International Journal of Computer Mathematics
Vol. 83, No. 3, March 2006, 357–370

An approximation algorithm for d_1-optimal motion of a rod robot with fixed rotations

MOHAMMAD ALI ABAM†‡ and MOHAMMAD GHODSI*†§¶

†Computer Engineering Department, Sharif University of Technology, Tehran, Iran
‡Computing Science Department, TU Eindhoven, PO Box 513, 5600 MB Eindhoven, The Netherlands
§School of Computer Science, Institute for Studies in Theoretical Physics and Mathematics (IPM), Tehran, Iran

(Received November 2004; reviewed 17 January 2006; revised 21 May 2006; accepted 1 June 2006)

Given a translating and rotating rod robot in a plane in the presence of polygonal obstacles with the initial and final placements of the rod known, the d_1-optimal motion planning problem is defined as finding a collision-free motion of the rod such that the orbit length of a fixed but arbitrary point F on the rod is minimized. In this paper we study a special case of this problem in which the rod can translate freely, but can only rotate by some pre-specified given angles around F. We first characterize the d_1-optimal motion of the robot under the given conditions and then present a $(1+\epsilon)$–approximation algorithm for finding the optimal path. The running time of the algorithm is bounded by a polynomial in terms of some parameters related to the problem input.

Keywords: Approximation algorithms; Robot motion planning; Rod robot; d_1-optimal motion

AMS Classification: 68U01; 68T20

1. Introduction

The problem of moving a robot in the presence of obstacles has received much attention. Most early published results in this area were concerned with the feasibility of motion planning. For example, comprehensive mathematical and algorithmic analyses of the general motion planning were given in [1–5] . Some authors have proposed efficient algorithms for the simple case of a rod moving in the polygonal space [6–8]. However, only a few researchers have algorithmically studied the optimal motion of a rotating robot. One major reason is that there is no well-defined objective function to measure the motion in general except for the simple planar case where the robot is a rod (directed line segment).

Let F be a fixed point on the rod. The curve traced by F in any continuous motion m of the rod is called the *orbit* of F. Several objective functions for measuring the motion of the rod are available. A commonly used function, the d_n function, is the average orbit length of n

*Corresponding author. Email: ghodsi@sharif.edu
¶This author's work was partially supported by a grant from IPM school of CS (No. CS1384-2-01).

($n \geq 1$) evenly distributed points on the rod. In particular, the d_1 function is the orbit length of one fixed arbitrary point on the rod. The aim of the d_n-optimal motion problem is to find the motion m with the least $d_n(m)$ of all collision-free motions between the given initial and final placements of the rod.

The d_2-optimal motion problem is to minimize the average lengths of the orbits of the two endpoints of the rod. If there are no obstacles, this problem is known as Ulam's problem. It was solved by Icking *et al.* [9] who proved that there is an optimal motion between the initial and final placements consisting of either at most three rotations, at most two rotations, and one straight-line motion, or one rotation between two straight-line motions. However, this problem has not been solved for the case when obstacles are present. O'Rourke [10] obtained a polynomial-time algorithm for the d_∞-optimal motion problem restricted to pure translation and rotation by $\pm 90°$.

The d_1-optimal motion problem has been proved to be NP-hard [11, 12]. Papadimitriou and Silverberg [13] studied the d_1-optimal problem where F is one endpoint of the rod with its motion restricted to straight lines between obstacle vertices. They obtained an $O(n^4 \lg n)$ time algorithm for this problem where n is the total number of obstacle edges. Sharir [14] improved this algorithm and obtained an $O(n^3 \alpha(n) \lg^2 n)$ time algorithm where $\alpha(n)$ is an extremely slowly growing function of Ackermann's inverse. Two approximation algorithms for the d_1-optimal motion problem were presented in [11, 15], some details of which are presented in section 4, but none of them is a $(1 + \epsilon)$–approximation algorithm. The existence of such an algorithm is unknown.

In this paper, we study a restricted case of the d_1-optimal motion problem and present a $(1 + \epsilon)$–approximation algorithm for this special case. The restriction is that the directed rod can translate and can only be positioned in one of the k allowable orientations in $\{\vec{\alpha}_1, \ldots, \vec{\alpha}_k\}$. More precisely, the rod can change its orientation from $\vec{\alpha}_i$ to $\vec{\alpha}_j$ by rotating around F through the smaller angle of $\vec{\alpha}_i$ and $\vec{\alpha}_j$. This implies that the rod can rotate clockwise or counter-clockwise, but it must sweep through the smaller angle to the destination orientation. We also assume that our workspace, which is a two-dimensional space, includes a set of disjoint convex polygonal obstacles.

Our result includes a characterization of the optimal motion under these constraints, and a $(1 + \epsilon)$–approximation algorithm for finding this motion. Our algorithm guarantees finding a motion m between any given initial and final placements of the rod satisfying $d_1(m) \leq (1 + \epsilon) d_1(m^*)$, where m^* is the optimal motion. The running time of our algorithm is bounded by a polynomial in n, k, L, and $1/\epsilon$, where n is the total number of obstacle edges, k is the number of allowable rotations, L is a bound on the number of bits for each input integer, and ϵ is an arbitrary positive number.

