Smoothing Imprecise 1.5D Terrains

Chris Gray

Maarten Löffler

Rodrigo I. Silveira

Technical Report UU-CS-2008-036
November 2008

Department of Information and Computing Sciences
Utrecht University, Utrecht, The Netherlands
www.cs.uu.nl

ISSN: 0924-3275

Department of Information and Computing Sciences
Utrecht University
P.O. Box 80.089
3508 TB Utrecht
The Netherlands

Smoothing imprecise 1.5D terrains *

Chris Gray† Maarten Löffler‡ Rodrigo I. Silveira‡

Abstract

We study optimization problems for polyhedral terrains in the presence of data imprecision. 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 terrains with a one-dimensional projection, usually referred to as 1.5-dimensional terrains, where an imprecise terrain is given by an x-monotone polyline, and the y-coordinate of each vertex is not fixed but only constrained to a given interval. Motivated mainly by applications in terrain analysis, in this paper we study five different optimization measures related to obtaining smooth terrains, for the 1.5-dimensional case. In particular, we present exact algorithms to minimize and maximize the average turning angle, as well as to minimize the maximum slope change. Furthermore, we also give approximation algorithms to minimize the largest turning angle and to maximize the smallest 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 [7]). 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 [26]. However, algorithms in computational geometry often assume that the height values are precise. Besides not necessarily representing the real terrain, this may lead to artifacts that compromise the reliability of the terrain model.

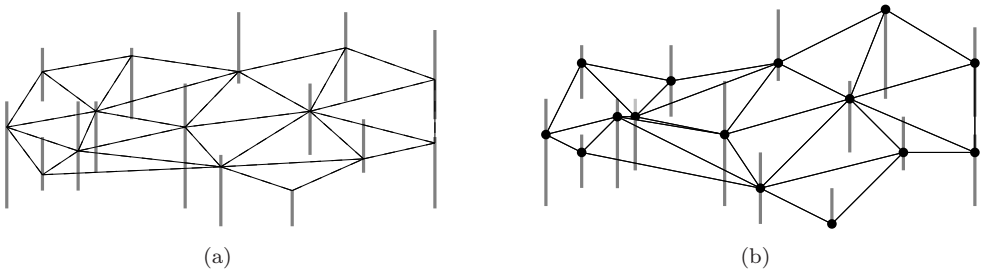
1.1 Imprecise terrains

Data imprecision is an important issue in computational geometry. In recent years there has been a growing interest in modeling imprecision in an exact way. An early model is *espilon*

*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



(a) (b)

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

geometry, introduced by Guibas *et al.* [12]: here the input is a set of points, but each point has an imprecision radius of ε. Abellanas *et al.* [1] study the *tolerance* of a geometric structure: the largest perturbation of the vertices such that the combinatorial structure remains the same. Bandyopadhyay and Snoeyink [4] compute the set of "almost-Delaunay simplices," which are the tuples of points that could define a Delaunay simplex if the entire point set is perturbed by at most $\varepsilon > 0$. Khanban and Edalat [15] develop a theory of computing partial structures on imprecise points, based on answering predicates with three possible answers: *yes*, *no*, or *maybe*. Löffler and Van Kreveld [20] consider the problem of determining the smallest and largest possible values for several geometric extent measures—such as the diameter or convex hull area—of a set of imprecise points.

More specifically focussed on imprecision in terrain data, Gray and Evans [11] 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 realization of the real terrain. Kholondyrev and Evans [16] also study this model. Silveira and Van Oostrum [25] 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 [11, 16]; each vertex has a height interval. 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.

Considering only the height information as imprecise can be seen as a simplifying assumption, since it can be argued that (x, y)-data is also inherently imprecise. However, assuming imprecision only in the height information is reasonable for many applications. Elevation data is often obtained from a different source than the (x, y)-data. For example, when laser altimetry techniques are used, the height information is obtained from measuring the time needed for a laser beam sent from an airplane to touch the Earth's surface and go back, whereas the (x, y)-data usually comes from a GPS device located on the airplane. Hence the imprecision in the elevation is completely independent from the one of the (x, y)-coordinate data. As a second example, consider a terrain model based on contour maps (that is, isolines). Any point between two consecutive contours has (rather) exact (x, y) coordinates, whereas the z-coordinate can only be obtained approximately by considering the heights of the two contours, giving rise to an imprecision interval.

The large number of different realizations of an imprecise terrain 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. 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, like for water run-off simulation. When simulating the way water flows through a terrain, artificial pits create artifacts where water accumulates, affecting the simulation. Therefore for the purpose of using the model for water run-off simulation, minimizing the number of pits is a criterion of interest.

In this paper, we study the problem of smoothing imprecise terrains. Many other interesting criteria to optimize exist, and to our knowledge, almost none have been studied in depth before. However, before concentrating on the smoothing problem, it is worth noting that we can deal with the second criterion mentioned, minimizing the number of pits, in an efficient and simple way. A *pit* or *local minimum* is a vertex that has no lower neighbors. If several such vertices are connected, we consider the whole group as one local minimum. As mentioned before, minimizing the number of local minima is important for some uses of terrain analysis, in particular for hydrologic applications. It is easy to see that a realization of an imprecise terrain that minimizes the number of local minima can be found by beginning with all the vertices as low as possible, and then lifting each local minimum until it disappears or until it cannot be lifted any further. As we will see soon, such simple solutions do not seem possible for other criteria like smoothing terrains.

1.2 Smooth terrains

From now on we focus on trying to obtain a *smooth* terrain, respecting the height intervals. There are several reasons to be interested in smooth terrains. A first one, already mentioned, is because some additional information about the topography of the terrain is known (that is, there are not too many sharp ridges in that area). Other reasons include visualization (smooth shapes usually look better), compression, and noise reduction. Finally, in terrain classification it is useful to have a measure of the smoothness of a certain piece of terrain: in an imprecise context, this means we are interested in tight bounds on the possible smoothness (determined by the smoothest and least smooth possible terrains that respect the imprecision).

Smoothing of terrains has been previously studied in the context of grid terrains [13, 26], where techniques from image processing can be applied. Smoothing TINs is a less well-studied topic, but still some results are known [14, 27]. These techniques aim at smoothing a terrain in general, trying to preserve the global shape but removing local "noise". However, there is no guarantee that in the process no important features of the terrain where removed.

When imprecision about the terrain is known, it is natural to ask for the smoothest possible terrain that respects the imprecision intervals. 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 height 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. Measures on spatial angles, in particular the angle between the surface normals of adjacent triangles, are known to be important for several GIS applications. In particular, they have been shown to be good for improving the quality of the approximation, for achieving good slope characteristics and for flow modeling [8, 9, 28].

1.3 1.5-Dimensional terrains

A *1.5D terrain* is the lower-dimensional analogue of a 2.5D terrain. It consists of a set of vertices whose vertical projections lie on a line, and consecutive vertices are linked with edges. In other words, it is an x-monotone polyline in the plane. Such 1.5D terrains are often studied to simplify problems on 2.5D terrains, but also have direct applications. Recently they have been studied in relation to guarding problems [5, 17, 18].

Here we study the smoothing problem on 1.5D terrains. The main reason is that, even though the 2.5D version is clearly more interesting, a simpler model is easier to handle and gives considerable insight into the difficulties of 2.5D terrains. As will become clear later, this restricted model is still challenging enough, and hopefully, the results can serve as building blocks for the 2.5D version. Some more comments on possible extensions are given in Section 8.

To smooth 1.5D imprecise terrains, we study several measures, but most importantly we try to minimize the largest turning angle at any vertex of the terrain. This will result in a smooth surface, without any sharp turns. This measure is discrete in nature; its continuous analogue would be to fit an x-monotone curve through the intervals, such that the maximum curvature of the curve in minimized. This problem is related to e.g. signal processing or non-holonomic robot motion planning [6]. The main difference is that by studying a discrete measure directly, we obtain a discrete terrain with provable guarantees, while when translating a continuous curve back to a discrete one it is not clear what properties will be preserved. It is worth mentioning that several discrete notions of curvature have been defined for polyhedral surfaces involving angles between triangles and vertices. Moreover, a notion of *concentrated curvature* for polygonal lines has been recently introduced [22], where the curvature at a vertex is defined by its turning angle. We further note that the discrete nature of these measures allows us to have a non-uniform sampling of the terrain, with more samples where the terrain is more curved. A consequence of this, though, is that the solution is not invariant under additional measurements.

1.4 Contribution and structure

This paper studies five different problems. The most interesting one, and also the most involved one, is minimizing the largest turning angle. This problem turns out to be hard from an algebraic point of view, therefore we propose an (additive) ε-approximation algorithm, for any $\varepsilon > 0$, which runs in linear time. We also study its counterpart to produce rough terrains: maximizing the smallest turning angle. Such a measure could be useful for modeling terrains that are known to have rough topography, but more importantly to get an idea of the worst possible case, if smoothness is the goal. For this problem we also give an ε-approximation algorithm, which is based on similar properties as the one for minimizing the largest angle. It is worth mentioning that even though the analysis of these two algorithms is rather involved, the algorithms themselves do not require any complicated technique or data structure, and should be relatively easy to implement in practice.

Since we are not able to solve exactly the optimization of the worst angle, we also study three other measures which also reflect the idea of smoothness (or roughness), but can be optimized exactly. Closely related to optimizing the worst (maximum/minimum) angle, we consider minimizing (or maximizing) the total turning angle. This is another natural criterion that deals with turning angles, which has been studied before in the context of bicriteria path problems [2]. Moreover, we also consider minimizing the maximum slope change of the edges. This also results in smooth terrains, although this measure behaves differently than the ones about angles. In particular, when edges can be very steep, the slopes grow towards infinity, and a small change in the turning angle at one of the vertices of such an edge can result in a huge change in the slope. Despite this, in some situations, for example when edges are known to have gentle slopes or when we want to discourage steep edges, such an alternative measure can be useful. For these three measures we give exact algorithms. The first two, for optimizing the total turning angle, run in linear time, whereas the third one, for minimizing the worst slope change, runs in quadratic time.

