Watchman Route in a Simple Polygon with a Rubberband Algorithm

Fajie Li* Reinhard Klette†

Abstract

So far, the best result in running time for solving the fixed watchman route problem (i.e., shortest path for viewing any point in a simple polygon with given start point) is $\mathcal{O}(n^3 \log n)$, published in 2003 by M. Dror, A. Efrat, A. Lubiw, and J. Mitchell. – This paper provides an algorithm with $\kappa(\varepsilon) \cdot \mathcal{O}(kn)$ runtime, where n is the number of vertices of the given simple polygon Π, and k the number of essential cuts; $\kappa(\varepsilon)$ defines the numerical accuracy in dependency of a selected constant $\varepsilon > 0$. Moreover, our algorithm is significantly simpler, easier to understand and implement than previous ones for solving the fixed watchman route problem.

1 Introduction

Let Π be a planar, simple, topologically closed polygon with n vertices, and $\partial\Pi$ be its frontier. A point $p \in \Pi$ is *visible* from point $q \in \Pi$ iff $pq \subset \Pi$. The *(floating) watchman route problem* (WRP) of computational geometry, as discussed in [2], is defined as follows: *Calculate a shortest route $\rho \subset \Pi$ such that any point $p \in \Pi$ is visible from at least one point on ρ.* If a start point of the route is given on $\partial\Pi$ then this refined problem is known as the *fixed* WRP. In the rest of this paper, let $s \in \partial\Pi$ be the starting point of the fixed WRP.

A simplified WRP of finding a shortest route in a simple isothetic polygon was solved in 1988 in [7] by presenting an $\mathcal{O}(n \log\log n)$ algorithm. In 1991, [8] claimed to have presented an $\mathcal{O}(n^4)$ algorithm, solving the fixed WRP. In 1993, [21] obtained an $\mathcal{O}(n^3)$ solution for the fixed WRP. In the same year, this was further improved to a quadratic time algorithm [22]. However, four years later, in 1997, [10] pointed out that the algorithms in both [8] and [21] were flawed, but presented a solution for fixing those errors. Interestingly, two years later, in 1999, [23] found that the solution given by [10] was also flawed! By modifying the (flawed) algorithm presented in [21], [23] gave an $\mathcal{O}(n^4)$ runtime algorithm for the fixed WRP. In 1995 and 1999, [17] and [6] gave an $\mathcal{O}(n^6)$ algorithm for the WRP respectively. This was improved in 2001 by an $\mathcal{O}(n^5)$ algorithm in [24]. So far the best known result for the fixed WRP is due to [9] by presenting in 2003 an $\mathcal{O}(n^3 \log n)$ runtime algorithm.

Given the time complexity of those algorithms for solving the WRP, finding efficient approximation algorithms became an interesting subject. Recall the following definition; see, for example, [11]: An algorithm is an *δ-approximation algorithm* for a minimization problem P iff, for each input of P, the algorithm delivers a solution that is at most δ times the optimum solution. In case of the WRP, the optimum solution is defined by the length of the shortest path.

In 1995, [14] published an $\mathcal{O}(\log n)$-approximation algorithm for solving the WRP. In 1997, [5] gave a 99·98-approximation algorithm with time complexity $\mathcal{O}(n \log n)$ for the WRP. In 2001, [25] presented a linear-time algorithm for an approximative solution of the fixed WRP such that the length of the calculated watchman route is at most twice of that of the shortest watchman route. The coefficient of accuracy was improved to $\sqrt{2}$ in [26] in 2004. Most recently, [27] presented a linear-time algorithm for the WRP for calculating an approximative watchman route of length at most twice of that of the shortest watchman route.

Let ESP denote the class of any Euclidean shortest path problem. Corresponding to the definition of δ-approximation algorithms, we introduce the following definition: A Euclidean path is a *δ-approximation (Euclidean) path* for an ESP problem iff its length is at most δ times the optimum solution.

The paper is organized as follows: Section 2 defines some notations for later usage. Section 3 proposes and discusses the main algorithm of this paper. Section 4 concludes.[1]

2 Preliminaries

We recall some definitions from [9] and [27]. A vertex v of Π is called *reflex* if v's internal angle is greater than 180°. Let u be a vertex of Π which is adjacent to a reflex vertex v. Assume that the straight line uv intersects an edge of Π at v'. Then the segment $C = vv'$ partitions Π into two parts. C is called a *cut* of Π if C makes a convex vertex at v in the part containing the starting

*College of Computer Science and Technology, Huaqiao University, Xiamen, Fujian, China, li.fajie@yahoo.com

†Computer Science Department, The University of Auckland, Private Bag 92019, Auckland 1142, New Zealand, r.klette@auckland.ac.nz

[1] An expanded version is MI-tech report no. 51 at www.mi.auckland.ac.nz/.

point s, and v is called a *defining vertex* of C. That part of Π which contains s is called *essential* part of C and is denoted by $\Pi(C)$. The other part of Π is called the *pocket* induced by cut C, and C is the *associated* cut of the pocket. A cut C *dominates* a cut C' iff $\Pi(C)$ contains $\Pi(C')$. A cut is called *essential* if it is not dominated by another cut. A pocket is called *essential* if it does not contain any other pocket. A pocket is essential iff its associated cut is essential.

If two points u and v are on two different edges of Π, such that the segment uv partitions Π into two parts, then we say that uv is a *general cut* of Π. We may arbitrarily select one of both endpoints of the segment uv to be its *start point*. In the rest of this paper, for an essential cut C of Π, we identify the defining vertex of C with its start point. If C_0, C_1, ..., C_{k-1} are all the essential cuts of Π such that their start points are ordered clockwise around on $\partial\Pi$, then we say that C_0, C_1, ..., C_{k-1} and Π *satisfy the condition of the fixed watchman route problem*. Let p, $q \in \Pi$; if $pq \subset \Pi$ then q *can see* p (with respect to Π), and p is a *visible point* of q. Let $q \in \Pi$ and assume a segment $s \subset \Pi$. If, for each $p \in s$, q can see p, then we say that q *can see* s. Let $q \in \Pi$, segment $s \subset \Pi$, $p \in s$, and p is not an endpoint of s. If q can see p, but for any sufficiently small $\varepsilon > 0$, q cannot see p', where $p' \in s$ and Euclidean distance $d_e(p, p') = \varepsilon$, then we say that p is a *visible extreme point* of q (with respect to s and Π). Let segment $s \subset \Pi$ and $q \in \Pi \setminus s$. If there exists a subsegment $s' \subseteq s$ such that q can see s', and each endpoint of s' is a visible extreme point of q or an endpoint of s, then we say that s' is a *maximal visible segment* of q (with respect to Π). Let s_0, s_1, ..., and s_{k-1} be k segments ($k \geq 2$) in three-dimensional Euclidean space (in short: $3D$), $p \in s_0$, and $q \in s_{k-1}$. Let $L_S(p, q)$ be the length of the shortest path, starting at p, then visiting segments s_1, ..., and s_{k-2} in order, and finally ending at q, where $S = \langle s_0, s_1, \ldots, s_{k-1} \rangle$. Let p, $q \in \Pi$. We denote by $L_\Pi(p, q)$ the length of the shortest path from p to q inside of Π. Let ρ be a polygonal path and $V(\rho)$ the set of all vertices of ρ; $|V(\rho)|$ is the number of vertices of ρ. Denote by $C(S)$ the convex hull of a set S. Let S_0, S_1, ..., and S_{k-1} be k non-empty sets; let $\prod_{i=0}^{k-1} S_i$ be the cross product of those sets. This ends our introduction of technical terms. We also recall in one place here two results which will be cited later in this paper:

- **Lemma 1** ([9], page 475) *A solution to the fixed watchman route problem (i.e., a shortest tour) visits the essential cuts in the same order as the defining vertices meet* $\partial\Pi$.
- **Theorem 2** ([27], Theorem 1) *Given a simple polygon* Π*; the set* $\mathcal{C}$ *of all essential cuts for the watchman route in* Π *can be computed in* $\mathcal{O}(n)$ *time.*

3 Algorithms

In this section, we describe and discuss now the promised algorithm for solving the fixed watchman route problem.

3.1 Two Procedures and Main Algorithm

The main algorithm uses two procedures; the second applies a 2D ESP algorithm (see [16], pages 639–641). We present the used procedures first, and the main algorithm later.