The paper is organized as follows. In section 2, we present the basic definitions and some known relevant results. Section 3 describes the structure of the optimal motion. Our proposed approximation algorithm is presented in section 4. Finally, section 5 contains the concluding remarks.

2. Notation and basic definitions

In general, the placement of a robot moving in a workspace is specified by a number of parameters. For example, the placement of a planar robot $\Re$ which can only translate can be specified by a pair (x, y) that are coordinates of one fixed point, known as the *reference point*, on the robot. This location is denoted $\Re(x, y)$. The parameter space of $\Re$, which is usually called its *configuration space*, is denoted by $C(\Re)$. A point p in $C(\Re)$ corresponds to a particular placement of $\Re$ in the workspace. For a translating and rotating robot, $C(\Re)$ is a three-dimensional $\mathbb{R}^2 * [0, 2\pi)$ space. A point (x, y, α) in this space corresponds to the placement $\Re(x, y, \alpha)$ in the workspace where x and y are coordinates of the reference point of $\Re$ and α specifies its orientation.

The points in $C(\Re)$ corresponding to the placements where the robot intersects one of the obstacles in the workspace are not permitted to be parts of any motion. The set of these points is called the robot's *forbidden space* and is denoted by $C_{forb}(\Re)$. The rest of $C(\Re)$ is called the robot's *free space* and is denoted by $C_{free}(\Re)$. A path of $\Re$ is a curve in the configuration space, and every placement along that path maps to its corresponding point in $C(\Re)$. Obviously, a collision-free path maps to a curve in the free space. We can map an obstacle O to a set of points p in $C(\Re)$ such that $\Re(p)$ intersects O. The resulting set is called the *forbidden space of obstacle O* and is denoted by $C_{obstacle}(O)$. $C_{forb}(\Re)$ is the union of $C_{obstacle}(O)$ for all obstacles.

The *Minkowski sum* of the two sets $S_1, S_2 \in \mathbb{R}^2$, denoted by $S_1 \oplus S_2$, is defined as

$$S_1 \oplus S_2 = \{p + q : p \in S_1, q \in S_2\}$$

where $p + q$ is the vector sum of two vectors p and q [16]. The following lemma is used to express $C_{obstacle}(O)$ in terms of Minkowski sum. For simplicity, we denote the reflection of a point, arc, or shape x about the origin by $\underline{x}$.

LEMMA 1 [16] *Let $\Re$ be a planar translating robot and let O be a planar obstacle. Then $C_{obstacle}(O)$ is $O \oplus \underline{\Re}(0, 0)$.*

In the remainder, let $\Re$ be a rod robot (a directed unit-sized vector $\overrightarrow{AB}$) moving around a two-dimensional environment workspace which consists of a set $\mathcal{O} = \{\mathcal{O}_1, \dots, \mathcal{O}_\ell\}$ of disjoint convex polygonal obstacles. The number of edges of $\mathcal{O}_i$, or its size, is assumed to be n_i, and the problem size n is defined to be $n = \sum_{i=1}^{\ell} n_i$. F is an arbitrary fixed point on $\Re$ as its reference point. The orientation of the rod must be one of the allowable orientations in $\{\vec{\alpha_1}, \dots, \vec{\alpha_k}\}$ (assuming that $\alpha_i \in [0, 2\pi)$ and $\alpha_1 < \alpha_2 < \cdots < \alpha_k$ where α_i is the angle of $\vec{\alpha_i}$ from the x axis) and the rod can change its orientation by rotating around F by sweeping through the corresponding smaller angle.

For any $Z = (p, \alpha)$, where p is a point in the plane and α is a real number, and any point q in the plane, we define $q[Z]$ to be a transformation by rotating q through the angle α around the origin followed by a translation under $\vec{p}$. Every motion m is specified by a function $m : [0, 1] \to C(\Re)$. For any point X on $\Re$, let $X_m : [0, 1] \to \mathbb{R}^2$, called the orbit of X under m, denote the function $X_m(x) = X[m(x)]$. In other words, $X[m(x)]$ denotes the curve traced by point X in the workspace under the motion m. More precisely, we define the motion m as follows.

A motion m of the rod from a placement $S = (s, \theta_s)$ to another location $T = (t, \theta_t)$ ($s, t \in \mathbb{R}^2$ and $\theta_s, \theta_t \in \{\alpha_1, \dots, \alpha_k\}$) is a function of the form $m : [0, 1] \to C(\Re)$, where $m(0) = S$ and $m(1) = T$, and F_m, the orbit of F under m, is a continuous and rectifiable function. We denote this motion by $S \overset{m}{\to} T$. For simplicity, we use $\overline{pq}$ to denote the line segment whose endpoints are two points p and q. The length of $S \overset{m}{\to} T$ is defined as follows:

$$d_1(m) = sup_{(a_0=0<a_1<\cdots<a_j=1)} \sum_{i=0}^{j-1} |\overline{F[m(s_i)]F[m(s_{i+1})]}|$$

where the supremum is taken over all finite subdivisions $a_0 < a_1 < \cdots < a_j$ of the interval $[0, 1]$ and $|\cdot|$ denotes the Euclidean length. We define the d_1 function between any two placements S and T to be $d_1(S, T) = inf_{S \overset{m}{\to} T} d_1(m)$ where the infimum is taken over all collision-free motions m between S and T. If m is a motion from S to T and $d_1(S,T) = d_1(m)$, we say that m is a d_1-*optimal motion*.