This paper is structured as follows. We begin in Section 2 by introducing some definitions and making some observations about 1.5D terrains. In Section 3 we present our main result: an ε-approximation algorithm to minimize the largest turning angle. Then, in Section 4 we study its counterpart to maximize the smallest turning angle. The next three sections are devoted to other related measures that can be optimized exactly. Section 5 deals with minimizing the sum of the turning angles of the terrain, and Section 6 with the maximization of the sum of turning angles. In Section 7 the goal is to minimize the maximum slope change. Finally, in Section 8 we give some conclusions and discuss directions for further research.

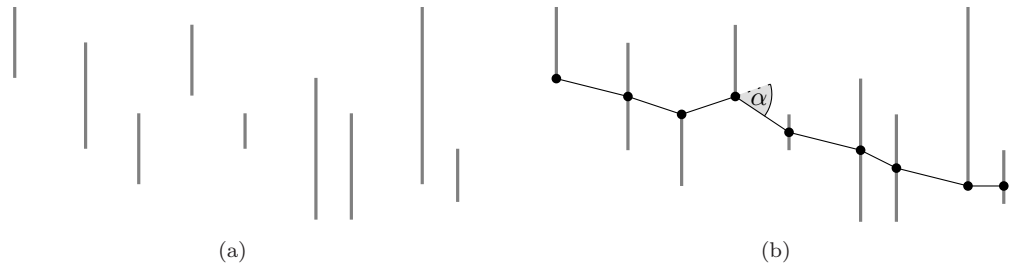


Figure 2: (a) An imprecise, 1.5D terrain. (b) A possible realization of the terrain, with the largest turning angle α highlighted in gray.

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. 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 2. In the discussion of the algorithms below, we sometimes treat realizations of 1.5D terrains directly as x-monotone polylines. Similarly, 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 other words, a saddle edge is the last edge of a sequence of consecutive left-turning edges, and – at the same time – the first edge of a sequence of right-turning edges (or vice versa). The two endpoints of a saddle edge are called *saddle vertices*. The notion of saddle edge can be generalized to a sequence of edges. A set of consecutive edges that are on a straight line with a left-turning first vertex and a right-turning last vertex (or vice versa) is called a *saddle group*.

When looking at a realization of an imprecise terrain, it will be useful to distinguish between two types of vertices: external and internal. 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 Figure 3(a)). The leftmost and rightmost chains might be only half-chains if the leftmost or rightmost vertices of the realization are not external. Chains can be further subdivided into *subchains*: a subchain consists of only left-turning or right-turning vertices (see Figure 3(b)). Subchains are connected by saddle edges into chains, and chains are connected by shared vertices into the final realization.

Finally, we note that for brevity, and when the context makes it clear, we will sometimes use

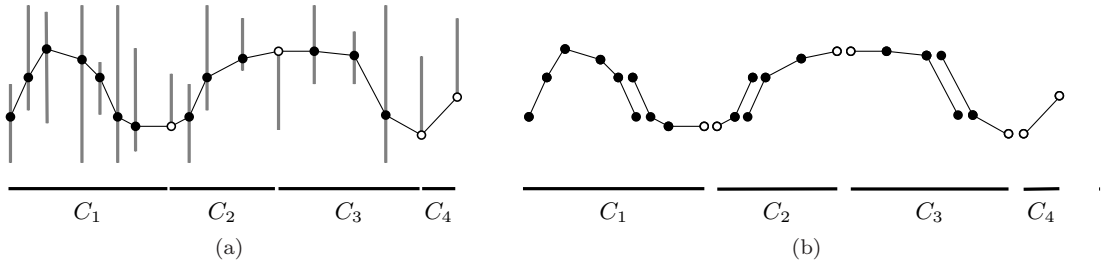


Figure 3: (a) Subdivision of a realization of terrain into chains $\{C_1, \cdots, C_4\}$ by external vertices (white circles). (b) Division of chains into subchains. Consecutive subchains in the same chain share a saddle edge.

terrain to refer to a *realization* of an imprecise terrain.

3 Minimizing the largest turning angle

In this section we consider the problem of minimizing the largest angle in the terrain. In fact, we will present an algorithm for a somewhat more general problem. Let T be a realization of the terrain. We denote by $V(T)$ the sorted vector of turning angles at the vertices of T. Angles along the vector occur from largest (first position) to smallest (last position). We will find the terrain T with the lexicographically smallest vector $V(T)$, that is, we will minimize the complete vector of turning angles $V(T)$, in lexicographical order. Figure 4 shows an example of a path that minimizes $V(T)$. It can be shown that the solution is unique, unless it consists of all vertices on a straight line.

However, computing the optimal terrain exactly is difficult due to algebraic reasons. To illustrate the difficulty, consider the following simpler subproblem. Suppose we have a part of the optimal solution that consists of several consecutive left-turning vertices with angles of θ (for $\theta \leq \pi/2$). Suppose that the leftmost point is fixed at the origin (and the previous point was on the negative x-axis), and the intervals are all vertical lines with x-coordinates x_i for $i = 1, 2, \ldots$. Then the height y_i at which the line at x_i is crossed is a function of θ with the following shape:

$$y_i(\theta) = y_{i-1}(\theta) + (x_i - x_{i-1}) \tan(i\theta) = \sum_{j=1}^{i} (x_j - x_{j-1}) \tan(j\theta) \qquad (*)$$

Figure 5 illustrates the situation. Even when the endpoint of such a chain is known, we need to compute the inverse function $\theta(y_i)$ to find the best possible angle. This is not possible under any reasonable model of computation, like the real-RAM model, standard in computational geometry.

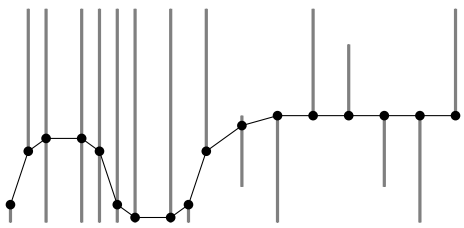


Figure 4: Optimal terrain for min max angle.

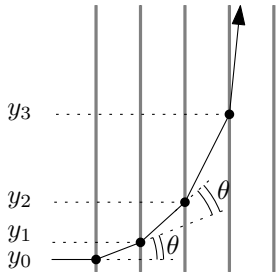


Figure 5: A simple case of our problem where the turn at each vertex is exactly θ. Even for this restricted case, algebraic difficulties arise.

We note that this type of algebraic hardness is present in several other computational geometry problems, especially in problems related to facility location [3].

Therefore, we present an algorithm to compute an approximate solution. 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 will incorporate 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 $O(k)$.

First, we discuss several properties of optimal terrains that we will need later. Then we present an algorithm for approximating a specific simplified problem. Finally, we show how to construct an algorithm for the original problem using this subroutine, and argue that this approximates the optimum within an error of ε. The final algorithm will run in $O(nk^4 \log k)$ time.

We define a subproblem $S(p_i, d_i, p_j, d_j)$ on two fixed points, p_i on the ith interval, and p_j on the jth interval (in fact, they will always lie on endpoints of their intervals), and two *cones*, 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 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.

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 $S(p_i, d_i, p_j, d_j)$ free. We will show later, in Lemma 3, that if a subproblem is free, we can ignore the imprecision intervals of its vertices, since the solution to the same subproblem with vertical lines instead of bounded intervals is the same. Free subproblems play an important role in our algorithm, and we will show later that we only have to be able to solve those. Unfortunately, we will not be able to solve free problems, but we will be able to solve *δ-free* problems instead. We call a subproblem *δ-free* if it is free and if any δ-approximation of the optimal solution is also free. This further step will finally allow us to show that we can obtain an ε-approximation of the optimal terrain.

3.1 Properties of chains

Recall that chains are parts of a realization between external vertices. Let T^* be an optimal realization of an imprecise terrain, possibly with extra conditions on the starting and ending angles (such as defined by subproblems). Any terrain, in particular T^*, is composed of chains. In this subsection we investigate some properties of chains of the optimal terrain. In what follows, C^* denotes some chain in T^*.

We begin by showing that chains of T^* cannot have many bends, but they are what we can call *S-shaped*.

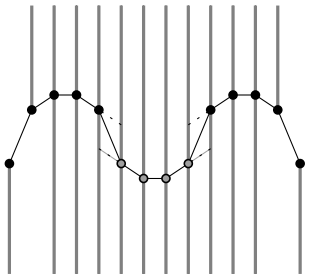


Figure 6: Moving the gray vertices upwards together does not increase any turning angle.

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

Proof: Suppose that the direction of curvature changes more than once along a chain. Because C^* is a chain, all its vertices (except for the first and last one) are allowed to move in either direction. Then take a part of consecutive vertices that all bend left (resp. right), while the neighbors bend right (resp. left). In Figure 6, such a part is marked by gray vertices. If we move this part as a whole up (resp. down), the only turning angles that change are the ones at the outermost vertices of this part and their neighbors (that is, the saddle vertices). But clearly all these angles only become smaller. This contradicts the assumption that C^* was part of the terrain with the optimal vector of turning angles. ⊠

Due to Lemma 1, we can see that every optimal chain has at most one saddle edge. Next, we show that all the vertices of an optimal chain, except possibly for one of the saddle vertices, have the same turning angle. For this we introduce the concept of *morphing* a chain. We say a chain C can be morphed into another chain C', if there exists a continuous transformation of C into C' such that the values in $V(C)$ only increase.

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

Proof: Recall that all vertices are internal, so we can move them both up and down a little bit. We will show that if the lemma is false, we can move some vertices over an infinitesimally small amount such that the value of the solution decreases. First we will show that each subchain only has vertices with the same turning angle, except for the saddle vertices which may have a smaller angle.

Let P^* be such a subchain. Suppose that not all vertices have the same turning angle, and let θ be the largest turning angle that occurs on P^*. Then there must exist a vertex v that is not a saddle vertex and that has an angle smaller than θ, and at least one neighbor w with an angle of θ. If v is left-turning, then so are its neighbors (since it is not a saddle vertex), so we can move v down by a small amount, which makes its angle worse but the angles of its neighbors better. Now w has a smaller angle, so the number of vertices with angle θ has decreased. Therefore the lexicographical value of P^* has decreased, which contradicts its optimality. Symmetrically, if v is right-turning, we can move it upwards a little bit.

