Smoothing Imprecise 1.5D Terrains*

Chris Gray[1], Maarten Löffler[2], and Rodrigo I. Silveira[2]

[1] Department of Computer Science, TU Braunschweig, Germany
gray@ibr.cs.tu-bs.de
[2] Dept. Computer Science, Utrecht University, The Netherlands
{loffler,rodrigo}@cs.uu.nl

Abstract. We study optimization problems in an imprecision model for polyhedral terrains. An imprecise terrain is given by a triangulated point set where the height component of the vertices is specified by an interval of possible values. We restrict ourselves to 1.5-dimensional terrains: an imprecise terrain is given by an x-monotone polyline, and the y-coordinate of each vertex is not fixed but constrained to a given interval. Motivated by applications in terrain analysis, in this paper we present two linear-time approximation algorithms, for minimizing the largest turning angle and for maximizing the smallest one. In addition, we also provide linear time exact algorithms for minimizing and maximizing the sum of the turning angles.

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 [4]. 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 defines a bivariate and continuous function, defining a surface that is often called a 2.5-dimensional (or 2.5D) terrain.

The height information used to construct TINs 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 [12]. However, algorithms in computational geometry often assume that the height values are precise.

* This research was partially supported by the Netherlands Organisation for Scientific Research (NWO) through the project GOGO and project no. 639.023.301. A preliminary version of this paper was presented at the 24th European Workshop on Computational Geometry (EuroCG 2008), under the title "Smoothing imprecise 1-dimensional terrains".

E. Bampis and M. Skutella (Eds.): WAOA 2008, LNCS 5426, pp. 214–226, 2009.

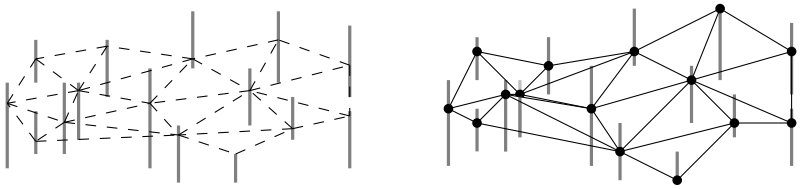

Fig. 1. (a) An imprecise terrain. (b) A possible realization of the real terrain.

In order to deal with this imprecision in terrains, one can use a more involved model that takes imprecision into account. Gray and Evans [5] propose a model where an interval of possible heights is associated with every vertex of the triangulation. Fig. 1 shows an example of an imprecise terrain, and a possible realization of the real terrain. Kholondyrev and Evans [8] also study this model. Silveira and Van Oostrum [11] also allow moving vertices of a TIN up and down to remove local minima, but do not assume bounded intervals. In this paper we use the same model as in [5,8]: each vertex has a height interval.

The imprecision model creates some freedom in the terrain: 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. This leads naturally to the problem of finding one that is best (i) according to the characteristics of the real terrain being modeled, and (ii) depending on the way the model will be used. As an example, consider a terrain model of an area that is known to have a smooth topography, like the dunes of some desert or hills in a relatively flat area. Then based on the information known about the real terrain, trying to find a realization that is *smooth* becomes important. On the other hand, terrain models are often used for terrain analysis, such as water run-off simulation. When simulating the way water flows through a terrain, artificial pits create artifacts where water accumulates, affecting the simulation. Therefore minimizing the number of pits is another criterion of interest.

There are several interesting applications for smoothing terrains. A first one, already mentioned, is because some additional information about the topography of the terrain may be known. Other applications include visualization (smooth shapes usually look better), compression and noise reduction. Smoothing of terrains has been previously studied in the context of grid terrains [12,7], 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 small. Measures on spatial angles are known to be important for several GIS applications [2,3,13]. In our context, we can try to find a height value for each vertex, restricted by the intervals, such that the resulting terrain minimizes some function of the spatial angles that models the notion of *smoothness*, like the largest spatial angle or the sum of all spatial angles.

We study several measures related to spatial angles, but only for 1.5D terrains. 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. As will become clear, this restricted model is still challenging enough and the results can serve as building blocks for the 2.5D version. Some comments on this are given in the conclusions. Moreover, 1.5D terrains are interesting in their own right, and have been recently studied in relation to guarding problems [1,9].