A glossary of the main notations is presented in the Appendix.

3. Structure of the optimal motion

The structure of the forbidden spaces is important for precise specification of the optimal motion in our approximation algorithm. Therefore we first give a detailed specification of the forbidden spaces and provide algorithms to compute them, and then present the structure of the optimal motion.

3.1 *Forbidden space*

The configuration space ($C(\Re) = \mathbb{R}^2 * \{\alpha_1, \ldots, \alpha_k\}$) is a three-dimensional space consisting of k planes $\{\mathcal{S}_1, \ldots, \mathcal{S}_k\}$, where $\mathcal{S}_i$ is the set of points corresponding to all placements where the orientation of the rod is $\vec{\alpha}_i$. Then $\mathcal{S}_i = \{(x, y, \alpha) | \alpha = \alpha_i\}$. In fact, we can assume there are k workspaces with the same obstacles and the rod has a specified orientation in each space. Figure 1 illustrates two workspaces $\mathcal{S}_i$ and $\mathcal{S}_j$. For simplicity, we assume that $\mathcal{S}_i = \{(x, y) | x, y \in \mathbb{R}\}$. We also use $\Re_\alpha(x, y)$ to represent $\Re(x, y, \alpha)$ and $\Re_\alpha$ to denote the rod with orientation $\vec{\alpha}$. In remainder, $\mathcal{S}_i$ is called the plane rather than the workspace provided that no confusion arises.

The rod performs a translating motion when it moves around in each plane and performs a rotating motion when it moves from one plane to another. The rod must not intersect any obstacle during rotation or translation. Therefore, there are two kinds of forbidden spaces in each plane.

1. *Translating forbidden space*, denoted by TFS_i, is defined as

$$TFS_i = \{(x, y) \in \mathcal{S}_i : \Re_{\alpha_i}(x, y) \cap (\cup_{h=1}^{\ell} \mathcal{O}_h) \neq \emptyset\}.$$

The boundary of TFS_i is denoted by $BTFS_i$.

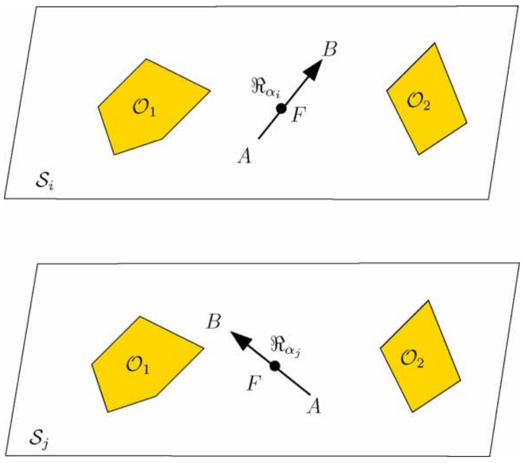

Figure 1. Two distinct orientations of the rod.

2. *Rotating forbidden space*, denoted by RFS_i^j $(i \neq j)$, is defined as

$$RFS_i^j = \{(x, y) \in \mathcal{S}_i : \exists\alpha \in I_{ij}, \Re_\alpha(x, y) \cap (\cup_{h=1}^{\ell} \mathcal{O}_h) \neq \emptyset\}$$

where I_{ij} is the interval [minimum(α_i, α_j), maximum(α_i, α_j)], if $|\alpha_i - \alpha_j| \leq \pi$. Otherwise, it is the interval [maximum(α_i, α_j), minimum$(\alpha_i, \alpha_j) + 2\pi$]. The boundary of RFS_i^j is denoted by $BRFS_i^j$.

Thus there is one translating forbidden space and $k - 1$ rotating forbidden spaces. By lemma 1, $TFS_i = \underline{\Re}_{\alpha_i}(0, 0) \oplus (\cup_{h=1}^{\ell} \mathcal{O}_h)$.

LEMMA 2 *The complexity of $BTFS_i$ is $O(n)$ and it can be computed in $O(n \log^2 n)$ time.*

Proof Since the complexity of $\Re$ is 2 (constant) and all obstacles are convex, the lemma is a simple conclusion from lemma 13.13 of [16]. ■

We now concentrate on computing $BRFS_i^j$. To move from plane $\mathcal{S}_i$ to plane $\mathcal{S}_j$, the rod rotates around F from angle α_i to angle α_j by sweeping through the smaller angle. During this rotation, the rod sweeps the area illustrated in figure 2(a). This area is denoted by sector$_{ij}$ and consists of two sectors: a lower sector with radius $|\overline{AF}|$ denoted by lsector$_{ij}$, and an upper sector with radius $|\overline{BF}|$ denoted by usector$_{ij}$. Thus we have

$$RFS_i^j = \underline{\text{sector}}_{ij} \oplus (\cup_{h=1}^{\ell} \mathcal{O}_h) = (\underline{\text{usector}}_{ij} \oplus \cup_{h=1}^{\ell} \mathcal{O}_h) \cup (\underline{\text{lsector}}_{ij} \oplus \cup_{h=1}^{\ell} \mathcal{O}_h).$$

Since the rod must sweep through the smaller angle to the destination orientation, usector$_{ij}$ and lsector$_{ij}$ are convex. The following algorithm describes how we can compute $\underline{\text{usector}}_{ij} \oplus \mathcal{O}_h$.