Now suppose that C^* has two subchains with different turning angles: assume without loss of generality that it first has a left-turning subchain with angles of θ, and then a right-turning subchain with angles of $\phi < \theta$. If we decrease the turning angle of the first chain a little bit (by moving vertices down), but keeping the first two vertices of the subchain (the external vertex and the first internal vertex) where they are, the last vertex of the subchain (the saddle vertex) will move downwards. However, if we increase the turning angle of the second chain a little bit (also by moving vertices down), keeping the last two vertices of the subchain where they are, the first vertex of the subchain (the other saddle vertex) will also move downwards. We can balance the

changes so that the turning angles at the saddle vertices do not change. However, we decreased the worst turning angle of C^*, which contradicts its optimality.

Finally, suppose both saddle vertices have a turning angle smaller than θ. Then we can decrease the turning angle of both subchains, and the saddle vertices will move away from each other, increasing their turning angles until one of them becomes θ too, again contradicting optimality. ⊠

As a direct consequence of the arguments described in the proof, we can establish the following corollary.

Corollary 1 *Any chain C can be morphed into the optimal chain C^**

Proof: By Lemma 2, we can morph P into another path P' such that P' is better than P, but none of the vertices have moved more than an arbitrarily small distance δ. We can again keep deforming it until all angles are the same. Since we are decreasing the vector of angles all the time, this process will terminate in a unique situation: that of P^*. ⊠

Another observation we can make is that the shape of C^* does not depend on the intervals it goes through, except for the first and last one.

Lemma 3 *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.*

Proof: Suppose all internal intervals are infinite. Let P^* be the optimal path considering infinite intervals. Now let P be any other path. If P has more than 2 subchains, we can take any subchain other than the first and last one, and move it upwards or downwards as in Lemma 1. We can continue to do so this until the subchain is reduced to a line segment tangent to its neighboring subchains. Eventually, only 2 subchains will be left.

Now, by Corollary 1, we can morph P into P^*. To prove the lemma, suppose that C^* is different from P^*, and consider the original imprecision intervals again. In this case, let $P = C^*$. The only reason why we would not be able to morph it into P^*, is that at some point a vertex cannot move any further. But this only happens when it is on an endpoint of its imprecision interval, which contradicts the fact that C^* is a chain, because a chain has only internal vertices (except for its endpoints). ⊠

Another observation we can make is related to the approximation setting we are in. It states that any long chain can be replaced by a number of shorter chains, without worsening the vector of turning angles too much.

Lemma 4 *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 $V(C^*)$.*

Proof: Take a stretch of the optimal solution of at least $3k$ internal vertices. The angle at these vertices must be less than ε, because $k = \lceil \frac{2\pi}{\varepsilon} \rceil$. Now we will construct a different piece of the solution, with the same beginning and end as this stretch, with only angles of at most ε, which touches an endpoint of an interval. Then we can replace the original stretch by this new piece and no angle has increased by more than ε.

To define this new piece, we start at the left starting position, and make a right-turn of ε at each new segment. After at most k segments, the direction of the terrain will be less than ε from going vertically down. In the same way, going from right to left, we start at the right starting position

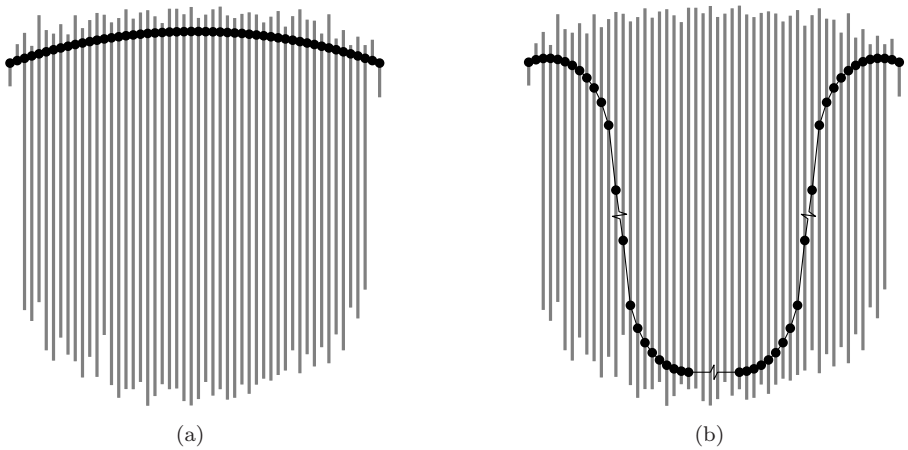


Figure 7: (a) The optimal solution. (b) An ε-approximation. The middle part can be moved down as far as we want, since the turning angles at the saddles can never become worse than ε.

and make a left-turn of ε at each new segment, until also this part of the terrain goes almost vertically down. Then, the middle part has at least k segments left, which means we can make a complete turn from going vertically down to going vertically up. We call this new chain C.

If P completely stays within all imprecision intervals, then we move this middle part of the terrain down until it hits an endpoint. No matter how far this is, the turning angles will stay less than ε. Figure 7 illustrates the optimal terrain and the new approximate terrain. On the other hand, if C does not stay within all intervals, we can start morphing it back to C^*, until all vertices are within their intervals.

The procedure may need to be applied several times, every time splitting a chain into at least two new chains, until all of them have at most $3k$ internal vertices. ⊠

3.2 Solving the subproblems

We now discuss how to solve the subproblems. Recall that a subproblem $S(p_i, d_i, p_j, d_j)$ is specified by two fixed points p_i and p_j, on endpoints of the ith and jth intervals, respectively, and two direction cones d_i and d_j, which bound the possible directions in which the solution may leave p_i and enter p_j. See Figure 8. Moreover, the properties from the previous section allow us to assume that the subproblem is free.

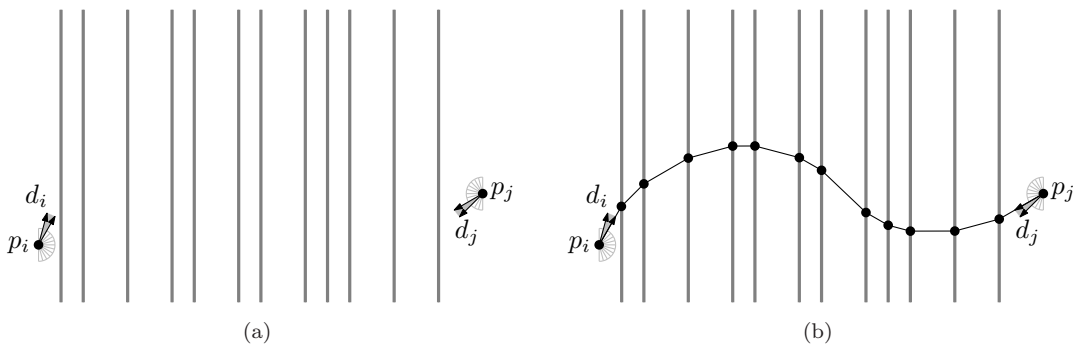


Figure 8: (a) A subproblem $S(p_i, d_i, p_j, d_j)$. (b) The optimal solution.

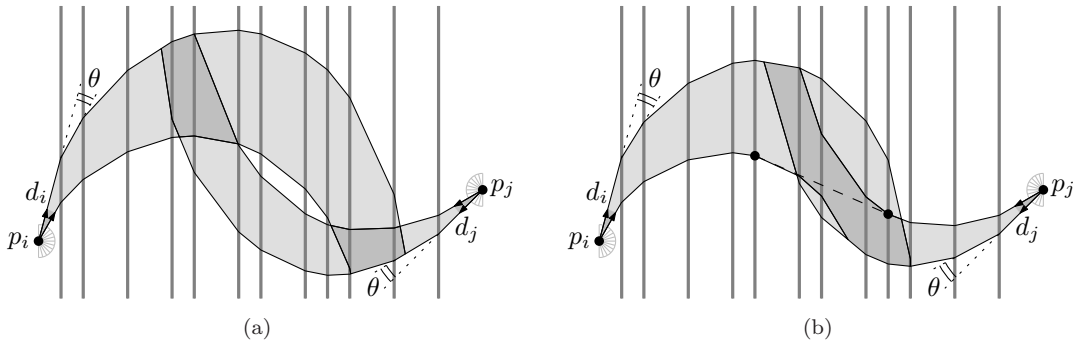


Figure 9: Horns for two values of θ. (a) This value of θ is not feasible. (b) This value of θ is feasible. The dashed line shows the inner tangent, which, together with part of the inner curves, gives a terrain with largest angle θ.

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 will present a δ-approximation algorithm for free subproblems (where the approximation is in the additive vector-of-angles sense, for any δ, independent of ε), which runs in $O((j-i)\log\frac{1}{\delta})$ time. 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 proceed as follows. First we observe that the optimal solution has an S-shape (by Lemma 1). This means that it has only two subchains. Note that this holds because we assumed that the subproblem was free, therefore its optimal solution consists of a single chain.

If possible, we will construct a solution that has this shape, and consists of a curve from p_i and a curve from p_j with consecutive angles of θ, and a piece of a straight line 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 will test them all.

For each combination, we still have freedom to choose in which direction (from the cone d_i) we will leave p_i. We will take the two directions that define the cone (the ones with smallest and largest slope inside the cone), and for each of them, we will start making turns of exactly θ. This results in two curves, also polylines, with turning angles of θ at each vertex. Note that any other starting direction will produce a polyline contained in the area between the other two. We call this area a *horn*. Symmetrically, we also define a horn that starts at p_j and grows from right to left. See Figure 9 for examples for two different values of θ.

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 9. If this tangent stays within both horns, θ is feasible. If not, it is not feasible (we test it for all configurations — left/right turning combination — 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 will find it, since there will always be a solution with this shape. Define the *shortest θ-terrain* as the shortest terrain with worst turning angle θ. Then this terrain will make turning angles of exactly θ at its ends, and have a straight part in the middle.