As described in Section B.1, the main idea of a *Rubberband Algorithm* (RBA) is as follows: In each iteration, we update (by finding a local minimum or optimal vertex) the second vertex p_i for every three-subsequent-vertices subsequence p_{i-1}, p_i, p_{i+1} in a *step set* $\{S_1, S_2, \ldots, S_k\}$. The first procedure below computes the maximal visible segment, which is actually an element of the step set of the used RBA. The second procedure is used for updating the vertices.

Procedure 1 Compute Maximal Visible Segment
Input: Polygon Π and a general cut C of Π; let v_1 and v_2 be two endpoints of C; two points p and q such that $p \in C$ and p is a visible point of $q \in \partial\Pi \setminus C$.
Output: Two points p'_1, $p'_2 \in C$ such that p is in the segment $p'_1p'_2$, and $p'_1p'_2$ is the maximal visible segment of q.

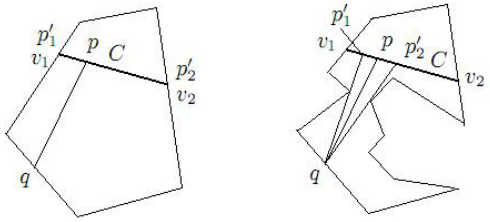

Figure 1: Illustration for Procedure 1.

We describe Procedure 1 informally. – Case 1: p is not an endpoint of C. For $i \in \{1, 2\}$, if q can see v_i, (see left, Figure 1), let p'_i be v_i; otherwise, let V_i be the set of vertices in $V(\partial\Pi)$ such that each vertex in V_i is in $\triangle qpv_i$. Apply the convex hull algorithm (see, e.g., [15] or Figure 13.7, [12]) to compute $C(V_i)$. Apply the tangent algorithm (see [20]) to find a point $p'_i \in C$ such that qp'_i is a tangent to $C(V_i)$ (see right of Figure 1). – Case 2: p is an endpoint of C. Without loss of generality, assume that $p = v_1$. Let p'_1 be p. Let V_2 be the set of vertices in $V(\partial\Pi)$ such that each vertex in V_2 is in $\triangle qpv_i$. Apply the convex hull algorithm to compute $C(V_2)$. Apply the tangent algorithm to find a point $p'_2 \in C$ such that qp'_2 is a tangent to $C(V_2)$.

Procedure 2 Handling of Three General Cuts
Input: Three general cuts C_1, C_2 and C_3 of Π; three points $p_i \in C_i$, for $i = 1, 2, 3$; and a degeneration accuracy constant $\varepsilon_2 > 0$.
Output: An updated shorter path $\rho(p_1, \ldots, p_2, \ldots, p_3)$ that might also contain vertices of the polygon Π.

1: For both $i \in \{1, 2\}$, let $\{p_i, p_{i+1}\}$ (where $p_i \in C_i$) be the input for the 2D ESP algorithm; the output is a set V_{ii+1} - the set of vertices of a shortest path from p_i to p_{i+1} inside of Π. Let V be $V_{12} \cup V_{23}$.
2: Find q_1 and $q_3 \in V$ such that $\langle q_1, p_2, q_3 \rangle$ is a subsequence of V (i.e., q_1, p_2, q_3 appear consecutively in V).
3: Let $C = C_2$, $p = p_2$, $q = q_i$, apply Procedure 1 to find the maximal visible segment $s_i = p_1' p_2'$ of q_i, $i = 1, 3$.
4: Find vertex $p_2' \in s_2 = s_1 \cap s_3$ such that $d_e(q_1, p_2') + d_e(p_2', q_3) = \min\{d_e(q_1, p') + d_e(p', q_3) : p' \in s_2\}$.
5: If $C_2 \cap C_1$ (or C_3) $\neq \emptyset$ and p_2' is the intersection point, then ε_2-transform p_2' into another point (still denoted by p_2') in C_2.
6: Update V by letting p_2 be p_2'.

Note that in Procedure 2, if C_1 or C_3 degenerates to a single point, then this procedure still works correctly.

Algorithm 1 Main Algorithm
Input: k essential cuts C_0, C_1, ..., C_{k-1}, and Π, which satisfy the condition of the fixed WRP, and points $p_i \in C_i$, where $i = 0, 1, 2, \ldots, k-1$; and an accuracy constant $\varepsilon > 0$ and a degeneration accuracy constant $\varepsilon_2 > 0$.
Output: An updated closed $\{1 + 4k[r(\varepsilon) + \varepsilon_2]/L\}$-approximation path $\rho(s, p_0, \ldots, p_1, \ldots, p_{k-1}, s)$, which may also contain vertices of Π, where L is the length of an optimal path, $r(\varepsilon)$ the upper error bound[2] for distances between p_i and the corresponding optimal vertex p_i': $d_e(p_i, p_i') \leq r(\varepsilon)$, for $i = 0, 1, \ldots, k-1$.

The following pseudo code is fairly easy to read, and we defer from providing another (more informal) high level description of Algorithm 1.

1: For $i \in \{0, 1, \ldots, k-1\}$, let p_i be the center of C_i.
2: Let V_0 and V be a sequence of points $\langle p_0, p_1, \ldots, p_{k-1} \rangle$; L_1 be $\sum_{i=-1}^{k} L_\Pi(p_i, p_{i+1})$; and L_0 be ∞ ($p_{-1} = p_k = s$).
3: **while** $L_0 - L_1 \geq \varepsilon$ **do**
4: **for** each $i \in \{0, 1, \ldots, k-1\}$ **do**
5: Let C_{i-1}, C_i, C_{i+1}, p_{i-1}, p_i, p_{i+1} and Π be the input for Procedure 2, which updates p_i in V_0. ($C_{-1} = C_k = p_{-1} = p_k = s$)
6: Let U_i be the sequence of vertices of the path $\rho(p_{i-1}, \ldots, p_i, \ldots, p_{i+1})$ with respect to C_{i-1}, C_i and C_{i+1} (inside of Π); let U_i be $\langle q_1, q_2, \ldots, q_m \rangle$.
7: Insert (after p_{i-1}) the points of sequence U_i (in the given order) into V_0; i.e., we have that $V_1 = \langle p_0, p_1, \ldots, p_{i-1}, q_1, q_2, \ldots, q_m, p_{i+1}, \ldots, p_{k-1} \rangle$. (Note: sequence V_1 is the updated sequence V_0, after inserting U_i)
8: **end for**
9: Let L_0 be L_1 and V_0 be V (Note: we use the updated original sequence V instead of V_1 for the next iteration).
10: Calculate the perimeter L_1 of the polygon, given by the sequence V_1 of vertices.
11: **end while**
12: Output sequence V_1, and the desired length equal to L_1.

[2]It is obvious to see that $\lim_{\varepsilon \to 0} r(\varepsilon) = 0$

3.2 Correctness and Time Complexity

Theorem 3 *If the chosen accuracy constant $\varepsilon > 0$ is sufficiently small, then Algorithm 1 outputs a unique $\{1+4k \cdot [r(\varepsilon)+\varepsilon_2]/L\}$-approximation (closed) path with respect to the step set $\langle S_0, S_1, \ldots, S_{k-1}, S_0 \rangle$, for any initial path.*

See Section C for the proof of Theorem 3.

Theorem 3 shows that Algorithm 1 outputs an approximate solution to the fixed WRP; we have the following:

Theorem 4 *Algorithm 1 outputs an $\{1 + 4k \cdot [r(\varepsilon) + \varepsilon_2]/L\}$-approximation solution to the fixed WRP.*

Proof. By Corollary 7, $\sum_{i=-1}^{k} L_\Pi(p_i, p_{i+1}) : \prod_{i=-1}^{k} C_i \to \mathbb{R}$ is a convex function, where $L_\Pi(p_i, p_{i+1})$ is defined as in Step 2 of Algorithm 1. Proposition 3 and Theorem 3 prove then the theorem. □

Regarding the time complexity of our solution to the fixed WRP, we first state the fact that Procedure 1 and Procedure 2 can be computed in time $\mathcal{O}(|V(\partial\Pi)|)$. Furthermore, note that the main computation is in the two stacked loops. The while-loop takes $\kappa(\varepsilon)$ iterations. By the stated fact, the for-loop can be computed in time $\mathcal{O}(k \cdot |V(\partial\Pi)|)$. Thus, Algorithm 1 can be computed in time

$$\kappa(\varepsilon) \cdot \mathcal{O}(k \cdot |V(\partial\Pi)|)$$

By Lemma 1 and Theorem 2, we may conclude that this paper provided an $\{1+4k \cdot [r(\varepsilon)+\varepsilon_2]/L\}$-approximation solution to the fixed WRP, having time complexity $\kappa(\varepsilon) \cdot \mathcal{O}(k \cdot |V(\partial\Pi)|)$, where k is the number of essential cuts, and L is the length of an optimal watchman route.