Our main result, presented in Sect. 3, is an approximation algorithm for minimizing the largest turning angle in the terrain. A similar algorithm can be used for the opposite problem of maximizing the smallest turning angle, leading to the *worst case* realization of the terrain, and is discussed in Sect. 4. Section 5 deals briefly with exact algorithms for minimizing and maximizing the sum of the turning angles. All our algorithms run in linear time. Due to space limitations, most results presented here are only sketched on a global or intuitive level; the interested reader can find more details in the full version of this paper [6].

2 Preliminaries

In this section we introduce imprecise 1.5D terrains more formally, together with a number of definitions and concepts that are used throughout the remainder of the paper.

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 a terrain is given by a sequence of n y-coordinates, one for each interval. Each y-coordinate must be within its corresponding interval.

In the discussion of the algorithms below, we sometimes treat realizations of 1.5D terrains directly as x-monotone polylines. We sometimes refer, with some abuse of notation, to the *vertices* and *edges* of the realization. In this context, we call a vertex *left-turning* when this polyline, seen from left to right, turns to the left (upwards). Similarly, we call the vertex *right-turning* if it turns to the right (downwards), and *straight* if it does not change direction. We call an edge *left-turning* if both its endpoints are left-turning vertices, *right-turning* if both its endpoints are right-turning, or a *saddle* edge if one of its endpoints is left-turning and the other is right-turning.

In a realization of a 1.5D terrain, we say that a vertex is *external* if it lies on one of the endpoints of its imprecision interval, and *internal* otherwise.

The external vertices of a realization partition it into a sequence of *chains*: each chain starts at an external vertex, then goes through a (possibly empty) sequence of internal vertices, and finally ends at an external vertex again (see Fig. 2(a)). The leftmost and rightmost chains might be only *half-chains* if the leftmost or rightmost vertices of the terrain are not external.

Chains can be further subdivided into *subchains*: a subchain consists of only left-turning or right-turning vertices (Fig. 2(b)). Subchains are connected by saddle edges into chains. Chains are connected by shared vertices into the final terrain.

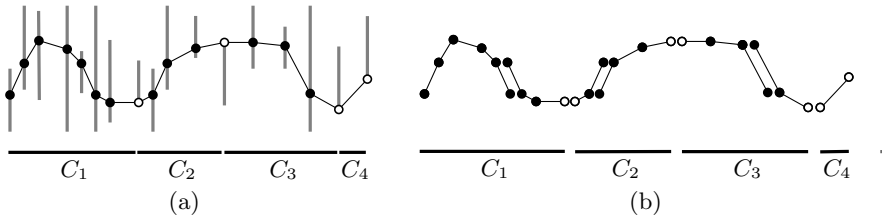

Fig. 2. (a) Division of a realization of terrain into chains $\{C_1, \cdots, C_4\}$ by external vertices (white circles). (b) Division of chains into subchains. Consecutive subchains in the same chain share a saddle edge.

3 Minimizing the Largest Turning Angle

In this section, we consider the problem of minimizing the largest angle in the terrain. Computing the optimal terrain under this measure exactly is difficult, since it involves computing the inverse of a non-algebraic function that cannot be computed under the real-RAM model commonly used in computational geometry. Therefore we present an algorithm to compute an approximate solution. In fact, we present an algorithm for a somewhat more general problem: we minimize the complete sorted vector of turning angles in lexicographical order. For a given terrain T, we denote this vector by $V(T)$. Figure 3 shows an example of a path that minimizes the angle vector $V(T)$. It can be shown that the solution is unique, unless it consists of all vertices on a straight line.

A solution T is said to be an ε-approximation of the optimal solution T^* if its vector of angles V is at most ε larger at every position, that is, if $V_i(T) \leq V_i(T^*) + \varepsilon$. Our solution incorporates approximation in different stages; for this purpose we define a smaller value $\varepsilon' = \varepsilon/4$. We also define $k = \lceil \frac{\pi}{2\varepsilon'} \rceil$; the algorithm relies on dividing the possible directions of edges in the terrain into k sectors, and on subdividing the terrain into independent pieces of length $O(k)$. Finally, we define $\delta = \varepsilon'/3k$. This is the factor to which we must approximate certain portions of the problem that appear later.

