Smoothing imprecise 1-dimensional terrains *

Chris Gray† Maarten Löffler‡ Rodrigo I. Silveira‡

Abstract

An imprecise 1-dimensional terrain is an x-monotone polyline where the y-coordinate of each vertex is not fixed but only constrained to a given interval. In this paper we study four different optimization measures for imprecise 1-dimensional terrains, related to obtaining smooth terrains. In particular, we present algorithms to minimize the largest and total turning angle, and to maximize the smallest and total turning angle.

1 Introduction

Terrain modeling is a central task in geographical information systems (GIS). Terrain models can be used in many ways, for example for visualization or analysis purposes (to compute features like watersheds or visibility regions [1]). One common way to represent a terrain is by means of a triangulated irregular network (TIN): a planar triangulation with additional height information on the vertices.

This height information is often collected by airplanes flying over the terrain and sampling the distance to the ground, for example using radar or laser altimetry techniques, or it is sometimes obtained by optically scanning contour maps and then fitting an approximating surface. These methods often return a height interval rather than a fixed value, or produce heights with some known error bound. 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 [7]. However, algorithms in computational geometry often assume that the height values are precise. This may lead to artifacts in the terrain.

An alternative to deal with this imprecision in terrains is to use a more involved model that takes the imprecision into account. Gray and Evans [2] propose a model where an interval of possible heights is associated with every vertex of the triangulation. Figure 1 shows an example of an imprecise terrain, and a possible instance of the real terrain. Kholondyrev and Evans [5] also study this model. Silveira and Van Oostrum [6] also allow moving vertices of a TIN up and down to remove local minima, but do not assume bounded intervals.

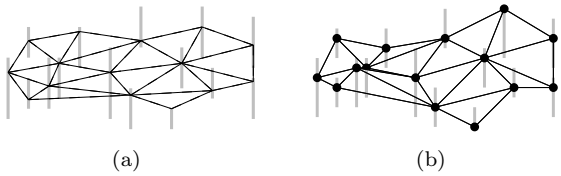

Figure 1: (a) An imprecise terrain. (b) A possible instance of the real terrain.

In this paper we use the same model as in [2, 5]; each vertex has a height interval. This leads to some freedom in the terrain: the real terrain is unknown. This paper deals with trying to obtain a *smooth* terrain, respecting the height intervals. This may be needed either because some additional information about the morphology of the terrain is known (that is, there are not too many sharp ridges in that area) or for visualization or compression purposes. Smoothing of terrains has been previously studied in the context of grid terrains [4, 7], where techniques from image processing can be applied, but not, to our knowledge, for imprecise TINs. In a TIN, a smooth terrain implies that the spatial angles between triangle normals are not too large. We can try to find a height value for each vertex, restricted by the intervals, such that the resulting terrain minimizes the largest spatial angle, or the sum of all spatial angles.

We study these measures, and two other ones, but only for 1-dimensional terrains. A 1-dimensional terrain is essentially an x-monotone polyline. An imprecise 1-dimensional terrain is the same thing, but with a y-interval for each vertex rather than a fixed coordinate—see Figure 2. This work constitutes a first step towards solving the 2-dimensional case.

We study four variants of the problem. In Section 2 we minimize the sum of the turning angles of the polyline, while in Section 3 we minimize the largest one. Both measures aim to smooth the terrain as much as possible; which is best to use depends on the situation at hand. In Section 4 we maximize the sum of the turning angles, and in Section 5 we maximize the smallest one. These measures aim to make the terrain as rough as possible: this gives some idea of the worst possible case.

*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 Computing Science, TU Eindhoven, the Netherlands, cgray@win.tue.nl

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

In the discussion of the algorithms below, we assume that the 1-dimensional terrain is an x-monotone polyline. In this context, we define a *left-turn* and *right-turn* as a vertex where this polyline, seen from left to right, turns to the left (upwards) or right (downwards) respectively.

2 Minimizing the total turning angle

To minimize the sum of (the absolute values of) the turning angles, we make the following observations. When we leave a certain vertex in a certain direction, and we enter another vertex from a certain direction, and the path between them is left-turning (or right-turning), then it does not matter how exactly this path goes: the total turning angle stays the same. This means there will most likely be many equivalent optimal solutions. An example is shown in Figure 2(a).