4 Concluding Remarks

This paper applies basic ideas of RBAs, which were proposed in digital geometry [3, 12] for the specific 3D ESP

of calculating shortest Euclidean "loops" in a sequence of cubes. We refined those ideas such that we now have a general "arc" version of an RBA; see Algorithm 2. This paper provides a simple and efficient way for solving the system (4) formed by partial differential equations. Algorithm 2 runs in time $\kappa(\varepsilon) \cdot \mathcal{O}(k)$, while the solution proposed by [7] has a time complexity which is doubly exponential in k. The basic idea of an RBA might be generalized to establish a whole class of rubberband algorithms (RBAs) for solving various Euclidean shortest path problems. The main algorithm of this paper (Algorithm 1) is just an example for such an RBA. As indicated in Note 1, in distinction to already published approximation algorithms, our algorithm offers a high accuracy. In some simple polygons, we find the exact solution to the fixed WRP, in the others we converge to the correct solution. A large number of experimental results also indicate that $\kappa(\varepsilon) = \mathcal{O}(k)$, where k is the number of essential cuts. It remains a challenge to prove a smallest upper bound for $\kappa(\varepsilon)$.

Altogether, our algorithm is not only faster than previously published solutions to the fixed WRP, but also significantly simpler, easier to understand and to implement.

References

[1] E. M. Arkin, J. S. B. Mitchell, and C. Piatko. Minimum-link watchman tours. Report, University at Stony Brook, 1994.

[2] T. Asano, S. K. Ghosh, and T. C. Shermer. Visibility in the plane. In *Handbook of Computational Geometry* (J.-R. Sack and J. Urrutia, editors), pages 829–876, Elsevier, 2000.

[3] S. Boyd and L. Vandenberghe. *Convex Optimization.* Cambridge University Press, Cambridge, UK, 2004.

[4] T. Bülow and R. Klette. Digital curves in 3D space and a linear-time length estimation algorithm. *IEEE Trans. Pattern Analysis Machine Intelligence*, **24**:962–970, 2002.

[5] S. Carlsson, H. Jonsson, and B. J. Nilsson. Approximating the shortest watchman route in a simple polygon. Technical report, Lund University, Sweden, 1997.

[6] S. Carlsson, H. Jonsson, and B. J. Nilsson. Finding the shortest watchman route in a simple polygon. *Discrete Computational Geometry*, **22**:377–402, 1999.

[7] W. Chin and S. Ntafos. Optimum watchman routes. *Information Processing Letters*, **28**:39–44, 1988.

[8] W.-P. Chin and S. Ntafos. Shortest watchman routes in simple polygons. *Discrete Computational Geometry*, **6**:9–31, 1991.

[9] M. Dror, A. Efrat, A. Lubiw, and J. Mitchell. Touring a sequence of polygons. In Proc. *STOC*, pages 473–482, 2003.

[10] M. Hammar and B. J. Nilsson. Concerning the time bounds of existing shortest watchman routes. In Proc. *FCT'97*, LNCS 1279, pages 210–221, 1997.

[11] D. S. Hochbaum (editor). *Approximation Algorithms for NP-Hard Problems.* PWS Pub. Co.,Boston, 1997.

[12] R. Klette and A. Rosenfeld. *Digital Geometry.* Morgan Kaufmann, San Francisco, 2004.

[13] H. Luo and A. Eleftheriadis. Rubberband: an improved graph search algorithm for interactive object segmentation. In Proc. *Int. Conf. Image Processing*, volume 1, pages 101–104, 2002.

[14] C. Mata and J. S. B. Mitchell. Approximation algorithms for geometric tour and network design problems. In Proc. *Ann. ACM Symp. Computational Geometry*, pages 360–369, 1995.

[15] A. Melkman. On-line construction of the convex hull of a simple polygon. *Information Processing Letters*, **25**:11–12, 1987.

[16] J. S. B. Mitchell. Geometric shortest paths and network optimization. In *Handbook of Computational Geometry* (J.-R. Sack and J. Urrutia, editors). pages 633–701, Elsevier, 2000.

[17] B. J. Nilsson. Guarding art galleries; Methods for mobile guards. Ph.D. Thesis, Lund University, Sweden, 1995.

[18] A. W. Roberts and V. D. Varberg. *Convex Functions.* Academic Press, New York, 1973.

[19] R. T. Rockafellar. *Convex Analysis.* Princeton University Press, Princeton, N.J., 1970.

[20] D. Sunday. Algorithm 14: Tangents to and between polygons. See `http://softsurfer.com/Archive/algorithm_0201/` (last visit: November 2008).

[21] X. Tan, T. Hirata, and Y. Inagaki. An incremental algorithm for constructing shortest watchman route algorithms. *Int. J. Comp. Geom. and Appl.*, **3**:351–365, 1993.

[22] X. Tan and T. Hirata. Constructing shortest watchman routes by divide-and-conquer. In Proc. *ISAAC*, LNCS 762, pages 68–77, 1993.

[23] X. Tan, T. Hirata, and Y. Inagaki. Corrigendum to 'An incremental algorithm for constructing shortest watchman routes'. *Int. J. Comp. Geom. and Appl.*, **9**:319–323, 1999.

[24] X. Tan. Fast computation of shortest watchman routes in simple polygons. *Information Processing Letters*, **77**:27–33, 2001.

[25] X. Tan. Approximation algorithms for the watchman route and zookeeper's problems. In Proc. *Computing and Combinatorics*, LNCS 2108, pages 201–206, Springer, Berlin, 2005.

[26] X. Tan. Approximation algorithms for the watchman route and zookeeper's problems. *Discrete Applied Mathematics*, **136**:363–376, 2004.

[27] X. Tan. A linear-time 2-approximation algorithm for the watchman route problem for simple polygons. *Theoretical Computer Science*, **384**:92–103, 2007.

A The Origin of Rubberband Algorithms

In this section, we recall the original rubberband algorithm, as defined for regular grids in 3D [3]. The ideas and basic steps of this algorithm will then guide us when solving the WRP. The *original* (or *first*) rubberband algorithm was defined in the context of 3D digital geometry, assuming a regular orthogonal grid in 3D.

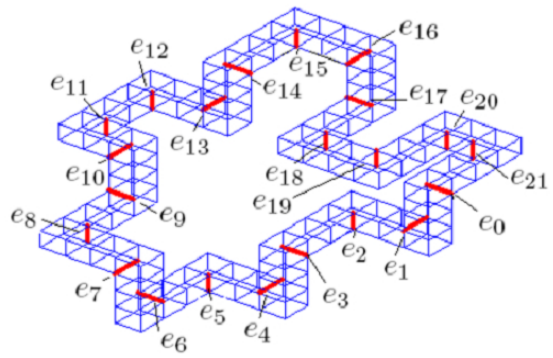

Figure 2: Example of a cube-curve which has 22 critical edges.

A *cube-curve* g is a loop of face-connected grid cubes in a 3D regular orthogonal grid; the union $\mathbf{g}$ of those cubes defines the *tube* of g. The original rubberband algorithm in [3] discusses ESPs in such tubes, which are also called *minimum-length polygonal curves* (MLPs). A *critical edge* of a cube-curve g is such a grid edge which is incident with exactly three different cubes contained in g. Figure 2 shows all the critical edges of a cube-curve.

The computation of 3D MLPs was at first published in [1, 2, 3, 5], proposing and discussing a *rubberband algorithm*[3]. This *original rubberband algorithm* is also published in the book [12].

Let $\rho = (p_0, p_1, \cdots, p_m)$ be a polygonal curve contained in a tube $\mathbf{g}$. Such a curve is *complete* if it intersects with every cube of g. A polygonal curve ξ is a *g-transform* of ρ iff ξ may be obtained from ρ by a finite number of steps, where each step is a replacement of a triple a, b, c of vertices by a polygonal sequence $a, b_1, \cdots, b_k, c$ such that the polygonal sequence $a, b_1, \cdots, b_k, c$ is contained in the same set of cubes of g as the polygonal sequence a, b, c.