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. Note, however, that for a δ-free problem, our algorithm is guaranteed to produce a valid solution.

Lemma 5 *Given a free subproblem $S(p_i, d_i, p_j, d_j)$, its optimal solution $T^*(p_i, d_i, p_j, d_j)$ can be approximated within a factor δ in $O((j-i)\log\frac{1}{\delta})$ time.*

Proof: By Lemma 2, all turning angles in the optimal solution T^* are equal. So we can do a binary search on that angle. For each value of θ, we check whether it is feasible in $O(m)$ time by applying the decision algorithm described above. If so, we try decreasing θ, and if not, we have to increase it. Once the difference between the last infeasible and feasible angles becomes less than δ, we stop, and we report a terrain with this angle as produced by the algorithm. ⊠

3.3 Main algorithm

Now we are ready to describe the algorithm for the original problem. The algorithm itself is simple; the correctness analysis is more involved. Recall that $k = \lceil \frac{\pi}{2\varepsilon'} \rceil$.

Each subproblem $S(p_i, d_i, p_j, d_j)$ is defined on two points and two directions. 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 Lemma 5, 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 within a factor of $\delta = O(\varepsilon^2)$ (details on this are given in the next subsection), so we spend $O(nk^4 \log k)$ 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 external vertex. We claim that the resulting terrain is an ε-approximation for the original problem; the analysis is done in the next subsection.

One complication occurs because the optimal terrain does not need to have external first and last vertices. In particular, if $j < 3k$, it is possible that there is no previous external interval. We handle this situation as a special case by computing the optimal *half-chain* up to j. A half-chain is similar to a chain, with the only difference that its first or last vertex can be internal. In the same way, we need to do something at the other end of the terrain. We explain here how to compute an optimal half-chain between I_1 and I_j. The last half-chain can be computed similarly. In particular, the following lemma can also be proved for the last chain in a symmetric way.

Lemma 6 *Let T^* be an optimal terrain, and let v_j be its first (leftmost) external vertex. If v_1 is not external, then all the vertices of T^* between v_1 and v_j lie on a straight line.*

Proof: Consider an optimal terrain and suppose no vertex between I_1 and I_{j-1} is external. Assume that v_2 is left-turning (resp. right-turning). Then we can move v_1 down (resp. up) until the angle at v_2 is zero, and improve the angle vector, contradicting the optimality of the terrain. Note that this can always be done unless while moving v_1 down, it becomes external, but that would reach a contradiction with our starting hypothesis. Therefore the angle at v_2 must be zero, and the three first vertices must be collinear. Now we can treat them as one single edge from v_1 to v_3, and apply the same reasoning again. Proceeding in this way, we conclude that, if there are no external vertices until I_j, then all the vertices up to v_j must be collinear. ⊠

In order to solve a subproblem from I_1 (internal) to v_j, with ending direction cone d_j, the algorithm will check if there exists a line from v_j that hits I_1 and stays within all intermediate intervals. If that is not the case, the subproblem can be discarded because we know from the previous lemma that there must be some other subproblem that contains the line in the optimal solution. Checking if such a line exists can be done in linear time by starting with the ray from v_j with minimum slope that d_j allows. Then we go through the intervals and increase the slope of the ray as to make it stab all the intervals, if possible.

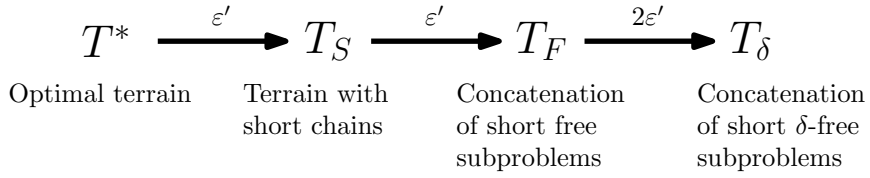


Figure 10: Series of approximations used to obtain an ε-approximation, by taking $\varepsilon = \varepsilon'/4$.

Note that in this special case of the first half-chain of the terrain, a subproblem $S(p_1, d_1, p_j, d_j)$ has a third option for the position of p_1: it can be internal. In that case d_1 will be undefined. A symmetric situation applies to $S(p_i, d_i, p_n, d_n)$, the last half-chain of the terrain.

3.4 Correctness analysis

Let T^* be the optimal terrain. We will argue the existence of several other terrains, each of which is a close approximation of T^* and has certain properties. Eventually, we will establish that one of the terrains in the class that we encounter in the algorithm, is also a close approximation of T^*. A summary of the different stages of the approximation is depicted in Figure 10.

By Lemma 4, we know that there exists a different terrain T_S that approximates T^* within ε', such that all chains of T_S are at most $3k$ long. Let T_S^* be the optimal terrain among all terrains for which all chains are at most $3k$ long. Clearly, T_S^* also approximates T^* within ε'.

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

Eventually we reach a terrain T_F with the property that T_F 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 optimal and free. Furthermore, T_F is within ε' from T_S^*. Let T_F^* be the optimal terrain with these properties. Then also T_F^* is within ε' from T_S^*, and therefore within $2\varepsilon'$ from T^*. Next, we will show that there exists a terrain T_δ that is a concatenation of δ-free subproblems, which approximates T_F^* within $2\varepsilon'$. Our algorithm encounters and approximately solves all δ-free subproblems, so it will compute the optimal terrain T_δ^* of this form. T_δ^* then also approximates T_F^* with $2\varepsilon'$, and therefore T^* within ε.

Next, let $S(p_i, d_i, p_j, d_j)$ 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 $S(p_i, d_i, p_j, d_j)$ has an error e, it is either δ-free, or there exists a sequence of smaller subproblems $S(p_{i_0}, d_{i_0}, p_{i_1}, d_{i_1})$, $S(p_{i_1}, d_{i_1}, p_{i_2}, d_{i_2})$, $\ldots$, $S(p_{i_{h-1}}, d_{i_{h-1}}, p_{i_h}, d_{i_h})$, where $i_0 = i$ and $i_h = j$, which 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 $e + \varepsilon'$.*

Proof: If $S(p_i, d_i, p_j, d_j)$ is δ-free, the Lemma is clearly true. If not, there must exist a solution R that is a δ-approximation of R^*, such that some vertex v_h of R lies outside its imprecision interval. We also know that R^* respects the imprecision intervals, and has only interior vertices,

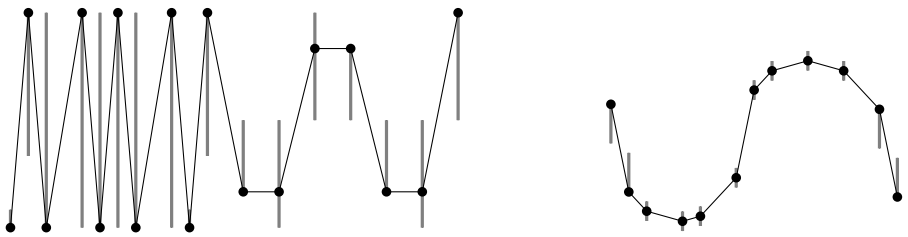


Figure 11: Optimal terrain for max min angle.

since the subproblem is free. Now, by Corollary 1, we can continuously transform R towards R^*, while the solution continuously improves. At some point, we will reach a terrain R' that goes through an endpoint p_h of the hth interval. If R misses any other intervals, we just continue the transformation until the last interval is hit. At this point, fix the cones in which R' enters and leaves p_h. Let R'^* be the solution we get from concatenating the optimal solutions of the subproblems from i to h and from h to j. Then R'^* is at least as good as R', except at vertex v_h, where it might be ε' worse. This means that R'^* is a δ-approximation for R^* except at v_h. Now, if the subproblems $S(p_i, d_i, p_h, d_h)$ and $S(p_h, d_h, p_j, d_j)$ are both free, we are done. Otherwise, we can split them into free subproblems again, possibly losing ε' at each vertex where the split takes place. ⊠

It now follows that we can approximate the optimal terrain by considering only δ-free subproblems.

Corollary 2 *If a free subproblem $S(p_i, d_i, p_j, d_j)$ of length m has an error e, it is either δ-free, or there exists a sequence of smaller subproblems $S(p_{i_0}, d_{i_0}, p_{i_1}, d_{i_1})$, $S(p_{i_1}, d_{i_1}, p_{i_2}, d_{i_2})$, ..., $S(p_{i_{h-1}}, d_{i_{h-1}}, p_{i_h}, d_{i_h})$, 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 $e + (m-1)\delta + \varepsilon'$.*

Proof: By Lemma 7, we can split free subproblem that are not δ-free into smaller free subproblems, such that their errors increase by at most δ. If these smaller subproblems are again not δ-free, we can apply the lemma on these problems again. This iterates until all our subproblems are δ-free, since the subproblems always become shorter and a subproblem of length two (without any intermediate intervals) is trivially δ-free. In each iteration, the error can increase by at most δ, and when the terrain is split at a vertex its error can increase by at most ε'. The corollary follows. ⊠

Now, consider T_F^*. We construct the terrain T_δ by replacing each free subproblem in T_F^* by a sequence of δ-free subproblems, according to Corollary 2. The errors in T_δ may have gotten at most $3k\delta + \varepsilon'$ worse than in T_F^*. By setting $\delta = \frac{\varepsilon'}{3k}$, we get the approximation factor sought.

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

We can also try to maximize the smallest turning angle to try to make the terrain as rough as possible. Figure 11(a) shows an example. As the figure shows, the optimal solution can use internal vertices. In fact, there could be multiple consecutive internal vertices, such that they all keep each other in balance, as in Figure 11(b). As in Section 3, computing the exact placements of the

vertices of such a chain involves computing the inverse of a function like the one in Equation $(*)$. Therefore, we will present an approximation algorithm.

As before, we do not only look at the worst angle in the terrain, but at the complete vector of turning angles. For a terrain instance T, define $V(T)$ to be the vector of turning angles in that instance, sorted from small to large. 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 smaller at every position, that is, if $V_i(T) \geq V_i(T^*) - \varepsilon$.

