ELSEVIER

Available online at www.sciencedirect.com

SCIENCE DIRECT®

Theoretical Computer Science 315 (2004) 371–404

Theoretical Computer Science

www.elsevier.com/locate/tcs

Approximating shortest path for the skew lines problem in time doubly logarithmic in 1/epsilon

D. Burago[a,1], D. Grigoriev[b,*,2], A. Slissenko[c,3]

[a] *Department of Mathematics, Penn State University, University Park, PA 16802, USA*

[b] *Institut Mathématique de Rennes, Université Rennes 1, Beaulieu 35042, Rennes, France*

[c] *Laboratory for Algorithmics, Complexity and Logic, Department of Informatics, University Paris 12, 61 Av. du Gén. de Gaulle, 94010, Créteil, France*

Abstract

We consider two three-dimensional situations when a polytime algorithm for approximating a shortest path can be constructed. The main part of the paper treats a well-known problem of constructing a shortest path touching lines in $\mathbb{R}^3$: given a list of straight lines $L = (L_1, \ldots, L_n)$ in $\mathbb{R}^3$ and two points $\boldsymbol{s}$ and $\boldsymbol{t}$, find a shortest path that, starting from $\boldsymbol{s}$, touches the lines L_i in the given order and ends at $\boldsymbol{t}$. We remark that such a shortest path is unique. We show that it can be length–position ε-approximated (i.e. both its length and its position can be found approximately) in time $(Rn/\tilde{d}\tilde{\alpha})^{16} + \mathrm{O}(n^2 \log\log 1/\varepsilon)$, where $\tilde{d}$ is the minimal distance between consecutive lines of L, $\tilde{\alpha}$ is the minimum of sines of angles between consecutive lines, and R is the radius of a ball where the initial approximation can be placed (such a radius can be easily computed from the initial data).

As computational model we take real RAM extended by square and cubic roots extraction. This problem of constructing a shortest path touching lines is known for quite some time to be a challenging problem. The existing methods for approximating shortest paths based on adding Steiner points which form a grid and subsequently applying Dijkstra's algorithm for finding a shortest path in the grid, provide a complexity bound which depends polynomially on $1/\varepsilon$, while our algorithm for the problem under consideration has complexity linear in $\log\log 1/\varepsilon$. Our algorithm is motivated by the observation that the shortest path in question is a geodesic in a certain length space of non-positive curvature (in the sense of A.D. Alexandrov), and it relies on the (elementary) theory of CAT(0)-spaces.

*Corresponding author.

E-mail addresses: burago@math.psu.edu (D. Burago), dima@math.univ-rennes1.fr (D. Grigoriev), slissenko@univ-paris12.fr (A. Slissenko).

[1] Partially supported by an Alferd P. Sloan Fellowship and NSF Grant DMS-9803129.

[2] Member of St-Petersburg Steklov Mathematical Institute, Russian Academy of Sciences, St-Petersburg, Russia.

[3] Member of St-Petersburg Institute for Informatics, Russian Academy of Sciences, St-Petersburg, Russia.

0304-3975/$ - see front matter
doi:10.1016/j.tcs.2004.01.014

In the second part of the paper we analyze very simple grid approximations. We assume that a parameter $a > 0$ describing separability of obstacles is given and the part of a grid with mesh size a outside the obstacles is built (for semi-algebraic obstacles all these pre-calculations are polytime). We show that there is an algorithm of time complexity $O((1/a)^6)$ which, given a-separated obstacles in a unit cube, finds a path (between given vertices $\boldsymbol{s}$ and $\boldsymbol{t}$ of the grid) whose length is bounded from above by $(84\pi^* + 96a)$, where π^* is the length of a shortest path. On the other hand, as we show by an example, one cannot approximate the length of a shortest path better than $7\pi^*$ if one uses only grid polygons (constructed only from grid edges). For semi-algebraic obstacles our computational model is bitwise. For a general type of obstacles the model is bitwise modulo constructing the part of the grid admissible for our paths. Observe that the existing methods for approximating shortest paths are not directly applicable for *semi-algebraic* obstacles since they usually place the Steiner points forming a grid on the edges of polyhedral obstacles.

Keywords: Shortest path; Skew lines problem

1. Introduction

We consider two three-dimensional situations when a polytime algorithm for approximating shortest path can be constructed. The main part of the paper concerns a known problem of constructing a shortest path touching lines in $\mathbb{R}^3$ *in a specified order*: given a list of straight lines $L = (L_1, \ldots, L_n)$ in three-dimensional space and two points $\boldsymbol{s}$ and $\boldsymbol{t}$, find a shortest path that starts from $\boldsymbol{s}$, touches the lines L_i in the given order and ends at $\boldsymbol{t}$. We will call this problem *the skew lines problem* (cf. [22]). The second situation is, in a way, more simple: we look for a length approximation of a shortest path (i.e. for a path such that its length, but not necessarily its position, is close to the length of shortest paths) amidst obstacles that are separated.

1.1. Related results

The general problem of constructing a shortest path or even a 'sufficiently good approximation' for such a path is well known [18]. It was shown to be NP-hard even for convex polyhedral obstacles [9] (by "polyhedral obstacles" we mean unions of polyhedra). The skew lines problem in $\mathbb{R}^3$ is also well known. This problem is mentioned in [22] as, presumably, representing the basic difficulties in constructing shortest paths in three-dimensional Euclidean space. The paper [22] states "For example, there is no efficient algorithm known for finding the shortest path touching several such lines in a specified order", where "such lines" means "skew lines in three-dimensional space". The same problem of finding a shortest path touching straight lines was mentioned in a survey [23] as presumably difficult, though a possibility of using numerical methods was mentioned without any elaboration.

In the problem of approximation of shortest paths we distinguish two types of approximations: length approximation and length–position approximation.

Length approximation means that, given an ε, we look for a path whose length is ε-close to the length of shortest path (ε-close may mean either additive or multiplicative error). *Position approximation* presumes that a path we wish to construct is in a given neighborhood of a shortest path. An ε-neighborhood of a path Π is a union of balls of radius ε centered at points of Π. To construct a length–position approximation means that, given an ε, we look for a path that is in an ε-neighborhood of a shortest path and whose length is ε-close to the length of shortest path.

In [9] one can find the following result that can be related to the complexity of position approximation: for the case of polyhedral obstacles, finding the sequence of edges touched by a shortest path is NP-hard. In the same article [9] it is proven that for polyhedral obstacles with the size of description N, determining $O(\sqrt{N})$ bits of the length of a shortest path is also NP-hard.

Concerning weaker approximations, for the case of polyhedral obstacles, paper [22] describes an algorithm that finds a path which is of length at most $(1+\varepsilon)$ times of the length of shortest path, and whose running time is polynomial in the size of the input data and $1/\varepsilon$. Considerable gaps in this paper have been fixed in [11] and the complexity of the algorithm [11] is roughly $O(n^4N/\varepsilon)$, where n is the total number of edges in the polyhedra. Further, another method was exhibited in [12] with complexity bound depending on n and on ε as $O(n^2/\varepsilon^4)$. An algorithm with complexity bound $O(n^4/\varepsilon^6)$ is designed in [15] which in addition, constructs for a fixed source $\boldsymbol{s}$ a preprocessed data structure allowing one to compute subsequently an approximation for any target $\boldsymbol{t}$ with the complexity bound $O(\log(n/\varepsilon))$.

For the easier problem of constructing a shortest path on a given polyhedral surface one can find a shortest path exactly. First, an algorithm with complexity $O(n^3 \log n)$ was exhibited in [24] in case of a convex surface, thereupon it was generalized in [19] to arbitrary polyhedral surfaces, and its complexity is $O(n^2 \log n)$. Finally, the latter algorithm was improved in [10] providing a complexity $O(n^2)$. To improve the quadratic complexity, several algorithms which give ε-approximations of a shortest path were produced: in [1] with complexity bound $O(n+1/\varepsilon^3)$ in case of a convex surface, then in [3] with complexity roughly $O(n/\varepsilon)$ in case of a simple polyhedral surface. In [15] for a fixed $\boldsymbol{s}$, a preprocessed data structure is constructed with complexity $O(n/\varepsilon^3)$ in case of a convex surface and with complexity $O(n^2+n/\varepsilon)$ in case of an arbitrary polyhedral surface, respectively, which allows one for any target $\boldsymbol{t}$ to output an approximation within complexity $O(\log(n/\varepsilon))$.

All the mentioned approximation results use an approach that places Steiner points on the edges of the polyhedral obstacles to form a grid and after that to find a shortest path in this grid applying the algorithm of Dijkstra [2]. Since the number of the placed Steiner points depends polynomially on n and on $1/\varepsilon$, the complexity of this approach should depend on n and on $1/\varepsilon$ as well.

The approach of [20] to solve the weighted region problem gives an algorithm whose running time depends polynomially on $\log 1/\varepsilon$. Combining the binary search with the local optimization as in [20] one may probably find an approximation for the skew lines problem; however, in any case, our algorithm is exponentially faster in $1/\varepsilon$ (and our text is shorter than that of [20]).

1.2. On computational models

In the present paper we also study the problem of approximating shortest paths in $\mathbb{R}^3$ trying to improve the computational complexity in some aspect. Even in simple two-dimensional situations, exact constructions of shortest paths involve substantial extensions of the computational model by operations like square root extraction or more powerful ones, see e.g. [13,14,17]. Note that a shortest path in $\mathbb{R}^3$ between two rational points, with even one cylindrical obstacle, may have transcendental length and necessitate using transcendental points in its construction [16]. The algorithm we propose for length–position approximation of the shortest path touching lines is based on methods of steepest descent. Each iteration of such a method involves measuring distances between points and other algebraic but not rational operations. If one uses a bitwise computational model, the error analysis becomes difficult and demands an extra considerable work. So to give a solution to the skew lines problem we exploit here an "extended" computational model over reals, which we make precise below.

As it was mentioned above, in this paper we present two results: the first one treats the skew lines problem, where we give a length–position approximation. The second result is an analysis of the complexity of a length approximation of a simple grid method to find a path amidst separated semi-algebraic obstacles. Though the obtained estimation shows that the quality of approximation is rather low, the simplicity of the method makes it worthy of analysis.

For our *main result*, concerning a length–position approximation of the shortest path touching lines in $\mathbb{R}^3$ our *computational model* is real RAMs [6] with square and cubic roots extraction. This model extends (by allowing also extracting cubic roots) the common model used (implicitly) in [1,3,10,12,15,19,20] which admits rational operations and extracting square roots. We mention also that in [11,22] a bitwise model was used which takes into account the bit complexity.

Our *second result* uses the *bitwise model* modulo constructing a grid; the latter can be done in polytime for semi-algebraic obstacles.

1.3. Our results and methods we use

We start with a remark that the shortest path Π^* that we seek, is unique. We show that this path Π^* can be length–position ε-approximated in time $(Rn/\tilde{d}\tilde{\alpha})^{16} + O(n^2 \log\log 1/\varepsilon)$, where $\tilde{d}$ is the minimal distance between consecutive lines of L, $\tilde{\alpha}$ is the minimum of sines of angles between consecutive lines, and R is the radius of a ball where the initial approximation can be placed (we have this formula under the condition that $\tilde{d} \leqslant 1$; it does not diminish the generality; without this condition one must take maximum of (35) and (39)). Such a satisfactory radius R can be easily computed from the initial data: $R \in [\sqrt{n}|\Pi^*|, 2n^{\frac{3}{2}}|\Pi^*|]$, where $|\Pi^*|$ is the length of Π^* (estimate (3) in Section 2). Observe that when $\tilde{\alpha} = 0$ or $\tilde{d} = 0$ it could happen that the gradient descent and Newton's method which are invoked in our algorithm, do not converge. (There are considerations how to treat these particular cases but they need another, geometrically more 'invariant' approach.)

The algorithm is based on the following geometric idea, which shows that there is a unique minimum for the problem in question, and it is a strict minimum (the latter is quantified by means of a convexity estimate). Given a sequence of lines, one can consider a sequence of copies of Euclidean space, and glue them into a "chain" by attaching "neighboring" spaces along the corresponding line. The resulting (singular) space happens to be of non-positive curvature (see [4,5] and the Section 2.1 below). Now a shortest path we want to construct is a geodesic in this new space, and this immediately implies its uniqueness (by the Cartan–Hadamard Theorem [4]). Furthermore, convexity comparisons for the distance functions in nonnegatively curved spaces allow us to estimate the rate of convergence in gradient descent methods. This approach is somewhat similar to applications of Alexandrov geometry to certain problems concerning hard ball gas models and other semi-dispersing billiard systems, see [7,8]. To avoid excessive use of Alexandrov geometry, we formulate the mentioned convexity property in terms of the Hessian of the distance function and prove it by direct computation.

The exposed geometric idea helps us to achieve the principal feature of our algorithm, namely that its complexity depends linearly on $\log\log(1/\varepsilon)$ rather than polynomially on $1/\varepsilon$ as in the existing algorithms from [1,3,11,12,15,22] based on the grid method (at the expense of a worse dependency on n and on other geometric parameters). It would be interesting to describe more situations when a better dependency of the complexity on $1/\varepsilon$ (logarithmic, cf. [20], or even double logarithmic) would be possible simultaneously with better dependency on geometric parameters.

