Minimizing Slope Change in Imprecise 1.5D terrains [*]

Chris Gray[†] Maarten Löffler[‡] Rodrigo I. Silveira[‡]

Abstract

A 1.5D terrain is an x-monotone polyline with n vertices. An *imprecise* 1.5D terrain is a 1.5D terrain with a y-interval at each vertex, rather than a fixed y-coordinate. A realization of an imprecise terrain is a sequence of n y-coordinates—one for each interval—such that each y-coordinate is within its corresponding interval. For certain applications in terrain analysis, it is important to be able to find a realization of an imprecise terrain that is *smooth*. In this paper we model smoothness by considering the change in slope between consecutive edges of the terrain. The goal is to find a realization of the terrain where the maximum slope change is minimized. We present an exact algorithm that runs in $O(n^2)$ time.

1 Introduction

A common way to model a terrain is by using a triangulated irregular network (TIN): a planar triangulation with height information on the vertices. This defines a bivariate and continuous function, defining a surface that is often called a 2.5-dimensional (or 2.5D) terrain.

Even though in computational geometry it is usually assumed that the input data is exact, in practice, terrain data is often imprecise. The sources of imprecision are many, starting from the methods used to acquire the data. Elevation data is often collected using remote sensing techniques, for example by airplanes flying over the terrain and sampling the distance to the ground, such as in LIDAR (Light Detection and Ranging), or it can be obtained by optically scanning contour maps and then fitting an approximating surface. Such methods often produce heights with a known error bound or return a height interval rather than a fixed height value. For example, in high-resolution terrains distributed by the United States Geological Survey, it is not unusual to have vertical errors of up to 15 meters [5].

In order to model the imprecision in the terrain, we follow the model used in [1–3], where the height of each terrain vertex is not precisely known, but only an interval of possible heights is available. This creates some freedom in the terrain: the real terrain is unknown, and any choice of a height for each vertex, as long as it is within its height interval, leads to a *realization* of the imprecise terrain.

The large number of different realizations of an imprecise terrain leads naturally to the problem of finding one that is best according to some criterion. In this paper we are concerned with *smoothness*. Obtaining a realization of an imprecise terrain that is smooth is interesting for several uses of terrains, including terrain analysis, visualization, compression, and noise reduction.

In this paper, however, we study smoothing for imprecise 1.5D terrains, that is 1.5D terrains with a y-interval at each vertex, rather than a fixed y-coordinate. Even though the 2.5D version is clearly more interesting, a simpler model is easier to handle and gives insight into the difficulties of 2.5D terrains.

This work is a continuation of [2]. In that paper, we investigated smoothness criteria related to turning angles, such as minimizing the maximum turning angle and minimizing the total turning angle. For the former criterion we gave a polynomial-time approximation scheme, whereas for the latter we presented an exact linear-time algorithm.

In this paper we look at the smoothness criterion of minimizing the maximum slope change. The main motivation for considering another criterion, after having studied turning angles, is that the minimization of the largest turning angle presents algebraic difficulties that prevent the design of exact algorithms under the real-RAM computation model. However, as we show in this paper, when the goal is the minimization of the maximum slope change, an optimal realization can be found in quadratic time.

It must be noted that the minimization of the maximum slope change yields different results than the minimization of the maximum turning angle in general. This is particularly clear in terrains with very steep edges, where a small change in the angle of the edge leads to a large change in slope. However, for terrains where no edges have extreme slopes, the slope change is a reasonable and intuitive measure of smoothness.

[*]This research was partially supported by the Netherlands Organisation for Scientific Research (NWO) through the project GOGO and project no. 639.023.301.

[†]Department of Computer Science, TU Braunschweig, Germany, gray@ibr.cs.tu-bs.de

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

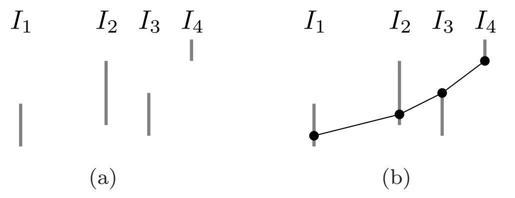

Figure 1: An example with four intervals, and a possible solution for $\delta = \frac{1}{4}$.

2 Preliminaries

In this section we introduce imprecise 1.5D terrains more formally, together with a number of definitions and concepts, most of them rather intuitive, which will be used throughout the remainder of the paper.

As mentioned in the introduction, a *1.5D terrain* is an x-monotone polyline with n vertices. An *imprecise 1.5D terrain* is a 1.5D terrain with a y-interval at each vertex rather than a fixed y-coordinate. More formally, an (imprecise) terrain T is given by a sequence of n *intervals* $\{I_1, I_2, \cdots, I_n\}$. Each interval I_i has an x coordinate x_i (with $x_i < x_j$ if $i < j$), and a closed interval of possible y coordinates, with the maximum y coordinate denoted by $\overline{y}_i$ and the minimum y coordinate denoted by $\underline{y}_i$. When, in addition to an imprecise terrain, we are given a sequence of n y-coordinates, one for each interval, we have a *realization* of the terrain. Each y-coordinate must be within its corresponding interval.