The main idea of the algorithm is very similar to the one in Section 3. Again, we use a value $\varepsilon' = \varepsilon/4$ and define $k = \lceil \frac{\pi}{2\varepsilon'} \rceil$. We still define subproblems of the form $S(p_i, d_i, p_j, d_j)$, which we solve for all combinations where $j - i$ is not too large, and then use dynamic programming to glue them together. However, several adaptations have to be made, because of the different nature of the problem.

4.1 Properties of chains

We discuss some important properties of chains. Let T^* be the optimal terrain, and C^* some chain in T^*. Some of the lemmata here are very similar to their counterparts in Section 3, but some of the properties that were true there are not true in this case and vice versa.

Lemma 8 *The leftmost and rightmost intervals of T^* are external.*

Proof: If the leftmost (or rightmost) interval is not external, its vertex can be moved in both directions, and one of them will necessarily increase the only angle affected by that vertex. Therefore it can be taken to one of the extremes of its interval, without decreasing the smallest turning angle. ⊠

Lemma 9 *C^* has at most two subchains.*

Proof: This lemma is the similar to Lemma 1, and almost the same proof still holds. Suppose that the direction of curvature changes more than once along a chain, and all vertices are allowed to move in either direction. Then take a part of consecutive vertices that all bend left (resp. right), while the neighbors bend right (resp. left). In Figure 6, such a part is marked by gray vertices. If we move this part as a whole down (resp. up), the only turning angles that change are the ones at the outermost vertices of this part and their neighbors. But clearly, if we move the part in the right direction, these angles only become larger. This contradicts the assumption that C^* was part of the terrain with the optimal vector of turning angles. ⊠

Note that Lemma 9 implies that a chain can have at most one saddle edge.

Lemma 10 *All vertices of C^* have the same turning angle, except possibly for one of the saddle vertices, which can have a larger angle.*

Proof: All vertices are internal, so we can move them up or down by a small amount. We will show that if the lemma is false, we can move some vertices an infinitesimally small amount such that the value of the solution increases. First we will show that each subchain only has vertices with the same turning angle, except for one of the saddle vertices which may have a larger angle.

Let P^* be such a subchain. Suppose that not all vertices have the same turning angle, and let θ be the smallest turning angle that occurs in P^*. Then there must exist a vertex v that is not a saddle vertex and that has an angle larger than θ, and at least one neighbor w with an angle of θ. If v is left-turning, then so are its neighbors (since it is not a saddle vertex), so we can move v up by a small amount, which makes its angle smaller but the angles of its neighbors larger. Now w has a

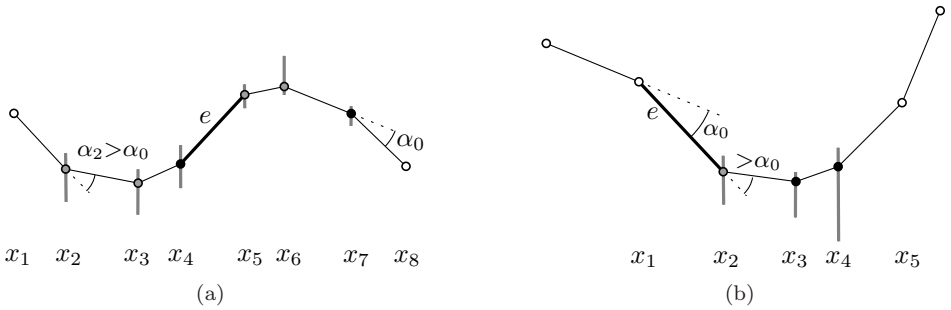


Figure 12: (a) Example of a chain comprised of two subchains, where e is the saddle edge. Black/white vertices have turning exactly α_0, while gray vertices have a larger angle. (b) Example in which the saddle edge e is the first edge of the chain, forcing x_2 to have an angle larger than the one of the other vertices (in an optimal solution).

larger angle, so the number of vertices with angle θ has decreased. Therefore the lexicographical value of P^* has increased, which contradicts its optimality. Symmetrically, if v is right-turning, we can move it downwards a little bit.

Now suppose that C^* has two subchains with different turning angles: say without loss of generality that it first has a left-turning subchain with angles of θ, and then a right-turning subchain with angles of $\phi > \theta$. If we increase the turning angle of the first chain a little bit, but keep the first vertex of the subchain (the external vertex) where it is, the last vertex of the subchain (the saddle vertex) will move upwards. However, if we decrease the turning angle of the second chain a little bit, keeping the last vertex of the subchain (the external vertex) where it is, the first vertex of the subchain (the other saddle vertex) will also move upwards. If we balance the changes so that the saddle vertices both move up by the same amount, the turning angles at the saddle vertices do not change.

Notice that the only vertices that might not be possible to move up (or down, in the case of a right-turning chain) are saddle vertices, and, by Lemma 8, there can be at most one saddle edge in the chain. Typically, such an edge will connect a left-turning subchain to a right-turning subchain (or the other way around). However, it could happen that the saddle edge is the first or last edge of the chain, in which case one of the saddle vertices will not be able to be moved up and can finish with an angle larger than the other vertices. See Figure 12 for an example. ⊠

Also, we can again formulate a corollary analogous to Corollary 1.

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

Lemma 11 *Suppose C^* consists of more than $2k$ vertices. Then there is another solution C that starts and ends in the same location and direction as C^*, has only external vertices, and such that C is an ε'-approximation of C^*.*

Proof: We know from Lemma 9 that C^* contains at most 2 subchains. By Lemma 10, we know that all vertices have the same turning angle θ, except for at most one saddle vertex v that may have a turning angle of $\phi > \theta$. But since the terrain is x-monotone, in total, the turning angle of each subchain cannot exceed π. Therefore, the turning angle at each vertex (that is, θ) is at most $\frac{\pi}{k} < \varepsilon'$. Clearly, the turning angles in C cannot be smaller than 0, so are always in the correct range for an ε-approximation, except possibly for vertex v. However, if we maximize the angle at v by moving it up as far as possible and its neighbors down, or the other way around, it will clearly have an angle of at least ϕ. Then we can place all remaining vertices anywhere we like. ⊠

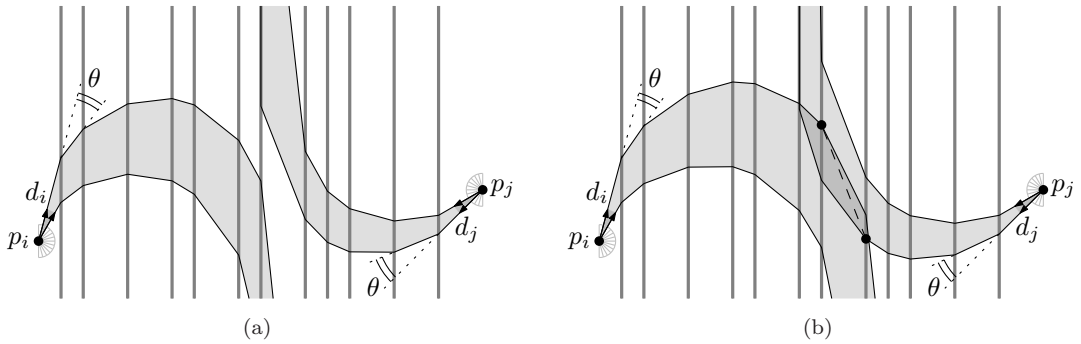


Figure 13: (a) This value of θ is not feasible. (b) This value of θ is feasible. The dashed line shows a connection between two consecutive vertices of the two outer chains, which realizes a terrain with smallest angle θ.

Finally, for this problem we need to introduce one more concept. A realization of an imprecise terrain is said to be *locally optimal* if its smallest angle cannot be increased by moving a single vertex over an infinitesimally small distance. The algorithm described in the next section will compute such a locally optimal solution.

4.2 Solving the subproblems

Let $S(p_i, d_i, p_j, d_j)$ be as before: a subproblem defined on two fixed external vertices, and two cones of directions. We will compute a locally optimal solution to the subproblem from I_i to I_j, under the assumption that all the intervals between I_i and I_j are internal.

When solving the subproblem, we ignore the imprecision intervals of the internal vertices. By Lemma 10, the shape of the optimal solution inside a chain can consist of at most two subchains. In order to find this solution after ignoring the intervals, however, we have to explicitly specify that we only allow two subchains (otherwise, we would return a solution that zigzags up and down arbitrarily far, yielding a turning angle of almost 180° at each vertex).

Again, because of algebraic difficulties we can only compute approximate solutions to the subproblems. We present a δ-approximation algorithm that runs in $O((j-i)\log\frac{1}{\delta})$ time.

Given a value θ, we ask the decision question: is there a solution terrain T, consisting of at most two subchains, for this subproblem that uses only angles of θ or more? Now the decision algorithm is very similar to the one in Section 3.2. 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 each combination, we compute a subinterval of every vertical line that we can reach by making turns of θ. Figure 13 shows two examples of the same subproblem for different values of θ.

To test feasibility, consider the outer curves of both horns. If they do not intersect, θ is not feasible. If they do, we take two consecutive vertices in their common intersection, and connect them. The terrain determined by the parts of the outer curves of the horns and this new saddle edge has only turning angles of θ or more, and has exactly two subchains (one of which may in fact be only a single edge, in some cases).

Lemma 12 *Given a free subproblem $S(p_i, d_i, p_j, d_j)$, we can approximate the locally optimal solution $T^*(p_i, d_i, p_j, d_j)$ of this problem within a factor delta in $O((j-i)\log\frac{1}{\delta})$ time.*

4.3 Main algorithm

With these ingredients, the algorithm itself works exactly the same as in Section 3, except that here we need no special treatment of the ends of the terrain, because of Lemma 8.

Each subproblem $S(p_i, d_i, p_j, d_j)$ is defined on two points and two directions. We now consider all $O(k^3 n)$ subproblems where $j - i \leq 2k$. By Lemma 12, each subproblem can be approximated within an error of δ in $O(k \log \frac{1}{\delta})$ time. We approximate all subproblems within a factor of $\delta = O(\varepsilon^2)$, so we spend $O(nk^4 \log k)$ in total.