However, our algorithm does not allow to solve the skew lines problem for bitwise models of computations. The latter would require, in particular, estimating the bitsize of the output data. For bitwise models one can consider the following two settings.

SkewLines *Bitwise exact*: Given a list of n lines and two points $\boldsymbol{s}$ and $\boldsymbol{t}$ in $\mathbb{R}^3$, find n algebraic numbers on the consecutive lines such that the path going through these points is the shortest path between $\boldsymbol{s}$ and $\boldsymbol{t}$ touching the lines.

SkewLines *Bitwise approximate*: Given a list of n lines and two points $\boldsymbol{s}$ and $\boldsymbol{t}$ in $\mathbb{R}^3$, and given an ε, find an ε-approximation to the shortest path between $\boldsymbol{s}$ and $\boldsymbol{t}$ touching the lines.

We believe that our result is a step towards solving (SkewLines Bitwise Approximate) problem for which it would be interesting to design an algorithm with complexity polynomial in $\log(1/\varepsilon)$.

Very likely the method we use can be generalized to find a shortest path touching cylinders in R^3 in a prescribed order.

It is straightforward (a rather general case of three-dimensional TSP with Neighborhoods [18]) that if the order of touching lines is not prescribed the problem of finding a shortest path becomes NP-hard. Indeed, take the lines to be parallel and place the points $\boldsymbol{s}$ and $\boldsymbol{t}$ at a plane orthogonal to these lines. Then the problem of finding a shortest path touching all the lines is equivalent to the Euclidean traveling salesman problem in which it is necessary to find a shortest path between $\boldsymbol{s}$ and $\boldsymbol{t}$ and passing all intersecting points of the lines with the plane and which is known to be NP-complete [21]. We can slightly incline lines if we wish not to have parallel ones.

Our second result is an analysis of very simple grid approximations. The question concerning possibilities of 'generalized' grid approximations is mentioned in the

conclusion of [11]. The matter is that the result of [11,22] is based on such a method. The initial idea used in [11,22] is straightforward: partition the edges of polyhedra that constitute the obstacles into sufficiently small segments and take them as vertices of a graph of visibility. Then determine whether two such segments are mutually visible, in which case connect them by edges. It is not a simple task to implement correctly this idea for the *bitwise* computational model because of the necessity to 'approximate approximations', and this was the main source of gaps in [22]. The primitive grid method we use avoids such difficulties even for semi-algebraic obstacles. But one cannot get approximations that are, in the worst case, better than $7\pi^*$, where π^* is the length of the shortest path. More precisely, our result is as follows.

When considering grid approximations we assume that a parameter $a>0$ describing a separability of obstacles and the part of a grid admissible for the paths is given, and $\boldsymbol{s}$ and $\boldsymbol{t}$ are vertices of the grid lying in the space admissible for the paths. We show that there is an algorithm that in the bitwise machine model has the running time $O((1/a)^6)$ and that for given a-separated obstacles in a unit cube finds a path (between given points $\boldsymbol{s}$ and $\boldsymbol{t}$) whose length is bounded from above as $(84\pi^* + 96a)$, where π^* is the length of shortest paths. On the other hand, we show by an example that one cannot approximate the length of shortest paths better than $7\pi^*$. We conjecture that 7 is the exact bound for the three-dimensional case for the method we consider.

We do not discuss in detail how to construct the parameter a and the part of the grid admissible for the paths. It depends on the type of obstacles. For semi-algebraic obstacles in $\mathbb{R}^3$ it can be done (precisely, in terms of algebraic numbers) in polytime by a bitwise machine. After that, the construction of a shortest path approximation deals only with natural numbers of a modest size.

As we mentioned above the methods of placing Steiner points and forming a grid developed in [1,3,11,12,15,22] cannot be *directly* applied to semi-algebraic obstacles since they place Steiner points just on the edges of polyhedral obstacles. On the other hand, in our simple grid method the dependency on the complexity of obstacles is reduced to the construction of the grid; after that the complexity of the algorithm depends only on the parameter a of the obstacles; this is in accordance with the proposal expressed in [3]: "...while studying the performance of geometric algorithms, geometric parameters (e.g. fatness, density, aspect ratio, longest, closest) should not be ignored...".

1.4. Structure of the paper

The paper is divided into two parts. Its structure is as follows.

The first and the main part of the paper deals with shortest paths touching lines. In Section 2 we give basic properties of the spaces under consideration, in particular, we explain what geometric properties ensure the uniqueness of the shortest path. Our main theorem is formulated in Section 2.5. In fact, this part is not needed for the proof as we reduce the problem to a problem of computing a minimum of a strictly convex function of many variables on a compact space. However, these geometric considerations, that are easy to apply, show the road to take, and they are very instructive indeed for the problem of shortest path that has rather various technical contexts and rarely

has a unique solution. Section 3 contains main technical estimates. The central point is to bound from below the eigenvalues of the Hessian of the path length function (Section 3.2, Proposition 2). This bound is crucial for the estimation of complexity of the gradient descent. The latter provides an initial approximation for the application of Newton's method that follows the gradient descent. Section 4 gives our approximation algorithm. It starts with a construction of an initial approximation, then applies a gradient descent and after that Newton's method. In the last Section 5 of Part 1 we estimate the complexity of the algorithm.

Part 2 presents our grid algorithm. The definition of separability is given in Section 6. The same section (Proposition 7) gives a lower bound for the probability to get well separated obstacles when randomly choosing balls as obstacles. Then in Section 7 we present the algorithm that constructs a length approximation.

Part 1: Shortest paths touching lines

2. Basic properties of shortest paths touching lines

Our construction is based on the following initial observation: we are seeking a shortest path in a simply connected metric space of non-positive curvature. This observation guarantees the uniqueness of the shortest path; moreover, it suggests that the distance functions in the space enjoys certain convexity properties, which will actually permit us to use the method of steepest descent to reach an approximate position of the shortest path.

As we mentioned in the Introduction, formally speaking we can omit these considerations and go directly to the estimations of convexity of the length function within our parametrization of paths. However, these geometric arguments are simple and direct, and can be used in other problems of construction of shortest paths. We remark that usually the uniqueness of the shortest path does not take place though the metric function is quite good (e.g. a sphere as an obstacle in $\mathbb{R}^3$).

First, we recall some known facts about spaces of non-positive curvature.

2.1. On spaces of non-positive curvature

Let $\mathscr{M}$ be a simply connected metric space. Denote by $|XY|_{\mathscr{M}}$, or by $|XY|$ if $\mathscr{M}$ is clear from the context, the distance between points X and Y in $\mathscr{M}$. A *path* in $\mathscr{M}$ between two points A and B is a continuous mapping φ of $[0,1]$ into $\mathscr{M}$ such that $\varphi(0)=A$ and $\varphi(1)=B$. The length of a path φ can be defined as the limit of sums $\sum_{0\leqslant i\leqslant(n-1)}|\varphi(i(1/n))\varphi((i+1)1/n)|$ as $n\to\infty$. A *shortest path* between two points is a path connecting these points and having the smallest length. We suppose that for any two points of $\mathscr{M}$ there exists a shortest path between them.

By the *angle* between two intervals going out of the same point on the plane $\mathbb{R}^2$ we mean the (nondirected) angle in $[0,\pi]$. Given 3 points A, B and C the angle between intervals AB and AC will be denoted by $\angle BAC$ or $\angle CAB$, the angle between 2 intervals ξ and ζ will be denoted by $\angle\xi\zeta$, and the angle between 2 vectors will

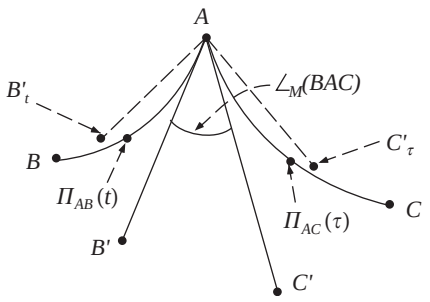

Fig. 1. Comparing triangles in $\mathscr{M}$ and on the plane.

be treated as the angle between two intervals obtained from the vectors by translating them to the same origin. If different spaces are considered in the same context the angle will be denoted by $\angle_{\mathscr{M}}$.

Take any three points A, B and C in $\mathscr{M}$ (Fig. 1). Let Π_{AB} and Π_{AC} be shortest paths respectively between A and B, and A and C. On these paths we take two arbitrary points $\Pi_{AB}(t)$ and $\Pi_{AC}(\tau)$, where t and τ are the values of parameters determining these points.

Then we choose any point A' in $\mathbb{R}^2$ and take in $\mathbb{R}^2$ two points B'_t and C'_τ that satisfy the equalities:

$$|A'B'_t|_{\mathbb{R}^2} = |\Pi_{AB}([0,t])|_{\mathscr{M}}, \quad |A'C'_\tau|_{\mathbb{R}^2} = |\Pi_{AC}([0,\tau])|_{\mathscr{M}},$$

$$|B'_tC'_\tau|_{\mathbb{R}^2} = |\Pi_{AB}(t)\Pi_{AC}(\tau)|_{\mathscr{M}}.$$

We are interested in the triangle $\triangle A'B'_tC'_\tau$ in the plane $\mathbb{R}^2$.

Denote by $\theta(t,\tau)$ the angle between the intervals $A'B'_t$ and $A'C'_\tau$ in $\mathbb{R}^2$. The *angle* between Π_{AB} and Π_{AC} at A is defined as $\lim_{t,\tau\to 0}\theta(t,\tau)$, if the latter exists.

Define the *angle* at A to B and C, or shorter $\angle_{\mathscr{M}}(BAC)$ as the supremum of angles at A between a shortest path from A to B and a shortest path from A to C.

Consider again three points A, B and C in $\mathscr{M}$. Draw in $\mathbb{R}^2$ a triangle $A'B'C'$ determined by the following conditions:

$$|A'B'|_{\mathbb{R}^2} = |AB|_{\mathscr{M}}, \quad |A'C'|_{\mathbb{R}^2} = |AC|_{\mathscr{M}} \quad \text{and} \quad \angle_{\mathscr{M}}(BAC) = \angle_{\mathbb{R}^2}(B'A'C').$$

By definition, the space $\mathscr{M}$ is of *non-positive curvature* if the angle $\angle_{\mathscr{M}}(BAC)$ exists and $|B'C'|_{\mathbb{R}^2} \leqslant |BC|_M$ for any three points A, B and C of $\mathscr{M}$.

It is known the following (see, for instance, [4]).

Uniqueness of the shortest path: In any simply connected space of non-positive curvature, there is a unique shortest path between any two points.

2.2. The configuration space

Let $L=(L_1,\ldots,L_n)$ be a list of straight lines in $\mathbb{R}^3$, and $\boldsymbol{s}$ and $\boldsymbol{t}$ two points. Without loss of generality, we assume that these points are not on the mentioned lines. Clearly,

the shortest paths touching the lines of L in the given order are polygonal chains (broken lines).

The space where we are looking for such a shortest broken line can be obtained by the following operation. For each pair (L_i, L_{i+1}) take a copy $\mathscr{R}_i$ of $\mathbb{R}^3$. Glue consecutively $\mathscr{R}_i$ and $\mathscr{R}_{i+1}$ along the line L_{i+1}. Denote the obtained space by $\mathscr{R}_L$. Each space $\mathscr{R}_i$ is a space of non-positive curvature, and they are glued along isometric convex sets. The Reshetnyak's theorem (see [5]) implies that the resulting space $\mathscr{R}_L$ is of non-positive curvature.

The next two paragraphs are not necessary to describe the main algorithm, but rather to clarify some geometric background behind it. Similarly to nonpositively curved spaces, one can consider spaces with other curvature bounds (say, spaces with curvature bounded above by -1). They feature even stronger convexity of distance functions, and algorithms for constructing shortest paths converge in such spaces even faster. Shortest paths in such spaces have remarkable *trapping properties*, as it is asserted by the classic Morse Lemma. The spaces constructed in this paper by gluing several copies of Euclidean space along a number of lines are nonpositively curved. Of course, they are *not* spaces of curvature bounded above by any negative number, for they have flat parts (regions in the Euclidean spaces used for this construction). However, if one considers two points A, B in different copies of Euclidean space and such that the shortest path between these points meets al least two lines used to glue the spaces together, and the gluing lines are not parallel, then a distance function $|A\cdot|$ restricted to a neighborhood of B has the same convexity properties as if the space was of curvature bounded above by a negative constant. This allows us to treat such shortest paths as if they were in a space with negative upper curvature bound.

Note that it is easy to see geometrically where this negative curvature comes from if the gluing lines are skew lines, and what happens if the gluing lines are parallel. First assume that the gluing lines are parallel. Consider the following plane lying in our space and passing through A and B. It is made of three parts: a half-plane containing A and bounded by the first gluing line (this part lies in the same copy of Euclidean spaces as A), then a strip between the two gluing lines (it lies in the copy of Euclidean space which the shortest path connecting A and B passes through), and finally a half-plane containing B and bounded by the second gluing line. These three parts indeed form a plane (a surface isometric to a plane) lying in our space totally geodesically. It contains A and B, and hence the convexity properties of the distance function $|A\cdot|$ restricted to a neighborhood of B are no better than in a flat space. To see where the negative curvature comes from in case of skew lines, consider a short Euclidean segment Γ containing B and parallel to the second gluing line. Connect all points of Γ with A. If the gluing lines were parallel, we would get just a part of the plane discussed above (more precisely, a triangular region). However, in case of skew lines the family of segments connecting A with Γ form a non-flat surface. This surface has two flat parts (a triangle with a vertex at A, and a trapezoid (one of the bases of the trapezoid is Γ, and the other one lies in the gluing line)). However, the third part of this surface is a ruled surface connecting two segments on the gluing lines, and this is indeed a negatively curved surface.