1. Find the reflection of usector$_{ij}$ about the origin. Let $\underline{B}_i$ and $\underline{B}_j$ be the reflections of B_i and B_j about the origin, respectively.
2. Compute the intersections of arc $\overset{\frown}{\underline{B}_i\underline{B}_j}$ with the half-lines originating from the origin and lying parallel to the outer normals of the edges of $\mathcal{O}_h$. Let $C_1, \ldots, C_r$ denote the intersection points (see figure 2(b)). Since each half-line intersects arc $\overset{\frown}{\underline{B}_i\underline{B}_j}$ at most once, $r \leq n_h$. The resulting polygon $C = F\underline{B}_iC_1 \ldots C_r\underline{B}_j$ is convex.
3. Compute $\mathcal{O}_h \oplus C$ using the algorithm given in [16].
4. Replace the edges of $\mathcal{O}_h \oplus C$ corresponding (parallel and equal) to $\overline{\underline{B}_iC_1}, \overline{C_1C_2}, \ldots, \overline{C_r\underline{B}_j}$ with the respective arcs of $\overset{\frown}{\underline{B}_iC_1}, \overset{\frown}{C_1C_2}, \ldots, \overset{\frown}{C_r\underline{B}_j}$.

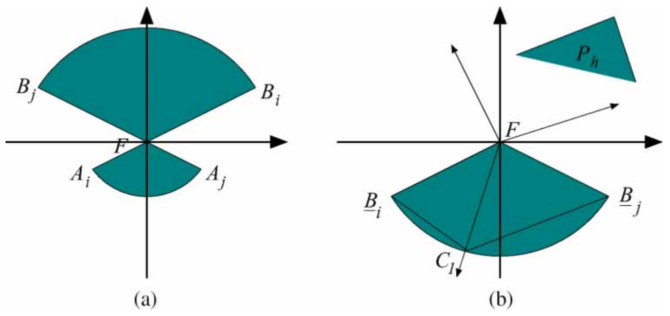

Figure 2. (a) The area swept by the rod in a rotation. (b) The Minkowski sum of a convex polygon and a part of a disc.

LEMMA 3 *The above algorithm correctly computes* $\underline{\text{usector}}_{ij} \oplus \mathcal{O}_h$. *Its time complexity and the complexity of* $\underline{\text{usector}}_{ij} \oplus \mathcal{O}_h$ *are both* $O(n_h)$.

Proof It is easy to see that step 1 of the algorithm correctly computes the reflection of usector$_{ij}$ about the origin. Let $v_1, \ldots, v_{n_h}$ be the vertices of $\mathcal{O}_h$ and let $\ell_1, \ldots, \ell_{n_h}$ be the half-lines originating from the origin and lying parallel to $\vec{d}_1, \ldots, \vec{d}_{n_h}$, the outer normals of the edges of $\mathcal{O}_h$. Since the order of the outer normals is the same as the order of edges in $\mathcal{O}_h$, some consecutive half-lines intersect $\overset{\frown}{\underline{B}_i\underline{B}_j}$. Without loss of generality, assume that $\ell_1, \ldots, \ell_r$ intersect $\overset{\frown}{\underline{B}_i\underline{B}_j}$ at $C_1, \ldots, C_r$, respectively. It is known that the extreme point on the Minkowski sum in direction $\vec{d}$ is the sum of the extreme points on the operands of the Minkowski sum in the same direction. If the half-line originating from the origin and lying parallel to $\vec{d}$ does not intersect $\overset{\frown}{\underline{B}_i\underline{B}_j}$, the extreme point on $\underline{\text{usector}}_{ij}$ in direction $\vec{d}$, which is F, $\underline{B}_i$, or $\underline{B}_j$, is the extreme point on C in the same direction. Thus the sum of the extreme points on $\mathcal{O}_h$ and $\underline{\text{usector}}_{ij}$ in direction $\vec{d}$ is equal to the sum of the extreme points on $\mathcal{O}_h$ and C in the same direction which are correctly computed in step 3 of the algorithm. Now let the half-line originating from the origin and lying parallel to $\vec{d}$ intersect $\overset{\frown}{\underline{B}_i\underline{B}_j}$ at X. Without loss of generality, assume that X is between C_1 and C_2, which implies that the extreme point on $\mathcal{O}_h$ in direction $\vec{d}$ must be v_2. Then the sum of the extreme points in direction $\vec{d}$ is $X \oplus v_2$. When the direction $\vec{d}$, which is initially equal to $\vec{d}_1$, moves toward $\vec{d}_2$, the extreme point on $\mathcal{O}_h$ does not change and still remains v_2 and the extreme point X on usector$_{ij}$ moves from C_1 towards C_2 on $\overset{\frown}{C_1C_2}$. This means that we must replace the edge corresponding to $\overline{C_1C_2}$ in $\mathcal{O}_h \oplus C$ with the arc that is equal to $\overset{\frown}{C_1C_2}$ (step 4 of the algorithm). Since the complexity of C is $O(n_h)$, the complexity of $\mathcal{O}_h \oplus C$ is $O(n_h)$. Therefore the complexity of $\mathcal{O}_h \oplus \underline{\text{usector}}_{ij}$ is $O(n_h)$. It is straightforward to see that step 1 of the algorithm requires $O(1)$ time and steps 2 and 4 require $O(n_h)$ time. By the properties of the Minkowski sum, the step 3 requires $O(n_h)$ time. Thus the time complexity of the algorithm is $O(n_h)$. ■

The same algorithm can be used to compute $\mathcal{O}_h \oplus \underline{\text{lsector}}_{ij}$. To compute their union for all h, we can use a simple divide and conquer approach. We thus obtain the following theorem.