We start by defining *subproblems*. In a subproblem, the objective is to smooth only a certain portion of the imprecise terrain. Our algorithm solves many small

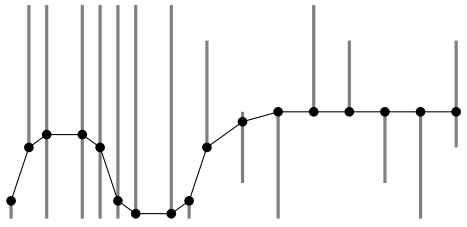

Fig. 3. Optimal terrain for min max angle

subproblems, and tries to combine them into a global solution. We define a subproblem $\mathsf{S}(p_i, d_i, p_j, d_j)$ on two fixed points p_i on the ith interval, and p_j on the jth interval, and two *sectors* d_i and d_j, which bound the possible directions in which the solution may leave p_i and enter p_j. Within this, we want to compute the locally optimal terrain $T^*(p_i, d_i, p_j, d_j)$ that leaves p_i in a direction from d_i, enters p_j in a direction from d_j, and otherwise optimizes the sorted vector of angles.

In the subproblems we consider, p_i and p_j are always external vertices, except for possibly p_1 and p_n. Furthermore, d_i and d_j are restricted to be one out of k possible sectors. If $T^*(p_i, d_i, p_j, d_j)$ has no external vertices other than p_i and p_j, we call it *free*. In this case we also call the subproblem $\mathsf{S}(p_i, d_i, p_j, d_j)$ free. We call a subproblem *δ-free* if it is free and any δ-approximation of the optimal solution is also free. We show that there is a good approximation of the optimal solution T^* that can be split into short δ-free subproblems, and provide an algorithm to efficiently compute it.

3.1 Properties of Optimal Terrains

First, we go over some properties of the shape of optimal terrains. Such terrains tend to have many consecutive small turns in the same direction, resulting in a smooth polyline. Let T^* be an optimal terrain for some subproblem. Let C^* be some chain of T^*. We establish a number of lemmata that should be reasonably intuitive, the formal proofs of which can be found in the full version [6].

Lemma 1. *C^* has at most two subchains.*

If a chain has more subchains, we could make a shortcut somewhere. Because of this lemma, every optimal chain has at most one saddle edge.

Lemma 2. *All vertices of C^* have the same turning angle, except possibly for one of the vertices of the saddle edge, which can have a smaller angle.*

If we have any other realization C of the same part of the imprecise terrain, then we can *morph* C to C^*: we can continuously deform the terrain such that the vector of turning angles $V(C)$ only decreases (lexicographically).

Lemma 3. *Any chain C can be morphed into the optimal chain C^*.*

This implies that there cannot be any locally optimal terrains that are not globally optimal. Because of this, we have the following important observation, which states that the imprecision intervals do not matter for internal vertices.

Lemma 4. *Given fixed directions for the first and last edges of a chain C^*, if we replace the imprecision intervals of the internal vertices of C^* by vertical lines (that is, if we allow any y-coordinate for each vertex), then C^* is still the optimal chain.*

Finally, we observe the following property, related to approximation. It states that any long chain can be replaced by a number of shorter chains, without damaging the vector of turning angles too much. Recall that $k = \lceil \frac{2\pi}{\varepsilon} \rceil$. Figure 4 gives the intuition behind the proof.

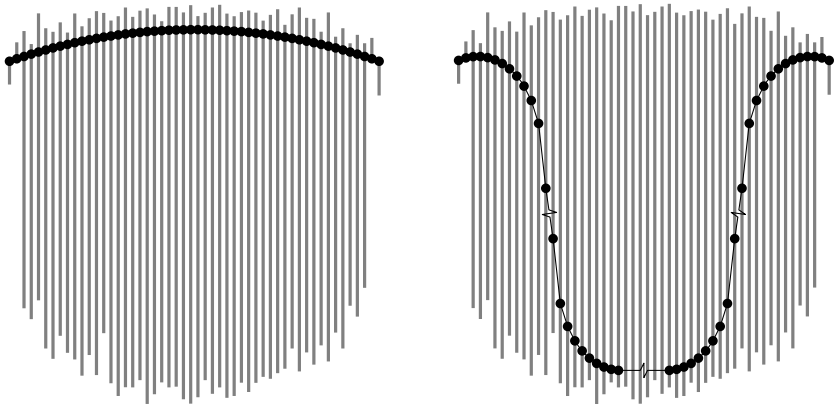

Fig. 4. (a) The optimal solution. (b) An ε-approximation. The middle part can be moved down as far as we want, since the turning angles at the saddles can never become worse than ε.

