Matching Terrains under a Linear Transformation[*]

Pankaj K. Agarwal[†] Boris Aronov[‡] Marc van Kreveld[§] Maarten Löffler[§] Rodrigo I. Silveira[§]

Abstract

We study the problem of matching two polyhedral terrains, where one can be changed vertically by a linear transformation of the third coordinate (scaling and translation). We give an algorithm that minimizes the maximum distance over all linear transformations in $O(n^{4/3}\text{polylog } n)$ expected time. We also study matching two 1-dimensional terrains, and give a $(1+\varepsilon)$-approximation algorithm for minimizing the area in between that runs in $O(n/\sqrt{\varepsilon})$ time, for any fixed $\varepsilon > 0$.

1 Introduction

In GIS, many data sets are bivariate scalar functions, for example annual precipitation, nitrate concentration, depth to groundwater, and elevation above sea level. Such functions are either stored by a regular square grid of values, or by a triangulation over some domain. In the case of a triangulation, one usually assumes linear interpolation over the values stored at the vertices. This gives a polyhedral terrain, that is, an xy-monotone polyhedral surface in $\mathbb{R}^3$.

Often, data sets for the same region but with a different theme are not independent: if there is more precipitation, the soil humidity will be higher, and if hill slopes are steeper, there will be more soil loss per year. Such relations can be analyzed by matching the representations of two data sets.

In this paper we assume that two bivariate scalar functions over the same domain are given as (polyhedral) terrains. We want to determine if there is a linear transformation applied to one to make it match well with the other. The linear transformation applies only to the scalar values, not to the world coordinates. Hence, a linear transformation has one scaling component s and one translation component t.

[*]This research has been partially funded by the Netherlands Organisation for Scientific Research (NWO) under FOCUS/BRICKS grant number 642.065.503, and also through the project GOGO. Work by B.A. has been partially supported by a grant from the U.S.-Israel Binational Science Foundation, by NSA MSP Grant H98230-06-1-0016, and NSF Grant CCF-08-30691.

[†]Department of Computer Science, Duke University, pankaj@cs.duke.edu

[‡]Department of Computer Science and Engineering, Polytechnic Institute of NYU, http://cis.poly.edu/~aronov

[§]Department of Information and Computing Sciences, Utrecht University, {marc,loffler,rodrigo}@cs.uu.nl

Given two terrains $f, g : D \to \mathbb{R}$, where D is a bounded, triangulated subset of $\mathbb{R}^2$, our goal is to find a linear relation between the two terrains. That is, we want to find values $s, t \in \mathbb{R}$ such that $sf + t$ matches g as well as possible. To compare the two terrains, we use two different measures:

$$\mu(s,t) = \max_{x,y} |sf(x,y) + t - g(x,y)|, \text{ and} \qquad (1)$$

$$\mu(s,t) = \int_{x,y} |sf(x,y) + t - g(x,y)| dxdy. \qquad (2)$$

With each measure, we want to minimize its value over s and t, namely

$$\min_{s,t} \mu(s,t).$$

In what follows we denote by n the total number of vertices in f and g, and sometimes use the O^* notation, which ignores polylogarithmic factors.

We say that f and g are *aligned* if their projections on the (x, y)-plane are the same, that is, they have exactly the same triangulation.

2 Minimizing the max distance of two 2-d terrains

In this section the goal is to find s, t to minimize $\mu(s,t)$ according to Eq. (1), namely, to minimize the maximum vertical distance between g and $sf + t$.

We begin by observing that, when the terrains are aligned, the maximum of μ is always obtained at a vertex of the terrain. When the terrains are not aligned, the maximum can occur either at a vertex of one of the terrains or at the intersection of two edges in the xy-projection. This implies that in both cases the problem is inherently discrete.

Let v be a vertex of one of the terrains or the intersection point of two terrain edges (always in the xy-projection). Let $d_v(s,t) = |sf(v) + t - g(v)|$, where $f(v)$ (resp. $g(v)$) gives the height of v in f (resp. g). Thus $d_v(s,t)$ defines a surface in the (s, t, μ)-space (here μ stands for the possible values for the objective function). This surface consists of two halfplanes with a V-shape (due to the absolute value). The collection of the halfplanes from all vertices gives an arrangement of planes where $\mu(s,t)$ can be viewed as the upper envelope of the planes. Since we are interested in the upper envelope only, we can consider the halfplanes to be full planes.