A realization of a terrain induces an x-monotone polyline with n vertices, with one vertex per interval. The vertex corresponding to interval I_i is denoted v_i, and has coordinates (x_i, y_i). See Figure 1.

3 Minimizing the maximum slope change

We describe an exact algorithm based on parametric search—a technique introduced by Megiddo [?]. This technique requires a subroutine that tests the feasibility of finding a solution to the overall problem with a given parameter. In our case, we must decide whether a realization of a terrain exists with a maximum slope change that is less than some fixed value δ. First, we present a naive decision algorithm that displays the main ideas and observations we use, but runs in $O(n^2)$ time. Then we show how to improve the running time to $O(n)$.

3.1 A naive decision algorithm

To determine whether a terrain can have a maximum slope change of δ while respecting the height constraints, we study the problem in *feature space*. This means that we add a dimension, λ, to each interval. This dimension represents the slope of the terrain. We define a set of rules for a path in feature space. A path

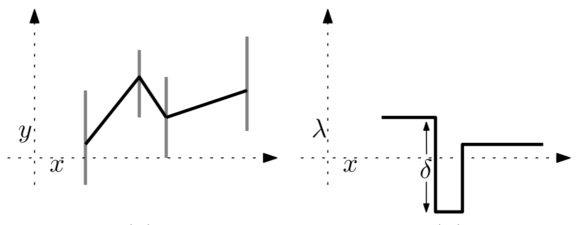

Figure 2: A path with maximum slope change δ shown in the (a) (x, y) plane and (b) (x, λ) plane.

moves from point $p_1 = (x_1, y_1, \lambda_1)$ to $p_2 = (x_2, y_2, \lambda_1)$ at a slope of λ_1. It is then allowed to move to $p_3 = (x_2, y_2, \lambda_2)$, where $|\lambda_2 - \lambda_1| \leq \delta$. From there it continues in the same manner until it reaches I_n or can no longer intersect an interval. As an example, the path $(0, 0), (1, 2), (3, 1)$ in an imprecise terrain becomes the path $(0, 0, 2), (1, 2, 2), (1, 2, -0.5), (3, 1, -0.5)$ in feature space. See Figure 2.

For every interval, except for the first and last, we consider two regions, called the *entrance region* and the *exit region*. They represent all the possible paths with maximum slope change at most δ that can go through the interval. The entrance region defines possible entry slopes (for the edge entering from the left), whereas the exit region represents the possible exit slopes (for the edge leaving on the right). For interval I_j, the entrance region is defined to be the set of points (x_j, y, λ) where $(x_j, y) \in I_j$ and where there exists a path consistent with $I_1, I_2, \ldots, I_j$ with maximum slope change δ that enters I_j at (x_j, y) via an edge with slope λ. The exit region of I_j is defined analogously: it is the set of points (x_j, y, λ_2) where (x_j, y, λ_1) is in the entrance region for I_j for some λ_1 satisfying $|\lambda_2 - \lambda_1| \leq \delta$. The exit region for the interval I_1 is defined to be the infinite strip (x_1, y, λ) where y is between the top and bottom y-values of interval I_1 and λ can take any value.

Clearly, the decision question is now equivalent to the question of whether the entrance region of I_n is empty or not. We proceed to describe how the entrance and exit regions can be computed recursively.

We recursively define some functions for each interval, which we later show bound the entrance and exit regions. For the first interval I_1, we define two constant functions $f_1^+(\lambda) = \overline{y}_1$ and $f_1^-(\lambda) = \underline{y}_1$. In fact, these two functions bound the exit region of I_1. Now, for any $j > 1$, we define six functions as follows.

- We first define intermediate functions $h_j^+(\lambda) = f_{j-1}^+(\lambda) + \lambda(x_j - x_{j-1})$ and $h_j^-(\lambda) = f_{j-1}^-(\lambda) + \lambda(x_j - x_{j-1})$.
- Next, we compute the highest and lowest feasible points $y_j^+ = \max\{y \mid h_j^+(y) \geq h_j^-(y)\}$ and $y_j^- = \min\{y \mid h_j^+(y) \geq h_j^-(y)\}$.