Lemma 5. *Suppose C^* consists of more than $3k$ internal vertices. Then there exists a sequence of j chains $C_1, C_2, \ldots, C_j$ such that each C_i has at most $2k$ vertices, C_1 starts at the same external vertex and in the same direction as C^* and C_j ends at the same external vertex and in the same direction as C^* such that the vector of angles $V(C_1 \cup C_2 \cup \ldots \cup C_j)$ is at most ε worse than $V(C^*)$.*

3.2 Solving the Subproblems

We are now ready to give an algorithm to solve the subproblems. If a subproblem is free, its optimal solution consists of only one chain, and by Lemma 1 this has at most two subchains. Because of the algebraic difficulties described in the beginning of this section, we cannot solve a subproblem optimally. However, we can approximate it arbitrarily well. We present a δ-approximation algorithm for free subproblems that runs in $O(m \log \frac{1}{\delta})$ time, where m is the number of vertices in the subproblem. The solution relies on a binary search on the worst angle, and on the following decision algorithm.

Given a value θ, we ask the decision question: is there a solution terrain T for this subproblem that uses only angles of θ or less? To decide this, we try to construct a solution that consists of a curve from p_i and a curve from p_j with consecutive angles of θ, and a straight line segment to connect them. We can choose whether the terrain starting from p_i turns left or right, and the same for p_j, resulting in four different combinations. We test them all. For such a combination, we still have some freedom in which direction (from the sector d_i) we leave p_i exactly. This results in a subinterval of the $(i+1)$th vertical line. Here we make a, say, right turn of θ. Then we reach the $(i+2)$th vertical line at a different subinterval, and so on. Figure 5 shows two examples of the same

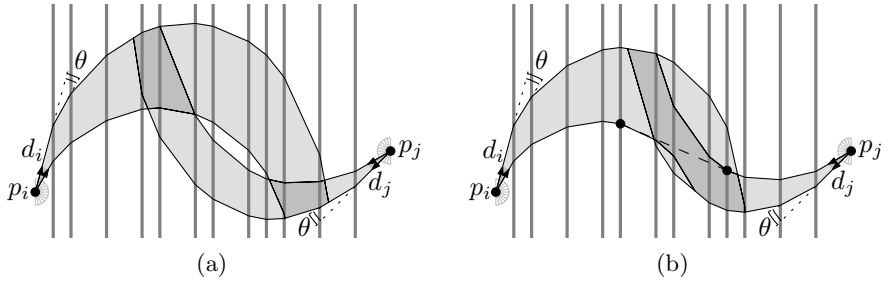

Fig. 5. (a) This value of θ is not feasible. (b) This value of θ is feasible. The dotted line shows the tangent between the two inner curves that realizes a terrain with worst angle θ.

subproblem for different values of θ. We call the regions between the curves that begin at extreme directions *horns*.

To test feasibility, consider the inner curves of both horns. If they intersect, θ is not feasible. If not, we compute their *inner tangent*. The two curves have two bitangents; we use the one with tangent points closest to p_i and p_j, see Figure 5. If this tangent stays within both horns, θ is feasible. If not, it is not feasible (we test feasibility for all combinations of left-turning and right-turning paths—and if any is feasible, θ is feasible). The reason is that the tangent touches the curves at vertices, thus it directly gives a correct solution. On the other hand, if there exists a solution with worst angle θ, then we find it, since there is always a solution with this shape. Define the *shortest θ-terrain* as the shortest terrain with worst turning angle θ. Then this terrain has turning angles of exactly θ at its ends, and a straight part in the middle.

Using this decision routine, a simple binary search on θ yields the following result:

Lemma 6. *Given a free subproblem $\mathsf{S}(p_i, d_i, p_j, d_j)$, we can approximate the locally optimal solution $T^*(p_i, d_i, p_j, d_j)$ of this problem within a factor δ in $O((j-i)\log\frac{1}{\delta})$ time.*

It is important to note that even though the subproblem we are solving (and therefore its optimal solution) is free, this does not guarantee that the approximate solution we compute stays within the imprecision intervals. In other words, the returned solution might be invalid. However, our algorithm is guaranteed to produce a valid solution for a δ-free problem.