LEMMA 4 *The complexity of* $BRFS_i^j$ *is* $O(n^2)$ *and it can be computed in time* $O(n^2 \log n)$.

Proof The number of edges of $\mathcal{O}_h \oplus \underline{\text{usector}}_{ij}$ and $\mathcal{O}_h \oplus \underline{\text{lsector}}_{ij}$ are both $O(n_h)$, for each h and $\sum_{h=1}^{\ell} n_h = n$. In the worst case, if each edge is intersected by all other edges, it is partitioned into $O(n)$ edges. Therefore the total number of edges of $BRFS_i^j$ is $O(n^2)$. Let $T(n)$ be the time needed by the algorithm. Since the merge step requires $O(n^2 \log n)$ (using the overlay algorithm presented in section 2.3 of [16]), we have $T(n) = 2T(n/2) + O(n^2 \log n)$ and thus $T(n) = O(n^2 \log n)$. ■

Each edge of $BTFS_i$ is a line segment but each edge of $BRFS_i^j$ can be either a line or an arc segment. An arc edge can be contained in a circle of either radius $|\overline{FA}|$ or radius $|\overline{FB}|$. Other simple and useful properties are as follows:

1. Since $\underline{\Re}_{\alpha_i}(0,0) \oplus \mathcal{O}_h \subset \underline{\text{sector}}_{ij} \oplus \mathcal{O}_h$, we have TFS$_i \subset$ RFS$_i^j$, i.e. the translating free space is a subset of the rotating free space.
2. Since sector$_{ij}$ = sector$_{ji}$, we have $BRFS_i^j = BRFS_j^i$.

3.2 *Characterization of the optimal motion*

The following theorem summarizes the characterization of optimal motion m between two placements $S = (s, \theta_s)$ and $T = (t, \theta_t)$.

THEOREM 1 *Any optimal motion m can be transformed into a motion m' such that $F_{m'}$ is a polygonal path whose vertices, except those around which the rod rotates, are the starting point s, the target point t, and the vertices of $BTFS_i$ $(i = 1, \ldots, k)$. The vertices that the rod rotates around are s and those on $BRFS_i^j$ $(i, j = 1, \ldots, k)$.*

Proof Since m is optimal, each submotion of m is also locally optimal. Therefore the submotion of m contained in $\mathcal{S}_i$ is a polygonal path whose inner vertices are, by lemma 15.1 of [16], the vertices of $BTFS_i$. Using this, we can say that F_m consists of a finite sequence $v_0 v_1 \cdots v_q$ where $v_h v_{h+1}$ is a line segment and $v_0 = s$, $v_q = t$. Let v_r be the vertex in which the rod rotates from α_i to α_j. If v_r does not belong to $BRFS_i^j$ and s, then $v_{r-1}v_r$ is parallel to $v_r v_{r+1}$. Otherwise, as illustrated in figure 3, we can find a shorter path. This is because there is a disc centred at v_r avoiding RFS_i^j and RFS_j^i $(RFS_i^j = RFS_j^i)$, and the submotion of m inside the disc can be replaced by translating along the line segment GH and rotating around G. Obviously, this path is shorter. Therefore, since $v_{r-1}v_r$ and $v_r v_{r+1}$ are parallel, the rotation around v_r can be performed later or earlier at either $BRFS_i^j$ or s. ■

The following theorem describes the vertices around which the rod rotates.

THEOREM 2 *If v_r is a vertex at which the rod changes its orientation from $\vec{\alpha}_i$ to $\vec{\alpha}_j$, one of the following holds.*

1. *v_r is a vertex of $BRFS_i^j$.*
2. *v_{r-1}, v_r and v_{r+1} are collinear.*
3. *$v_r v_{r+1}$ is the reflection of $v_{r-1}v_r$ with respect to the normal at vertex v_r of one edge of $BRFS_i^j$.*

Proof Let C, A, B be a curve and two distinct points in the plane, respectively. Let X be a point on C such that $|\overline{AX}| + |\overline{BX}|$ is minimum. If C intersects the segment AB in point D, it is obvious that $X = D$ (figure 4(a)). Otherwise, consider an ellipse whose foci are A, B and whose fixed value is $|\overline{AB}|$. We increase the fixed value of the ellipse until it touches the curve C. The touching point X is either one endpoint of C or an interior point of C. In the latter case, the ellipse and C are tangent at X, which means that the tangent line to the ellipse at X is tangent to C. As shown in figure 4(b), we know that $\angle AXF = \angle BXE$, i.e. BX is the reflection of AX with respect to the normal of C at X. Assume that v_r is on edge e of

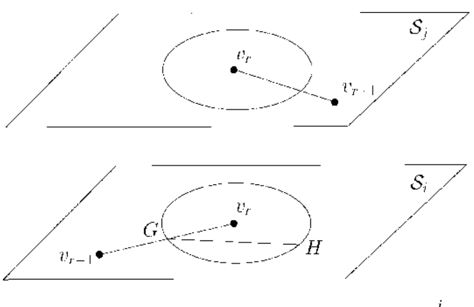

Figure 3. Performing rotations on s and $BRFS_i^j$.