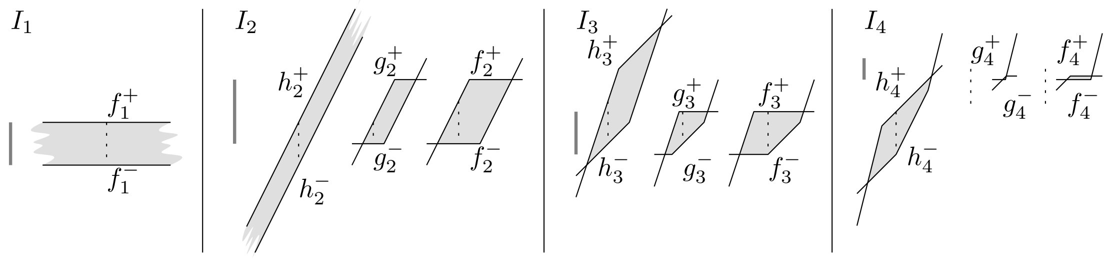

Figure 3: The regions as they are computed in the process for $\delta = \frac{1}{4}$, for the same example as shown in Figure 1. The figures are drawn in the (λ, y)-plane. The dotted lines correspond to $\lambda = 0$. For each interval (except the first), three regions are shown: the first one is an intermediate region used to construct the entrance region (it is a sheared copy of the previous exit region). The second region is the entrance region. The third one is the exit region: the entrance region expanded by having the bounding functions moved apart a distance of 2δ.

- We truncate $h_j^+(\lambda)$ and $h_j^-(\lambda)$ so that their y-values do not go above the top or below the bottom y-value of I_j or above or below the just computed extreme feasible values. The resulting functions are $g_j^+(\lambda) = \min(h_j^+(\lambda), y_j^+, \overline{y}_j)$ and $g_j^-(\lambda) = \max(h_j^-(\lambda), y_j^-, \underline{y}_j)$. We show in Lemma 1 that the region enclosed by $g_j^+(\lambda)$ and $g_j^-(\lambda)$ is the entrance region.
- Finally, we expand the area between these functions by moving the functions horizontally δ away from each other in opposite directions. We define $f_j^+(\lambda) = g_j^+(\lambda+\delta)$ and $f_j^-(\lambda) = g_j^-(\lambda-\delta)$. We show in Lemma 1 that the region enclosed by $f_j^+(\lambda)$ and $f_j^-(\lambda)$ is the exit region.

From this construction, we can immediately observe that all the functions involved are monotonically increasing. Also, we can see that the functions f_j^-, g_j^-, and h_j^- are convex, while the functions f_j^+, g_j^+, and h_j^+ are concave.

Lemma 1 *The entrance region for interval I_j is the set of points bounded by the functions g_j^+ and g_j^-, that is, those points (λ, y) for which $g_j^-(\lambda) \leq y \leq g_j^+(\lambda)$. Similarly, the exit region of I_j is bounded by the functions f_j^+ and f_j^-.*

Proof. First note that for any point (x, y, λ) in the entrance region, $(x, y, \lambda + \mu)$ is in the exit region for every $-\delta \leq \mu \leq \delta$.

Now assume that the region bounded by g_j^+ and g_j^- is non-empty and that $p_j = (x_j, y_j, \lambda_j)$ is a point inside it. We describe a procedure for finding a path through the intervals that has a maximum slope change of δ, which shows that p_j is in the entrance region of I_j.

We begin at interval I_j. We then construct a line segment from p_j to the interval I_{j-1} that has slope λ_j. Let the point where the line segment intersects the interval I_{j-1} be $p' = (x_{j-1}, y_{j-1}, \lambda_j)$. Note that y_{j-1} is easy to compute: it is $y_j - \lambda_j(x_j - x_{j-1})$. Because of the procedure for propagating the regions, the point p' is inside the region bounded by f_{j-1}^+ and f_{j-1}^-.

If p' is outside the region bounded by g_{j-1}^+ and g_{j-1}^-, then we find $-\delta \leq \mu \leq \delta$ such that $p'' = (x_{j-1}, y_{j-1}, \lambda_j + \mu)$ is inside that region, and set λ_{j-1} to be $\lambda_j + \mu$. We can then recursively apply this procedure until we have found a path from I_j to I_1 whose maximum slope change is δ. This path is formed by the sequence of points $(x_1, y_1), (x_2, y_2), \ldots, (x_j, y_j)$. See Figure 2.

The proof that no valid path exists to p_j if p_j is outside the region bounded by g_j^+ and g_j^- is similar, and we omit it in the interest of space. $\square$

To compute the running time of the algorithm above, we must bound the running time taken when propagating a region from one interval to another. From this information, finding the total running time is simple.

We begin with a lemma that characterizes the shape and complexity of an entrance region.

Lemma 2 *The entrance and exit regions of interval I_j are convex polygons with complexity at most $2j$.*

This result, proved in the full version of the paper, almost immediately implies that this algorithm takes quadratic time.