A similar construction can be used to obtain a space of non-positive curvature to seek shortest paths consecutively touching convex cylinders $Z=(\zeta_1,\dots,\zeta_n)$ in $\mathbb{R}^N$. In this case the gluings are made along the corresponding cylinders.

Consider the shortest path problem in $\mathscr{R}_L$, where two points $\boldsymbol{s}$ and $\boldsymbol{t}$ are fixed.

From the uniqueness of shortest path between two points in a simply connected nonpositively curved space we immediately conclude that:

Claim 1. *There is a unique shortest path between any two given points in $\mathscr{R}_L$; hence a shortest path touching the lines in a prescribed order and connecting two given points is unique.*

Furthermore, a standard distance comparison argument with Euclidean development of a broken line representing an approximate solution implies that length-approximation guarantees position-approximation (this is proven in detail in Lemma 4, Section 4.1):

Claim 2. *If the length of the actual shortest path is L, every path whose length is less than $L+\varepsilon$ belongs to the $\sqrt{L\varepsilon/2}$-neighborhood of the actual shortest path.*

To simplify our constructions we will assume that (see Assumption 2.5 in Section 2.5) *the consecutive lines of L are pairwise disjoint*. The case with intersecting consecutive lines needs a particular attention when our approximation to the shortest path is close to a point of intersection of lines.

By a *path* in $\mathscr{R}_L$ we mean a polygon P consisting of $n+1$ links (straightline intervals), denoted by P_i, $1\leqslant i\leqslant n$, connecting $\boldsymbol{s}$ with $\boldsymbol{t}$ via consecutive lines, i.e. the link P_1 connects $\boldsymbol{s}$ with L_1, the link P_{i+1} connects L_i with L_{i+1} $(1\leqslant i\leqslant n-1)$, and the link P_{n+1} connects L_n with $\boldsymbol{t}$. When speaking about the *order of points* on a path we mean the order corresponding to going along the path starting from $\boldsymbol{s}$. Thus any link of a path is directed from $\boldsymbol{s}$ to $\boldsymbol{t}$.

Notice that for a shortest path P the angle between P_i (incident ray) and L_i must be equal to the angle between L_i and P_{i+1} (emergent ray). Clearly, the incident angle being fixed, the emergent rays constitute a cone (which may have up to two intersection points with L_{i+1}—this means that a geodesic in our configuration space can branch, having two different continuations after passing through a line L_i).

2.3. Technical notations

Throughout the text we use some notations that we summarize here—the reader can consult the list when coming across a new notation. The list is divided into 4 parts: notations concerning vectors, lengths, angles, paths. Some more local notation will appear later.

Notation 1 (Vectors, lines, points).

- $\langle U,V\rangle$ is the scalar product of vectors U and V.

- $|U|$ or $\|U\|$ is the $\mathbb{L}_2$-norm of a vector U (the length of U).
- $\|A\|$, where A is a matrix, is the spectral norm of A. For positive definite symmetric matrices (our main case) $\|A\| = \sup\{\langle AU, U\rangle : \|U\| = 1\}$.
- $L = (L_1, \ldots, L_n)$ is the list of straight lines in $\mathbb{R}^3$ we consider, and $\boldsymbol{s}$ and $\boldsymbol{t}$ are respectively the first and the last point to connect by a polygon touching the lines of L in the order of the list.
- Λ_i^0 is a fixed vector determining a point on L_i. We use it as the origin of coordinates on L_i.
- Λ_i is a unit vector defining the direction on L_i.
- $T = (t_1, \ldots, t_n)$ is a list of n reals that will serve as coordinates respectively on $L_1, \ldots, L_n$. Our applications of gradient and Newton's methods take place in the space $\mathbb{R}^n$ of such points T.
- To make the notations more uniform we assume that $\Lambda_0^0 = \boldsymbol{s}$, $\Lambda_{n+1}^0 = \boldsymbol{t}$, and that always $t_0 = t_{n+1} = 0$.
- $W_i(t_i) =_{\mathrm{df}} (\Lambda_i^0 + t_i \Lambda_i)$; clearly, it is a point on L_i determined by a single real parameter t_i. Note that $W_0(t_0) = \boldsymbol{s}$ and $W_{n+1}(t_{n+1}) = \boldsymbol{t}$.
- $V_i = V_i(t_i, t_i) =_{\mathrm{df}} W_{i+1}(t_{i+1}) - W_i(t_i)$ is a vector connecting W_i with W_{i+1} (in this order), $0 \leqslant i \leqslant n$.
- D_0 is a vector from $\boldsymbol{s}$ to L_1 *perpendicular* to L_1, D_n is a vector from L_n to $\boldsymbol{t}$ *perpendicular* to L_n, and D_i is a vector connecting L_i and L_{i+1} and *perpendicular* to both of them.
- For a directed interval σ in $\mathbb{R}^3$ we denote by σ^- and σ^+ its beginning and its end, respectively.

Notation 2 (Lengths).

- ρ will be used to denote various $\mathbb{L}_2$-distances, e.g. $\rho(\boldsymbol{s}, L_i)$ will denote the distance between $\boldsymbol{s}$ and L_i.
- $v_i =_{\mathrm{df}} |V_i|$ is the length of V_i, $0 \leqslant i \leqslant n$.
- d_i is the *square* of the distance between L_i and L_{i+1} for $1 \leqslant i \leqslant n-1$, i.e. $d_i = \rho(L_i, L_{i+1})^2$, $d_0 =_{\mathrm{df}} \rho(\boldsymbol{s}, L_1)^2$, $d_n =_{\mathrm{df}} \rho(L_n, \boldsymbol{t})^2$.
- $d =_{\mathrm{df}} \min\{d_i : 0 \leqslant i \leqslant n\}$.
- $\tilde{d} =_{\mathrm{df}} +\sqrt{d}$. To simplify formulas we assume that $\tilde{d} \leqslant 1$; we do not loose the generality as we can change the coordinates in an appropriate way, and within our model of computation this change does not affect the complexity. In the proofs we will also give formulas without this assumption.

Notation 3 (Angles).

- $\angle ABC$, where A, B and C are points, is the angle at B in the triangle determined by these 3 points. In fact, we will consider angles in $\mathbb{R}^3$, though formally speaking we are in configuration space $\mathscr{R}_L$ defined in Section 2.2.
- $\angle VV'$, where V and V' are vectors, is the angle in $[0, \pi]$ between these vectors.
- $\alpha_i =_{\mathrm{df}} (\sin \angle \Lambda_i \Lambda_{i+1})^2$, $1 \leqslant i \leqslant n-1$, $\alpha =_{\mathrm{df}} \min\{\alpha_i : 1 \leqslant i \leqslant n-1\}$.
- $\tilde{\alpha} =_{\mathrm{df}} +\sqrt{\alpha}$.
- $\beta_i =_{\mathrm{df}} (\cos \angle \Lambda_i \Lambda_{i+1})^2$, $\tilde{\beta}_i =_{\mathrm{df}} \cos \angle \Lambda_i \Lambda_{i+1}$, $1 \leqslant i \leqslant n-1$.

Notation 4 (Paths).

- A path $\Pi(T)$ is determined by a list of reals T that gives the consecutive points $W(t_i)$ of $\Pi(T)$ on the lines of L_i connected by this path. Thus, $W_0(t_0)=\boldsymbol{s}$, $W_i(t_i)=\Pi(T)\cap L_i$ for $1\leqslant i\leqslant n$ and $W_{n+1}(t_{n+1})=\boldsymbol{t}$, where W_i, t_i and L_i are defined in Notations 1.
- By P_i for a path P we denote the ith link of P. We consider P_i as an interval directed from L_i to L_{i+1}, $0\leqslant i\leqslant n$. Regarded as a vector, it coincides with V_i.
- Π^* is the shortest path between $\boldsymbol{s}$ and $\boldsymbol{t}$.
- $T^*=(t_1^*,\ldots,t_n^*)$ is the point in $\mathbb{R}^n$ defining Π^*, i.e. $\Pi^*=\Pi(T^*)$.
- Π^0 is the *initial* approximation of Π^* defined below in Section 2.4.
- T^0 is the point in $\mathbb{R}^n$ defining Π^0, i.e. $\Pi^0=\Pi(T^0)$.
- $r=_{\mathrm{df}}|\Pi^0|$ is the length of the initial approximation. It will be clear from its construction that $\tilde{d}<r$. Moreover, without loss of generality, just to simplify some formulas that will appear in the estimations of complexity, we suppose that $r>1$.
- $R=_{\mathrm{df}} r\sqrt{n}$ is the radius of a ball in $\mathbb{R}^n$ that will contain all paths in question.
- $\mathscr{B}$ is the closed ball in $\mathbb{R}^n$ of radius R centered at T^0; it contains all paths under consideration.

2.4. Initial approximation for the shortest path

Denote by P_i^0 the base of the perpendicular from $\boldsymbol{s}$ to L_i.

Take as initial approximation to the shortest path the polygon Π^0 that starts at $\boldsymbol{s}$, then goes to P_1^0, then to P_2^0 and so on to P_n^0 and finally to $\boldsymbol{t}$. Clearly $\Pi^0\subseteq\mathscr{B}$ and

$$|P_iP_{i+1}| \leqslant (|\boldsymbol{s}P_i|+|\boldsymbol{s}P_{i+1}|) \quad \text{for } 1\leqslant i\leqslant n-1. \tag{1}$$

The length of Π^0 can be bounded in terms of the length of the shortest path Π^* as

$$|\Pi^*| \leqslant |\Pi^0| = r \leqslant 2\sum_{i=1}^{n-1}\rho(\boldsymbol{s},L_i)+|\boldsymbol{st}| \leqslant 2n|\Pi^*|. \tag{2}$$

The first inequality follows from the fact that the length of any path connecting $\boldsymbol{s}$ and $\boldsymbol{t}$ and touching lines from L is greater than or equal to $|\Pi^*|$, the second inequality is implied by (1), and the third one is due to the fact that the distance from $\boldsymbol{s}$ to any line is not greater than $|\Pi^*|$.

For R estimations (2) give

$$\sqrt{n}|\Pi^*| \leqslant R \leqslant 2n^{3/2}|\Pi^*|. \tag{3}$$

We remark that if $L_i=\{W_i^0+t\Lambda_i : t\in\mathbb{R}\}$, then $P_i=W_i^0+t_i^0\Lambda_i$ is determined by $\langle P_i,\Lambda_i\rangle=0$, and thus,

$$P_i = W_i^0-\langle W_i^0,\Lambda_i\rangle\Lambda_i. \tag{4}$$

The parameter $R=|\Pi^0|\sqrt{n}$ intervenes in our estimation of complexity, and one may ask what is the dependence of R on the length of the input. Suppose that the input lines are represented by W_i^0 and Λ_i, and the maximum of the lengths of involved numbers is v. Then the length of the input is $O(nv)$. Clear that in the worst case the value R is exponential in this input length. But it can be as well linear, logarithmic and so on. The precise bound on R is an open question.

2.5. Main Theorem

Hereafter we *suppose* that

Assumption. $\tilde{\alpha} > 0$ and $\tilde{d} > 0$.

The first inequality says that *there are no consecutive parallel lines among* L_i, and the second one says that *the consecutive lines do not intersect and the points* $\boldsymbol{s}$ *and* $\boldsymbol{t}$ *are not on the lines.*

Now we can formulate our main result:

Main Theorem. *There is an algorithm that finds a length–position* ε*-approximation of the shortest path touching lines in time*

$$\left(\frac{Rn}{\tilde{d}\tilde{\alpha}}\right)^{16} + \mathrm{O}\left(n^2 \log\log\frac{1}{\varepsilon}\right),$$

where R, $\tilde{d}$ *and* $\tilde{\alpha}$ *are defined above in Notations* 4, 2, 3.

Remark 1. The complexity bound of the Main Theorem depends actually on ε if the latter is sufficiently small (see the end of Section 5.2), namely, $\varepsilon < \mathrm{O}(\tilde{d}^4\tilde{\alpha}^2/r^3\sqrt{n})$. Otherwise, the second summand of the estimation which contains ε is dominated by the first summand in the complexity bound. As we remarked in the Introduction, in order to simplify the first summand we supposed, without loss of generality, that $\tilde{d} \leqslant 1$; without this assumption one must take maximum of (35) and (39) which have a similar form.

3. Bounds on eigenvalues of the Hessian of the path length

This section contains technical bounds on the norm of derivatives of the path function, in particular the main estimation, namely, a lower bound on the eigenvalues of the Hessian ('second derivative') of the path length. The fact that the length function is at "least as convex as in Euclidean space" follows from the curvature bound. We will make direct computation though, due to the fact that we use a particular coordinate system.

Consider a path $\Pi(T) = \Pi(t_1, \ldots, t_n)$ represented by points W_i, $0 \leqslant i \leqslant n+1$ (Notations 4 from Section 2.3).

Notations 5.