3.3 Main Algorithm

Now we are ready to describe the algorithm for the original problem. The algorithm itself is very simple; the correctness analysis is more involved.

Each subproblem $\mathsf{S}(p_i, d_i, p_j, d_j)$ is defined on two points and two sectors. Consider all subproblems where $j - i \leq 3k$. There are $O(k^3 n)$ such problems,

since there are n possible choices for i, $3k$ possible choices for j, 2 possible choices for both p_i and p_j (each can be either at the top or at the bottom of its interval), and k possible choices for both d_i and d_j. By Theorem 6, each subproblem can be approximated within an error of δ in $O(k \log \frac{1}{\delta})$ time, because $j - i \leq 3k$. We approximate all subproblems of size at most $3k$ within a factor of $\delta = O(\varepsilon^2)$, so we spend $O(nk^4 \log k)$ time in total. Next, we discard any solutions that do not respect the imprecision intervals (we establish that those subproblems are not δ-free). Then we invoke dynamic programming to compute the best concatenation of subproblems, processing them from left to right and computing for each position (i, p_i, d_i) the best solution so far by minimizing over all possible placements of the previous point. We claim that the resulting terrain is an ε-approximation for the original problem; the analysis is done in the next subsection.

Note that the optimal terrain does not need to have external first and last vertices. Thus if $j < 3k$, it is possible that there is no previous external interval. To solve these special subproblems, where $i = 1$ or $j = n$, we exploit a property about the beginning and end of an optimal terrain. In particular, we prove in the full version that if the optimal terrain does not start with an external vertex, then all the vertices between the first vertex and the first external vertex lie on a straight line. A symmetric argument holds for the last vertex.

Using this property, the algorithm solves the subproblems from the beginning (internal) interval to v_j, with ending direction sector d_j, by checking if there exists a line from v_j that hits the first interval and stays within all intermediate intervals. If that is not the case, the subproblem can be discarded because we know there must be some other subproblem that contains the line in an optimal solution. Checking if such a line exists can be done in linear time.

3.4 Correctness Analysis

Let T^* be the optimal terrain. We argue the existence of several other terrains, each of which is a close approximation of T^* and has certain properties. Eventually, we establish that one of the terrains in the class that we encounter in the algorithm is also a close approximation of T^*. By Lemma 5, we know that there exists a different terrain S that approximates T^* within ε' such that all chains of S are at most $3k$ long. Let S^* be the optimal terrain among all terrains for which all chains are at most $3k$ long. Clearly, S^* also approximates T^* within ε'.

The terrain S^* is partitioned into short chains. For each external vertex in S^*, we divide its circle of directions into $2k$ sectors, and we locate its two outgoing edges in this sector set. We fix these sectors. Observe that any terrain C that respects these sectors is within an error of ε' from S^* at these vertices. Therefore, if we take the terrain apart into a sequence of independent subproblems, each restricted by two vertices and sectors, and we solve each subproblem optimally, this results in a terrain C^* that is an ε'-approximation of S^*. These optimal subsolutions may again have external vertices at places where S^* had none. In this case, we again fix these vertices and their sectors, and subdivide the terrain

further. Each vertex gets fixed at most once, and in the other steps can only get a better turning angle, so the terrain remains an ε'-approximation of S^*.

Eventually we reach a terrain D with the property that D is a concatenation of smaller subproblems, each of length at most $3k$ such that each subproblem, defined on a pair of vertices and a pair of directions, is locally optimal and free. Furthermore, D is within ε' from S^*. Let D^* be the optimal terrain with these properties. Then also D^* is within ε' from S^*, and therefore within $2\varepsilon'$ from T^*. Next, we show that there exists a terrain E that is a concatenation of δ-free subproblems, that approximates D^* within $2\varepsilon'$. Our algorithm encounters and approximately solves all δ-free subproblems, so it computes the optimal terrain E^* of this form. E^* is better than E, so it also approximates D^* within $2\varepsilon'$, and therefore T^* within ε.

Let $\mathsf{S}(p_i, d_i, p_j, d_j)$ be a free subproblem, and let R^* be its optimal solution. We define the *error* of a vertex in a terrain as the difference between its turning angle and the corresponding turning angle in T^*. We define the error vector $e(p_i, d_i, p_j, d_j)$ as the sorted vector of errors in R^*.