3.2 A faster decision algorithm

In the previous subsection, we described how the regions in the (λ, y)-plane for each interval propagate, and how to compute them incrementally. Since the complexity of one region can be linear, we spent quadratic time in total. However, in the end we are only interested in whether the last entrance region is empty or not. It is possible to save a linear factor by keeping track of the region boundaries implicitly.

Consider the vertices of the polyline that describes the top boundary of the exit region of I_j, that is, f_j^+.

When computing the exit region for I_{j+1}, these vertices undergo two transformations: a shear and a translation. In the clipping step, vertices may be thrown away and/or new vertices may be added, but the existing vertices do not move. In this subsection, we describe a scheme where we do not explicitly perform the shears and translations on the vertices. Rather, we compute one transformation T^+ that is the composition of the sequence of the shears and translations. We can then apply T^+ to the original locations of the vertices to get their actual locations. Note that T^+ can be represented in constant space. It follows from the theory of affine transformations that a sequence of shear and translation operations can be represented as a single shear and a single translation.

Now we can describe the new algorithm. Suppose that we have processed I_j and we have computed two lists of vertices L_j^+ and L_j^-, together with two transformations T_j^+ and T_j^-, such that applying T_j^+ to L_j^+ gives a representation of f_j^+ and applying T_j^- to L_j^- gives a representation of f_j^-. Then we proceed to I_{j+1} as follows.

First, we need to apply the shear operation $(y, \lambda) \mapsto (y, y + \lambda(x_{j+1} - x_j))$ to the exit region; we do this by composing T_j^+ and T_j^- with this transformation. This step takes constant time.

Next, we need to clip the region between two horizontal lines y_j^+ and y_j^-. We walk over the lists L_j^+ and L_j^- starting from the first vertex, and for each vertex v we apply T_j^+ (resp. T_j^-) to it, and check whether the resulting v' lies below y_j^-. If it does, we throw v away and proceed to the next vertex. If it does not, we keep v. We create a new vertex where y_j^- intersects the two lists, we compute those vertices and then apply the inverse of T_j^+ (resp. T_j^-) to make sure they are in the same space as the remaining vertices. In the same way, we walk over L_j^+ and L_j^- starting from the last vertex to clip against y_j^+. This step takes constant time per vertex that is thrown away.

Finally, we expand the entrance region by translating its vertices outward by δ (in the λ-dimension) to get the exit region. Since f_j^+ and f_j^- are monotonically increasing, this means all vertices in L_j^+ move δ to the left, and all vertices in L_j^- move δ to the right. We apply these translations to T_j^+ and T_j^-. This step also takes constant time.

Lemma 3 *The time complexity of the feasibility-testing algorithm is $O(n)$.*

Proof. The second step takes constant time per vertex that is thrown away. Since each vertex can be thrown away at most once, this is linear in total. For the rest, each step takes constant time, so we spend linear time overall. □

This feasibility-testing algorithm, together with parametric search, gives us an algorithm that runs in time quadratic in the number of input intervals.

Theorem 1 *Given an imprecise 1.5D terrain with n intervals, a realization that minimizes the maximum slope change can be computed in $O(n^2)$ time.*

Proof. By Lemma 3, the feasibility-testing algorithm takes $O(n)$ time. We apply parametric search with this algorithm. Since the feasibility test is sequential, the parametric search squares the running time of the feasibility-testing algorithm, giving us $O(n^2)$ time. □

4 Conclusions

We presented an $O(n^2)$-time algorithm to find a realization of an imprecise 1.5D terrain that minimizes the maximum slope change. This constitutes an interesting result, given that a similar criterion like minimizing the maximum turning angle presents algebraic difficulties that prevent exact algorithms under the real-RAM computation model [2]. The main question left open is whether the feasibility-testing algorithm can be made parallel. If that would be the case, then together with parametric search, it could lead to an improvement in the running time to $O(n \log n)$.

Acknowledgments

The authors would like to thank several anonymous referees for their useful comments.

References

[1] C. Gray and W. Evans. Optimistic shortest paths on uncertain terrains. In *Proc. 16th Canadian Conference on Comput. Geom.*, pages 68–71, 2004.

[2] C. Gray, M. Löffler, and R. I. Silveira. Smoothing imprecise 1.5D terrains. In *Proc. Sixth International Workshop on Approximation and Online Algorithms*, pages 214–226, 2009.

[3] Y. Kholondyrev and W. Evans. Optimistic and pessimistic shortest paths on uncertain terrains. In *Proc. 19th Canadian Conference on Comput. Geom.*, pages 197–200, 2007.

[4] J. S. Salowe. *Parametric search*, pages 683–695. CRC Press, Inc., 1997.

[5] T. Tasdizen and R. T. Whitaker. Feature preserving variational smoothing of terrain data. In *Proceedings of the 2nd International IEEE Workshop on Variational, Geometric and Level Set Methods in Computer Vision*, 2003.