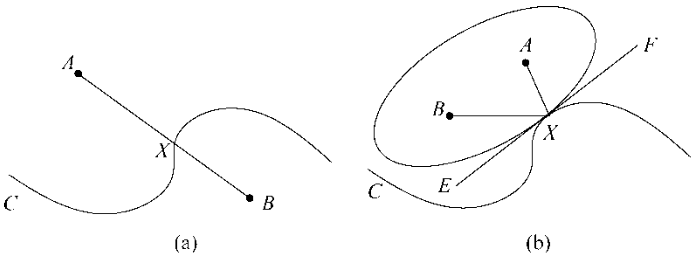

Figure 4. Shortest path between A and B passing through curve C.

$BRFS_i^j$, $v_{r-1}v_r$ avoids TFS_i, and v_rv_{r+1} avoids TFS_j. Also, assume that e is a curve. Since $|\overline{v_{r-1}v_r}| + |\overline{v_rv_{r+1}}|$ is a minimum, using the above characterization, one of the three features of the theorem must hold. ■

4. The approximation algorithm

The d_1-optimal motion problem is known to be NP-hard [11, 12]. The environment used in the reduction of a 4CNF-satisfiability problem to the decision version of the d_1-optimal motion problem consists of the assembly of certain prefabricated modules. There is one basic module (figure 5) from which other modules are fabricated. It is easy to see that the rod can lie in two particular orientations in this module. Therefore we can conclude that our problem is also NP-hard.

The output motion m of the approximation algorithm presented in [11] satisfies the following condition:

$$d_1(m) \leq (1+\epsilon)d_1(m^*) + O(n^2\epsilon')$$

where ϵ, ϵ' are two positive numbers and m^* is the optimal motion. The running time of the algorithm is

$$O\left(n^4\alpha(n)\frac{L-\lg\epsilon'}{\lg(1+\epsilon L^{-L})}\right).$$

This algorithm was improved in [15] by introducing a pseudo ϵ–approximation algorithm and converting it into an ϵ–approximation algorithm. The pseudo ϵ–approximation algorithm satisfies

$$d_1(m) \leq d_1(m_R^*) + \epsilon R$$

where $R > 0$ is an arbitrary number and m_R^* is the d_1-optimal when the orbit of F is restricted to a disc of radius R centred at the starting point s. If R is large enough, $m_R^* = m^*$.

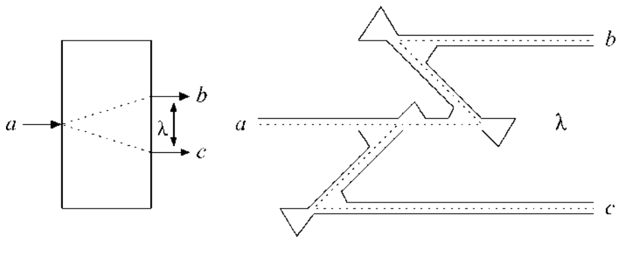

Figure 5. Schematic and detail description of a wide-beam splitter module.

As in [11] and [17], our algorithm breaks up edges of $BRFS_i^j$ into smaller segments. A *monotone edge e* is an edge whose endpoints are the points closest to and farthest from the starting point s. Each line edge and arc edge can be partitioned into at most three monotone edges as follows.

- Let $e = \overline{pq}$ be a line edge and let x be the closest point on e to s. If $x = p$ or $x = q$, then e is already a monotone edge. Otherwise, we partition e into two monotone edges $\overline{px}$ and $\overline{xq}$.
- Let $e = \widehat{pq}$ be an arc edge. The line passing through s and the centre of a circle containing e intersects e in at most two points which partition e into at most three arcs. It is easy to see that the three arcs are monotone.

Observation 1 Let e be a monotone edge and let x be the closest point on e to s (x is one of the endpoints of e). If a point y initially lying on x moves towards the other endpoint of e, $|sy|$ increases strictly. Moreover, if e is a line edge, then $|sy| \geq |xy|$. Otherwise, $|sy| \geq 1/2|xy|$.

Observation 2 If $E = \widehat{pq}$ is a monotone edge, then $|\overline{pq}| \geq 2/\pi |\ \widehat{pq}\ |$.

Observation 3 After making all edges of $BRFS_i^j$ monotone, the asymptotic complexity of $BRFS_i^j$ does not change and still remains $O(n^2)$.