It follows that the lowest value of μ can be computed using linear programming (LP) in three dimensions. For the case of aligned terrains, each terrain vertex generates two planes, resulting in a system with $O(n)$ inequalities. Standard methods for linear programming in fixed dimension allow to solve the problem in $O(n)$ time [4].

When the terrains are not aligned, the same approach can be used if all the intersection points between terrain edges are considered vertices. However, their number may become $\Omega(n^2)$, hence solving the linear program would take quadratic time. Note, however, that we have $O(n)$ running time for non-aligned *fat* terrains, since the number of intersections is linear for two such terrains [6]. In what follows we show how to reduce the potentially quadratic running time for arbitrary terrains by considering only a linear number of intersection points. From now on we will refer to both the terrain vertices and the intersection points in the xy-projection as *vertices*.

We will avoid computing the quadratic set of constraints explicitly by applying a random sampling approach. Our method is similar to the LP algorithm in [5], and is sketched in Algorithm 1. Recall that each vertex defines two constraints.

Algorithm 1 MinMaxDistance

```
 1: $R \leftarrow$ constraints induced by random subset of $3n$
    vertices of the projection
 2: **repeat**
 3:     Solve LP problem constrained to $R$
 4:     $V \leftarrow$ set of violated constraints
 5:     **if** $|V| > 2n$ **then**
 6:         $R \leftarrow$ constraints induced by a new random
            subset of $3n$ vertices of the projection
 7:     **else**
 8:         $R \leftarrow R \cup V$
 9:     **end if**
10: **until** $V = \emptyset$
11: **return** Solution from step 3
```

Several parts of the algorithm are explained in more detail below.

First a random subset of the vertices must be chosen. This is done in any way that guarantees that all subsets with r elements have the same probability of being chosen. This can be done using the two-level hereditary segment tree mentioned below.

Then the linear program that has the constraints induced by the vertices in R is solved. This is a 3-dimensional LP with $O(n)$ constraints that can be solved in $O(n)$ time using standard algorithms [4]. The solution gives us concrete values of s_0, t_0 and μ_0. That is, after that we have two concrete terrains, g and $f_0 = s_0 f + t_0$.

Computing the violated constraints. Since the solution (s_0, t_0) from line 3 only considers the $O(n)$ constraints in R, some of the $O(n^2)$ original constraints may be violated by this solution. A violated constraint implies that there is some point (original vertex or intersection point) in g that is at a vertical distance larger than μ_0 from f_0. To find the violated constraints we need to find the vertices at distance larger than μ_0 from the other terrain. The distances between vertices of f_0 and triangles in g, or the other way around, can easily be computed in $O(n \log n)$ time. More challenging is finding the pairs of edges with vertical distance larger than μ_0, since we want to report these constraints only if there are at most $2n$ of them, without spending more time otherwise.

To report this type of violated constraint we proceed in two steps. First we use a data structure based on hereditary segment trees to reduce the vertical distance problem between line segments to a problem of reporting lines in 3D that are at least μ_0 apart. Then we solve that problem by mapping the lines in space to Plücker points and Plücker hyperplanes, and solving a halfspace range reporting problem in 5D projective space. By using the trade-off techniques for halfspace range reporting we find all the violated constraints in $O^*(n^{4/3} + |V|)$ time, with the option to stop reporting if $|V| > 2n$. A more detailed description follows.

For the first part, we use two-level hereditary segment trees [3]. We consider the set of segments of the projection of the two terrains. They can be seen as a set of red and a set of blue segments (one set from each terrain; segments from the same terrain cannot intersect). As in [3], the segment tree can be augmented to produce a bipartite clique decomposition of the intersecting pairs of edges. Within each clique of segments, we extend the line segments to lines, without adding or missing any intersection/distance. The complete data structure can be built in $O(n \log^2 n)$ time.