We discard any solutions that do not respect the imprecision intervals. Note that the solutions computed for the subproblems may not return the best realization of the subproblem between I_i and I_j, because it returns a *locally optimal* solution that does not use external vertices. This means that it will be an optimal solution only if the optimal solution for that subproblem does not use external vertices either. However, if the optimal solution really does use external vertices, it will be found at some later step of the algorithm.

After solving all subproblems, we invoke dynamic programming again 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 external vertex.

4.4 Correctness analysis

Let T^* be the optimal terrain. As before, we will argue the existence of several other terrains, each of which is a close approximation of T^* and has certain properties. By Lemma 11, we know that there exists a different terrain T_S that approximates T^* within ε', such that all chains of T_S are at most $2k$ long. Let T_S^* be the optimal terrain among all terrains for which all chains are at most $2k$ long. Then T_S^* is an ε'-approximation of T^*.

The terrain T_S^* can be partitioned into short chains. For each external vertex in T_S^*, we divide its circle of directions into $2k$ cones, and we locate its two outgoing edges in this cone set. We fix these cones, and split the problem into subproblems. Let T_C be the terrain we get by solving these subproblems optimally. If T_C has new external vertices, repeat. Eventually we reach a terrain T_F with the property that T_F is a concatenation of optimally solved independent smaller subproblems of length at most $2k$. Let T_F^* be the optimal such terrain. Then T_F^* is an ε'-approximation of T_S^*.

Next, we will show that there exists a terrain T_δ that is a concatenation of δ-free subproblems, which approximates T_F^* within $2\varepsilon'$. Our algorithm encounters and approximately solves all δ-free subproblems, so it will compute the optimal terrain T_δ^* of this form. T_δ^* then also approximates T_F^* with $2\varepsilon'$, and therefore T^* within ε. Next, let $S(p_i, d_i, p_j, d_j)$ be a free subproblem, and let R^* be its optimal solution.

Note that Lemma 7 and Corollary 2 from Section 3.4 still hold in our current situation without adaptation. Again, we construct the terrain T_δ by replacing each free subproblem in T_F^* by a sequence of δ-free subproblems, according to Corollary 2. The errors in T_δ may have gotten at most $2k\delta + \varepsilon'$ worse than in T_F^*. By setting $\delta = \frac{\varepsilon'}{2k}$, we get the approximation factor sought.

Theorem 2 *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.*

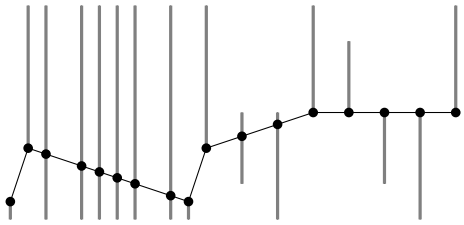


Figure 14: Optimal terrain for min total turning angle.

5 Minimizing the total turning angle

We now study a different measure, related also to smoothness, which can be optimized exactly. We consider the problem of finding a realization of the imprecise terrain that minimizes the *total turning angle*, defined as the sum of the absolute values of the turning angles at each vertex (except for the first and last one). Given a realization of a terrain, we use $\measuredangle(e_i, e_j)$ to denote the total turning angle between two edges e_i and e_j. We start by making the following observation.

Observation 1 *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 them.*

In other words, the location of the vertices in between does not matter, as long as they make the path be left-turning (or right turning). This implies that there will most likely be many equivalent optimal solutions. An example is shown in Figure 15.

Moreover, the next lemma shows that 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 (see Figure 16(a)) is an optimal solution.

Lemma 13 *Let v_1 and v_n be the first and last vertices of an imprecise terrain. If the positions of v_1 and v_n are fixed, then the shortest path from v_1 to v_n that stays inside the corridor induced by the intervals I_2 to I_{n-1} minimizes the total turning angle.*

Proof: Let ρ denote the shortest path from v_1 to v_n, as stated in the lemma. Consider the global shape of ρ. It can be seen as consisting of an alternating sequence of consecutive right-turning

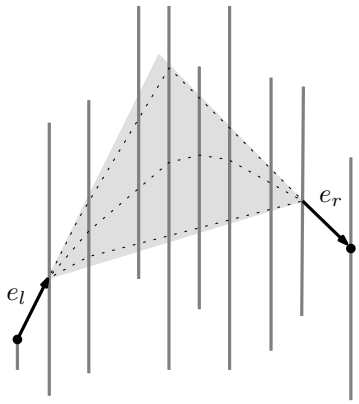
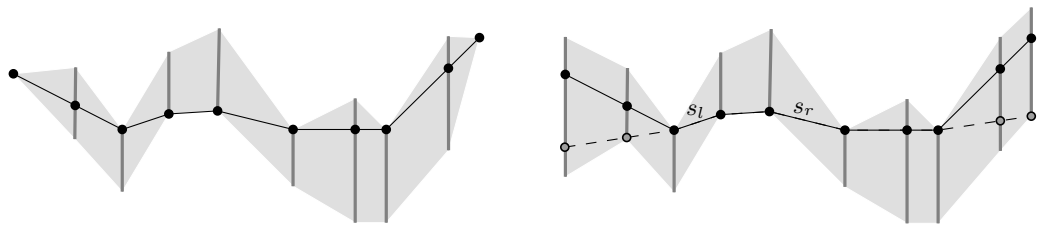


Figure 15: Illustration for Observation 1. Any right-turning path starting and ending with the highlighted directions (thick arrows) will have the same total turning angle, namely $\measuredangle(e_l, e_r)$.



(a) (b)

Figure 16: (a) The shortest path (solid) through the corridor between the intervals (shaded) minimizes the total turning angle when the endpoints are fixed. (b) When the endpoints are not fixed, the optimal path (dashed) can differ from the shortest path (solid) in the parts from the first vertex to the leftmost saddle edge (s_l), and from the rightmost saddle edge (s_r) to the last vertex.

vertices and consecutive left-turning vertices, connected by saddle edges (if there are saddle groups, we treat them as one saddle edge each).

The proof is based on observing that the angles between consecutive saddle edges in ρ cannot be avoided: any path will incur those or worse angles. In other words, these turns (or turns with a worse angle) are present in any realization of the terrain. The same applies to the angles between the first vertex and first saddle edge, and between the last saddle edge and the last vertex. Since the total angular change of ρ is exactly the sum of these angles, it follows that it must be optimal. Details are given next.

First we observe that since ρ is the shortest path, each of its turns must be at external vertices: a right turn must lie on the lower endpoint of its interval, and a left turn must lie on the upper endpoint of its interval, otherwise there would be a shorter path. Now consider a saddle edge $s_k = (v_i, v_{i+1})$, together with the next saddle edge $s_{k+1} = (v_{j-1}, v_j)$. Assume without loss of generality that s_k connects left-turning edges to right turning-edges. Then v_i must be at the upper endpoint of I_i and v_{i+1} at the lower endpoint of I_{i+1}.

We know from Observation 1 that any optimal path connecting s_k to s_{k+1} incurs the same angular change as ρ between these two edges, namely $\measuredangle(s_k, s_{k+1})$. This is because a path between s_k and s_{k+1} that is part of an optimal realization cannot have bends between these two saddle edges: we know from ρ that they can be avoided, and avoiding them would decrease the total turning angle.

It remains to verify that no path that leaves v_i or enters v_j in a different direction can have better total angular change between v_i and v_j. Note that between s_k and s_{k+1} the path is right-turning. Any change (within the intervals) in the position of v_i or v_{i+1} (that is, moving v_i down or v_{i+1} up) increases the slope of s_k, thus also $\measuredangle(s_k, s_{k+1})$. Symmetrically, any change in the position of v_{j-1} or v_j decreases the slope of s_{k+1}, increasing $\measuredangle(s_k, s_{k+1})$.

We conclude that no path can have a total angular change smaller than the one of ρ between v_i and v_j. Since this applies to every pair of consecutive saddle vertices (and to the paths between the first vertex and first saddle edge, and between the last saddle edge and the last vertex), it follows that any path must have at least the total turning angle of ρ. Therefore ρ has minimum total angular change. ⊠

When the leftmost and rightmost intervals are not fixed at single points but are 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, it may contain some unnecessary turns at the ends. We can identify the leftmost and rightmost saddle edges of the path, and note that the minimum total turning angle would be achieved by just continuing in a straight line from these segments, towards the leftmost and rightmost intervals —see Figure 16(b)— since then we

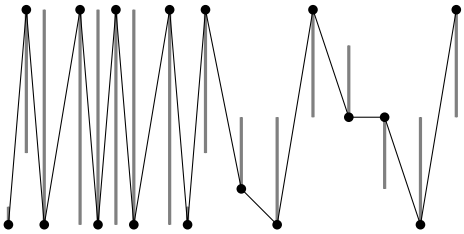


Figure 17: The terrain maximizing the total turning angle.

make no extra turns. However, it can be that this is not possible.

Take the rightmost saddle edge s_r (the situation for the leftmost one is symmetric). Ideally, we would like to replace the last part of the shortest path with the straight line segment extending s_r until the rightmost interval I_n, avoiding all turns between s_r and I_n. However, this may produce an invalid realization (violating some intervals), and some turns may be necessary. Observe that necessary turns will be always in the same direction, because there is no other saddle edge to the right of s_r, hence the remaining edges are all right-turning or all left-turning.

To find the path from s_r that minimizes the turning angle we can proceed by starting with an angle of 0 with respect to s_r (that is, starting with a line segment that is an extension of s_r). Then we go through the intervals from s_r to I_n. Every time the current path is below (resp. above) the current interval, we add the minimum turn needed to make it go through it. Once we reach the last interval, we know the position of the rightmost vertex, but the path can still be outside the corridor. To fix this, we do the same as before but going from I_n to s_r, adapting the path when it goes above (below) the current interval. The resulting path is inside the corridor and minimizes the total angular change.

As for the running time of the algorithm, the shortest path can be computed in linear time [19]. The adaptations needed to fix the beginning and end of the path, as described in the previous paragraph, can also be made in linear time. We obtain the following result.