Lemma 7. *If a free subproblem $\mathsf{S}(p_i, d_i, p_j, d_j)$ has an error e, then it is either δ-free, or there exists a sequence of smaller subproblems $\mathsf{S}(p_{i_0}, d_{i_0}, p_{i_1}, d_{i_1})$, $\mathsf{S}(p_{i_1}, d_{i_1}, p_{i_2}, d_{i_2})$, ..., $\mathsf{S}(p_{i_{h-1}}, d_{i_{h-1}}, p_{i_h}, d_{i_h})$, where $i_0 = i$ and $i_h = j$, that are all free and have errors of at most $e + \delta$, and such that the turning angles of their external vertices have an error of at most $e + \varepsilon'$.*

This lemma implies that we can actually split a subproblem into *δ-free* subproblems, not just into free problems, at the cost of a slightly worse error vector.

Corollary 1. *If a free subproblem $\mathsf{S}(p_i, d_i, p_j, d_j)$ of length m has an error e, it is either δ-free, or there exists a sequence of smaller subproblems $\mathsf{S}(p_{i_0}, d_{i_0}, p_{i_1}, d_{i_1})$, $\mathsf{S}(p_{i_1}, d_{i_1}, p_{i_2}, d_{i_2})$, ..., $\mathsf{S}(p_{i_{h-1}}, d_{i_{h-1}}, p_{i_h}, d_{i_h})$, where $i_0 = i$ and $i_h = j$, that are all δ-free and have errors of $e + m\delta$, and such that the turning angles of their external vertices have an error of at most $e + (m-1)\delta + \varepsilon'$.*

Now, consider D^*. We construct the terrain E by replacing each free subproblem in D^* by a sequence of δ-free subproblems, according to Corollary 1. The errors in E may have gotten at most $3k\delta + \varepsilon'$ worse than in D^*. If we set $\delta = \frac{\varepsilon'}{3k}$, this is exactly what we need.

Theorem 1. *Given an imprecise 1.5D terrain, a realization of the terrain that minimizes the vector of turning angles lexicographically can be computed approximately within an error of ε in $O(\frac{n}{\varepsilon^4} \log \frac{1}{\varepsilon})$ time.*

4 Maximizing the Smallest Turning Angle

In this section, we discuss a different variant of the problem: instead of minimizing the largest angle, we *maximize* the *smallest* angle. This results in a terrain that is "as rough as possible" given the imprecision constraints. An example is shown in Fig. 6. While this is not something one would want to do in practice, it does give some insight into the worst case for the terrain at hand.

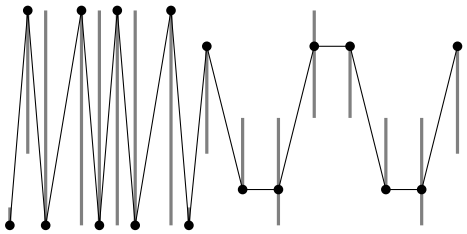

Fig. 6. Optimal terrain for max min angle.

As the example shows, the optimal solution can use internal vertices, because moving such a vertex in either direction decreases one of the neighboring vertices. In fact, there could be multiple consecutive such internal vertices, that all keep each other in balance. Therefore, as in Sect. 3, we cannot solve the problem exactly, but we can solve it approximately in roughly the same way. Here we just list the differences; we give a more detailed description in the full version.

Because of the nature of the problem, there are some important differences. When maximizing the smallest angle, an optimal terrain will try to zig-zag as much as possible, as opposed to the optimal terrain in the previous section, which tries to avoid zig-zagging. This actually makes the problem easier: whenever the terrain is able to zig-zag, it will use external vertices, which results in many very short chains without internal vertices. However, there are situations where zig-zagging is not possible, and the optimal terrain must use internal vertices to balance the angles, as is shown in Fig. 7.

Such chains with internal vertices can still have at most two subchains, and still have the same turning-angle almost everywhere. However, such a situation is only locally optimal: if we would move the vertices a small distance, the solution would get worse, but if we would move them a large distance, the solution would get better again. This means that Lemma 4 does not hold here. However, we can still compute (δ-approximations of) the locally-optimal chains.