Assume a polygonal curve $\rho = (p_0, p_1, \cdots, p_m)$ and three pointers addressing vertices at positions $i-1, i$ and $i+1$ in this curve. There are three different *options* that may occur, and which define a specific g-transform:

(O_1) Point p_i can be deleted iff $p_{i-1}p_{i+1}$ is a line segment within the tube. Then the subsequence (p_{i-1}, p_i, p_{i+1}) is replaced in the curve by (p_{i-1}, p_{i+1}). In this case, the algorithm continues with vertices $p_{i-1}, p_{i+1}, p_{i+2}$.

(O_2) The closed triangular region $\triangle(p_{i-1}, p_i, p_{i+1})$ intersects more than just three critical edges of cube-curve g (i.e., a simple deletion of p_i would not be sufficient anymore). This situation is solved by calculating a convex arc and by replacing point p_i by a sequence of vertices $q_1, \cdots, q_k$ on this convex arc between p_{i-1} and p_{i+1} such that the sequence of line segments $p_{i-1}q_1, \ldots, q_kp_{i+1}$ lies within the tube. In this case, the algorithm continues with a triple of vertices starting with the calculated new vertex q_k.

If (O_1) and (O_2) do not lead to any change, the third option may lead to an improvement (i.e., a shorter polygonal curve which is still contained and complete in the given tube). Here, l_e denotes the straight line defined by extending an edge e at both ends to infinity:

(O_3) Point p_i may be moved on its critical edge to obtain an optimum position p_{new} minimizing the total length of both line segments $p_{i-1}p_{new}$ and $p_{new}p_{i+1}$. First, find $p_{opt} \in l_e$ such that

$$|p_{opt} - p_{i-1}| + |p_{opt} - p_{i+1}| = min_{p \in l_e} L(p)$$

with $L(p) = |p - p_{i-1}| + |p - p_{i+1}|$. Then, if p_{opt} lies on the closed critical edge e, let $p_{new} = p_{opt}$. Otherwise, let p_{new} be that vertex bounding e and lying closest to p_{opt}.

The authors showed in various previous publications (see, for example, [6]) that the basic idea of (O_3) can be generalized to establish a whole class of rubberband algorithms (RBAs) for solving various Euclidean shortest path problems. The main algorithm of the report is also just some kind of adaptation of this original rubberband algorithm.

B Sequences of Line Segments in 3D

In this section, we present a simple rubberband algorithm which receives as input a finite sequence of line segments in 3D. Later it will be generalized and then becomes the main algorithm; see Section 3. We discuss the simple RBA without degenerate cases in Subsections B.1 and B.2, leaving degenerate cases in Subsection B.3 with which we can simply deal with, without affecting the time complexity of our algorithms.

B.1 The Algorithm

The numerical accuracy of results obtained by a rubberband algorithm is controlled by a chosen accuracy constant $\varepsilon > 0$. For example, with respect to current computer technology, a constant such as $\varepsilon = 10^{-15}$ is appropriate. However, whenever ε is mentioned, have in mind that its value may further decrease with the progress in computer technology.

Algorithm 2 (RBA for a sequence of pairwise disjoint 3D line segments)
Input: A sequence of k pairwise disjoint line segments $S_1, S_2, \ldots, S_k$ in 3D; two points p, $q \notin \bigcup_{i=1}^{k} S_i$, and an accuracy constant $\varepsilon > 0$.
Output: A sequence $\langle p, p_1, p_2, \ldots, p_k, q \rangle$ of an $[1 + 4(k+1)r(\varepsilon)/L]$-approximation path which starts at p, then visits (i.e., passes through) segments S_i at p_i in the given order, and finally ends at q, where L is the length of an optimal path, $r(\varepsilon)$ the upper error bound[4] the for distances between p_i and the corresponding optimal vertex p'_i: $d_e(p_i, p'_i) \leq r(\varepsilon)$, for $i = 1, \ldots, k$, where d_e denotes the Euclidean distance.

[3] Not to be confused with a 2D image segmentation algorithm of the same name [13].

[4] It is obvious to see that $\lim_{\varepsilon \to 0} r(\varepsilon) = 0$

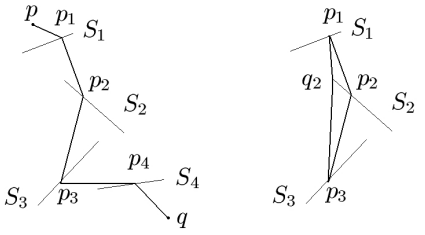

Figure 3: Illustration for Algorithm 2.

We provide an informal specification of the algorithm. The algorithm consists of two parts: initialization and iteration step. In the initialization part, we select an initial path and calculate its length. For example, take arbitrarily one point in each segment and connect those points into a sequence, to obtain an initial path. We could also take the center or one of the endpoints in each segment. In each iteration cycle, we update all the vertices of the path in sequence: For every three subsequent vertices p_{i-1}, p_i and p_{i+1} in the path, we consider the first and third vertices, p_{i-1} and p_{i+1}, as being fixed and slide p_i freely in segment S_i into an optimal point with respect to this local configuration; we update p_i by replacing it with this newly detected, locally optimal point. That is, we apply

$$\begin{aligned} & d_e(p_{i-1}, p_i) + d_e(p_i, p_{i+1}) \\ = \min\{ & d_e(p_{i-1}, p) + d_e(p, p_{i+1}) : p \in S_i\} \end{aligned}$$

At the end of the iteration cycle we compute the difference between the length of the previous path to that of the current (i.e., updated) path; if this is less than or equals ε then this terminates the algorithm. Otherwise, we go to the next iteration cycle.

Figure 3 shows on the left an initial path $\langle p, p_1, p_2, p_3, p_4, q\rangle$ for Algorithm 2; on the right it shows an optimal point $q_2 \in S_2$ for the given positions of p_1 and p_3, defining the new position of p_2. Note that optimal points are not necessarily at endpoint positions.

B.2 Proof of Correctness

We have to show that these repeated local optimizations of Algorithm 2 ensure that the calculated path converges against the shortest path.

If an expression is derived from a finite number of polynomials in x by only applying operations "+", "-", "×", "÷", or "$\sqrt{}$" finitely often, then we say that this expression is a *simple compound of polynomials* in x. Let f be a function, mapping $\mathbb{R}$ into $\mathbb{R}$. If interval $J \subseteq I$, then we say that J is a *subinterval* of interval I. If f is monotonous in J, then we say that J is a *monotonous interval of f in I*. If x_0 satisfies $f(x_0) = 0$, and for a sufficiently small numbers $\delta > 0$ and all x_1 in the interval $(x_0 - \delta, x_0 + \delta)$, $f(x_1) \neq 0$, then we say that x_0 is an *isolated solution* of $f(x)$. If $I \subset \mathbb{R}$ is a bounded interval, and for all x in I, $f(x) = 0$, then we say that I is an *interval solution* to $f(x)$. We generalize those two definition for the multi-variable case: Let f be a function from $\mathbb{R}^m$ into $\mathbb{R}$, for $m \geq 1$. At a point $(x_{1_0}, x_{2_0}, \ldots, x_{m_0})$, assume that $f(x_{1_0}, x_{2_0}, \ldots, x_{m_0}) = 0$, and for a sufficiently small number $\delta > 0$ and all $(x_{1_1}, x_{2_1}, \ldots, x_{m_1})$ such that $i = 1, 2, \ldots, m$, and

$$x_{i_1} \in (x_{i_0} - \delta, x_{i_0} + \delta) \setminus \{x_{i_0}\}$$

we have that $f(x_{1_1}, x_{2_1}, \ldots, x_{m_1}) \neq 0$; then point $(x_{1_0}, x_{2_0}, \ldots, x_{m_0})$ is an *isolated solution* of $f(x_1, x_2, \ldots, x_m)$. If $I_1 \subset \mathbb{R}$ is a bounded interval, and for $i = 2, 3, \ldots, m$, $I_i \subset \mathbb{R}$ is a bounded interval or a single point (i.e., a degenerated bounded interval), and for all $x_{1_1} \in I_1$, there exists an x_{i_1} in I_i such that $f(x_{1_1}, x_{2_1}, \ldots, \ldots x_{m_1}) = 0$, then we say that $\langle I_1, I_2, \ldots, I_m\rangle$ is an *interval* solution to $f(x_1, x_2, \ldots, x_m)$. We say that $(x_{1_0}, x_{2_0}, \ldots, x_{m_0})$ is an *isolated* solution to the system formed by $f_j(x_1, x_2, \ldots, x_m) = 0$, for $j = 1, 2, \ldots, m$, if for any of those j, $(x_{1_0}, x_{2_0}, \ldots, x_{m_0})$ is an isolated solution to $f_j(x_1, x_2, \ldots, x_m)$. We say that $\langle I_1, I_2, \ldots, I_m\rangle$ is an *interval solution* to the system formed by $f_j(x_1, x_2, \ldots, x_m) = 0$, for $j = 1, 2, \ldots, m$, if for any of those j, $\langle I_1, I_2, \ldots, I_m\rangle$ is an *interval* solution to $f_j(x_1, x_2, \ldots, x_m)$.

