Finding Planar Regions in a Terrain

Stefan Funke [1], Theocharis Malamatos, Rahul Ray

Max-Planck-Institut für Informatik, Stuhlsatzenhausweg 85, 66123 Saarbrücken, Germany

Abstract

We consider the problem of computing large connected regions in a triangulated terrain of size n for which the normals of the triangles deviate by at most some small fixed angle. In previous work an exact near-quadratic algorithm was presented, but only a heuristic implementation with no guarantee was practicable. We present a new approximation algorithm for the problem which runs in $O(n/\epsilon^2)$ time and—apart from giving a guarantee on the quality of the produced solution—has been implemented and shows good performance on real data sets representing fracture surfaces with around half a million triangles.

1. Introduction

A terrain is a surface in $\mathbb{R}^3$ defined by a function $f : \mathbb{R} \times \mathbb{R} \to \mathbb{R}$. If f is piecewise linear and the surface consists of a collection of triangles, the terrain is called a *triangulated irregular network (TIN)*. Given a TIN $\mathcal{T}$, the goal is to find large, nearly planar regions in $\mathcal{T}$. We define 'near planarity' as follows: for a real parameter $\delta > 0$, we say that a subset of triangles T in $\mathcal{T}$ is *δ-planar* if (i) the adjacency graph of the triangles in T is connected, and (ii) there is a vector $\overrightarrow{r}$ such that for each $t \in T$, $\angle(n_t, r) \leq \delta$, where n_t denotes the normal of triangle t. We call $\overrightarrow{r}$ the *reference normal.* (Throughout the paper $\angle(v, u)$ denotes the angle between two vectors $\overrightarrow{v}$ and $\overrightarrow{u}$.)

Researchers in the material sciences examine surface topographies of materials as they provide useful information about the generation process and the internal structure of the material. Surfaces generated by fracture, wear, corrosion and machining are of interest. Among many other criteria, they want to examine feature-related parameters like facets in brittle fracture surfaces. This application motivated our work.

Email addresses: `funke@mpi-sb.mpg.de` (Stefan Funke), `tmalamat@mpi-sb.mpg.de` (Theocharis Malamatos), `rahul@mpi-sb.mpg.de` (Rahul Ray).

[1] Part of this research was conducted while the author was visiting the University of Illinois at Urbana-Champaign, USA.

Related work. Assume that each triangle in $\mathcal{T}$ is assigned a weight and that any set of triangles has weight equal to the sum of the weights of its triangles. Smid, Ray, Wendt and Lange [2] considered the problem of finding a δ-planar subset of $\mathcal{T}$ of maximum weight. They presented an $O(n^2 \log n (\log\log n)^3)$ algorithm, where n is the number of triangles in $\mathcal{T}$. Since this is impractical they proposed a heuristic which computes nearly planar regions quite quickly but with no guarantee. Also they suggested that finding a δ-planar set with weight at least a constant fraction of the optimum may be equally difficult as solving the problem exactly, see [2].

Our results. In this paper we adopt the following notion of approximation. Given a real parameter $\epsilon > 0$, we say that a subset of triangles T of $\mathcal{T}$ is *ϵ-approximate* δ-planar if it is $\delta(1+\epsilon)$-planar and has weight at least as large as an optimal δ-planar set. Under this notion, we present a new approximation algorithm that runs in $O(n/\epsilon^2)$ time which is independent of δ. For n sufficiently large, the algorithm uses optimal $O(n)$ space. Recently Har-Peled and Mazumdar [1] have used a similar notion of approximation for the problem of computing the smallest k-enclosing disk.

We have implemented and empirically evaluated the algorithm on real test data from the application domain consisting of fracture surface terrains with more than 500,000 triangles. It provides very good quality results in reasonable time.

2. Preliminaries

Let $\mathcal{T}$ be a TIN. We associate with $\mathcal{T}$ an undirected weighted graph $G_{\mathcal{T}}(V, E)$ as follows. Each triangle t in $\mathcal{T}$ has an associated weight $w(t)$ and corresponds to a vertex v_t in V. An edge connects two vertices of V if and only if the corresponding triangles in $\mathcal{T}$ are adjacent. Note that $G_{\mathcal{T}}$ is the dual graph of $\mathcal{T}$, is planar and has degree three. Each vertex $v_t \in V$ is assigned the weight of its associated triangle $w(t)$ which can be, for example, equal to the area of the triangle t or one (when we want to maximize the area of the detected region or the number of triangles, respectively). The weight $w(V')$ of any subset V' of V is defined as the sum of the weights of the vertices in V'.