- $f(T) =_{\mathrm{df}} |\Pi| = \sum_{i=0}^{n} v_i$ is the length of $\Pi(T)$, where v_i is defined in Notations 2.
- $g(T) =_{\mathrm{df}} f'(T) =_{\mathrm{df}} \mathbf{grad} f(T) = (\frac{\partial f}{\partial t_1}(T), \ldots, \frac{\partial f}{\partial t_n}(T))$ is its gradient (first derivative).

$$\begin{bmatrix}
\frac{\partial^2 v_0}{\partial t_1^2}+\frac{\partial^2 v_1}{\partial t_1^2} & \frac{\partial^2 v_1}{\partial t_1 \partial t_2} & 0 & \dots & 0 & 0 \\
\frac{\partial^2 v_1}{\partial t_1 \partial t_2} & \frac{\partial^2 v_1}{\partial t_2^2}+\frac{\partial^2 v_2}{\partial t_2^2} & \frac{\partial^2 v_2}{\partial t_2 \partial t_3} & \dots & 0 & 0 \\
0 & \frac{\partial^2 v_2}{\partial t_2 \partial t_3} & \frac{\partial^2 v_2}{\partial t_3^2}+\frac{\partial^2 v_3}{\partial t_3^2} & \dots & 0 & 0 \\
\dots\dots & \dots\dots & \dots\dots & \dots & \dots\dots & \dots\dots \\
0 & 0 & 0 & \dots & \frac{\partial^2 v_{n-1}}{\partial t_{n-1} \partial t_n} & \frac{\partial^2 v_{n-1}}{\partial t_n^2}+\frac{\partial^2 v_n}{\partial t_n^2}
\end{bmatrix}$$

Fig. 2. Hessian (of path length) Δ.

Let us begin with the formula for g, that is the first variation of length. For $1 \leqslant i \leqslant n$ we set

$$\pi_i = \frac{\partial f}{\partial t_i} = \frac{\partial}{\partial t_i}(v_{i-1}+v_i) = \frac{\partial v_{i-1}}{\partial t_i} + \frac{\partial v_i}{\partial t_i} = \frac{\langle V_{i-1}, \Lambda_i\rangle}{v_{i-1}} - \frac{\langle V_i, \Lambda_i\rangle}{v_i} \tag{5}$$

and

$$g = (\pi_1(t_1,t_2), \pi_2(t_1,t_2,t_3), \dots, \pi_{n-1}(t_{n-2},t_{n-1},t_n), \pi_n(t_{n-1},t_n)) \tag{6}$$

(which stresses the variables on which each component of g depends).

3.1. The second variation formula for the path length

The Hessian of $|\Pi|$, which we will denote $\Delta = \Delta(T)$ (Jacobian of $|\Pi|'$), looks as shown in Fig. 2.

The matrix of Δ is symmetric 3-diagonal with positive diagonal entries. Here are the formulas for derivatives involved in Δ in arbitrary coordinates (for further references).

$$\begin{aligned}
\frac{\partial^2 v_i}{\partial t_i \partial t_{i+1}} &= \frac{\partial}{\partial t_i}\left(\frac{\langle V_i, \Lambda_{i+1}\rangle}{v_i}\right) = \frac{\langle \frac{\partial V_i}{\partial t_i}, \Lambda_{i+1}\rangle}{v_i} - \frac{\langle V_i, \Lambda_{i+1}\rangle\langle V_i, \frac{\partial V_i}{\partial t_i}\rangle}{v_i^3} \\
&= -\frac{\langle \Lambda_i, \Lambda_{i+1}\rangle}{v_i} + \frac{\langle V_i, \Lambda_{i+1}\rangle\langle V_i, \Lambda_i\rangle}{v_i^3}
\end{aligned} \tag{7}$$

$$= -\frac{1}{v_i}(\cos \angle \Lambda_i \Lambda_{i+1} - \cos \angle V_i \Lambda_{i+1} \cdot \cos \angle V_i \Lambda_i \rangle), \tag{8}$$

$$\frac{\partial^2 v_i}{\partial t_i^2} = \frac{1}{v_i}\left(1 - \frac{\langle V_i, \Lambda_i\rangle^2}{v_i^2}\right) = \frac{(\sin \angle V_i, \Lambda_i)^2}{v_i}, \tag{9}$$

$$\frac{\partial^2 v_i}{\partial t_{i+1}^2} = \frac{1}{v_i}\left(1 - \frac{\langle V_i, \Lambda_{i+1}\rangle^2}{v_i^2}\right) = \frac{(\sin \angle V_i, \Lambda_{i+1})^2}{v_i}. \tag{10}$$

3.2. Lower and upper bounds on the eigenvalues of the Hessian of the path length

Decompose Δ into a sum of matrices with 2×2-blocks corresponding to links of Π plus two 1×1-blocks for the first and the last links.

Notation 6.

- $\Delta_0 =_{\mathrm{df}} \left[\dfrac{\partial^2 v_0}{\partial t_1^2}\right]$.
- $\Delta_i =_{\mathrm{df}} \begin{bmatrix} \dfrac{\partial^2 v_i}{\partial t_i^2} & \dfrac{\partial^2 v_i}{\partial t_i \partial t_{i+1}} \\ \dfrac{\partial^2 v_i}{\partial t_i \partial t_{i+1}} & \dfrac{\partial^2 v_i}{\partial t_{i+1}^2} \end{bmatrix}$ for $1 \leqslant i \leqslant n-1$.
- $\Delta_n =_{\mathrm{df}} \left[\dfrac{\partial^2 v_n}{\partial t_n^2}\right]$.
- *$\tilde{\Delta}_i$ is the $n \times n$-matrix consisting of Δ_i situated at its proper place in Δ and with zeros at all other places. More precisely, $(1,1)$-element of Δ_i is placed at (i,i) for $1 \leqslant i \leqslant n$ and at $(1,1)$ for $i = 0$.*

Within these notations

$$\Delta = \sum_{0 \leqslant i \leqslant n} \tilde{\Delta}_i. \tag{11}$$

3.2.1. Lower bound on eigenvalues of Δ

Strict convexity of metric implies that the second derivative Δ is positive definite. However, we need constructive upper and lower bounds on eigenvalues of Δ. So we do not use explicitly the just mentioned property of the metric—it will follow from the estimations below.

We start with a more difficult question of obtaining a lower bound on the least eigenvalue of Δ, i.e. on $\inf\{\langle \Delta U, U\rangle : \|U\| = 1\}$. To do it we reduce this problem to the same problem for matrices Δ_i.

Lemma 1. *The only eigenvalue of Δ_0 and that of Δ_n is greater than $d/2r^3$, and both eigenvalues of Δ_i for $1 \leqslant i \leqslant n-1$ are greater than $d\alpha/2r^3$. Thus $d\alpha/2r^3$ is a lower bound on the norm of Δ_i for all i (and for any t_i).*

Proof. As supposed in the beginning of Section 2.5 there are no consecutive parallel lines among L_i. Thus vectors Λ_i, Λ_{i+1} and D_i (see notations in Section 2.3) constitute a basis in $\mathbb{R}^3$, and we can represent the link V_i as $V_i = t_i \cdot \Lambda_i + t_{i+1} \cdot \Lambda_{i+1} + D_i$ (here t_i and t_{i+1} are different from those used in notations in Section 2.3).

To simplify computations rewrite this formula as $W = xA + yB + D$, where $W = V_i$, $x = t_i$, $A = \Lambda_i$, $y = t_{i+1}$, $B = \Lambda_{i+1}$ and $D = D_i$. For $i = 0$ we put $x = 0$, and for $i = n$ we put $y = 0$. And let $v = \sqrt{\langle W, W\rangle}$.

Within these notations $d_i = \langle D, D\rangle$, $D \perp A$ and $D \perp B$.

For $i = 0$ we have that the only eigenvalue of Δ_0 is its element $\partial^2 v_0/\partial t_1^2 = (1/v_0)(1 - \langle W, B\rangle^2/v_0^2)$. Here $\langle W, B\rangle = \langle yB + D, B\rangle = y$, $\langle W, W\rangle = v_0^2 = \langle yB + D, yB + D\rangle = y^2 + d_0$ and thus

$$\|\Delta_0\| = \left|\frac{1}{v_0}\left(1 - \frac{y^2}{v_0^2}\right)\right| = \frac{1}{v_0^3}(y^2 + d_0 - y^2) = \frac{d_0}{v_0^3} > \frac{d}{r^3}.$$

Similarly for $i = n$ we have $\|\Delta_n\| \geqslant d_n/v_n^3 > d/r^3$.

Consider $i \in \{1, \ldots, n-1\}$. Notice that in our notations

$$v^2 = \langle xA + yB + D, xA + yB + D\rangle = x^2 + 2xy\tilde{\beta}_i + y^2 + d_i$$

and

$$\Delta_i = \begin{bmatrix} \dfrac{\partial^2 v}{\partial x^2} & \dfrac{\partial^2 v}{\partial x \partial y} \\ \dfrac{\partial^2 v}{\partial x \partial y} & \dfrac{\partial^2 v}{\partial y^2} \end{bmatrix}.$$

Now we compute the second derivative of v:

$$\begin{aligned} \frac{\partial^2 v}{\partial x^2} &= \frac{1}{v}\left(1 - \frac{\langle xA + yB + D, A\rangle^2}{v^2}\right) = \frac{1}{v^3}(v^2 - (x + y\tilde{\beta}_i)^2) \\ &= \frac{1}{v^3}(y^2 + d_i - (y\tilde{\beta}_i)^2) = \frac{1}{v^3}(y^2\alpha_i + d_i), \end{aligned}$$

$$\begin{aligned} \frac{\partial^2 v}{\partial y^2} &= \frac{1}{v}\left(1 - \frac{\langle xA + yB + D, B\rangle^2}{v^2}\right) = \frac{1}{v^3}(v^2 - (x\tilde{\beta}_i + y)^2) \\ &= \frac{1}{v^3}(x^2\alpha_i + d_i), \end{aligned}$$

$$\begin{aligned} \frac{\partial^2 v}{\partial x \partial y} &= -\frac{1}{v_i}\left(\tilde{\beta}_i - \frac{\langle xA + yB + D, B\rangle\langle xA + yB + D, A\rangle}{v^2}\right) \\ &= -\frac{1}{v^3}(\tilde{\beta}_i(x^2 + 2xy\tilde{\beta}_i + y^2 + d_i) - (x\tilde{\beta}_i + y)(x + y\tilde{\beta}_i)) \\ &= -\frac{1}{v^3}((x^2\tilde{\beta}_i + 2xy\tilde{\beta}_i^2 + y^2\tilde{\beta}_i + d_i\tilde{\beta}_i) - (x^2\tilde{\beta}_i + xy + xy\tilde{\beta}_i^2 + y^2\tilde{\beta}_i)) \\ &= -\frac{1}{v^3}(xy\tilde{\beta}_i^2 + d_i\tilde{\beta}_i - xy) = -\frac{1}{v^3}(d_i\tilde{\beta}_i - xy\alpha_i). \end{aligned}$$

Let us estimate the determinant

$$\begin{aligned}\det \Delta_i &= \frac{1}{v^6}\left((y^2\alpha_i + d_i)(x^2\alpha_i + d_i) - (d_i\tilde{\beta}_i - xy\alpha_i)^2\right)\\ &= \frac{1}{v^6}(x^2y^2\alpha_i^2 + d_i\alpha_i(x^2+y^2) + d_i^2 - d_i^2\tilde{\beta}_i^2 - x^2y^2\alpha_i^2 + 2xy\alpha_i d_i\tilde{\beta}_i)\\ &= \frac{1}{v^6}(d_i\alpha_i(x^2+y^2) + d_i^2\alpha_i + 2xy\alpha_i d_i\tilde{\beta}_i)\\ &= \frac{\alpha_i}{v^6}(d_i(x^2+y^2) + d_i^2 + 2xyd_i\tilde{\beta}_i)\\ &\geqslant \frac{\alpha_i}{v^6}(d_i(x^2+y^2) + d_i^2 - 2|x||y|d_i)\\ &\geqslant \frac{d_i^2\alpha_i}{v^6}.\end{aligned}$$

Note that $\operatorname{Tr}\Delta_i = (1/v^3)(\alpha_i(x^2+y^2)+2d_i)$. The smallest of two eigenvalues of Δ_i is not less than

$$\frac{\det\Delta_i}{\operatorname{Tr}\Delta_i} \geqslant \frac{v^3d_i^2\alpha_i}{v^6(\alpha_i(x^2+y^2)+2d_i)} \geqslant \frac{d_i^2\alpha_i}{v^3 2d_i} = \frac{d_i\alpha_i}{2v^3} > \frac{d\alpha}{2r^3}. \qquad \square$$

Proposition 2. *All eigenvalues of $\Delta(T)$ are greater than $d\alpha/2r^3$. Thus $\Delta(T)$ is positive definite, the metric function is strictly convex and $\|\Delta(T)\| \geqslant d\alpha/r^3$, and $\|\Delta^{-1}(T)\| \leqslant r^3/d\alpha$ for any T.*

Proof. Now we estimate $\langle \Delta U, U\rangle$ for $\|U\| = 1$. Formula (11) gives

$$\langle \Delta U, U\rangle = \sum_i \langle \tilde{\Delta}_i U, U\rangle = \sum_i \langle \Delta_i U_i, U_i\rangle,$$

where U_i is the appropriate 2-vector. From Lemma 1 we have

$$\langle \Delta U, U\rangle \geqslant \sum_i \frac{d\alpha}{2r^3}\langle U_i, U_i\rangle = \frac{d\alpha}{2r^3}\sum_i \langle U_i, U_i\rangle = \frac{d\alpha}{2r^3}\cdot 2 = \frac{d\alpha}{r^3}.$$

This inequality gives a lower bound on eigenvalues of Δ. Hence, $\|\Delta\| \geqslant d\alpha/r^3$, and $\|\Delta^{-1}\| \leqslant r^3/d\alpha$. $\square$

3.2.2. Upper bound on the norm of Δ

The matrix Δ is symmetric, therefore, using formulas (8)–(10), Notations 2 and standard inequalities for the norms one gets

$$\begin{aligned}\|\Delta\| = \|\Delta(T)\| &\leqslant \sqrt{n}\cdot\|\Delta\|_1 = \sqrt{n}\cdot\max_i \sum_j |\Delta_{i,j}|\\ &\leqslant 4\sqrt{n}\max_i\left\{\frac{1}{v_i}\right\} \leqslant \frac{4\sqrt{n}}{\tilde{d}}\end{aligned} \tag{12}$$

for all T.