Let e be an edge of $BRFS_i^j$ for some i and j. Let cap(e) be the area surrounded by e and the chord connecting the two endpoints of e. Edge e is called a *visible* edge if cap(e) and TFS_i are interiorly disjoint. If e is a visible edge and p and q are two points on e, the rod whose reference point lies on p can move towards q with minimum cost just by translating along $\overline{pq}$. If e is a line edge, clearly e is a visible edge. However, if e is an arc edge, cap(e) and TFS_i are not necessarily interiorly disjoint. In the following, we explain how e can be partitioned into arc edges $e_1, \ldots, e_\ell$ such that cap(e_h) and TFS_i are interiorly disjoint for any $h = 1, \ldots, \ell$.

Let u and v be the endpoints of e where e is in the left side of $\overrightarrow{uv}$ (figure 6). Find vertices of $BTFS_i$ lying inside cap(e), among these vertices find vertex w such that the angle of $\overrightarrow{uw}$ from the x-axis is minimum. Let u_1 be the other intersection of e and the line passing through u and w, the cap corresponding to arc $\widehat{uu_1}$ does not have interior intersection with TFS$_i$. We can now recursively perform these steps for arc $\widehat{u_1v}$. In figure 6, e is partitioned into three arc edges.

LEMMA 5 *After making all edges of $BRFS_i^j$ visible, the asymptotic complexity of $BRFS_i^j$ does not change and still remains $O(n^2)$.*

Proof Let V_e be a set of vertices of $BTFS_i$ lying inside cap(e) where e is an edge of $BRFS_i^j$. After making e visible, e is partitioned into at most $|V_e|$ visible edges. We know that if

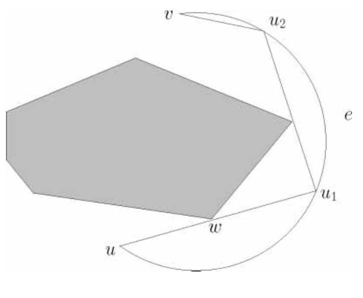

Figure 6. Partitioning the arc edge e into visible arc edges.

e and e' are two edges of $BRFS_i^j$, cap(e) and cap(e') are interiorly disjoint which means that $V_e \cap V_{e'} = \emptyset$. Since the complexity of $BTFS_i$ is $O(n)$, in total at most $O(n)$ new edges are created which does not change the $O(n^2)$ asymptotic complexity of $BRFS_i^j$. ■

The steps of the algorithm are described below. Throughout this algorithm, $D(\sigma, \sigma')$ denotes the shortest distance between two segments σ and σ' (each point is a segment of length 0).

1. Compute $BRFS_i^j$ and $BTFS_i$ for all $i, j = 1, \ldots, k$.
2. Make the edges of $BRFS_i^j$ monotone for all i and j.
3. Make the edges of $BRFS_i^j$ visible for all i and j.
4. Divide each edge e of $BRFS_i^j$ into a set of segments by introducing points on it, such as $b_0, b_1, b_2, \ldots$; each segment is then of the form $\sigma = [p, q]$ where p, q are consecutive points introduced on e. Let ϵ_1 be $\epsilon/8\mu$ where μ is the total number of edges of all $BRFS_i^j$, s, t, and the vertices of all $BTFS_i$.
   (a) Let b_0 be the closest point on e to the starting point s, i.e. $D(s, e) = |\overline{sb_0}|$. Notice that b_0 is one of the endpoints of e (figure 7).
   (b) Let b_i be at distance $D(s, e)(1 + \epsilon)^{i-1}$ $(i > 0)$ from b_0 on edge e. Notice that if e is an arc edge, $|\overset{\frown}{b_0 b_i}| = D(s, e)(1 + \epsilon)^{i-1}$.
   (c) The segment $\overline{b_0 b_1}$ is uniformly subdivided into segments of length $\epsilon_1 D(s, E)$. This further introduces points $c_1, c_2, \ldots$.

   If e is an arc, the arc $\overset{\frown}{b_i b_{i+1}}$ $(i > 0)$ is uniformly subdivided into arcs of length $1/\pi |\overset{\frown}{b_i b_{i+1}}|$.
5. Construct the visibility graph G as follows. The nodes of G are the segments from step 4 as well as vertices of $BTFS_i$ $(i = 1, \ldots, k)$, t in the target plane, and s in every plane $\mathcal{S}_j$ where $BRFS_h^j$ (let $\theta_s = \alpha_h$) does not contain s. Two segments σ and σ' contained in $\mathcal{S}_i$ are visible to each other when there exists $x \in \sigma$ and $x' \in \sigma'$ such that the line segment $\overline{xx'}$ avoids the interior of TFS_i. The edges of G are pairs (σ, σ') of nodes such that σ and σ' are in the same plane and visible to each other or are corresponding segments on $BRFS_i^j$ and $BRFS_j^i$ that are equal (notice that $BRFS_i^j = BRFS_j^i$). Notice that t, s, and the vertices of TFS_i are taken as segments of length zero.
6. For each edge (σ, σ'), compute and assign a nominal cost: If σ and σ' are in the same plane, this cost is the Euclidean distance between the midpoints of σ and σ'. Otherwise, the cost is zero.
7. Apply Dijkstra's shortest path algorithm to the weighted graph G. Let $\sigma_1, \ldots, \sigma_q$ be the nodes lying on the shortest path. The output of the algorithm is a sequence of points $u_1, w_1, \ldots, u_q, w_q$ where $u_j, w_j \in \sigma_j$, $u_1 = w_1 = s$, $u_q = w_q = t$, and w_j, u_{j+1} are visible to each other.