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

6 Maximizing the total turning angle

In this section with study another measure related to making the terrain as rough as possible: maximizing the total turning angle. This measure turns out to be much easier to optimize than the one considered in Section 4. We make the following simple observation.

Lemma 14 *There is always a realization of an imprecise 1.5D terrain that maximizes the total turning angle using only external vertices.*

Proof: 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 17 for an example. ⊠

After making this observation, it is easy to verify that the problem can be solved in linear time. We simply go over the terrain once, from left to right, and for each segment, defined by the upper/lower endpoints of two consecutive intervals, we store the optimal solution to the left of

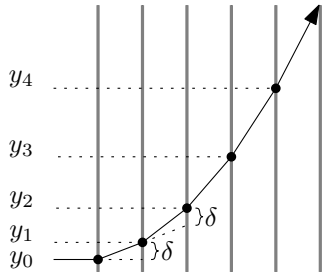


Figure 18: A simple case of a terrain with constant slope change δ. The resulting function is simpler than the one of Equation $(*)$.

that segment that uses it. There are $4n$ such segments, and computing a value involves matching it with the two possible stored solutions immediately to the left of it and taking the best one. The next lemma summarizes the result.

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

7 Minimizing the maximum slope change

In Section 3, we gave an approximation algorithm for the problem of minimizing the maximum turning angle in a realization of an imprecise 1.5D terrain. In this section, we show that we can give an exact solution if we attempt to minimize the change in slope between successive edges of a realization, rather than the change in angle between successive edges. The main difference between the two problems is that the function that we minimize is much simpler in the former problem than in the latter. This will allow us to give an exact algorithm for this problem.

To illustrate the difference, consider the following simpler subproblem, analogous to the one of Section 3. Suppose we have a part of the optimal solution that consists of several consecutive left-turning vertices with constant slope change of δ (for $0 \leq \delta$). Suppose that the leftmost point is fixed at the origin (and the previous point was on the negative x-axis), and the intervals are all vertical lines with x-coordinates x_i (for $i = 1, 2, \ldots$). Figure 18 illustrates the situation.

Then the height y_i at which the line at x_i is crossed is a function of δ with the following shape:

$$y_i(\delta) = y_{i-1}(\delta) + (x_i - x_{i-1})i\delta = \delta \sum_{j=1}^{i} (x_j - x_{j-1}) \cdot j \qquad (**)$$

Note that, as opposed to the case of Equation $(*)$, the inverse of Equation $(**)$ can be easily computed, since the summation is independent of δ.

The broad outline of the technique we use is to find a solution to the feasibility problem of whether there exists a realization of a given terrain with maximum slope change of δ. See Figure 19 for an example. We then apply parametric search [21], which uses this solution as a subroutine.

7.1 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

Figure 19: An example with four intervals, and a possible solution for $\delta = \frac{1}{4}$.

each interval. This dimension represents the slope of the terrain. We define a set of rules for a path in feature space. A path 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.

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 paths that go through the interval. The entrance region defines possible entry slopes (for the edge entering from the left), whereas the exit region possible exit slopes (for the edge leaving on the right). The entrance region of an interval I_j is bounded from above (resp. below) by a function $g_j^+(\lambda)$ (resp. $g_j^-(\lambda)$). For a given λ, $[g_j^+(\lambda), g_j^-(\lambda)]$ gives the subinterval of I_j within which a path must enter I_j, if it does so with a slope of λ. Similarly, the exit region of an interval I_j is bounded from above and below by functions $f_j^+(\lambda)$ and $f_j^-(\lambda)$. The meaning of these functions is analogous to the one of $g_j^+(\lambda)$ and $g_j^-(\lambda)$.

For the first interval I_1, only the exit region is defined. It is an infinite strip bounded by the functions $f_1^+(\lambda)$, which is a constant function with a y-value equal to the y-value of the top of the first interval, and $f_1^-(\lambda)$, a constant function with a y-value equal to the y-value of the bottom of the first interval. The regions of interval I_j are the regions of I_{j-1} modified in the following manner (see Figure 20):

- We define intermediate functions $h_j^+(\lambda)$ and $h_j^-(\lambda)$ to be $f_{j-1}^+(\lambda) + \lambda(x_j - x_{j-1})$ and $f_{j-1}^-(\lambda) + \lambda(x_j - x_{j-1})$ respectively.
- 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. The resulting functions are $g_j^+(\lambda)$ and $g_j^+(\lambda)$. The region between them is the entrance region of I_j.
- If the entrance region of I_j is not empty, then we treat it as a polygon P in the (λ, y)-plane. We compute the Minkowski sum of P and a horizontal line segment that has length δ. This amounts to moving the non-horizontal edges of P horizontally by a distance of δ away from the interior of the polygon. After this, we define $f_j^+(\lambda)$ to be the segments of P that have the interior of P below them and $f_j^-(\lambda)$ to be the segments of P that have the interior of P above them. The region between these two functions is the exit region of I_j.

Each dimension of a point $p = (x, y, \lambda)$ inside the entrance region for an interval I corresponds to one aspect of an incoming path. The x-coordinate of p corresponds to the x-coordinate of I, the y-coordinate corresponds to the height of the path along I, and the λ-coordinate corresponds to the slope of the previous segment of the path. A path thus travels from one interval to the next at a slope (in the y-dimension) equal to its λ-coordinate. At each interval, the path can change its λ-coordinate by at most δ.

Lemma 15 *A path with maximum slope change δ exists if and only if the entrance region for interval I_n is non-empty.*

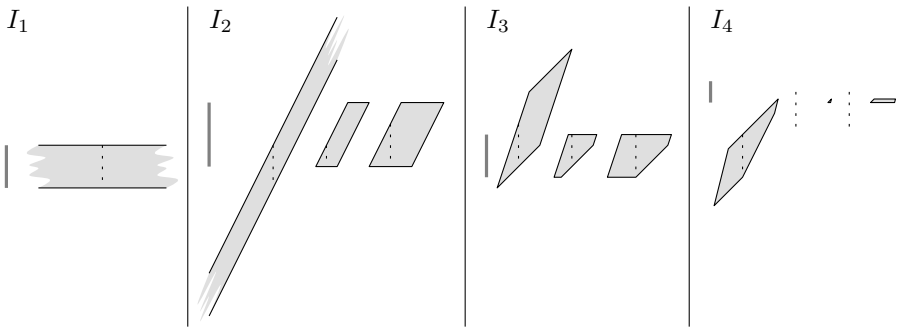


Figure 20: The regions as they are computed in the process for $\delta = \frac{1}{4}$. The figures are drawn in the (λ, y)-plane. The dotted lines correspond to $\lambda = 0$. For each interval (except the first), three regions are shown: the first one is an intermediate region used to construct the 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 a Minkowski sum with a horizontal line segment of length δ.

Proof: Recall that the *exit region* of I_j is the same as its *entrance region* after being expanded by the Minkowski sum with a horizontal line segment. See Figure 20. Note that for any point $p = (x, y, \lambda)$ in the exit region, $p' = (x, y, \lambda + \mu)$ is in the entrance region for some $-\delta \leq \mu \leq \delta$.

Assume that the entrance region for the interval I_n is non-empty and that $p_n = (x_n, y_n, \lambda_n)$ is a point inside it. We describe a procedure for finding a path through the intervals that has a maximum slope change of δ.

We begin at interval I_n. We then construct a line segment from p_n to the interval I_{n-1} that has slope λ_n. Let the point where the line segment intersects the interval I_{n-1} be $p_{n-1} = (x_{n-1}, y_{n-1}, \lambda_n)$. Note that y_{n-1} is easy to compute: it is $y_n - \lambda_n(x_n - x_{n-1})$. Because of the procedure for propagating the regions, the point $p' = (x_{n-1}, y_{n-1}, \lambda_n)$ is inside the exit region for the interval I_{n-1}.

If the point $(x_{n-1}, y_{n-1}, \lambda_n)$ is outside the entrance region of interval I_{n-1}, then we find $-\delta \leq \mu \leq \delta$ such that $p'' = (x_{n-1}, y_{n-1}, \lambda_n + \mu)$ is inside the entrance region, and set λ_{n-1} to be $\lambda_n + \mu$. We can then recursively apply this procedure until we have found a path from I_n to I_1 whose maximum slope change is δ. This path is formed by the sequence of points $(x_1, y_1), (x_2, y_2), \ldots, (x_n, y_n)$. See Figure 21.

To see that a path with maximum slope change δ exists only if the entrance region for the interval I_n is non-empty, assume that the entrance region at some interval I_j is empty and that I_j is the first such interval. This implies that the exit region for I_{j-1} was propagated to a region that has y-values that are either too high or too low and were thus truncated. This means that any path whose maximum slope change is δ will either be above or below interval I_j if the path started at I_1 and respected all previous intervals. ⊠

Thus, if the entrance region for interval I_n is non-empty, a path with maximum slope change δ has been determined. We are then ready to apply parametric search[1] in the standard manner.

7.2 Time analysis

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

[1]We intentionally omit any details of parametric search. We simply use it as a black box that we apply to a feasibility-testing algorithm and an instance of our problem. Interested readers can see [24] for a more extensive treatment of the subject.

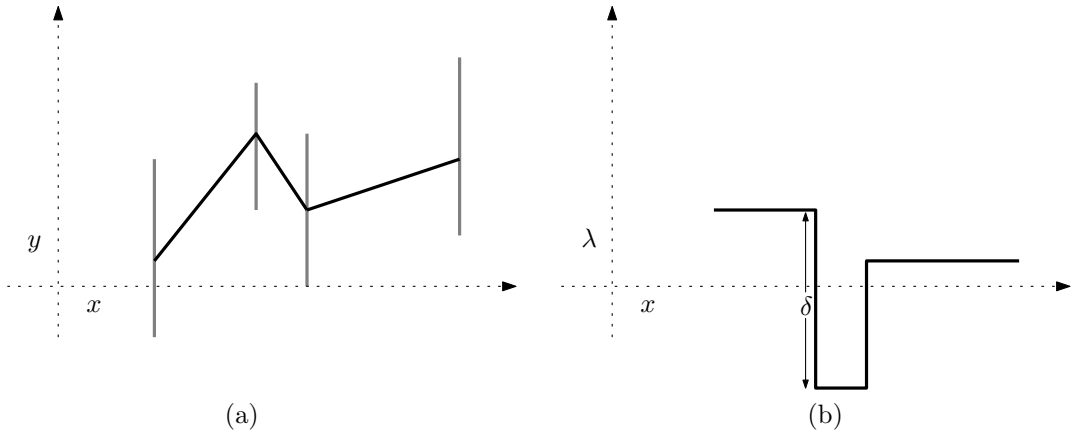


Figure 21: A path with maximum slope change δ shown in the (a) (x, y) plane and (b) (x, λ) plane.

running time is simple.