3.3. Newton's method for the search of zero of the gradient of path length

Our algorithm consists of 3 phases described below in Section 4. The third phase is an application of Newton's method to approximate the zero of the Jacobian f' of the length function. The main difficulty, however, is to find an appropriate approximation for Newton iterations. That will be done by a gradient descent described in Section 4.3. We will use the bounds we just have obtained to estimate the rate of convergence of Newton's method.

First let us recall standard facts about Newton's method. Consider a mapping $H = (h_1, \dots, h_n)$ from $\mathbb{R}^n$ into $\mathbb{R}^n$. Letter X without or with superscripts and subscripts will denote vector-columns from $\mathbb{R}^n$. Newton's iterations are defined by the formula

$$X^{k+1} = X^k - (H'(X^k))^{-1} H(X^k), \tag{13}$$

where H' is the Jacobian of H.

Proposition 3 gives a sufficient condition to ensure a fast convergence of Newton's iterations (it can be found in textbooks on numerical methods).

Proposition 3. *Suppose that X_0 is a zero of H, i.e. $H(X_0)=\bar{0}$, where $\bar{0}=(0,\dots,0)\in\mathbb{R}^n$. Let a, a_1, a_2 be reals such that $0<a$ and $0\leqslant a_1, a_2<\infty$, and denote $\Omega_a = \{X : \|X - X_0\| < a\}$, $c = a_1 a_2$ and $b = \min\{a, 1/c\}$.*

If $X_0 \in \Omega_b$ *and*

(A) $\|(H'(X))^{-1}\| \leqslant a_1$ *for* $X \in \Omega_a$,

(B) $\|H(X_1) - H(X_2) - H'(X_2)(X_1 - X_2)\| \leqslant a_2 \|X_1 - X_2\|^2$ *for* $X_1, X_2 \in \Omega_a$ *then*

$$\|X^k - X_0\| \leqslant \frac{1}{c} (c\|X^0 - X_0\|)^{2^k}. \tag{14}$$

In Proposition 3 there are two constants a_1 and a_2: the norm of the inverse of H' has to be at most a_1 (condition (A)); and a_2 is, in fact, an upper bound on the norm of the second derivative of H'' (condition (B)). Both bounds must hold in a-neighborhood of the zero X_0 of H. In our case a, that appears only in (B), will be 'big' (more precisely, we will take $a = R$), so we can ignore it for now. Then the convergence is determined by two things: a parameter $b = 1/a_1 a_2$, and the choice of initial approximation X^0 that must be in an open b-neighborhood of the zero X_0. We are interested in making $c\|X^0 - X_0\|$ smaller than 1 with a known upper bound less than 1. We will construct X^0 such that $c\|X^0 - X_0\| \leqslant \frac{1}{2}$.

We will take the bound from Proposition 2 for a_1. As for a_2, it is not hard to estimate it using formulas for elements of Δ. We will choose X^0 in Section 4.3.

3.3.1. Choosing parameters a_1 and a_2 to satisfy (A) and (B) of Proposition 3

We start with calculating a_2. To satisfy condition (B) it is sufficient to show that the (partial) second derivatives of f are bounded in some neighborhood of X_0. We use Taylor's formula with the third derivatives of f. Take any $T_1, T_2 \in \mathscr{B}$. In our case

$$H(X_1) - H(X_2) - H'(X_2)(X_1 - X_2) = f'(T_1) - f'(T_2) - \Delta(T_2)(T_1 - T_2). \tag{15}$$

We assume that the vectors involved in (15) are represented as columns. To bound from above the norm of this vector, firstly estimate its components. Using notations $T_j=(t_1^{(j)},\dots,t_n^{(j)})$, $j=1,2$, we can write the ith component of (15) as

$$\pi_i(t_{i-1}^{(1)},t_i^{(1)},t_{i+1}^{(1)})-\pi_i(t_{i-1}^{(2)},t_i^{(2)},t_{i+1}^{(2)})+\sum_j\frac{\partial\pi_i(t_j^{(2)})}{\partial t_j}(t_j^{(1)}-t_j^{(2)}). \tag{16}$$

Taylor's formula says that (16) is equal to

$$\frac{1}{2}\left(\sum_{j=i-1,i,i+1}\frac{\partial^2\pi_i(\xi)}{\partial t_j\partial(t_j'-t_j)^2}+\sum_{j,k=i-1,i,i+1}\frac{\partial^2\pi_i(\xi)}{\partial t_j\partial t_k}(t_j'-t_j)(t_k'-t_k)\right) \tag{17}$$

for some vector $\xi=(\xi_j)_j$ whose jth component is between $t_j^{(1)}$ and $t_j^{(2)}$.

Equalities (7)–(10) show that the second derivatives of π_i (notation from (6)) involved in (17) are of the form

$$\frac{1}{v_i^2}\sum(\sin^{\mathrm{O}(1)}\varphi,\cos^{\mathrm{O}(1)}\psi,(\mathrm{O}(1)))+\frac{1}{v_{i-1}^2}\sum(\sin^{\mathrm{O}(1)}\varphi,\cos^{\mathrm{O}(1)}\psi,(\mathrm{O}(1))),$$

where $\sum(\sin^{\mathrm{O}(1)}\varphi,\cos^{\mathrm{O}(1)}\psi,(\mathrm{O}(1)))$ is a sum of expressions in parenthesis ("arguments of Σ"). Thus the absolute value of each second derivative of π_i is bounded by $\mathrm{O}(1/d)$. Hence, the absolute value of each component of vector (15) is bounded by $\mathrm{O}(1/d)$. This implies that in our case (see (15))

$$\begin{aligned}\|H(X_1)-H(X_2)-H'(X_2)(X_1-X_2)\| &= \|f'(T_1)-f'(T_2)-\Delta(T_2)(T_1-T_2)\|\\ &\leqslant C_1\left(\frac{\sqrt{n}}{d}\right)\|T_1-T_2\|^2\end{aligned} \tag{18}$$

for some constant $C_1>0$ and for $T_1,T_2\in\mathscr{B}$.

Thus we set

$$a_2=C_1\cdot\frac{\sqrt{n}}{d}. \tag{19}$$

As in our case $H'=\Delta$, Proposition 2 permits to take as a_1 the lower bound for $\|\Delta^{-1}\|$ from this proposition.

Now we can define all the *parameters for Newton's method*:

- $a=R$ (justified by (18)),
- $a_1=r^3/d\alpha$ (see Proposition 2), $a_2=C_1(n/d)$ (see (18)),
- $c=a_1a_2=C_1(nr^3/\alpha d^2)$ for the constant C_1 from (18).

Our assumptions about r (see Notations 4) imply that $a=r\geqslant 1/c$.

Hence we take

$$b=\frac{1}{c}=\frac{\alpha d^2}{C_1nr^3}=\frac{\tilde{\alpha}^2\tilde{d}^4}{C_1nr^3}. \tag{20}$$

For further references we rewrite (20) as (we use Notations 2 and 3)

$$\bullet\ \frac{b}{2} = c_2 \frac{\tilde{\alpha}^2 \tilde{d}^4}{nr^3} = c_2 \frac{\alpha d^2}{nr^3}, \tag{21}$$

where $c_2 =_{\mathrm{df}} 1/2C_1$.

To ensure fast convergence we will need a 'good' initial approximation X^0 for Newton's method, namely such that

$$X^0 \in \Omega_{b/2}. \tag{22}$$

For such X^0 we have

$$c\|X^0 - X_0\| \leqslant \tfrac{1}{2}, \tag{23}$$

and the rate of convergence guaranteed by Proposition 2 will be $(\frac{1}{2})^{2^k}$, where k is the number of iterations (see Proposition 3).

4. Algorithm for the shortest path touching lines

The algorithm takes as *input* a list L of lines, points $\boldsymbol{s}$ and $\boldsymbol{t}$ and ε, $1 \geqslant \varepsilon > 0$.

The algorithm *outputs* a path, represented as the list of points where it meets the lines of L. The output path is in the ε-neighborhood of the shortest path and its length is ε-close to the length of the shortest path.

We assume that the lines of L are represented by vectors Λ_i^0 (point on the line L_i) and Λ_i (unit vector directed along the line L_i) introduced in Section 2.3. The complexity of transforming another usual representation of lines and points into this form is of linear complexity for our computation model.

The algorithm consists of three phases:

Phase 1: Preliminary computations.

Phase 2: Application of a gradient method to find an approximation for Newton's method.

Phase 3: Application of Newton's method.

Below in this section we will describe the phases. Their descriptions are rather short within the technique developed before, however some new notions will be needed. Together with this description we will make some estimations of complexity that concerns 'local' computations. The global estimation of the complexity will be done in the next Section 5.

4.1. Length approximation from a position approximation

The algorithm seeks an approximation of the position of the shortest path. An appropriate approximation of the length in our case is 'automatic' as follows from Lemma 4 below.

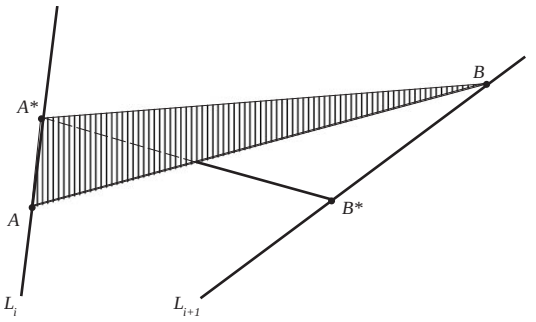

Fig. 3. Comparing the lengths of two paths.

Lemma 4. $|\Pi(T)| - |\Pi(T^*)| \leqslant 2\|T - T^*\|_1 \leqslant \sqrt{n}\|T - T^*\|$.

Proof. The last inequality is standard. Let us prove the first one. We compare $\Pi = \Pi(T)$ and $\Pi^* = \Pi^*(T^*)$ linkwise, see Fig. 3, where $\Pi_i = AB$, $\Pi_i^* = A^*B^*$, $|AA^*| = |t_i - t_i^*|$, $|BB^*| = |t_{i+1} - t_{i+1}^*|$. The triangle inequality immediately implies that

$$|AB| - |A^*B| \leqslant |AA^*|, \quad |A^*B| - |A^*B^*| \leqslant |BB^*|$$

and thus, $|AB| - |A^*B^*| \leqslant |AA^*| + |BB^*|$.

Similarly, we get $|A^*B^*| - |AB| \leqslant |AA^*| + |BB^*|$. Hence, $\|AB| - |A^*B^*\| \leqslant |AA^*| + |BB^*|$.

It remains to notice that we use each $|AA^*| = |t_i - t_i^*|$ twice for $1 \leqslant i \leqslant n$. □

Thus, to obtain a length-approximation with precision ε, we will construct a position $\varepsilon/\sqrt{n}$-approximation.

4.2. Preliminary computations

Phase 1: Compute the path Π^0 and the values d, $\tilde{d}$, α, $\tilde{\alpha}$ r, R and save also all intermediate values that appear within these calculations (all that will be used in the estimations that govern the two other phases).

Our construction of the initial approximation Π^0 was described in Section 2.4. The values d, $\tilde{d}$ are defined in Notations 2, the values α, $\tilde{\alpha}$ are defined in Notations 3 and the values r, R are defined in Notations 4.

4.3. Initial gradient descent to initial approximation of Newton's method

Recall that $f(t_1, \ldots, t_n)$ denotes the length of a path Π represented by points on the lines of L, i.e. $f(t_1, \ldots, t_n) = f(T) = |\Pi(T)|$ (see Notations 5). This function f is convex (see Proposition 2), and the point at which it attains its minimum is denoted by T^* (Section 2.3, Notations 4). Proposition 2 gives a positive lower bound for the norm of the Hessian Δ of f; we denote this lower bound by δ. Let $\tilde{\delta}$ be the upper bound on the norm of Δ from (12).

Consider $g =_{\mathrm{df}} \operatorname{grad} f = ((\partial f/\partial t_1), \ldots, (\partial f/\partial t_n))$. Notice that $g(T^*) = 0$. We have $g(T) = \int_{T^*}^{T} \Delta$, where one can integrate Δ along any path from T^* to T.