Next we must find the pairs of red-blue lines that are more than μ_0 vertically apart. This second problem is transformed into a halfspace range reporting problem as follows. Testing whether two lines ℓ_1, ℓ_2 in 3D lie at a vertical distance of more than μ_0 from each other can be formulated (after translating one of the lines by μ_0) as testing whether the Plücker point of ℓ_1 lies in one of the halfspaces bounded by the Plücker halfplane of ℓ_2. We map the lines from, say, g, to Plücker points and perform a halfspace reporting query with each of the halfspaces bounded by the Plücker hyperplanes of the lines induced by edges of f_0. To solve the halfspace reporting problem we apply standard trade-off techniques for geometric range searching (see for example [1]). We build a data structure of size $O^*(n^{4/3})$ that allows to answer halfspace reporting queries in $O(n^{1/3} + k)$ time, where k is the output size. In our context, k is the number of pairs of segments with distance more than μ_0. Note that in line 4 of the algorithm, we do not need to report all violated constraints, but we can stop when their number exceeds $2n$. Thus line 4 takes $O(n^{4/3})$ time.

Choosing a new random sample. The standard random-sampling argument in the Clarkson-Shor framework (see [5]) states that if we choose a random sample of size r among m constraints, and solve the LP with only those constraints, the expected number of violated constraints will be roughly dm/r, where d is the dimension of the LP problem. In our case we have $d = 3$, $r = 3n$ and $m \approx n^2$. Thus the expected number of violated constraints is approximately n, and on average the random subset of constraints will have to be generated twice.

Number of iterations. It can be shown that once $|V|$ has fewer than $2n$ constraints, the loop of Algorithm 1 will be executed at most four times. This is because every time the LP problem is solved, one of the three constraints that define the optimal solution is added to the set of violated constraints (see [5] for a proof), and from then on, it will be part of R. Therefore after executing line 3 at most three times, the set of constraints R will contain the three vertices that define the optimum, hence in the next (fourth) iteration the optimum will be found.

Theorem 1 *Given two terrains with n vertices, a linear transformation minimizing the maximum distance between the terrains can be found in $O^*(n^{4/3})$ expected time.*

3 Minimizing the area between two 1-d terrains

Let f, g be two 1-dimensional terrains, or, the graphs of two piecewise linear functions. This section discusses the problem of aligning f and g in order to minimize the area in between. For any value of s and t, let $\mu(s,t)$ denote the area between $sf + t$ and g over their domain. We will study the problem in the solution space, the (s,t)-plane.

Analytic form of μ. The bivariate function $\mu(s,t)$ has an analytic form that depends on which edges of $sf + t$ and g intersect. If $x_1, \ldots, x_n$ is the increasing order of the x-coordinates of all vertices of f and g, then we may as well assume that both f and g have vertices at these x-coordinates. At worst, this doubles the number of vertices of the input. Let $(x_1, a_1), \ldots, (x_n, a_n)$ be the vertices of f and let $(x_1, b_1), \ldots, (x_n, b_n)$ be the vertices of g.

Consider any vertical slab defined by two consecutive x-coordinates of vertices x_i and x_{i+1}, see Figure 1. Let f_i, g_i, and μ_i be the functions f, g, and μ restricted to the domain $[x_i, x_{i+1}]$. Unless f_i is constant,

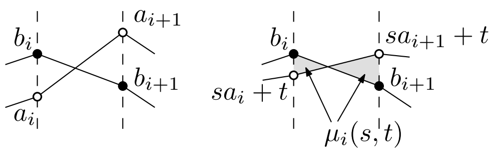

Figure 1: An edge of f and an edge of g, and the function $\mu_i(s,t)$ illustrated.

there is exactly one pair s, t that makes $sf_i + t \equiv g_i$, and $\mu_i(s,t) = 0$. This happens when both $sa_i + t = b_i$ and $sa_{i+1} + t = b_{i+1}$.