Let s_0, s_1 and s_2 be three (closed) pairwise disjoint line segments; the two endpoints of s_i be $a_i = (a_{i_1}, a_{i_2}, a_{i_3})$ and $b_i = (b_{i_1}, b_{i_2}, b_{i_3})$. A point $p_i \in s_i$, for $i = 0, 1, 2$, may be written as

$$\begin{aligned} p_i(t_i) &= a_i + (b_i - a_i)t_i \\ &= (a_{1_1} + (b_{1_1} - a_{1_1})t_i, \\ &\quad a_{1_2} + (b_{1_2} - a_{1_2})t_i, \\ &\quad a_{1_3} + (b_{1_3} - a_{1_3})t_i) \end{aligned}$$

where $t_i \in [0, 1]$. Let
$d(t_0, t_1, t_2) = d_e(p_1(t_1), p_0(t_0)) + d_e(p_1(t_1), p_2(t_2))$. Then we have the following

Corollary 1

$$\frac{\partial d(t_0, t_1, t_2)}{\partial t_1} = 0 \tag{1}$$

implies that t_2 is a simple compound of polynomials of t_0 and t_1. All the t_0, t_1 and t_2 are in [0, 1].

Proof. The formula
$d_e(p_1(t_1), p_0(t_0)) =$

$$\sqrt{\sum_{i=1}^{3}\{[a_{1_i} + (b_{1_i} - a_{1_i})t_1)] - [a_{0_i} + (b_{0_i} - a_{0_i})t_0)]\}^2} \tag{2}$$

can be simplified: Without loss of generality, we can assume that s_1 is parallel to one of the three coordinate axes. It follows that only one element of the set $\{b_{1_i} - a_{1_i} : i = 1, 2, 3\}$ is not equal to 0, and the other two are equal to 0. Thus, we can assume that the expression inside the square root in Equation (2) can be written as

$$\begin{aligned} & \sum_{i=1}^{3}\{[a_{1_i} + (b_{1_i} - a_{1_i})t_1)] - [a_{0_i} + (b_{0_i} - a_{0_i})t_0)]\}^2 \\ &= \{[a_{1_1} + (b_{1_1} - a_{1_1})t_1)] - [a_{0_1} + (b_{0_1} - a_{0_1})t_0)]\}^2 \\ &\quad + \{a_{1_2} - [a_{0_2} + (b_{0_2} - a_{0_2})t_0)]\}^2 \\ &\quad + \{a_{1_3} - [a_{0_3} + (b_{0_3} - a_{0_3})t_0)]\}^2 \end{aligned}$$

Thus, we have that

$$d_e(p_1, p_0) = |A_1|\sqrt{(t_1 + B_0t_0 + C_0)^2 + D_0{t_0}^2 + E_0t_0 + F_0}$$

where A_1 is a function of a_{1_i} and b_{1_i}; B_0, C_0, D_0, E_0 and F_0 are functions of a_{0_i}, b_{0_i}, a_{1_i} and b_{1_i}, for $i = 0, 1, 2$. – Analogously, we have that

$$d_e(p_1, p_2) = |A_1|\sqrt{(t_1 + B_2 t_2 + C_2)^2 + D_2 {t_2}^2 + E_2 t_2 + F_2}$$

where B_2, C_2, D_2,E_2 and F_2 are functions of a_{1_i}, b_{1_i}, a_{2_i} and b_{2_i} for $i = 0, 1, 2$. By Equation (1) or the following,

$$\frac{\partial(d_e(p_1, p_0) + d_e(p_1, p_2))}{\partial t_1} = 0$$

we have that

$$\frac{t_1 + B_0 t_0 + C_0}{\sqrt{(t_1 + B_0 t_0 + C_0)^2 + D_0 {t_0}^2 + E_0 t_0 + F_0}} + \frac{t_1 + B_2 t_2 + C_2}{\sqrt{(t_1 + B_2 t_2 + C_2)^2 + D_2 {t_2}^2 + E_2 t_2 + F_2}} = 0$$

This equation can be written as

$$A {t_2}^2 + B t_2 + C = 0$$

where A, B, and C are polynomials of t_0, t_1 (and a_{0_i}, b_{0_i}, a_{1_i}, b_{1_i}, a_{2_i} and b_{2_i} for $i = 0, 1, 2$). To keep t_2 inside of [0, 1], let $t_2 = 0$ if we have to satisfy $t_2 < 0$; and let $t_2 = 1$ if we have to satisfy $t_2 > 1$. This proves the corollary. □

Analogously, we have

Corollary 2 *Equation* (1) *uniquely implies that* t_1 *is a continuous function in* t_0 *and* t_2.

Proof. We may translate two points $p_0(t_0)$ and $p_2(t_2)$, and line segment s_1 such that the endpoint a_1 of s_1 is identical to the origin. Then rotate $p_0(t_0)$, $p_2(t_2)$, and s_1 such that the other endpoint b_1 of s_1 is (also) on the x-axis. Let $p_0(t_0) = (p_{0_1}, p_{0_2}, p_{0_3})$, $p_2(t_0) = (p_{2_1}, p_{2_2}, p_{2_3})$. After translation and rotation, we have that $a_1 = (0, 0, 0)$ and $b_1 = (b_{1_1}, 0, 0)$. Thus, $p_1(t_1) = (b_{1_1} t_1, 0, 0)$, and

$$d_e(p_1, p_0) = \sqrt{(b_{1_1} t_1 - p_{0_1})^2 + {p_{0_2}}^2 + {p_{0_3}}^2}$$

$$d_e(p_1, p_2) = \sqrt{(b_{1_1} t_1 - p_{2_1})^2 + {p_{2_2}}^2 + {p_{2_3}}^2}$$

Equation (1) is equivalent to

$$\frac{\partial(d_e(p_1, p_0) + d_e(p_1, p_2))}{\partial t_1} = 0$$

From this we obtain that

$$\frac{b_{1_1} t_1 - p_{0_1}}{\sqrt{(b_{1_1} t_1 - p_{0_1})^2 + {p_{0_2}}^2 + {p_{0_3}}^2}} + \frac{b_{1_1} t_1 - p_{2_1}}{\sqrt{(b_{1_1} t_1 - p_{2_1})^2 + {p_{2_2}}^2 + {p_{2_3}}^2}} = 0$$

This equation has a unique solution

$$t_1 = \frac{p_{0_1}\sqrt{{p_{2_2}}^2 + {p_{2_3}}^2} + p_{2_1}\sqrt{{p_{0_2}}^2 + {p_{0_3}}^2}}{b_{1_1}(\sqrt{{p_{2_2}}^2 + {p_{2_3}}^2} + \sqrt{{p_{0_2}}^2 + {p_{0_3}}^2})}$$

Again, to keep t_2 inside of [0, 1], let $t_2 = 0$ if we have to satisfy $t_2 < 0$; and let $t_2 = 1$ if we have to satisfy $t_2 > 1$. This proves the corollary. □

Let $s_0, s_1, \ldots,$ and s_{k+1} be $k + 2$ (closed) line segments. Let the two endpoints of s_i be $a_i = (a_{i_1}, a_{i_2}, a_{i_3})$ and $b_i = (b_{i_1}, b_{i_2}, b_{i_3})$. Points $p_i \in s_i$, for $i = 0, 1, 2, \ldots, k + 1$, can be written as follows:

$$\begin{aligned} p_i(t_i) &= a_i + (b_i - a_i)t_i \\ &= (a_{1_1} + (b_{1_1} - a_{1_1})t_i, \\ &\quad a_{1_2} + (b_{1_2} - a_{1_2})t_i, \\ &\quad a_{1_3} + (b_{1_3} - a_{1_3})t_i) \end{aligned}$$

where $t_i \in [0, 1]$.