Denote by η the distance from T^* to T. Since Δ is positive definite and its norm is at least δ, we have $\langle \Delta(V), V/\|V\| \rangle \geqslant \delta \|V\|$ for all vectors V. Hence the projection of the integral of Δ along the segment from T^* to T to this segment is at least $\delta\eta$, and in particular the norm of the integral itself is at least $\delta\eta$. Recalling that this integral is equal to the gradient of f at T, we conclude that the absolute value $|(\partial f/\partial t_i)(T)|$ of one of the coordinates of the gradient $g(T)$ is at least $\delta \cdot \eta/\sqrt{n}$.

Choose i such that

$$\left| \frac{\partial f}{\partial t_i}(T) \right| \geqslant \frac{\delta \cdot \eta}{\sqrt{n}}. \tag{24}$$

Let us estimate how one can decrease the length of the path by changing t_i (which geometrically represents moving the point where the path meets L_i).

To study how f depends on t_i, introduce a function in one variable τ given by $\varphi(\tau) = f(t_1, \ldots, t_{i-1}, \tau, t_{i+1}, \ldots, t_n)$. Function φ is the restriction of f to a straight line and hence convex. Let φ attain its minimum at τ_0.

We summarize the just introduced notations for further references:

Notations 7.

- $\tilde{\delta} = 4\sqrt{n}/\tilde{d}$ is an upper bound for Δ (formula (12)).
- $\delta = d\alpha/r^3$ is a lower bound for Δ (Proposition 2).
- $g =_{\mathrm{df}} \operatorname{grad} f = ((\partial f/\partial t_1), \ldots, (\partial f/\partial t_n))$.
- η is the distance from T^* to T, i.e. $\eta =_{\mathrm{df}} \|T^* - T\|$, where T^* is a point where the length of our path $\Pi(T^*)$ is minimal and T is the current point in the space of parameters determining our path.
- $\varphi(\tau) = f(t_1, \ldots, t_{i-1}, \tau, t_{i+1}, \ldots, t_n)$ (f is defined in Notations 5).
- i is chosen such that (24).

We wish to estimate $|\varphi(t_i) - \varphi(\tau_0)|$ from below. Notice that our bounds on the Hessian of f imply that $\delta \leqslant \varphi'' \leqslant \tilde{\delta}$. We need the following lemma.

Lemma 5. *Let h be a strictly convex function, i.e. $h'' > 0$ on a segment σ. Suppose that h attains its minimum at a point ζ_0, and assume that $h'' \leqslant \tilde{\delta}$.*

Then

$$h(\zeta) - h(\zeta_0) \geqslant \frac{(h'(\zeta))^2}{4\tilde{\delta}} \tag{25}$$

for all $\zeta \in \sigma$.

Proof. Suppose without loss of generality that $\zeta_0 < \zeta$, and let a point ζ_1 be such that $h'(\zeta_1) = h'(\zeta)/2$. This defines ζ_1 uniquely since the derivative h' of h is strictly increasing.

Then

$$h(\zeta) - h(\zeta_1) \geqslant (\zeta - \zeta_1) \cdot \frac{h'(\zeta)}{2}. \tag{26}$$

On the other hand, integrating h'' from ζ_1 to ζ yields

$$\frac{h'(\zeta)}{2} \leqslant \tilde{\delta} \cdot (\zeta - \zeta_1). \tag{27}$$

Multiplying (26) and (27) we obtain (25). □

Lemma 5 gives

Lemma 6. *In the notations introduced above* (*Notations* 7)

$$|\varphi(t_i) - \varphi(\tau_0)| \geqslant \frac{\delta^2\eta^2}{4n\tilde{\delta}}.$$

This Lemma 6 describes what one gains (in terms of shortening the path in question) by setting $t_i = \tau_0$. To use this lemma we introduce

Notations 8.

- $gain =_{\mathrm{df}} |\varphi(t_i) - \varphi(\tau_0)|$ is the gain in the shortening of the path $\Pi(T)$ after setting $t_i = \tau_0$, where τ_0 is the point where φ attain its minimum.
- $UpBnd =_{\mathrm{df}} (gain \cdot 4n\tilde{\delta})^{1/2}/\delta$ is an upper bound on η (the latter denotes the distance between T^* and T—see Notations 7).

Phase 2: Gradient method procedure.
Phase 2.1: Find i such that

$$\left|\frac{\partial f}{\partial t_i}(T)\right| = \max_j \left\{\left|\frac{\partial f}{\partial t_j}(T)\right|\right\}.$$

(See formulas (5) for $\partial f/\partial t_i = \partial|\Pi|/\partial t_i$.) This is done by using formulas (5), which involve only arithmetic operations and square root extractions.

Claim 3. *The complexity of Phase* 2.1 *is linear in n.*

Phase 2.2: Find the point τ_0 where $\varphi(\tau)$ attains its minimum. Since φ is strictly convex, this happens at the point where its derivative vanishes. Using our notations (see Notations 5 for φ and Notations 2 of Section 2.3 for v_i) we have

$$\varphi(\tau) = v_0(t_1) + v_1(t_1, t_2) + \cdots + v_{i-1}(t_{i-1}, \tau) + v_i(\tau, t_{i+1}) + \cdots + v_n(t_n).$$

Hence, using (5), we have

$$\begin{aligned}\frac{\mathrm{d}\varphi}{\mathrm{d}\tau}(\tau) &= \frac{\mathrm{d}v_{i-1}(t_{i-1}, \tau)}{\mathrm{d}\tau} + \frac{\mathrm{d}v_i(\tau, t_{i+1})}{\mathrm{d}\tau} \\ &= \frac{\langle V_{i-1}(t_{i-1}, \tau), \Lambda_i\rangle}{v_{i-1}} - \frac{\langle V_i(\tau, t_i), \Lambda_i\rangle}{v_i}.\end{aligned} \tag{28}$$

Equating expression (28) to zero, then squaring both sides and expressing the square of the length of a vector v_j by the scalar product of the vector by itself, we get

$$\frac{\langle V_{i-1}(t_{i-1},\tau),\Lambda_i\rangle^2}{\langle V_{i-1}(t_{i-1},\tau),V_{i-1}(t_{i-1},\tau)\rangle}=\frac{\langle V_i(\tau,t_i),\Lambda_i\rangle^2}{\langle V_i(\tau,t_i),V_i(\tau,t_i)\rangle}. \tag{29}$$

Notice that both $V_{i-1}(t_{i-1},\tau)$ and $V_i(\tau,t_i)$ depend in τ linearly (see Notations 1). Thus, Eq. (29) is of the 4th degree in τ. The classical method of L. Ferrari reduces this equation to equations of the 2nd and of the 3rd degree. This method uses a fixed number of arithmetical operations. The solutions of these equations of the 2nd and 3rd degree (formulas of G. Cardano for the latter) can be found using explicit formulas over arithmetic operations and square and cubic root extraction—all these operations are admissible in our computational model (see the Introduction).

Hence,

Claim 4. *The complexity of Phase* 2.2 *is* O(1).

Note that, for a standard convexity reason, the modification of the path described in this phase cannot make it leave a ball where the path lied. Hence the path stays in the ball $\mathscr{B}$ (Notations 4).

Phase 2.3: Calculate (we use Notations 8)

gain, the point $T'=(t_1,\dots,t_{i-1},\tau_0,t_i,\dots,t_n)$ and *UpBnd*, where τ_0 is from Phase 2.2 just above.

If $\varepsilon/\sqrt{n}<UpBnd<b/2$ (cf. (21) and (22)) then go to Phase 3 with T' as the initial approximation for Newton's method.

Otherwise, repeat Phase 2 with $T=T'$. Thus, the recalculation of *gain*, T' and *UpBnd* is iterated while $UpBnd\geqslant\max\{\varepsilon/\sqrt{n},b/2\}$.

As one can see from the formulas defining the values of *gain*, T' and *UpBnd* (Notations 8, 5 and the respective notations from Section 2.3),

Claim 5. *The complexity of computing gain*, T' *and UpBnd is linear in* n.

4.4. Application of Newton's method

Phase 3 (*Newton's method*). Apply Newton's method with the value of T, found by Phase 2.3, as X^0. Iterate (13) until obtaining a sufficiently close approximation; see Section 5 below, formula (46). The rate of convergence is estimated in Proposition 3, formula (14).

5. Complexity of the algorithm

Now we summarize the comments on the complexity made in the previous sections to count the number of steps used by the algorithm to find a length–position ε-

approximation of the shortest path touching the lines of L. By C_i, c_j we will denote various positive constants (using capital C for upper bounds, and lower case c for lower bounds).

5.1. Complexity of preliminary computations

Phase 1 finds perpendiculars from $\boldsymbol{s}$ to lines L_i. These perpendiculars fall at points $P_i \in L_i$. To find a point P_i we solve a linear equation with one unknown t_i: $\langle W_i(t_i) - \boldsymbol{s}, \Lambda_i\rangle = 0$. This can be done using only arithmetic operations. The complexity of solving one equation is constant, thus the total complexity of finding all P_i's is linear in n. The points P_i represent the polygon Π^0, and computing its length $|\Pi^0|$ also involves square root extraction. However, the complexity is still linear in n.

Thus, *the parameter R has been computed with linear complexity*.

To compute sines $\tilde{\alpha}_i = \sin \angle \Lambda_i\Lambda_{i+1}$ and cosines $\tilde{\beta}_i = \cos \angle \Lambda_i\Lambda_{i+1}$ we compute the scalar products $\langle \Lambda_i, \Lambda_{i+1}\rangle$, and use the standard formulas that involve arithmetic operations and square root extractions only. The complexity is again linear in n.

The next parameter to find is the minimal distance between consecutive lines. For two consecutive lines L_i and L_{i+1}, the distance between them is the length of the segment connecting the lines and perpendicular to both of them; this is again a standard "analytic geometry" computation, which yields $\tilde{d}$ and d in linear time. The calculation of δ and $\hat{\delta}$ add only O(1) to the complexity (see formulas in Notations 5). Hence,

Claim 6. *The complexity of Phase* 1 *is linear in n.*

5.2. Complexity of gradient descent

The complexity of one iteration of the gradient descent of Phase 2 was estimated from above at the end of Section 4.3. This complexity is O(n).

So we are to *estimate a sufficient number of iterations*.

Recall that $\|T - T^*\| = \eta \leqslant UpBnd$ (Notations 5 and Lemma 6). Phase 2.3 iterates the calculation of *UpBnd* until the latter becomes less than $\max\{\varepsilon/\sqrt{n}, b/2\}$.

It is evident that initially we have $gain \leqslant r$ (r is the length of the initial approximation to the shortest path—see Notations 4). Hence, initially for the initial *gain* (we use the expression for *UpBnd* from Notations 8)

$$UpBnd \leqslant \frac{(gain \cdot 4n\tilde{\delta})^{1/2}}{\delta} \leqslant c_3 \frac{(rn \cdot n^{1/2})^{1/2} r^3}{\tilde{d}^{1/2} d\alpha} = c_3 \cdot \frac{r^{7/2} \cdot n^{3/4}}{\tilde{d}^{5/2} \cdot \alpha} =_{\mathrm{df}} B_0. \tag{30}$$

(We replaced d by $\tilde{d}^2$ and $\tilde{\delta}, \delta$ by their expressions using, respectively, Notations 2 and Notations 7.)

So we can estimate the number of iterations in the Phase 2.3 as the ratio of the right-hand side value in formula (30) over a lower bound for *gain* that is valid while *UpBnd* goes down to the demanded value. Such a lower bound can be found from the

condition $UpBnd \geqslant \max\{\varepsilon/\sqrt{n}, b/2\}$ that controls the continuation of the iterations:

$$UpBnd = \frac{(gain \cdot 4n\tilde{\delta})^{1/2}}{\delta} \geqslant \max\left\{\frac{\varepsilon}{\sqrt{n}}, \frac{b}{2}\right\}. \tag{31}$$

This condition (31) gives

$$gain \geqslant \frac{\delta^2}{4n\tilde{\delta}} \cdot \max\left\{\frac{\varepsilon^2}{n}, \frac{b^2}{4}\right\} = \frac{(d\alpha)^2 \cdot \tilde{d}}{r^6 \cdot 4n \cdot 4\sqrt{n}} \cdot \max\left\{\frac{\varepsilon^2}{n}, \frac{b^2}{4}\right\} \tag{32}$$

$$= \frac{\tilde{d}^5 \cdot \alpha^2}{r^6 \cdot 16n^{3/2}} \cdot \max\left\{\frac{\varepsilon^2}{n}, \frac{b^2}{4}\right\},$$

$$= \begin{cases} \dfrac{\tilde{d}^5\alpha^2}{r^6 \cdot 16n^{3/2}} \cdot \dfrac{b^2}{4} =_{\mathrm{df}} g_1 \text{ if } \dfrac{\varepsilon^2}{n} < \dfrac{b^2}{4} \\ \dfrac{\tilde{d}^5\alpha^2}{r^6 \cdot 16n^{3/2}} \cdot \dfrac{\varepsilon^2}{n} =_{\mathrm{df}} g_2 \text{ if } \dfrac{\varepsilon^2}{n} \geqslant \dfrac{b^2}{4}. \end{cases} \tag{33}$$

To estimate the number of iterations of recalculations of $UpBnd$ in Phase 2 we estimate B_0/g_1 and B_0/g_2. Denote the number of iterations by $NbIter$, and consider 2 cases corresponding to the cases in (33).