Lemma 2 *The region in the (s,t)-plane where $sf_i + t$ and g_i intersect is a double wedge whose bounding lines are $\ell_i : t = -a_i s + b_i$ and $\ell_{i+1} : t = -a_{i+1}s + b_{i+1}$. These lines intersect in the apex $(\bar{s}_i, \bar{t}_i)$, where $\bar{s}_i f_i + \bar{t}_i \equiv g_i$ and $\mu_i(\bar{s}_i, \bar{t}_i) = 0$.*

Above ℓ_i and ℓ_{i+1} (in the direction of t), μ_i is linear in s and t, and the same is true below ℓ_i and ℓ_{i+1}. Below ℓ_i and above ℓ_{i+1} in the (s,t)-plane, we have

$$\mu_i(s,t) = (x_{i+1} - x_i) \times \frac{(b_i - sa_i - t)^2 + (sa_{i+1} - b_{i+1} + t)^2}{2(sa_{i+1} - sa_i - b_{i+1} + b_i)} .$$

A similar expression exists for the wedge in the (s,t)-plane above ℓ_i and below ℓ_{i+1}.

We observe that inside the double wedge defined by ℓ_i and ℓ_{i+1}, μ_i is a fraction that has the unknown s in the denominator. To minimize μ, we must solve

$$\min_{s,t} \sum_{i=1}^{n-1} \mu_i(s,t) ,$$

which is a sum of fractions and involves solving a polynomial of linear degree, if the minimum occurs where many edges of f and g intersect. It appears that we cannot hope to get an exact solution in this case.

Approximation for μ. We continue to derive an approximation algorithm. For any given $\varepsilon > 0$, we will determine values of s and t such that the value of μ is at most $(1+\varepsilon)$ times μ^*, the minimum possible value of μ. Our approach is to linearize each function μ_i, so that the sum we get is linear as well.

We first analyze the function μ_i closer. For a fixed scaling factor $\hat{s}$, consider the function $\mu_i(\hat{s}, t)$. From Lemma 2 we know that below ℓ_i and ℓ_{i+1}, or above both of them, $\mu_i(\hat{s}, t)$ is linear in t with slope $x_i - x_{i+1}$ and $x_{i+1} - x_i$. Between ℓ_i and ℓ_{i+1}, $\mu_i(\hat{s}, t)$ is a quadratic function in t. In fact, $\mu_i(\hat{s}, t)$ is symmetric, differentiable everywhere, and consists of three pieces, see Figure 2. Since this holds for any scaling factor $\hat{s}$ except the one of the apex of the double wedge (where it has two pieces and is not differentiable in the apex), we obtain useful properties of the whole function μ_i.

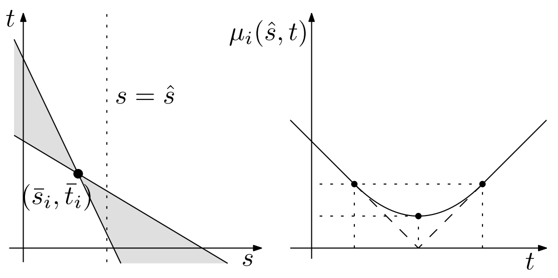

Figure 2: A double wedge in the s, t-plane, and a vertical cross-section of it that shows $\mu_i(\hat{s}, t)$.

Lemma 3 *μ_i is a convex function, and the restriction of μ_i to a ray starting at $(\bar{s}_i, \bar{t}_i)$ is a linear function.*

Corollary 4 *μ is a convex function.*

To obtain a $(1 + \varepsilon)$-approximation, we will replace μ_i by a piecewise linear function. Since μ_i is already linear outside the double wedge, we only need to do so inside it. We do so by first approximating the quadratic part of $\mu_i(\hat{s}, t)$. We choose $\lceil 2/\sqrt{\varepsilon} \rceil$ extra points equally-spaced on $\mu_i(\hat{s}, t)$ such that the maximum distance between the polygonal line through the chosen points and $\mu_i(\hat{s}, t)$ itself is at most $\varepsilon \mu_i(\hat{s}, t)$, see Figure 3. We can extend this 1-dimensional approximation to a 2-dimensional one by choosing lines through the new points and $(\bar{s}_i, \bar{t}_i)$, giving a partition of the plane into $\Theta(1/\sqrt{\varepsilon})$ wedges. Lemma 3 ensures that the implied piecewise linear function—denoted $\tilde{\mu}_i$—is a $(1 + \varepsilon)$-approximation of μ_i everywhere.