Let

$$d(t_0, t_1, t_2, \ldots, t_{k+1}) = \sum_{i=0}^{k} d_e(p_i(t_i), p_{i+1}(t_{i+1})) \quad (3)$$

Assume that both s_0 and s_{k+1} degenerate into single points p and q. Then we have that $t_0 = t_{k+1} = 0$. We also have the following

Corollary 3 *For each* $i \in \{1, 2, \ldots, k\}$,

$$\frac{\partial d(t_0, t_1, t_2, \ldots, t_k, t_{k+1})}{\partial t_i} = 0 \quad (4)$$

is equivalent to

$$\frac{\partial d(t_{i-1}, t_i, t_{i+1})}{\partial t_i} = 0 \quad (5)$$

where t_1, t_2, …, t_k *are in [0, 1].*

Note that Equation (4) is related to a global minimum property of the Euclidean path $\langle p, p_1, p_2, \ldots, p_k, q\rangle$ while Equation (5) is related to a local minimum property of the same path. Therefore, Corollary 3 describes a relationship between global and local minimum properties of the same path.

Corollary 4 *The equational system formed by Equation* (4) (*where* $i = 1, 2, \ldots, k$) *implies a unary equation* $f(t_1) = 0$ *which has only a finite number of isolated or interval solutions in* $[0, 1]$.

Proof. By Corollary 3 and Corollary 1, t_{i+1} is a simple compound of polynomials in t_{i-1} and t_i, denoted by $t_{i+1} = f_i(t_{i-1}, t_i)$. Thus, the system formed by Equation (4) (where $i = 1, 2, \ldots, k$) implies an equational system formed by $t_2 = f_2(t_0, t_1)$, $t_3 = f_3(t_1, t_2)$, $t_4 = f_4(t_2, t_3)$, $\cdots$, $t_k = f_k(t_{k-2}, t_{k-1})$, and $t_{k+1} = f_{k+1}(t_{k-1}, t_k)$. Now note that $t_0 = t_{k+1} = 0$. Therefore, $f(t_1)$ is a simple compound of polynomials in t_1. Note that function $f(t_1)$ has only a finite number of monotonous intervals in [0, 1], and $f(t_1)$ is differentiable in each of those monotonous intervals. Thus, $f(t_1)$ can be approximately expressed as a linear function in a finite number of monotonous subintervals in [0, 1]. Therefore, Function $f(t_1)$ has only a finite number of isolated or interval solutions in [0, 1]. This proves the corollary. □

Corollary 5 *Algorithm 2 defines a continuous function* $f_{RBA}(p_1, p_2, \ldots, p_k)$ *in* $\prod_{i=1}^{k} S_i = S_1 \times S_2 \times \ldots \times S_k$, *or a function* $f_{RBA}(t_1, t_2, \ldots, t_k)$ *in* $\prod_{i=1}^{k} I_i = I_1 \times I_2 \times \ldots \times I_k = [0, 1]^k$. *And* $f_{RBA}(t_1, t_2, \ldots, t_k)$ *has only a finite number of values.*

Proof. For each $(p_1, p_2, \ldots, p_k) \in \prod_{i=1}^{k} S_i$ or each $(t_1, t_2, \ldots, t_k) \in [0,1]^k$, Algorithm 2 outputs the vertices of an approximate path. It can also output the length of the approximate path, which is a positive real. In this way, Algorithm 2 defines a mapping from $\prod_{i=1}^{k} S_i$ to $\mathbb{R}$, or from $[0,1]^k$ into $\mathbb{R}$. By Corollary 2, and because Algorithm 2 will terminate after a finite number of steps, thus, $f_{RBA}(t_1, t_2, \ldots, t_k)$ is continues in its domain $[0,1]^k$.

To prove the second conclusion of the corollary, it is sufficient to prove that for each interval solution J to the equational system formed by Equation (4) (where $i = 1, 2, \ldots, k$), the following function

$$f_{RBA}(t_1, t_2, \ldots, t_k) : J \to \mathbb{R}$$

has only a finite number of values. Suppose that $f(t_1) \equiv 0$, where t_1 is in an interval $I \subseteq [0,1]$, and $f(t_1)$ is defined as in Corollary 4. By Corollary 4, $d(t_0, t_1, t_2, \ldots, t_{k+1})$ implies a unary length function $L(t_1)$, where t_1 is in an interval $I' \subseteq I$, $d(t_0, t_1, t_2, \ldots, t_{k+1})$ is defined as in Equation (3), and

$$\frac{d[L(t_1)]}{dt_1} \equiv 0$$

($t_1 \in I' \subseteq I$). This implies that the length function $L(t_1) \equiv$ *constant*, where $t_1 \in I' \subseteq I$. Thus, function

$$f_{RBA}(t_1, t_2, \ldots, t_k) : J \to \mathbb{R}$$

has only a finite number of values. We have proven the corollary. □

Theorem 5 *If the chosen accuracy constant ε is sufficiently small, then, for any initial path, Algorithm 2 outputs a unique $[1 + 4(k+1) \cdot r(\varepsilon)/L]$-approximation path.*

Proof. By Corollary 5, Algorithm 2 defines a function $f_{RBA}(p_1, p_2, \ldots, p_k)$ in $\prod_{i=1}^{k} S_i$ which is continuous and only maps into a finite number of positive real numbers (i.e., the lengths of paths), for any points $p_1, p_2, \ldots, p_k$ sampled in $\prod_{i=1}^{k} S_i$. Therefore, $f_{RBA}(p_1, p_2, \ldots, p_k)$ must be a singleton.

For each $i \in \{1, 2, \ldots, k-1\}$, the error of the difference between $d_e(p_i, p_{i+1})$ and $d_e(v_i, v_{i+1}))$ is at most $4 \cdot r(\varepsilon)$ because of $d_e(p_i, v_i) \leq r(\varepsilon)$. Let $p = p_0 = v_0$ and $q = p_{k+1} = v_{k+1}$. We obtain that

$$L \leq \sum_{i=0}^{k} d_e(p_i, p_{i+1}) \leq \sum_{i=0}^{k} [d_e(v_i, v_{i+1}) + 4r(\varepsilon)] = L + 4(k+1)r(\varepsilon)$$

Thus, the output path is an $[1 + 4(k+1) \cdot r(\varepsilon)/L]$-approximation path. This proves the theorem. □

Note 1 *In the proof of Theorem 5, it is possible to find the explicit expression for $r(\varepsilon)$. And it is obvious that $\lim_{\varepsilon \to 0} r(\varepsilon) = 0$. Therefore, unlikely the approximation algorithms mentioned in Section 1, Algorithm 2 could have very high accuracy.*

Based on

Proposition 1 (see [4, 7, 8]) *The shortest path from p to q, which passes through the interior points of a sequence $\langle S_1, S_2, \ldots, S_k \rangle$ of line segments in the given order, is unique.*

and also based on our experiments we conclude that the equational system formed by Equation (4) (where $i = 1, 2, \ldots, k$) has only isolated solutions in [0, 1].

We implemented[5] Algorithm 2, and were running the program several thousands of times. For each run we took a random configuration of line segments $S_1, S_2, \ldots, S_k$.

For example, Table 1 shows the results for three random configurations of 5,000 line segments. Each column summarizes results corresponding to one configuration. For each configuration of line segments, we ran Algorithm 2 fifty times with 50 random initial paths when starting the program. The table shows that for each configuration, although the lengths of initial paths are different, the lengths of final paths are approximately identical.

B.3 A Degenerate Case

In this section we study a degenerate case: when applying (O_3) of the original rubberband algorithm (see Section A), assume that at least two vertices of the obtained updated polygonal path are identical. In this case, RBAs may not work properly. Unfortunately, this may actually occur sometimes when working with RBAs, and we show (one possible way) how to handle such degenerate cases.

Having an option for dealing with such degenerate cases, we may even remove "pairwise disjoint" from the input conditions for the 3D line segments in Algorithm 2. The more general algorithm is now as follows:

Algorithm 3 (RBA for a sequence of arbitrary 3D line segments)
Input: A sequence of k line segments $S_1, S_2, \ldots, S_k$ in 3D; two points p and q which are both not in $\bigcup_{i=1}^{k} S_i$, and an accuracy constant $\varepsilon > 0$.
Output: Exactly the same as for Algorithm 2.

However, before specifying this algorithm, we discuss at first three examples, which will help to understand the issue of degenerated cases, and will then be used to motivate this modified RBA.