Some parts of the algorithm become simpler. We do not need to split long chains into shorter chains anymore, since any chain that is longer than $2k$ must have turning angles of less than ε. But if this is the case in the optimal

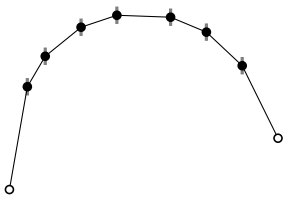

Fig. 7. A chain. Moving any vertex down makes its own angle smaller, moving it up makes its neighbors' angles smaller. Note however, that if we would be able to move a vertex very far down (or up), all angles involved would get larger.

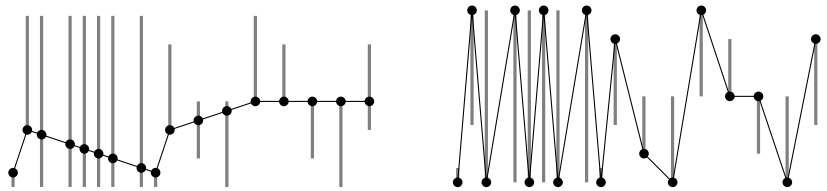

Fig. 8. (a) A terrain minimizing the sum of its turning angles. (b) A terrain maximizing the sum of its turning angles.

solution, since we are maximizing the smallest angle, any terrain would be an ε-approximation. Also, the leftmost and rightmost vertices are always external here, which simplifies the situation.

Other than these small differences, the same algorithm works. We can still solve the subproblems in $O(m \log \frac{1}{\delta})$ time, although the actual conditions become a bit different (a vertex of the saddle edge should now have a turning angle that is *larger* than the rest, rather than smaller). The dynamic programming based on the subproblems goes through unchanged.

Theorem 2. *Given an imprecise 1.5D terrain, a realization of the terrain that maximizes the vector of turning angles lexicographically can be computed approximately within an error of ε in $O(\frac{n}{\varepsilon^4} \log \frac{1}{\varepsilon})$ time.*

5 Minimizing and Maximizing the Total Turning Angle

Another measure of smoothness or roughness that we may wish to optimize is the sum of the turning angles of a terrain. Figure 8 shows the optimal terrain in these cases for the same example that was used in the previous sections. We show that the terrain with the minimum sum of turning angles is related to the shortest path that respects the intervals. We use this observation to design a linear-time algorithm that computes a terrain that minimizes this sum. We also observe that the terrain that maximizes the total turning angle has the property that all the vertices are external. This property allows us to design a linear-time algorithm to find such a terrain.

We begin by looking at the terrain that minimizes the total turning angle. The first observation that we can make is that the total turning angle of a left-turning (or right-turning) path between two vertices depends only on the starting and ending directions, and is the turning angle between those directions. In other words, the location of the vertices in a left-turning chain does not matter as long as the entire chain remains left-turning. This implies that there are many terrains that minimize the total turning angle.

Moreover, we observe that if the leftmost and rightmost intervals were single points, the shortest path between those points through the corridor defined by

the upper and lower endpoints of the intervals would be an optimal solution. Our algorithm, the details of which are omitted, is to find this shortest path and then fix the ends of the terrain by making them as straight as possible in a relatively simple procedure. We prove in the full version:

Theorem 3. *Given an imprecise 1.5D terrain, a realization of the terrain that minimizes the total turning angle can be computed in linear time.*

The observation that all the vertices of a path that maximize the total turning angle must be external also leads us to a simple algorithm. We can use dynamic programming to find the best terrain from left to right. We show in the full-version:

Theorem 4. *Given an imprecise 1.5D terrain, a realization of the terrain that maximizes the total turning angle can be computed in linear time.*

6 Discussion and Future Challenges

We studied several measures to compute the smoothest or roughest possible 1.5D terrain, when height information is imprecise. We presented approximation algorithms that run in linear time for optimizing the worst turning angle in the terrain (either smallest or largest). They find a terrain where the worst turning angle is at most ε away from the one in the optimal terrain, for any $\varepsilon > 0$. We highlight that the depth of the algorithms lies in the correctness analysis, and not in the algorithms themselves, which are relatively simple and should be easy to implement. As a supplement to these results, we also studied algorithms for optimizing the total turning angle. We sketched two exact algorithms that also run in linear time. These algorithms should also be fairly simple to implement.