Now we are ready to consider all edges from f and g. If we were considering the exact function μ, we would get $n - 1$ double wedges in the (s, t)-plane. They form an arrangement of $O(n^2)$ cells, and in each cell we can write down a closed expression for μ as the sum of the μ_i. However, this gives us the expression that we cannot optimize exactly. Using the piecewise linear approximations $\tilde{\mu}_i$, we get an arrangement of $O(n/\sqrt{\varepsilon})$ lines which partition the (s, t)-plane in $O(n^2/\varepsilon)$ cells. The sum of at most $n - 1$ linear functions is of course linear, and with little extra work we can easily find the minimum of $\tilde{\mu}(s, t)$ in $O(n^2/\varepsilon)$ time.

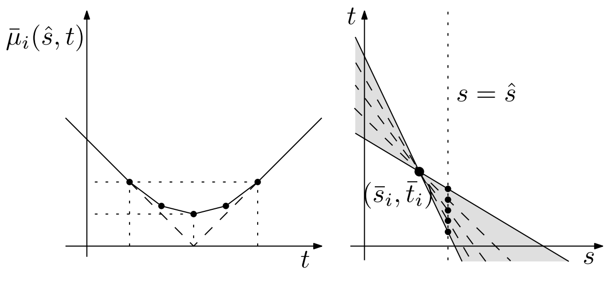

Figure 3: The approximation $\bar{\mu}_i(\hat{s}, t)$ and the subdivision into double wedges in the s, t-plane.

A more efficient solution is obtained using cuttings for a divide-and-conquer approach. This works because all $\tilde{\mu}_i$ are convex functions, so $\tilde{\mu}$ is convex as well. Let $n' = n/\sqrt{\varepsilon}$, and consider the $O(n')$ lines that define the boundaries of the cells over which $\tilde{\mu}$ is linear. We compute a $(1/2)$-cutting of size $O(1)$ in $O(n')$ time [2]. On each of the $O(1)$ edges in the cutting, we find the minimum in $O(n')$ time, which will reveal in which triangle of the cutting the global optimum of $\tilde{\mu}$ lies. We recurse on the lines that intersect this triangle; their number is halved by the definition of $(1/2)$-cuttings.

After $O(\log n')$ phases of divide-and-conquer, which together take $O(n')$ time, we find the triangle that lies in a single cell in the arrangement of all $O(n')$ lines that contains the global optimum.

Theorem 5 *Let f and g be two one-dimensional terrains with n vertices. For any fixed $\varepsilon > 0$, a linear transformation of the image of f that yields at most area $(1+\varepsilon)A^*$ between the images of g and the transformed f can be determined in $O(n/\sqrt{\varepsilon})$ time, where A^* is the minimum area achievable.*

4 Conclusions and open problems

The obvious open problem is extending the result on one-dimensional terrains to two-dimensional terrains, which is what we are still working on. Another extension we consider is a third measure for matching, namely the integral of squared difference between the functions.

References

[1] P. K. Agarwal and J. Erickson. Geometric range searching and its relatives. In B. Chazelle, J. E. Goodman, and R. Pollack, editors, *Advances in Discrete and Computational Geometry*, volume 223 of *Contemporary Mathematics*, pages 1–56. American Mathematical Society, Providence, RI, 1999.

[2] B. Chazelle. Cutting hyperplanes for divide-and-conquer. *Discrete & Computational Geometry*, 9:145–158, 1993.

[3] B. Chazelle, H. Edelsbrunner, L. J. Guibas, and M. Sharir. Algorithms for bichromatic line segment problems and polyhedral terrains. *Algorithmica*, 11:116–132, 1994.

[4] K. L. Clarkson. Linear programming in $O(n3^{d^2})$ time. *Inf. Process. Lett.*, 22(1):21–24, 1986.

[5] K. L. Clarkson. Las Vegas algorithms for linear and integer programming when the dimension is small. *J. ACM*, 42(2):488–499, 1995.

[6] M. de Berg, H. J. Haverkort, S. Thite, and L. Toma. I/O-efficient map overlay and point location in low-density subdivisions. In *ISAAC*, volume 4835 of *Lecture Notes in Computer Science*, pages 500–511, 2007.