Example 1 *Let the input for Algorithm 4 be as follows (see also Figure 4):*

$$S_1 = q_1q_2,\ S_2 = q_2q_3,\ q_1 = (0,0),\ q_2 = (2,4),\ q_3 = (3,0),\ p = (1,0),\ \text{and}\ q = (2,0).$$

To initialize, let p_1 and p_2 be the centers of S_1 and S_2, respectively [i.e., $p_1 = (1,2)$, and $p_2 = (2{\cdot}5, 2)$]. We obtain that the length of the initialized polyline $\rho = \langle p, p_1, p_2, q \rangle$ is equal to 5·5616. Algorithm 4 finds the shortest path $\rho = \langle p, p_1', p_2', q \rangle$ where $p_1' = (0{\cdot}3646, 0{\cdot}7291)$, $p_2' = (2{\cdot}8636, 0{\cdot}5455)$ and the length of it is equal to 4·4944 (see Table 2, which lists resulting δs for the number t of iterations).

[5] The source code can be downloaded at `www.mi.auckland.ac.nz/`; follow the link at the 2009 MI-tech Report 51.

min iterations	2039	2888	2133
max iterations	3513	3243	8441
min run time	44·11 s	62·922 s	47·188 s
max run time	77·657 s	70·094 s	187·672 s
min initial length	827430	822952	822905
max initial length	846928	841860	839848
min final length	516994·66273890162	513110·99723050051	512768·28438387887
max final length	516994·66273896693	513110·99723056785	512768·28457121132

Table 1: Three examples of experimental results, for three randomly generated sequences of 5,000 line segments in 3D space.

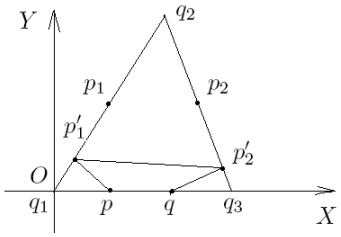

Figure 4: Illustration of a degenerate case of a rubberband algorithm.

t	δ
1	-0·8900
2	-0·1752
3	-0·0019
4	-1·2935e-005
5	-8·4435e-008
6	-5·4930e-010
7	-3·5740e-012

Table 2: Number t of iterations and resulting δs, for Example 1, illustrated by Figure 4, with $p_1 = (1, 2)$ and $p_2 = (2{\cdot}5, 2)$ as initialization points.

Example 2 *Now we modify Example 1 such that $p_1 = p_2 = q_2$; in this case, the output of Algorithm 4 will be false: the calculated path equals $\rho = \langle p, p'_1, p'_2, q\rangle$, where $p'_1 = q_2$ and $p'_2 = q_2$, and its length equals 8·1231.*

We call a situation as in the previous example a *degenerate case* when applying a rubberband algorithm. In general, it is defined by the occurrence of at least two identical vertices of the updated polygonal path. Such a degenerate case causes Algorithm 2 to fail.

A degenerate case can be solved approximately: we will not allow that a case $p_2 = q_2$ is happening. To do so, we can remove sufficiently small endsegments from both segments S_1 and S_2. The following example shows how to handle such a degenerate case.

Example 3 *We modify the initialization step of Example 2 as follows: Let the accuracy be*

$$\varepsilon = 1{\cdot}0 \times 10^{-100}$$

and let

$$\begin{array}{rcll} \delta' & = & 2{\cdot}221 \times 10^{-16} & \\ x_1 & = & 2 - \delta' & \text{and} \quad y_1 = 2 \times x_1 \\ x_2 & = & 2 + \delta' & \text{and} \quad y_2 = -4 \times (x_2 - 3) \\ p_1 & = & (x_1, y_1) & \text{and} \quad p_2 = (x_2, y_2) \end{array}$$

The length of the initialized polyline $\rho = \langle p, p_1, p_2, q\rangle$ is equal to 8·1231. Algorithm 4, to be defined below, will calculate the shortest path $\rho = \langle p, p'_1, p'_2, q\rangle$, where $p'_1 = (0{\cdot}3646, 0{\cdot}7291)$ and $p'_2 = (2{\cdot}8636, 0{\cdot}5455)$, and its length equals 4·4944 (see Table 3 for resulting δs in dependence of the number I of iterations).

Of course, if we leave the accuracy to be $\varepsilon = 1{\cdot}0 \times 10^{-10}$, then the algorithm will stop sooner, after less iterations. For example, the algorithm was implemented on a Pentium 4 PC using Matlab 7.04. If we change the value of δ' into

$$\delta' = 2{\cdot}22 \times 10^{-16}$$

then we obtain the same false result as that of Example 1. This is because this particular implementation was not able to recognize a difference between x_1 and $x_1 \mp 2{\cdot}22 \times 10^{-16}$. However, for practical applications in general, the value

$$\delta' = 2{\cdot}221 \times 10^{-16}$$

should be small or accurate enough for this implementation.

We summarize the method for handling a degenerate case with the modified rubberband algorithm (thus further preparing for defining Algorithm 2):

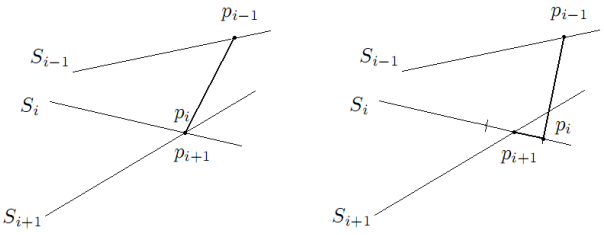

Figure 5: Handling a degenerate case of a rubberband algorithm.

t	δ	t	δ	t	δ	t	δ
1	-5·4831e-007	7	-1·2313	13	-7·0319e-010	19	8·8818e-016
2	-6·2779e-006	8	-2·0286	14	-4·5732e-012	20	8·8818e-016
3	-7·7817e-005	9	-0·2104	15	-3·0198e-014	21	-8·8818e-016
4	-9·6471e-004	10	-0·0024	16	-8·8818e-016	22	8·8818e-016
5	-0·0119	11	-1·6550e-005	17	8·8818e-016	23	-8·8818e-016
6	-0·1430	12	-1·0809e-007	18	-8·8818e-016	24	0

Table 3: Number t of iterations and resulting δs, for the example shown in Figure 4, with $p_1 = (2-\delta', 2(2-\delta'))$ and $p_2 = (2+\delta', -4((2+\delta')-3))$ as initialization points and $\delta' = 2{\cdot}221\text{e-}16$.

Let S_{i-1}, S_i and S_{i+1} be three continuous segments in the input such that $S_i \cap S_{i+1} \neq \emptyset$. Assume that p_{i-1}, p_i and p_{i+1} are three continuous vertices of the updated polygonal path such that p_i and p_{i+1} are identical (see left of Figure 5). Let ε_2 be a sufficiently small positive number. There are at most two possible points p in S_i such that $d_e(p, p_{i+1}) = \varepsilon_2$. Select one such point p such that $d_e(p, p_i) + d_e(p, p_{i+1})$ is smaller, and update the polygonal path by letting $p_i = p$ (see right of Figure 5). We say that p_i is ε_2-*transformed* to be p in S_i. Analogously to the explanation of Equation (6), the total error of this ε_2-transform is $4(k-1)\varepsilon_2$, for handling the degenerate case, and ε_2 is called a chosen *degeneration accuracy constant*.

To finalize this section, we provide a pseudo code of Algorithm 4 which also handles degenerate cases as discussed above. Let $p_0 = p$ and $p_{k+1} = q$. The output of this algorithm is a sequence $\langle p, p_1, p_2, \ldots, p_k, q\rangle$ of an $\{1+4[(k+1)r(\varepsilon)+(k-1)\varepsilon_2]/L\}$-approximation path which starts at p, then visits segments S_i at p_i in the given order, and finally ends at q, where L and $r(\varepsilon)$ are defined as in Algorithm 2, and ε_2 is a chosen degenerative accuracy constant. The pseudo code is listed below:

Algorithm 4 (RBA for a sequence of arbitrary 3D line segments)
Input: A sequence of k line segments $S_1, S_2, \ldots, S_k$ in 3D; two points p and q which are both not in $\bigcup_{i=1}^{k} S_i$, an accuracy constant $\varepsilon > 0$, and a degeneration accuracy constant $\varepsilon_2 > 0$.
Output: Modified from the output of Algorithm 2 (see Section B.3).