Case 1: $\varepsilon^2/n < b^2/4$. Replace $b^2/4$ by its value from (21)

$$g_1 = \frac{\tilde{d}^5\alpha^2}{r^6 \cdot 16n^{3/2}} \cdot c_2 \frac{\alpha d^2}{nr^3} = c_2' \cdot \frac{\tilde{d}^9 \cdot \alpha^3}{r^9 \cdot n^{5/2}}. \tag{34}$$

Divide B_0 from (30) by g_1 from (34)

$$NbIter \leqslant \mathrm{O}\left(\frac{r^{7/2} \cdot n^{3/4} \cdot r^9 \cdot n^{5/2}}{\tilde{d}^{5/2} \cdot \alpha \cdot \tilde{d}^9 \cdot \alpha^3}\right) = \mathrm{O}\left(\frac{r^{12.5} \cdot n^{3.25}}{\tilde{d}^{11.5} \cdot \alpha^4}\right). \tag{35}$$

We take into account that $\tilde{d} \leqslant 1 < r$ (Notations 2 and 4), $\alpha \leqslant 1$ and $\alpha = \tilde{\alpha}^2$ (Notations 3) and $n \geqslant 1$, and rewrite (35) as

$$NbIter \leqslant \mathrm{O}\left(\left(\frac{r}{\tilde{d}}\right)^{12.5} \cdot \left(\frac{n}{\alpha}\right)^4\right) \leqslant \mathrm{O}\left(\frac{rn}{\tilde{d}\tilde{\alpha}}\right)^{13}. \tag{36}$$

Case 2: $\varepsilon^2/n \geqslant b^2/4$. Divide B_0 from (30), this time, by g_2 from (33):

$$NbIter \leqslant \frac{B_0}{g_2} = \mathrm{O}\left(\frac{r^{7/2} \cdot n^{3/4} \cdot r^6 \cdot n^{3/2} \cdot n}{\tilde{d}^{5/2} \cdot \alpha \cdot \tilde{d}^5 \cdot \alpha^2 \cdot \varepsilon^2}\right) = \mathrm{O}\left(\frac{r^{19/2} \cdot n^{13/4}}{\tilde{d}^{15/2} \cdot \alpha^3 \cdot \varepsilon^2}\right). \tag{37}$$

In Case 2 we can bound $1/\varepsilon^2$ using Case 2 condition:

$$\frac{1}{\varepsilon^2} \leqslant \frac{4}{b^2 \cdot n} = \mathrm{O}\left(\frac{r^6 \cdot n}{d^4 \cdot \alpha^2}\right) = \mathrm{O}\left(\frac{r^6 \cdot n}{\tilde{d}^8 \cdot \alpha^2}\right), \tag{38}$$

where we used also (21) to replace b and Notations 2 to replace d by $\tilde{d}^2$. From (38) and (37) we get

$$NbIter \leqslant \mathrm{O}\left(\frac{r^{19/2}\cdot n^{13/4}\cdot r^6\cdot n}{\tilde{d}^{15/2}\cdot\alpha^3\cdot\tilde{d}^8\cdot\alpha^2}\right)=\mathrm{O}\left(\frac{r^{31/2}\cdot n^{17/4}}{\tilde{d}^{31/2}\cdot\alpha^5}\right)=\mathrm{O}\left(\frac{r^{15.5}\cdot n^{4.25}}{\tilde{d}^{15.5}\cdot\alpha^5}\right) \tag{39}$$

$$\leqslant \mathrm{O}\left(\left(\frac{r}{\tilde{d}}\right)^{15.5}\cdot\left(\frac{n}{\alpha}\right)^5\right)\leqslant \mathrm{O}\left(\frac{rn}{\tilde{d}\tilde{\alpha}}\right)^{16}. \tag{40}$$

The bound in (40) majorizes the bound in (36), so we can take the bound from (40) for further references. Hence, in any case

$$NbIter \leqslant \mathrm{O}\left(\frac{rn}{\tilde{d}\tilde{\alpha}}\right)^{16}. \tag{41}$$

From this bound (41) and Claim 5 that says that the complexity of calculating of *gain*, T' and *UpBnd* in Phase 2.3 is linear we get

Claim 7. *The complexity of Phase* 2.3 *is* $\mathrm{O}(rn/\tilde{d}\tilde{\alpha})^{16}$ *that is majorized by* $\mathrm{O}(Rn/\tilde{d}\tilde{\alpha})^{16}$.

From Claims 3, 4 and 7 we deduce

Claim 8. *The complexity of Phase* 2 *is* $\mathrm{O}(rn/\tilde{d}\tilde{\alpha})^{16}$ *or, in other terms,* $\mathrm{O}(Rn/\tilde{d}\tilde{\alpha})^{16}$.

5.3. Complexity of Newton's method

Once $\varepsilon/\sqrt{n}\leqslant b/2$, and this is the case we are mainly interested in, Phase 3 of our algorithm applies Newton's method. Formula (14) for the convergence of Newton's method estimates the error after k iterations by

$$\frac{1}{c}(c\|X^0-X_0\|)^{2^k}, \tag{42}$$

which we want to be smaller than $\varepsilon/\sqrt{n}$. For our choice of initial approximation for Newton's method, see (23), and for our value of $1/c$, see (20), value (42) takes the form

$$\frac{1}{c}\left(\frac{1}{2}\right)^{2^k}=b\left(\frac{1}{2}\right)^{2^k}=\frac{\alpha d^2}{C_1 nr^3}\left(\frac{1}{2}\right)^{2^k}. \tag{43}$$

Thus we need

$$\left(\frac{1}{2}\right)^{2^k}<\frac{C_1\sqrt{n}r^3\varepsilon}{\alpha d^2}. \tag{44}$$

Inequality (44) is equivalent to

$$2^k < \log \frac{1}{\varepsilon} + \frac{1}{2}\log n + 3\log r + \log \frac{1}{\alpha} - 2\log d + \mathrm{O}(1)$$
$$\leqslant \log \frac{1}{\varepsilon} + \mathrm{O}\left(\log r + \log \frac{1}{\alpha}\right) \tag{45}$$

where all *log*'s are with base 2; recall also that (without loss of generality) we assumed $n \leqslant r$. From (45) we can bound k as

$$k < \log\left(\log \frac{1}{\varepsilon} + \mathrm{O}\left(\log r + \log \frac{1}{\alpha}\right)\right). \tag{46}$$

It remains to estimate the complexity of computing the kth iterate X^{k+1} using formula (13). In our case $F(T) = \mathbf{grad} f(T)$ and $F'(T)^{-1} = \Delta(T)^{-1}$. The complexity of calculating $F(T)$ is shown to be linear in n. Computing the inverse of a 3-diagonal $n \times n$-matrix $\Delta(T)$ takes $\mathrm{O}(n^2)$ operations. Note that the complexity of computing one element of $\Delta(T)$ is constant. Thus the total complexity of Newton's method is

$$\mathrm{O}\left(n^2 \log\left(\log \frac{1}{\varepsilon} + \mathrm{O}\left(\log r + \log \frac{1}{\alpha}\right)\right)\right). \tag{47}$$

Taking into consideration that $\log(x+y) \leqslant (\log x + \log y)$ for $x, y \geqslant 2$, we can simplify this estimation (47), obtaining the following bound for the complexity of Newton's method

Claim 9. *The complexity of Phase* 3 *is*

$$\mathrm{O}\left(n^2 \log\log \frac{1}{\varepsilon}\right) + \mathrm{O}\left(n^2\left(\log r + \log \frac{1}{\alpha}\right)\right). \tag{48}$$

Combining complexity estimates for the three phases stated in Claims 6, 8 and 9 and observing that the term $\mathrm{O}(n^2(\log r + \log 1/\alpha))$ is much smaller than the bound on the complexity of the gradient procedure,

we finally obtain the estimation of the Main Theorem.

Thus, the Main Theorem is proved.

Part 2: A simple grid approximation of shortest paths amidst separated obstacles

6. Separated obstacles

We consider obstacles in $\mathbb{R}^3$ though all the notions and constructions can be easily generalized to an arbitrary dimension. The dimension under consideration influences only the constants that appear in estimations of the complexity of algorithms.

The problem we study here is, as in the previous part, to construct a path that connects two given points $\boldsymbol{s}$ and $\boldsymbol{t}$, does not intersect given obstacles W and whose length is close to that of a shortest path.

6.1. Obstacles and paths

Let W be an arbitrary set in $\mathbb{R}^3$, and $\boldsymbol{s}$ and $\boldsymbol{t}$ be two points in its compliment.

Denote by $cl(S)$, $int(S)$ and $bnd(S)$, respectively, the *closure*, the *interior* and the *boundary* of a set S.

Denote by $B_v(a)$, where $a \in \mathbb{R}_{\geqslant 0}$ and $v \in \mathbb{R}^3$, the ball of radius a centered at v.

The boundary $bnd(W) = cl(W) \setminus int(W)$ may contain 'degenerate' pieces. For example, an isolated point or a point with a neighborhood homeomorphic to a segment. Such pieces can hardly be considered as obstacles. So we assume that every point of $bnd(W)$ has a two-dimensional neighborhood. For technical reasons it is convenient to assume a neighborhood of each point of $bnd(W)$ intersects $int(W)$. To achieve this we can 'slightly inflate' W. It is known how to do it efficiently for semi-algebraic obstacles, see e.g. [13].

A *path* is a continuous piecewise smooth image of a closed segment. A *simple* path or a *quasi-segment* is a path without self-intersections (which is, clearly, homeomorphic to a segment).

The set $\mathbb{R}^3 \setminus W$ will be called the *free space*, and its closure will be called the *space admissible for trajectories*.

We consider only paths lying in the admissible space and not intersecting the interior of W.

6.2. Separability and random separated balls

We say that obstacles W are *a-separated* if for any $v \in \mathbb{R}^3$ the set $W \cap B_v(a)$ is connected.

For example, if each connected component of W is convex and the distance between each two connected components is greater than $2a$, then W is a-separated. However, the convexity of the obstacles is not assumed in the general case. What is imposed by a-separation is a certain smoothness of concave (from the point of view of an observer outside the obstacles) pieces of the boundary—a ball of radius $2a$ that goes inside 'fjords' of an obstacle cannot touch or intersect two pieces of the obstacles that are 'remote' if to follow the boundary.

Our main goal is to describe an algorithm that constructs a length approximation to a shortest path under the condition of separability of obstacles. But now we will make a digression estimating the expectation of separability of obstacles constituted by n randomly chosen balls. The centers of the balls are independently chosen in the unit cube $[0,1]^3$, and their radii are independently chosen from an interval $[0,r]$ under the uniform distribution.

Proposition 7. *There exist constants $c_1, c_2 > 0$ such that for any $r < c_1 n^{-2/3}$ the union of n randomly chosen balls in $[0,1]^3$ is $(c_2 n^{-2/3})$-separated with probability greater than $\frac{2}{3}$.*

Proof. We claim that n randomly chosen points in the unit cube are $c_3/n^{2/3}$-separated with probability greater than $\frac{2}{3}$ for some constant $c_3 > 0$.

Indeed, a choice of n points in $[0,1]^3$ is equivalent to a choice of a point $p=(x_1,y_1,z_1,\ldots,x_n,y_n,z_n)\in[0,1]^{3n}$. For any pair of natural numbers i,j such that $1\leqslant i<j\leqslant n$, the measure of those points p for which $|x_i-x_j|,|y_i-y_j|,|z_i-z_j|\leqslant c_3n^{-2/3}$ is not greater than $(\sqrt{2}c_3n^{-2/3})^3$. This is true for any constant c_3. Hence the measure of those p for which there exist a pair $1\leqslant i<j\leqslant n$ such that this condition holds is not greater than $[n(n-1)/2](\sqrt{2}c_3n^{-2/3})^3\leqslant\frac{1}{3}$ for an appropriate constant c_3.

To finish the proof of the lemma we choose c_1 and c_2 so that $2c_1+c_2<c_3$. □

7. A grid algorithm for a shortest path approximation

In this section we assume that the obstacles $W\subseteq[0,1]^3$ are a-separated and that $\boldsymbol{s},\boldsymbol{t}\in[0,1]^3$.

7.1. Approximation algorithm and its complexity

Denote by $L\subseteq[0,1]^3$ a cubic grid with mesh (edge of the basic cube) a. Without loss of generality we assume that $1/a$ is an integer and that $\boldsymbol{s}$ and $\boldsymbol{t}$ are nodes of the grid.

Consider a graph G whose vertices are those nodes of the grid that do not belong to W and whose edges are those edges of the grid that do not intersect W.

Assume that the length of every edge in G is a.

The length of a path P will be denoted by $|P|$.

Theorem on Simple Grid Method. *There is an algorithm of time complexity* $\mathrm{O}((1/a)^6)$ *that, using the graph G defined above, constructs a path* Π_G *connecting* $\boldsymbol{s}$ *and* $\boldsymbol{t}$ *and whose length satisfies the inequality* $|\Pi_G|\leqslant 392|\Pi^*|+448a$, *where* Π^* *is a shortest path amidst the obstacles W connecting* $\boldsymbol{s}$ *and* $\boldsymbol{t}$.

Remark 2. If a is not known but it is possible to construct the part of the grid admissible for paths, an algorithm can consecutively try $a=\frac{1}{2},\frac{1}{4},\frac{1}{8},\ldots$ until it obtains a path with the bound from Theorem on Simple Grid Method.