For a point $u \in \mathbb{R}^3$ we denote by $\overrightarrow{u}$ the vector $\overrightarrow{Ou}$, where O is the origin. Let $\mathbb{S}^2$ denote the unit sphere, i.e., the boundary of the three-dimensional ball of radius one centered at the origin. For each triangle $t \in \mathcal{T}$, we can associate a point $p_t \in \mathbb{S}^2$ which represents the normalized normal of triangle t. Specifically, $\overrightarrow{p_t} = \overrightarrow{n_t}/|\overrightarrow{n_t}|$. Our goal is to approximate the space $\mathbb{S}^2$ of all possible normals by a finite set of points $\mathbb{V} \subset \mathbb{S}^2$ such that for any $p \in \mathbb{S}^2$, there is a point $v \in \mathbb{V}$ nearby.

Definition 1 *A set of point $\mathbb{V} \subset \mathbb{S}^2$ is called a* $\delta\epsilon$-discretization *of $\mathbb{S}^2$ if $\forall p \in \mathbb{S}^2 : \exists v \in \mathbb{V}$ with $\angle(v, p) \leq \delta \cdot \epsilon$.*

Lemma 2 *There exists a $\delta\epsilon$-discretization of $\mathbb{S}^2$ of size $O(1/(\delta\epsilon)^2)$ which can be computed in the same time.*

The following construction yields a $\delta\epsilon$-discretization of $\mathbb{S}^2$ as needed in Lemma 2. (We omit its proof in this abstract.) Consider a cube L with sidelength two centered at the origin. Note that $\mathbb{S}^2 \subset L$. Place a 2-dimensional grid of size $k \times k$ with $k = \lceil \sqrt{2}/(\delta\epsilon) \rceil$ over each of the six faces of L. This generates k^2 equally sized square grid cells on each face of L, where each cell has sidelength at most $(\delta\epsilon\sqrt{2})$, and $6k^2 + 2$ grid points overall. See Figure 1. Let A be the set consisting of these grid points. Our $\delta\epsilon$-discretization $\mathbb{V}$ of $\mathbb{S}^2$ is defined as

$$\mathbb{V} = \left\{ \frac{\overrightarrow{z}}{|\overrightarrow{z}|} : z \in A \right\},$$

that is, for each gridpoint z we shoot a ray from the origin through z and we include the point where the ray leaves $\mathbb{S}^2$ into set $\mathbb{V}$.

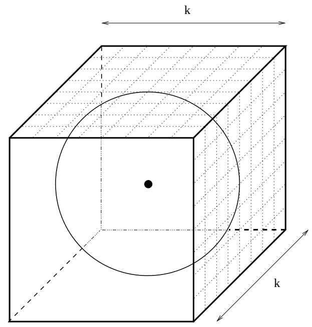

Fig. 1. Cube with sidelength two containing $\mathbb{S}^2$ and with a $k \times k$ grid on each of its faces.

3. Finding Large Planar Regions

3.1. *The Basic Algorithm*

We first describe a simple algorithm for the problem that computes an ϵ-approximate solution in $O(n/(\delta\epsilon)^2)$ time. The algorithm proceeds as follows:

1. Compute a $\delta\epsilon$-discretization $\mathbb{V}$ of $\mathbb{S}^2$.
2. For each $p \in \mathbb{V}$,
   (a) Compute the set V_p of vertices v_t with $\angle(p, n_t) \leq (1 + \epsilon) \cdot \delta$.
   (b) Consider the subgraph of $G_{\mathcal{T}}$ induced by set V_p and determine its heaviest connected component C_p.
3. Report the set of triangles T corresponding to the heaviest component C_p found in Step 2 and the associated reference normal $\overrightarrow{p}$.

We now discuss the running time and correctness of the algorithm. Computing the $\delta\epsilon$-discretization takes $O(1/(\delta\epsilon)^2)$ by Lemma 2. For each element $p \in \mathbb{V}$ we determine the subgraph induced by V_p and compute its connected components, which can be done in $O(n)$ time. So the total running time of Step 2 is $O(n/(\delta\epsilon)^2)$ which also dominates the overall running time.

For correctness, observe first that clearly the computed set T is $\delta(1 + \epsilon)$-planar. It remains to show that the weight of T is at least that of an optimal δ-planar set T^*. For set T^* there exists a vector $\overrightarrow{r^*}$ such that for all triangles $t \in T^*$, $\angle(r^*, n_t) \leq \delta$. Let p be a point in $\mathbb{V}$ for which $\angle(p, r^*) = \min_{u \in \mathbb{V}} \angle(u, r_{\text{opt}})$. By the definition of $\mathbb{V}$, the angle $\angle(p, r^*)$ must be at most $\delta\epsilon$. Then, for any triangle $t \in T^*$ the angle between $\overrightarrow{n_t}$ and $\overrightarrow{p}$ is at most $\delta + (\delta\epsilon) = \delta(1+\epsilon)$. Therefore for all $t \in T^*$,

$v_t \in V_p$ and hence, the algorithm will find a connected component with at least the same weight.