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

Lemma 16 *The entrance and exit regions of interval I_j are convex polygons with complexity at most $2j$.*

Proof: We prove by induction. We consider, for now, only the entrance region. The entrance region of interval I_2 is a convex polygon with complexity at most 4, so the proposition holds. We now assume that the entrance region of interval I_{k-1} is a convex polygon with complexity at most $2k-2$ and will proceed to prove that the entrance region of interval I_k is a convex polygon with complexity at most $2k$.

When the entrance region from interval I_{k-1} is propagated to I_k, the edges that form the exit region of interval I_{k-1} are sheared, possibly truncated, and translated. In all of these operations, all of the edges remain line segments. It is also clear that if the entrance region is convex, then at most two new edges are added as a result of the truncations to the top and bottom functions.

From the above, it is easy to see that the entrance region at interval I_k is a polygon with complexity at most $2k$. It remains to show that the region is convex. Since I_{k-1} is a convex region, the shear transformation preserves this property. The clipping also does, since it is an intersection with a convex set (the intersection of two halfplanes). Finally, the Minkowski sum of two convex polygons is again convex, therefore the entrance region for I_k is convex.

Regarding the exit region, its convexity follows by construction. As for its complexity, we note that the Minkowski sum only will create two new vertices if the clipping applied to produce the entrance region did not create new vertices. Therefore the total number of vertices for the exit region will increase, in any case, by at most two with respect to the previous exit region. ⊠

Lemma 17 *The time complexity of the feasibility algorithm is $O(n^2)$.*

Proof: The feasibility-testing algorithm requires the regions to be propagated n times. Each propagation takes $O(n)$ time by Lemma 16. Therefore, the feasibility-testing algorithm takes $O(n^2)$ time. ⊠

Using this feasibility-testing algorithm with parametric search, we get an algorithm to solve our original problem. Unfortunately, our feasibility-testing algorithm is inherently sequential, so we

do not get the benefit of the speedup for parallel algorithms that parametric search gives. Thus the running time is $O(n^4)$. Fortunately, we can solve the feasibility problem more efficiently, as we show in the next section.

7.3 A faster 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. The following follows from using affine transformations:

Observation 2 *A sequence of shear and translation operations can be represented as a single shear and a single translation.*

Another observation that will make the description of the algorithm easier, is that f^+ and f^- are monotonically increasing functions. The reason for this is that they start as horizontal lines, and in each transformation step, they are either sheared by a positive factor (in which case the slopes of the functions increase), or clipped (in which case a new horizontal part is added).

Observation 3 *The functions f_j^+ and f_j^- that bound the exit region of I_j, are monotonically increasing.*

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 $(y, \lambda) \mapsto (y, \lambda + (x_{j+1} - x_j)y)$ 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 t_j and b_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 b_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 b_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 t_j. This step takes constant time per vertex that is thrown away.

Finally, we translate the vertices describing the entrance region over a distance of δ in each direction, 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 18 *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. ⊠

7.4 Solving the original problem

Theorem 5 *Given an imprecise 1.5D terrain, a realization that minimizes the maximum slope change can be computed in $O(n^2)$ time.*

Proof: By Lemma 18, the feasibility-testing algorithm takes $O(n)$ time. We apply parametric search with this algorithm. Since the feasibility test is inherently sequential, the parametric search squares the running time of the feasibility-testing algorithm, resulting in total $O(n^2)$ time. ⊠

Note that we could alternatively use the feasibility-testing algorithm to get an approximate solution. If we just do a binary search on δ, we can find an ε-approximation in $O(n \log \frac{1}{\varepsilon})$ time.

8 Conclusions

We studied several measures to compute the smoothest or roughest possible 1.5D terrain, when height information is imprecise. For optimizing the worst turning angle in the terrain (either smallest or largest), we presented approximation algorithms that run in linear time. 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 easy to implement. As a supplement to these results, we also studied algorithms for optimizing the total turning angle, and for minimizing the maximum slope change. We sketched two exact algorithms that also run in linear time, and one that runs in quadratic time. These algorithms should also be fairly simple to implement.

There are several other problems that are worth studying for our model of 1.5D imprecise terrains, in addition to smoothness. One interesting topic is visibility or guarding. A problem already studied for (precise) 1.5D terrains is placing a (small) set of guards such that they can see all the terrain [5, 17]. Taking the problem to an imprecise setting gives rise to questions like finding a small set of guards that is guaranteed to see all the terrain for any possible realization, or given a set of guards, finding a realization that hides the maximum amount of terrain. Another interesting topic is terrain simplification (e. g. [10]). For example, we may want to find a realization that can be specified by the smallest number of vertices (that is, where many vertices are collinear and can be removed).

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 that is quite sharp can be created at the intersection of three or more faces. A promising approach involves using one of the definitions of discrete curvature specifically defined for polyhedral surfaces, like for example, the mean or Gaussian Ccurvature introduced by Mesmoudie *et al.* [22]. Even when the objective 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.* [23]. 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.

Acknowledgements

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

References

[1] M. Abellanas, F. Hurtado, and P. A. Ramos. Structural tolerance and Delaunay triangulation. *Inf. Proc. Lett.*, 71:221–227, 1999.

[2] E. M. Arkin, J. S. B. Mitchell, and C. D. Piatko. Bicriteria shortest path problems in the plane. In *Proc. 3rd Canadian Conference on Comput. Geom.*, pages 153–156, 1991.

[3] C. Bajaj. The algebraic degree of geometric optimization problems. *Discrete Comput. Geom.*, 3:177–191, 1988.

[4] D. Bandyopadhyay and J. Snoeyink. Almost-Delaunay simplices: Nearest neighbour relations for imprecise points. In *Proc. 15th ACM-SIAM Symposium on Discrete Algorithms*, pages 410–419, 2004.

[5] B. Ben-Moshe, M. J. Katz, and J. S. B. Mitchell. A constant-factor approximation algorithm for optimal 1.5D terrain guarding. *SIAM Journal on Computing*, 36(6):1631–1647, 2007.

[6] J.-D. Boissonnat and S. Lazard. A polynomial-time algorithm for computing a shortest path of bounded curvature amidst moderate obstacles. *Internat. J. Comput. Geom. Appl.*, 13:189–229, 2003.

[7] 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.

[8] N. Dyn, D. Levin, and S. Rippa. Data dependent triangulations for piecewise linear interpolation. *IMA Journal of Numerical Analysis*, 10:137–154, 1990.

[9] R. Feciskanin. Optimization of triangular irregular networks for modeling of geometrical structure of georelief. In *Proceedings of the International Cartographic Conference 2007*, 2007.

[10] M. Garland and P. Heckbert. Fast polygonal approximation of terrains and height fields. Technical Report CMU-CS-95-181, Computer Science Department, Carnegie Mellon University, Pittsburgh, PA, September 1995.

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

[12] L. J. Guibas, D. Salesin, and J. Stolfi. Constructing strongly convex approximate hulls with inaccurate primitives. *Algorithmica*, 9:534–560, 1993.

[13] 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.

[14] G. Jordan. Adaptive smoothing of valleys in DEMs using TIN interpolation from ridgeline elevations: An application to morphotectonic aspect analysis. *Comput. Geosci.*, 33(4):573–585, 2007.

[15] A. A. Khanban and A. Edalat. Computing Delaunay triangulation with imprecise input data. In *Proc. 15th Canadian Conference on Comput. Geom.*, pages 94–97, 2003.

[16] 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.

[17] J. King. A 4-approximation algorithm for guarding 1.5-dimensional terrains. In *LATIN 2006: Theoretical Informatics, 7th Latin American Symposium, Valdivia, Chile, March 20-24, 2006*, pages 629–640, 2006.

[18] J. King. VC-dimension of visibility on terrains. In *Proc. 20th Canadian Conference on Comput. Geom.*, pages 27–30, 2008.

[19] D. T. Lee and F. P. Preparata. Euclidean shortest paths in the presence of rectilinear barriers. *Networks*, 14:393–410, 1984.

[20] M. Löffler and M. van Kreveld. Largest bounding box, smallest diameter, and related problems on imprecise points. In *Proc. 10th Workshop on Algorithms and Data Structures*, LNCS 4619, pages 447–458, 2007.

[21] N. Megiddo. Applying parallel computation algorithms in the design of serial algorithms. *J. ACM*, 30(4):852–865, 1983.

[22] M.M.Mesmoudi, L. Floriani, and P.Magillo. Morphological analysis of terrains based on discrete curvature and distortion. In *Proceedings of the 16th ACM International Conference on Geographic Information Systems (ACM-GIS)*, pages 415–418, 2008.

[23] E. Moet, M. van Kreveld, and A. F. van der Stappen. On realistic terrains. In *Proc. 22nd Annu. ACM Sympos. Comput. Geom.*, pages 177–186, 2006.

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

[25] 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.

[26] 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.

[27] G. Taubin. Discrete surface signal processing: The polygon as the surface element. In *Proceedings of the International NSF-ARPA Workshop on Object Representation in Computer Vision*, pages 167–175, London, UK, 1995. Springer-Verlag.

[28] K. Wang, C.-P. Lo, G. A. Brook, and H. R. Arabnia. Comparison of existing triangulation methods for regularly and irregularly spaced height fields. *International Journal of Geographical Information Science*, 15(8):743 – 762, 2001.