From Proposition 7 one concludes

Corollary 8. *When the set of obstacles W is a union of n random balls whose radii are independently chosen from* $[0,c_1n^{-2/3}]$, *then* $a\leqslant c_2n^{-2/3}$ *with probability greater than* $\frac{2}{3}$. *Hence the algorithm will compute an approximation* (*as described in Theorem on Simple Grid Method*) *to a shortest path with probability greater than* $\frac{2}{3}$ *within time* $\mathrm{O}(n^4)$.

We say that a path is a *grid polygon* if it goes only along the edges of the grid.

To prove the Theorem, it suffices to show that the length of a shortest grid polygon Π_G connecting $\boldsymbol{s}$ and $\boldsymbol{t}$ and satisfies

$$|\Pi_G|\leqslant 84|\Pi^*|+96a. \tag{49}$$

Then this grid polygon can be constructed in quadratic time as a shortest path Π_G in the graph G by using any polytime algorithm for a shortest path in a graph (e.g. see [2]).

Estimate (49), and hence Theorem on Simple Grid Method will follow from Lemmas 9 and 10.

We say that a cube of the grid is *visited* by a path P if the closure of this cube without vertices intersects with P. Notice that P does not determine uniquely the order of visited cubes as P may visit two cubes simultaneously by going along their common edge.

Lemma 9. *If two nodes v_1 and v_2 of the grid are connected by a path P that visits s distinct cubes of the grid, then one can connect v_1 and v_2 by a grid polygon whose length is not greater than $\leqslant 12sa$.*

Proof. For each cube K, we can consider a maximal segment of P contained in K. Thus P is subdivided into intervals, each contained in one cube of the grid and such that its continuation in either direction leaves the cube. Consider such a segment (a subpath) $[wu]$ for a cube K, that is P enters K via a point w and leaves it via a point u (of course, P can make several visits like that to a cube). Let u lie in a face with vertices u_1, u_2, u_3, u_4. Then one of the four intervals uu_i, $1 \leqslant i \leqslant 4$, does not intersect the obstacles W.

Indeed, assume the contrary. Then, because of a-separability of W, the intersection W with any cube of the grid and, thus, with any of its faces is a convex set. The fact that a segment lying in the face intersects W means that there are points of the segment in the interior of W. But $u \notin int(W)$. So if u is different from any u_i, then our assumption implies that $u \in int(W)$, for u is in the convex hull of the intersection of the face with $int(W)$. If $u = u_i$ for some i then $uu_i = u$ have the desired property.

Now replace P by a path P' obtained by a sequence of the following modifications. Take any cube K visited by P, together with a maximum segment $[wu]$ of P in K. Let i, $1 \leqslant i \leqslant 4$, be such that the segment uu_i does not intersect W. Insert the intervals uu_i and u_iu into P to force P to visit a vertex of the cube by going there and back. Repeating the same procedure for the entry point w (which is in its turn an exit point for some other cube), we modified our segment so that it ends and begins at a vertex of K. We will say that the visit of P to K *terminates* at u_i. Perform this operation for all cubes visited by P. This gives P', a new path which visits the same cubes, and for which every maximum subpath contained in a cube begins at and leaves the cube via a vertex.

We remark that though the number of cubes visited by P' is the same as for P, i.e. s, the length of P' might have increased.

Delete from P' all loops connecting the same vertex, thus obtaining a new path P''. Notice that the number of vertices visited by P'' is at most $8s$, for any vertex of any of the visited cubes is visited at most once. Moreover, P'', still connects v_1 and v_2.

Now consider a maximum subpath of P'' contained in a grid cube K and connecting two vertices of K. Denote by W_1 the (convex) intersection $W \cap K$. Consider the following homotopy of this subpath in K: pull each of its points v in the direction *from*

the point of W_1 nearest to v. (The nearest point is unique since W_1 is convex.) Let the magnitude of the velocity of v (with respect to the parameter of the homotopy) be equal to the distance from v to the boundary ∂K of K. This way we "push" the maximum subpath of P'' in question away from W_1 until we transform it into a path lying in the boundary ∂K of K, connecting the same vertices of K and not intersecting W. Of course, its length still might have increased. Repeat the same procedure for each maximal subpath contained in one cube, having constructed a path going along faces.

Having repeated again the same operation for each maximum subpath contained in one boundary square of a grid cube (and now using a homotopy pushing this subpath to the boundary of this square), we end up with a polygonal path P''' that consists of edges of the grid, connects the same vertices v_1 and v_2, and is contained in at most s cubes. After removing all loops from this path, we obtain a path P'' that traverses each edge at most once. Since s cubes together have at most $12s$ edges, we have constructed a path with required properties and whose length is at most $12as$. □

Lemma 10. *If a path P connecting nodes v_1 and v_2 of the grid has visited s different cubes of the grid, then $|P| \geqslant \lfloor (s-2)/7 \rfloor a$.*

Proof. Mark the points when P visits for the first time the 1st, the 9th, the 16th,..., the $(7l+2)$th,... cube (we count only the visits to *new* cubes). The pieces of P between two consecutive marked points will be called *intervals*. Clearly, the number of intervals is at least $\lfloor (s-2)/7 \rfloor$.

It remains to show that the length of each interval is at least a. Reasoning by contradiction, note that the projection of an interval whose length is less than a to each coordinate axis is a segment of a length less than a. Hence it is contained in the interior of the union of two adjacent intervals of the form $[ka,(k+1)a]$. Now the product of these intervals, which is the interior of the union of eight grid cubes with a common vertex, contains the interval. This contradicts the assumption that the interval visited at least 9 cubes. □

Recall that s denotes the number of different cubes of the grid visited by a shortest path Π^*. Lemma 10 gives $|\Pi^*| \geqslant \lfloor (s-2)/7 \rfloor a \geqslant ((s-2)/7 - \frac{6}{7})a$. On the other hand, for the length of a shortest grid polygon Π_G Lemma 9 gives $|\Pi_G| \leqslant 12sa$. Estimate (49) immediately follows from these two inequalities, and this concludes the proof of Theorem on Simple Grid Method. □

Remark 3. The estimation given by Theorem on Simple Grid Method is, obviously, far from the exact one. We conjecture that $|\Pi_G| \leqslant 7|\Pi^*|$. The latter bound cannot be essentially improved as one can see from the following example. Denote by v_1, v_2, v_3, v_4 the vertices of the bottom face of the unit cube (ordered counter clockwise), and by u_1, u_2, u_3, u_4 the respective vertices of the top face, see Fig. 4. Let W_0 be the polyhedron with the following 5 vertices: 3 vertices at the middles of edges v_1v_2, v_3v_4, u_3u_4, and 2 vertices on edges u_1v_1 and u_2v_2 close to points v_1 and v_2, respectively. To obtain W, apply to W_0 a homothety (scaling) with coefficient $(1+\varepsilon)$, for a small enough $\varepsilon > 0$ centered at the center of the cube.

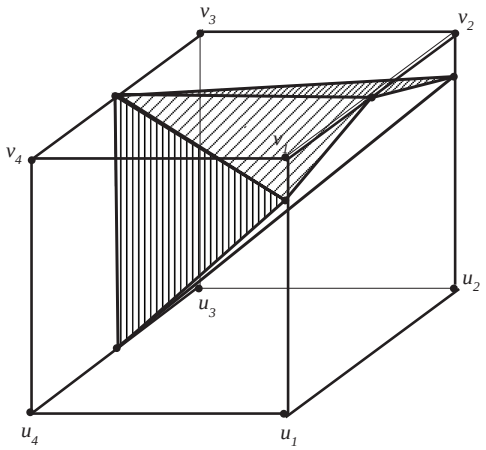

Fig. 4. Some faces of W_0.

For this obstacle the only grid polygon connecting v_1 and v_2 visits consecutively v_1, v_4, u_4, u_1, u_2, u_3, v_3, v_2. On the other hand, one can see that v_1 and v_2 can be connected by a path (avoiding W and contained in the cube) of length close to 1.

Acknowledgements

We are thankful to Misha Gromov for sharing his geometric insight, and to Yuri Burago for his useful comments and his help with the references. We are also grateful to anonymous referee for many useful remarks that helped to improve the presentation.

References

[1] P.K. Agarwal, S. Har-Peled, M. Sharir, K.R. Varadarajan, Approximate shortest paths on a convex polytope in three dimensions, J. Assoc. Comput. Mach. 44 (1997) 567–584.
[2] A. Aho, J. Hopcroft, J. Ullman, The Design and Analysis of Computer Algorithms, Addison-Wesley, Reading, MA, 1976.
[3] L. Alexandrov, M. Lanthier, A. Maheshwari, J.-R. Sack, An ε-approximation algorithm for weighted shortest paths on polyhedral surfaces, in: Proc. 6th Scandinavian Workshop on Algorithm Theory, Lecture Notes in Computer Science, Vol. 1432, Springer, Berlin, 1998, pp. 11–22.
[4] W. Ballmann, Lectures on spaces of nonpositive curvature, DMV Seminar, Band 25, Birkhäuser, Basel, Boston, Berlin, 1995.
[5] V. Berestovsij, I. Nikolaev, Multidimensional generalized Riemann spaces, in: Yu. Reshetnyak (Ed.), Geometry 4, Non-regular Riemann Geometry, Encyclopaedia of Mathematical Sciences, Vol. 70, Springer, Berlin, 1993.
[6] L. Blum, M. Shub, S. Smale, On a theory of computation and complexity over real numbers: NP-completeness, recursive functions and universal machines, Bull. Amer. Math. Soc. 1 (1989) 1–46.
[7] D. Burago, Hard balls gas and Alexandrov spaces of curvature bounded above, Proc. ICM, Invited Lecture, Vol. 2, Berlin, 1998, pp. 289–298.

[8] D. Burago, S. Ferleger, A. Kononenko, Uniform estimates on the number of collisions in semi-dispersing billiards, Ann. Math. 147 (1998) 695–708.
[9] J. Canny, J. Reif, New lower bound technique for robot motion planning problems, in: Proc. 28th Ann. IEEE Symp. on Foundations of Computer Science, Los Angeles, 1987, pp. 49–60.
[10] J. Chen, Y. Han, Shortest paths on a polyhedron, in: Proc. 6th Ann. ACM Symp. on Computational Geometry, Berkeley, June 6–8, New York, 1990, pp. 360–369.
[11] J. Choi, J. Sellen, C.-K. Yap, Approximate Euclidean shortest path in 3-space, in: Proc. 10th ACM Symp. on Computational Geometry, Stony Brook, New York, 1994, pp. 41–48.
[12] K.L. Clarkson, Approximate algorithms for shortest path motion planning, in: Proc. 19th Ann. ACM Symp. on Theory of Computing, New York, 1987, pp. 56–65.
[13] D. Grigoriev, A. Slissenko, Polytime algorithm for the shortest path in a homotopy class amidst semi-algebraic obstacles in the plane, in: Proc. of the 1998 Internat. Symp. on Symbolic and Algebraic Computations (ISSAC'98), ACM Press, New York, 1998, pp. 17–24.
[14] D. Grigoriev, A. Slissenko, Computing minimum-link path in a homotopy class amidst semi-algebraic obstacles in the plane, St. Petersburg Math. J. 10 (2) (1999) 315–332.
[15] S. Har-Peled, Constructing approximate shortest path maps in three dimensions, SIAM J. Comput. 28 (4) (1999) 1182–1197.
[16] J. Heintz, T. Krick, A. Slissenko, P. Solernó, Une borne inférieure pour la construction de chemins polygonaux dans $\boldsymbol{r}^n$, Publications du département de mathématiques de l'Université de Limoges, l'Université de Limoges, France, 1993, pp. 94–100.
[17] J. Heintz, T. Krick, A. Slissenko, P. Solernó, Search for shortest path around semialgebraic obstacles in the plane, J. Math. Sci. 70 (4) (1994) 1944–1949 (translation into English of the paper published in Zapiski Nauchn. Semin. *LOMI* 192 (1991) 163–173).
[18] J.S.B. Mitchell, Geometric shortest paths and network optimization, in: J.-R. Sack, J. Urrutia (Eds.), Handbook of Computational Geometry, Elsevier, North-Holland, Amsterdam, 2000, pp. 633–701.
[19] J.S.B. Mitchell, D. Mount, C.H. Papadimitriou, The discrete geodesic problem, SIAM J. Comput. 16 (4) (1987) 647–668.
[20] J. Mitchell, C. Papadimitriou, The weighted region problem: finding shortest paths through a weighted planar subdivision, J. Assoc. Comput. Mach. 38 (1991) 18–73.
[21] C.H. Papadimitriou, Euclidean TSP is NP-complete, Theoret. Comput. Sci. 4 (1977) 237–244.
[22] C.H. Papadimitriou, An algorithm for shortest-path motion in three dimensions, Inform. Process. Lett. 20 (1985) 259–263.
[23] J.T. Schwartz, M. Sharir, Algorithmic motion planning in robotics, in: J. van Leeuwen (Ed.), Handbook of Theoretical Computer Science, Vol. A, Elsevier, Amsterdam, 1990, pp. 391–430.
[24] M. Sharir, A. Schorr, On shortest paths in polyhedral spaces, SIAM J. Comput. 15 (1986) 193–215.