Lemma 3 *Given a triangulated irregular network $\mathcal{T}$, and two real parameters $\delta > 0$ and $\epsilon > 0$ we can compute in $O(n/(\delta\epsilon)^2)$ time an ϵ-approximate δ-planar set of triangles in $\mathcal{T}$.*

The running time of the basic algorithm is optimal in terms of n. But one may ask whether the dependence on ϵ or δ can be improved. In the following we will refine the algorithm to achieve $O(n/\epsilon^2)$ running time, which is independent of δ.

3.2. *The Refined Algorithm*

The improvement in the running time of the algorithm comes from two factors. First we determine a set of reference normals $\mathbb{V}'$ of size at most $O(n/\epsilon^2)$ that contains all relevant reference normals, avoiding the inspection of $\Omega(1/(\delta\epsilon)^2)$ potential reference normals. Second by a bucketing scheme, we reduce significantly the number of times each triangle is considered. The refined algorithm proceeds as follows:

1. For each triangle $t \in \mathcal{T}$ with normal n_t, let p_t be a point in $\mathbb{V}$ for which $\angle(p_t, n_t) = \min_{u\in\mathbb{V}} \angle(u, n_t)$; store t in the bucket associated with p_t.

2. Determine a set $\mathbb{V}' \subset \mathbb{V}$ of potential reference normals as $\mathbb{V}' = \{p \in \mathbb{V} : \exists p_t$ with non-empty bucket and $\angle(p, p_t) \leq (1 + 2\epsilon) \cdot \delta\}$.

3. For each $v \in \mathbb{V}'$,
   - (a) Collect the set of triangles N_v which are contained in buckets of reference normals $r' \in \mathbb{V}'$ with $\angle(r', v) \leq (1 + 2\epsilon) \cdot \delta$.
   - (b) Prune N_v keeping only those triangles t with $\angle(n_t, v) \leq (1 + \epsilon) \cdot \delta$. Let N_v' be the pruned set.
   - (c) Consider the subgraph of $G_{\mathcal{T}}$ induced by the vertices corresponding to triangles in N_v' and determine its heaviest connected component C_v.

4. Output the heaviest connected component C_v found in Step 3.

Before analysing the algorithm, we state a small lemma which informally says that in the $\delta\epsilon$-discretization, for any point p there is a small number of other points nearby.

Lemma 4 *Let p be a point in the $\delta\epsilon$-discretization $\mathbb{V}$. Then the number of points $p' \in \mathbb{V}$ with $\angle(p, p') < (1 + 2\epsilon) \cdot \delta$ is at most $O(1/\epsilon^2)$.*

The proof of this lemma follows easily from our construction of $\mathbb{V}$ and it is omitted. We note also that given a triangle normal n_t we can compute the point p_t with $\angle(n_t, p_t) = \min_{u\in\mathbb{V}} \angle(n_t, u)$ in constant time by first determining which face of the cube L is hit by the ray $\overrightarrow{n_t}$ and then locating the position of the intersection point within the grid on that face. We now state the main result of this section:

Theorem 5 *Given a triangulated irregular network $\mathcal{T}$, and two real parameters $\delta > 0$ and $\epsilon > 0$ we can compute in $O(n/\epsilon^2)$ time an ϵ-approximate δ-planar set of triangles in $\mathcal{T}$.*

PROOF. We claim that the output of the refined algorithm give us such a set. To show correctness, it suffices to prove that the algorithm computes the same solution as the basic algorithm. We leave this proof to the reader. We focus now on the running time. Step 1 of the algorithm takes $O(n)$ since for each t we can in constant time determine p_t and access the associated bucket using a hashing scheme. Step 2, where we form the set $\mathbb{V}'$, takes $O(n/\epsilon^2)$ since there are at most n non-empty buckets. For each of the non-empty buckets, we explore $O(1/\epsilon^2)$ points in the neighborhood by Lemma 4. Finally, for the overall running time of Step 3, observe that again by Lemma 4, each triangle can be collected by at most $O(1/\epsilon^2)$ reference normals and that the running time of one iteration of Step 3 is $O(|N_v|)$. Therefore Step 3 takes $O(n/\epsilon^2)$ time in total. $\square$

4. Experimental Results

We have implemented the refined algorithm of Section 3.2 in `C++` using the LEDA library [3]. Experiments were carried out on a 1.8 GHz Pentium 4 machine with 256 MB of RAM. We used several data sets representing fracture surfaces from the application domain in material sciences. Input data were given as 512×512 raster images with the intensity of each pixel corresponding to its height value. To obtain the TIN, we triangulated the point set by creating triangles $(i, j), (i+1, j), (i+1, j+1)$ and $(i, j), (i, j + 1), (i + 1, j + 1)$. See Figure 2.