In fact, if the leftmost and rightmost intervals would just be single points, the shortest path between those points through the corridor between the polylines defined by the upper and lower endpoints of the intervals is one of the optimal solutions. To see why, consider the global shape of the shortest path. This will be a polyline, with a number of left and right turns. Each right turn must lie on the lower endpoint of its imprecision interval, otherwise there would be a shorter path possible. Similarly, each left turn must lie on the upper endpoint of its interval. Now call a segment of the shortest path *critical* when it has one left and one right turn. It is not hard to see that we cannot get a better turning angle than the sum of the turning angles between pairs of consecutive critical segments. However, the total turning angle of the shortest path achieves exactly this lower bound, and hence is optimal.

When the leftmost and rightmost intervals are not single points but real intervals, we need to make one additional observation. If we compute the shortest path from some point on the leftmost interval to some point on the rightmost interval, then this may contain some unnecessary turns at the ends. We can identify the leftmost and rightmost critical segments of the path, and note that the best thing to do is just continue in a straight line from these segments towards the leftmost and rightmost intervals since then we make no extra turns. It can be that this is not possible, but in that case we just make the turn at those extreme critical segments as small as possible.

The shortest path can be computed in linear time [3]. The adaptations that are necessary at the ends are also easy to do in linear time.

3 Minimizing the largest turning angle

It appears that the problem of minimizing the maximum angle in a realization of an uncertain terrain is difficult. The main problem is that finding a solution requires finding the inverse of a non-algebraic function. Therefore, we present an approximate solution. If the best realization of a given uncertain terrain has a maximum turning angle α, we find a terrain with maximum turning angle at most $\alpha + \varepsilon$, for any given ε. Our solution is a dynamic-programming algorithm that works by discretizing the range of angles that we consider.

Figure 2(b) shows an example of a path which minimizes the largest turning angle.

Let D be a set of $k = \lceil \pi/\varepsilon \rceil$ angles, evenly spaced from $-\pi/2$ to $\pi/2$. Our algorithm is based on solving subproblems of the form $S[p_1, p_2, d_1, d_2]$, where p_1 and p_2 are endpoints of intervals and d_1 and d_2 are directions from D. It is also based on the following observation: let P be the optimal path through a set of intervals between points p_1 and p_2 that are assumed to have infinite length. If we remove the assumption that the intervals are of infinite length and P no longer passes through all of the intervals, then the optimal path must pass through the endpoint of at least one of the intervals in the set. This allows us to find the optimal path recursively by trying all endpoints of all intervals between p_1 and p_2 and all directions from D.

We begin by describing the process of finding the optimal path through a set of intervals, assuming that the lengths of all the intervals are infinite. Let P^* be the optimal path between p_1 and p_2 that leaves p_1 at angle d_1 with respect to horizontal and enters p_2 at angle d_2 with respect to horizontal. It is easy to see that at almost all of the vertices of P^*, the turning angle is the same (call it θ^*). There may be one vertex at which the turning angle is smaller than θ^*, if the direction of curvature changes, but this happens at most once. We can find an ε-approximation to P^* by binary searching for θ^*.

We perform the binary search by constructing a path P_1 starting at p_1 and a path P_2 from right to left starting at p_2. We construct P_1 and P_2 so that all the turning angles are some common θ. The angle between P_1 and P_2 at their intersection allows us to determine whether we should raise or lower θ. When the angle at the intersection of P_1 and P_2 becomes less than ε, we stop the binary search. Since the range of angles for θ is bounded above by $\pi/2$, the time complexity for the binary search is $O(n \log 1/\varepsilon)$.

Note that the direction of the turns in P_1 and P_2 is not specified. In fact, there are four options depending on whether P_1 or P_2 are left-turning or right-turning. We perform the binary search for each of these options, keeping the search which returns the lowest θ.

