–64.