Clearly, the major open problem suggested by this work is to smooth 2.5D terrains, which are encountered more often in practice. Such terrains pose challenges on two different levels. On the modeling level, it is unclear how to define the problem correctly—it is difficult to define a smooth terrain in a way that ensures that all the features are smooth. For example, even if all of the solid angles between faces of the terrain are small, a peak can be created at the intersection of three or more faces that is quite sharp. Even when this would be clear, though, designing efficient algorithms is still challenging. For example, fitting a smooth terrain through a few known fixed points, when the remaining points have no bounded intervals, is already a non-trivial task. With the algorithms presented in this paper, we show that the 1.5-dimensional case is already more challenging than it looks at first; one cannot expect the 2.5-dimensional case to be any easier. At the same time, we hope that our solutions will provide the necessary insight required to solve these problems eventually.

A tool recently introduced in the analysis of terrains is the *realistic terrain* model proposed by Moet *et al.* [10]. In this model, restrictions are placed on four properties of the triangles of a terrain. These four properties are the minimum angle of each triangle, the ratio of the size of the largest triangle to the size of the

smallest triangle, the aspect ratio of the bounding rectangle of the terrain, and the steepness of each triangle. In all cases, the properties are restricted to be a constant. Most of these restrictions have to do with the underlying triangulation of the terrain. However, the restriction that the steepness of any triangle in the terrain is bounded deals directly with the heights of the vertices of the terrain. We wonder whether the first three restrictions make the last any easier to satisfy in an uncertain terrain. That is, given an imprecise terrain whose triangulation is "realistic", can an algorithm set the heights of the vertices in such a way that the steepness of the steepest triangle is minimized? This question, among many others, is interesting to study in the context of imprecise terrains.

References

1. Ben-Moshe, B., Katz, M.J., Mitchell, J.S.B.: A constant-factor approximation algorithm for optimal 1.5d terrain guarding. SIAM Journal on Computing 36(6), 1631–1647 (2007)
2. Dyn, N., Levin, D., Rippa, S.: Data dependent triangulations for piecewise linear interpolation. IMA Journal of Numerical Analysis 10, 137–154 (1990)
3. Feciskanin, R.: Optimization of triangular irregular networks for modeling of geometrical structure of georelief. In: Proc. International Cartographic Conference 2007 (2007)
4. de Floriani, L., Magillo, P., Puppo, E.: Applications of computational geometry in Geographic Information Systems. In: Sack, J., Urrutia, J. (eds.) Handbook of Computational Geometry, pp. 333–388. Elsevier, Amsterdam (1997)
5. Gray, C., Evans, W.: Optimistic shortest paths on uncertain terrains. In: Proc. 16th Canadian Conf. Comput. Geom., pp. 68–71 (2004)
6. Gray, C., Löffler, M., Silveira, R.I.: Smoothing imprecise 1.5D terrains. Tech. Rep. UU-CS-2008-036, Department of Information and Computing Sciences, Utrecht University (2008)
7. Hofer, M., Sapiro, G., Wallner, J.: Fair polyline networks for constrained smoothing of digital terrain elevation data. IEEE Trans. Geosc. Remote Sensing 44, 2983–2990 (2006)
8. Kholondyrev, Y., Evans, W.: Optimistic and pessimistic shortest paths on uncertain terrains. In: Proc. 19th Canadian Conf. Comput. Geom., pp. 197–200 (2007)
9. King, J.: A 4-approximation algorithm for guarding 1.5-dimensional terrains. In: Correa, J.R., Hevia, A., Kiwi, M. (eds.) LATIN 2006. LNCS, vol. 3887, pp. 629–640. Springer, Heidelberg (2006)
10. Moet, E., van Kreveld, M., van der Stappen, A.F.: On realistic terrains. In: Proc. 22nd Annu. ACM Sympos. Comput. Geom., pp. 177–186 (2006)
11. Silveira, R.I., van Oostrum, R.: Flooding countries and destroying dams. In: Dehne, F., Sack, J.-R., Zeh, N. (eds.) WADS 2007. LNCS, vol. 4619, pp. 227–238. Springer, Heidelberg (2007)
12. Tasdizen, T., Whitaker, R.T.: Feature preserving variational smoothing of terrain data. In: Proc. 2nd International IEEE Workshop on Variational, Geometric and Level Set Methods in Computer Vision (2003)
13. Wang, K., Lo, C.P., Brook, G.A., Arabnia, H.R.: Comparison of existing triangulation methods for regularly and irregularly spaced height fields. International Journal of Geographical Information Science 15(8), 743–762 (2001)