As in Section 2, we initially reduce the first and last intervals to one of their endpoints. We call these points p_1 and p_n. We then find the

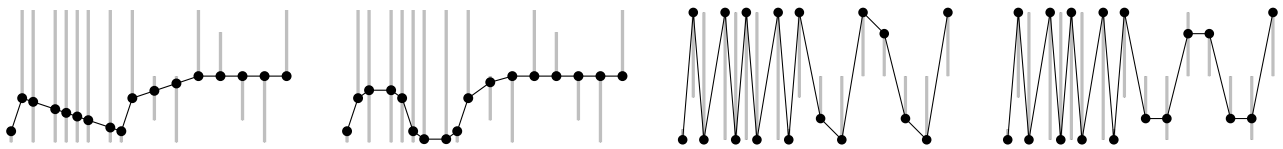

Figure 2: Four different measures applied to the same imprecise terrain. Optimal terrains for (a) min total turning angle, (b) min max angle, (c) max total turning angle and (d) max min angle.

lowest value of $S[p_1, p_n, d_1, d_2]$ over all values of d_1 and d_2. We construct the table S by combining the results of recursively-computed subproblems. We set $S[p_i, p_k, d_i, d_k] = \min_{p_j, d_j, d'_j} \max\{|d_j - d'_j|, S[p_i, p_j, d_i, d_j], S[p_j, p_k, d'_j, d_k]\}$.

Constructing the table S takes $O(n^2k^2)$ space and $O(nk^2 + n \log k)$ per entry. Therefore the entire algorithm takes $O(n^3(1/\varepsilon)^4)$ time.

4 Maximizing the total turning angle

When analyzing imprecise terrains, we may also be interested in the worst possible case for the real terrain. When our goal is to maximize the sum of the turning angles, we make the simple observation that we only need to consider the endpoints of the imprecision intervals. Indeed, if we have a solution that uses some point on the interior of an interval, there is at least one direction in which we can move the point such that the total turning angle does not decrease: if it lies on a left-turning or right-turning chain, moving it will not change the total angle at all, and if it does not, then it is always good to make the turn sharper, since this increases both its own turning angle and that of (one of) its neighbors. See Figure 2(c) for an example.

After making this observation, it is not hard to come up with a linear time algorithm to solve the problem. We simply apply 1-dimensional dynamic programming: for each segment, defined by the upper/lower endpoints of two consecutive intervals, we store the optimal solution to the left of that segment that uses it. There are $4n$ such segments, and computing a value involves matching it with the two possible stored solutions to the left of it and picking the best.

5 Maximizing the smallest turning angle

We can also try to maximize the smallest turning angle to try to make the terrain as rough as possible. Figure 2(d) shows an example.

For this problem we present a linear time approximation algorithm. The terrain computed by the algorithm will have a minimum angle at least $(\alpha^*_{min} - \varepsilon)$, where α^*_{min} is the minimum angle in the optimal terrain and ε is any constant. It is based on going through the intervals from left to right, and computing the solutions of partial subproblems defined between the first (leftmost) interval and the current one.

Given an instance of an imprecise terrain, we call an interval *extremal* if its vertex is positioned at one of the endpoints of the interval. An non-extremal interval is called *internal*. With some abuse of notation, we will also refer to extremal/internal *vertices*.

We base the algorithm on the following observations:

- Let T^* denote the optimal terrain, with minimum angle α^*_{min}, and let α^*_i denote the angle of the vertex x_i of I_i in T^*. We assume that $\alpha^*_{min} > \varepsilon$, otherwise any terrain will be within the approximation factor.
- Let I_i be an internal interval. Then $\alpha^*_i = \alpha^*_{min}$. Moreover, I_i is part of a left-turning/right-turning chain of angle α^*_{min}.
- The leftmost and rightmost intervals of T^* are extremal.
- If there is an interval I_i in T^* connecting a left-turning chain to a right-turning chain which is immediately followed by another left-turning chain (or *vice versa*), then I_i is extremal. If it is not, we can take the middle right-turning chain as a whole and move it up, until I_i becomes extremal, and balance all the angles again, increasing the overall minimum angle.
- The maximum number of vertices of any left-turning or right-turning chain in T^* is $K = \lceil \pi/\varepsilon \rceil$. Therefore the maximum number of consecutive intervals that are internal is $2K$.

The previous observations imply that if a subproblem has more than $2K$ intervals, there must be at least one extremal interval among them. Extremal intervals allow us to separate the problem into independent subproblems. As in the algorithm of Section 3, we will consider only $k = \lceil 2\pi/\varepsilon \rceil$ possible directions.