To speed-up our program (while preserving the same guarantee), we have come up with the following three heuristics:

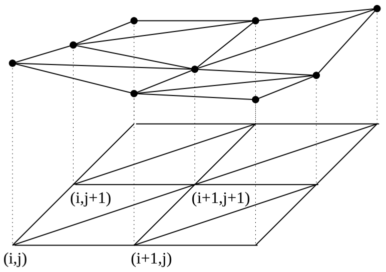

Fig. 2. Triangulation scheme for the array of height values.

(i) With each potential reference normal we associate the weight of all triangles in buckets at distance at most $(1 + 2\epsilon) \cdot \delta$ and examine the reference normals in decreasing order of weight. If the current best solution exceeds the associated weight of the next reference normal, we can stop examining further.

(ii) If the normal of a triangle t forms an angle larger than $2(1+\epsilon)\delta$ with the normal of each of the three adjacent triangles, we can prune t out (at Step 1). It can only form a singleton solution.

(iii) In Step 3(b) where we compute the set of relevant triangles N'_v we only check triangles in buckets at angle distance more than δ, thus saving some expensive floating-point operations.

In our experiments, the three heuristics combined reduce the running time by nearly a factor of two. For our test instances consisting of roughly $500,000$ triangles and a choice of $\delta = 0.2$ (about 11.5 degrees) and $\epsilon = 0.2$, the running time varied between 70 and 110 sec. The best solution was in all cases found within the first 20 seconds due to the prioritization scheme. Thus most of the running time was spent on checking that no better solution exists. (A heuristic could just report the solution computed, for example, after 30 seconds.)

We also ran experiments to determine the variations of the running time as a function of n and ϵ. The dependence graph on ϵ for a fixed data set is shown in Figure 3. The upper curve denotes the total running time and the lower curve denotes the time when the reported solution was detected. In Figure 4 we examine the dependence on n (number of triangles). Note that again the reported solution was found after only few seconds.

For the heuristic implementation in [2], running times reported are in the range of 30–40 seconds but as it is mentioned the solution computed may be far from optimal.

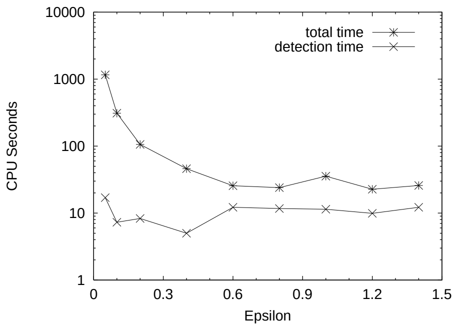

Fig. 3. CPU time versus ϵ for $n = 522$K, $\delta = 0.2$.

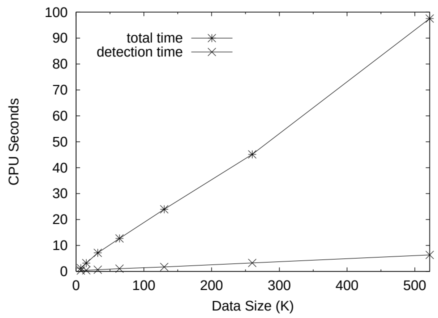

Fig. 4. CPU time versus n for $\delta = 0.2$, $\epsilon = 0.2$.

References

[1] S. Har-Peled and S. Mazumdar. Fast algorithms for computing the smallest k-enclosing disc. In *Proc. 11th Annu. European Sympos. Algorithms*, Lecture Notes Comput. Sci. Springer-Verlag, 2003.

[2] K. Lange, R. Ray, M. Smid, and U. Wendt. Computing large planar regions in terrains. In *Proceedings of the 8th International Workshop on Combinatorial Image Analysis (IWCIA)*, Electronics Notes in Computer Science, Volume 46, 2002.

[3] K. Mehlhorn and S. Näher. *LEDA: A Platform for Combinatorial and Geometric Computing*. Cambridge University Press, Cambridge, U.K., 1999.

[4] U. Wendt, K. Lange, M. Smid, R. Ray, and K. Tönnies. Surface topography quantification by integral and feature-related parameters. *Materialwissenschaft und Werkstofftechnik*, 33(10):621–627, October 2002.