1: For each $i \in \{1, 2, \ldots, k\}$, let p_i be the center of S_i such that p_i (if $S_i \cap S_{i\mp1} \neq \emptyset$, then select p_i such that p_i is not the intersection point).
2: Calculate $L_1 = \sum_{i=0}^{k-1} L_S(p_i, p_{i+1})$; and let L_0 be 0.
3: **while** $L_1 - L_0 \geq \varepsilon$ **do**
4: **for** each $i \in \{1, 2, \ldots, k\}$ **do**
5: Compute a point $q_i \in S_i$ such that $d_e(p_{i-1}, q_i) + d_e(q_i, p_{i+1}) = \min\{d_e(p_{i-1}, p) + d_e(p, p_{i+1}) : p \in S_i\}$
6: **if** $S_i \cap S_{i\mp1} \neq \emptyset$ and q_i is the intersection point **then**
7: ε_2-transform q_i to be another point (still denoted by q_i) in S_i.
8: **end if**
9: Update the path $\langle p, p_1, p_2, \ldots, p_k, q\rangle$ by replacing p_i by q_i.
10: **end for**
11: Let L_0 be L_1 and calculate $L_1 = \sum_{i=0}^{k-1} L_S(p_i, p_{i+1})$.
12: **end while**
13: Return $\{p, p_1, p_2, \ldots, p_k, q\}$.

In other words, Algorithm 4 is modified from Algorithm 2 by adding Steps 6–8 in this pseudo code for handling the degenerate case. An informal specification of Algorithm 4 can also be obtained by modifying the informal specification of Algorithm 2 in Section B.1 as follows:

For each updated point p_i in Equation (1), if it is the intersection point between S_i and S_{i-1} or S_{i+1}, then ε_2-transform p_i into another point.

We call

$$\{S_1, S_2, \ldots, S_k\}$$

the *step set* of the rubberband algorithm, and each S_i is a *step element* of the rubberband algorithm, where $i = 1, 2, \ldots, k$.

B.4 Time Complexity

The time complexity of Algorithm 2 and Algorithm 4 can be analyzed as follows: The main computation occurs in two stacked loops. Each iteration of the inner for-loop runs in time $\mathcal{O}(k)$. In theory, the outer while-loop might take $\kappa(\varepsilon) = \frac{L_0 - L}{\varepsilon}$ times, where L is the length of an optimal path, L_0 is the length of an initial path. Thus, Algorithm 2 and Algorithm 4 will run in time $\kappa(\varepsilon)\mathcal{O}(k)$. We will see that $\frac{L_0-L}{\varepsilon}$ is usually too large to estimate $\kappa(\varepsilon)$. If we let L_m be the length of m-th updated path, where $m = 1, 2, \ldots$, then we have $\frac{L_0-L}{\varepsilon} =$

$$\frac{L_0 - L_1 + L_1 - L}{\varepsilon} > 1 + \frac{L_1 - L}{\varepsilon} > \cdots > m + \frac{L_m - L}{\varepsilon}$$

As the sequence $\{m + \frac{L_m - L}{\varepsilon}\}$ is monotonously decreasing and lower bounded by 0, it converges to $\kappa(\varepsilon)$.

Note 2 *It is obvious that $\kappa(\varepsilon)$ depends on the selection of initial path. By Theorem 5, we can take each vertex of the initial path as the center of each segment. Then $\kappa(\varepsilon)$ only depends the chosen accuracy constant ε.*

Algorithm 2 has been implemented and tested for a large number of various inputs. We let the chosen accuracy constant to be $\varepsilon = 10^{-15}$, and generated input for $k = 5000$, k=10,000, or k=20,000.

For each of those different numbers of segments, we were running the program several thousands of times. According to the resulting observations, the program often terminates

min iterations	2605	1522	2926
max iterations	3227	2741	7573
min work time	158·771 s	93·235 s	178·516 s
max work time	196·584 s	167·844 s	461·938 s
min initial length	3·32576e+006	3·33183e+006	3·33079e+006
max initial length	3·36785e+006	3·37652e+006	3·37889e+006
min final length	2085786·2964211311	2083340·4955095584	2068552·0753370232
max final length	2085786·2964214147	2083340·4955139237	2068552·0753745015

Table 4: Experiment results for three random configurations of 20,000 line segments.

after k iterations. These cases occurred at more than 90% of all inputs. So far, the worst case for all the tested inputs was $7 \cdot 2k$ iterations, and worst cases in this order occurred at less than 0·01% of all inputs.

Based on these thousands of runs, we conclude that Algorithm 2 runs practically with $\kappa(\varepsilon) = \mathcal{O}(k)$, or, equivalently, in time $\mathcal{O}(k^2)$.

For example, Table 4 shows the results for three random configurations of 20,000 line segments. Each column shows summary results corresponding to one configuration. For each configuration of line segments, we ran Algorithm 2 fifty times with 50 random initial paths for starting the program. The table shows that for each configuration, although the lengths of initial paths are different, the lengths of final paths are approximately identical.

C Proof of Theorem 5

We apply basic results of convex analysis; see, for example, [3, 18, 19]:

- **Theorem 6** ([19], Theorem 3.5) *Let S_1 and S_2 be convex sets in $\mathbb{R}^m$ and $\mathbb{R}^n$, respectively. Then $S_1 \times S_2$ is a convex set in $\mathbb{R}^{m+n}$, where m, $n \in \mathbb{N}$.*
- **Proposition 2** *Each line segment is a convex set* ([3], page 27)*; each norm on $\mathbb{R}^n$ is a convex function* ([3], page 72)*; a nonnegative weighted sum of convex functions is a convex function* ([3], page 72)*.*
- **Proposition 3** ([19], page 264) *Let f be a convex function. If x is a point where f has a finite local minimum, then x is a point where f has its global minimum.*

By Theorem 6 and Proposition 2, we have the following

Corollary 6 *$L_S(p, q)$: $s_0 \times s_{k-1} \to \mathbb{R}$ is a convex function.*

Let C_1, C_2, and Π satisfy the condition of the fixed watchman route problem. By Corollary 6, we have the following

Corollary 7 *$L_\Pi(p, q)$: $C_1 \times C_2 \to \mathbb{R}$ is a convex function.*

Line segment s_2 in Step 4 of Procedure 2 is called *associated to* the updated (optimal) point p_2. Let $s_i \subseteq C_i$ be the line segment associated to the final updated point $p_i \in C_i$ in Algorithm 1, where $i = 0, 1, 2, \ldots, k-1$. Analogous to Theorem 5, we have Theorem 3. Regarding the proof of Theorem 3, at first we define that Algorithm 2 is also called an *arc version* of an RBA. If we modify Algorithm 2 such that p and q are not specified by finding a shortest closed path which passes through line segments $\langle S_1, S_2, \ldots, S_k, S_1 \rangle$ in order, then we obtain a *curve version* of Algorithm 2 (which also allows to calculate a loop). Basically, following the same way as demonstrated with the proof of Theorem 5, we can prove that the curve version outputs a closed $\{1 + 4k \cdot [r(\varepsilon) + \varepsilon_2]/L\}$-approximation path. Thus, Algorithm 1 defines a closed $\{1+4k\cdot[r(\varepsilon)+\varepsilon_2]/L\}$-approximation path to the step set $\langle S_0, S_1, \ldots, S_{k-1}, S_0 \rangle$; we skip the proof due to given similarities.

References

[1] T. Bülow and R. Klette. Rubber band algorithm for estimating the length of digitized space-curves. In Proc. *Intern. Conf. Pattern Recognition*, volume 3, pages 551–555, 2000,

[2] T. Bülow and R. Klette. Approximation of 3D shortest polygons in simple cube curves. In Proc. *Digital and Image Geometry*, LNCS 2243, pages 281–294, Springer, Berlin, 2001.

[3] T. Bülow and R. Klette. Digital curves in 3D space and a linear-time length estimation algorithm. *IEEE Trans. Pattern Analysis Machine Intelligence*, **24**:962–970, 2002.

[4] J. Choi, J. Sellen, and C.-K. Yap. Precision-sensitive Euclidean shortest path in 3-space. In Proc. *Ann. ACM Symp. Computational Geometry*, pages 350–359, 1995.

[5] R. Klette and T. Bülow. Critical edges in simple cube-curves. In Proc. *Discrete Geometry Computational Imaging*, LNCS 1953, pages 467–478, Springer, Berlin, 2000.

[6] F. Li and R. Klette. Rubberband algorithms for solving various 2D or 3D shortest path problems. Invited talk, in IEEE Proc. *Computing: Theory and Applications*, The Indian Statistical Institute, Kolkata, pages 9 - 18, 2007.

[7] M. Sharir and A. Schorr. On shortest paths in polyhedral spaces. *SIAM J. Comput.*, **15**:193–215, 1986.

[8] C.-K. Yap. Towards exact geometric computation. *Computational Geometry: Theory Applications.*, **7**:3–23, 1997.