Assume for now the following subproblem can be solved in constant time. $OptimalChain(p_i, p_j, d_i, d_j)$, for $i < j, (j - i) < 2K$, returns the solution to the

subproblem from I_i to I_j, with fixed positions p_i and p_j, and fixed incoming directions at I_i and I_j, given by d_i and d_j, under the assumption that all the intervals between I_i and I_j are internal. Notice that since there are at most $2K$ intervals, the whole subproblem has constant size.

5.1 Main algorithm

The algorithm goes through the intervals from left to right. Solutions to partial subproblems are stored in a table with entries of the shape $S[j, p, d]$. Such an entry stores the value (and information to reconstruct the terrain) of a solution for the terrain going from I_1 to I_j, finishing at I_j at position p (top/bottom extreme) with direction d. We explain how to compute the value of $S[j, p, d]$, assuming that the values for $S[i, p, d]$, $i < j$, for all possible positions p and directions d, have been already computed.

Assume that we are computing $S[j, p, d]$ for p one of the two extremes of I_j, and some incoming direction d (incoming at I_j). In order to find the value of $S[j, p, d]$ we need to know which is the first extremal interval found when going from I_j to I_1. The previous observations show that such an interval lies between $I_{(j-2K)}$ and I_{j-1}. We will consider each of them. For each interval choice I_r, we will also consider both choices for the position at I_r, p_r, and all k choices for the incoming direction d_r. This gives rise to a total of $2K \cdot 2 \cdot k$ combinations. For each of them we must solve the subproblem between I_r and I_j. To solve it we apply the algorithm $OptimalChain(p_r, p_j, d_r, d)$. The solution of $OptimalChain(p_r, p_j, d_r, d)$ is combined with the entry $S[r, p_r, d_r]$ to obtain the total value of this alternative. The best value among all the ones considered is stored at $S[j, p, d]$.

5.2 Solving the subproblems

The previous observations imply that any optimal solution for a series of intervals that does not use any extremal point must be a left-turning chain followed by a right-turning chain, or *vice versa*, with all vertices having turning angle α^*_{min}. It can also be comprised of only one left-turning or right-turning chain, but we see it as a degenerate case of the previous ones, hence we consider two possible shapes: left-turning/right-turning and right-turning/left-turning.

To find an approximation of the optimal chain, we follow an approach similar to the one used in Section 3. We first guess the minimum angle α_{min} and the shape of the chains. Once the general shape of the chain is known, we construct a chain that at every vertex turns exactly α_{min}. If the final chain reaches the last interval of the subproblem and it does it with an angle of at least α_{min}, we are done. Otherwise we need to try another value of α_{min}. Recall that the number of possible angles to try is only $\lceil \pi/\varepsilon \rceil$, and that the size of the subproblem is constant. Therefore the subproblem can be solved in constant time.

It is easy to verify that the angle of the solution computed by the algorithm is at most ε away from α^*_{min}. The running time is $O(n \cdot (1/\varepsilon)^4 \log(1/\varepsilon))$.

6 Conclusions

We studied several measures to compute the smoothest or least smooth possible 1-dimensional terrain, when height information is imprecise. We gave efficient algorithms for three cases, and a somewhat less efficient algorithm for the fourth case. Two of the algorithms are approximate.

Future work includes trying to improve the $O(n^3)$ algorithm for minimizing the maximum angle. We would also like to tackle 2-dimensional terrains. This poses challenges both at the modeling level (a definition of a *smooth* TIN is not straightforward) and also at the algorithm level.

References

[1] L. de Floriani, P. Magillo, and E. Puppo. Applications of computational geometry in Geographic Information Systems. In J. Sack and J. Urrutia, editors, *Handbook of Computational Geometry*, pages 333–388. Elsevier, Amsterdam, 1997.

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

[3] L. J. Guibas, J. Hershberger, D. Leven, M. Sharir, and R. E. Tarjan. Linear-time algorithms for visibility and shortest path problems inside triangulated simple polygons. *Algorithmica*, 2:209–233, 1987.

[4] M. Hofer, G. Sapiro, and J. Wallner. Fair polyline networks for constrained smoothing of digital terrain elevation data. *IEEE Trans. Geosc. Remote Sensing*, 44:2983–2990, 2006.

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

[6] R. I. Silveira and R. van Oostrum. Flooding countries and destroying dams. In *Proc. 10th Workshop on Algorithms and Data Structures*, LNCS 4619, pages 227–238, 2007.

[7] 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.