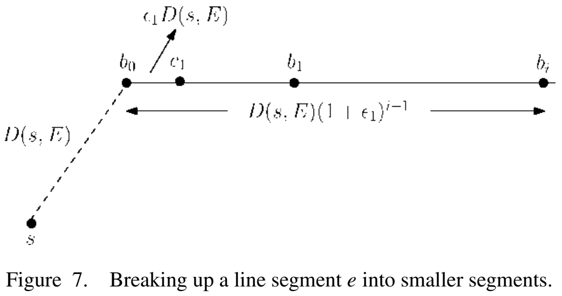

Figure 7. Breaking up a line segment e into smaller segments.

As mentioned above, the main aim of the algorithm is to break up the edges of $BRFS_i^j$ into small segments. The following lemmas shows that the strategy adopted to split the edges of $BRFS_i^j$ creates segments that have useful properties.

LEMMA 6

1. *If σ is a segment computed in step 4 of the algorithm, its length satisfies $|\sigma| \leq \epsilon_1 D(s, \sigma)$.*
2. *There are at most $O(L/\log(1+\epsilon_1))$ segments on each edge e and a total of $O(\mu L/\log(1+\epsilon_1))$ segments on all edges. Recall that μ is the total number of edges of all $BRFS_i^j$, point s, point t, and the vertices of all $BTFS_i$ and L is a bound on the number of bits for each input integer.*
3. *μ is bounded by $O(k^2n^2)$. Recall that k is the number of available orientations.*

Proof

1. Let σ belong to an edge e. If one endpoint of σ is c_i for some i, the length of σ is equal to $\epsilon_1 D(s, e)$. Also, since $D(s, e) \leq D(s, \sigma)$, we have
$$|\sigma| = \epsilon_1 D(s, e) \leq \epsilon_1 D(s, \sigma).$$
If e is a line edge and $\sigma = \overline{b_h b_{h+1}}$ $(h > 0)$, then we have
$$|\sigma| = b_{h+1} - b_h = D(s, e)(1+\epsilon_1)^h - D(s, e)(1+\epsilon_1)^{h-1} = \epsilon_1 D(s, e)(1+\epsilon_1)^{h-1}.$$
Since $D(s, \sigma) = |\overline{sb_h}| \geq |\overline{b_0 b_h}| = D(s, e)(1+\epsilon_1)^{h-1} = 1/\epsilon_1 |\sigma|$, then $|\sigma| \leq \epsilon_1 D(s, \sigma)$. If e is an arc edge, the arc $\overset{\frown}{b_h b_{h+1}}$ is divided into segments of length $1/\pi |\overset{\frown}{b_h b_{h+1}}|$. If σ is one of these segments, then we have
$$D(s, \sigma) \geq |\overline{sb_h}| \geq \frac{1}{2}|\overline{b_0 b_h}| \geq \frac{1}{\pi}|\overset{\frown}{b_0 b_h}| \geq \frac{1}{\pi} D(s, e)(1+\epsilon_1)^{h-1} = \frac{1}{\epsilon_1}|\sigma|.$$
We used observations 1 and 2 to prove some of the above inequalities.
2. We have assumed that the number of bits of each input integer is bounded by L. Then the maximum length of e is 2^L and the minimum distance between two different points is 2^{-L}. Consequently, we have
$$D(s, e)(1+\epsilon_1)^{h-1} \leq 2^L \Rightarrow h \leq \frac{L - \log D(s, e)}{\log(1+\epsilon_1)} + 1 \leq \frac{2L}{\log(1+\epsilon_1)} + 1.$$
The segment $b_0 b_1$ is divided into $1/\epsilon_1$ segments and the segment $b_i b_{i+1}$ $(i > 0)$ is divided into at most π segments. Therefore the number of segments on e is $1/\epsilon_1 + 2\pi L/\log(1+\epsilon_1) + \pi$. Since $\epsilon_1 \geq \ln(1+\epsilon_1)$, the number of segments on e is $O(L/\log(1+\epsilon_1))$. With μ edges, the total number of segments is $O(\mu L/\log(1+\epsilon_1))$.
3. There are k^2 rotating forbidden spaces with complexity of $O(n^2)$ (lemma 5) and we have k translating forbidden spaces with complexity of $O(n)$ (lemma 2). Consequently, the complexity of the forbidden spaces is $O(k^2n^2)$ in total, i.e. $\mu = O(k^2n^2)$. ■

The output motion m and the optimal motion m^* pass through each vertex of $BTFS_i$ $(i = 1, \ldots, k)$, s, and t at most once. Otherwise, Dijkstra's algorithm returns a shorter path and m^* is not an optimal path. Let e be an edge of $BRFS_i^j$ for some i and j. Although e may be partitioned into some segments and for each there is a corresponding node in graph G, e is visited by m and m^* at most once. Otherwise, let p and q be distinct points on e visited by motion m^* (or m). Since e is a visible edge, $\overline{pq}$ avoids TFS_i and we can always find a shorter path by translating along $\overline{pq}$.

LEMMA 7 *Every edge of $BRFS_i^j$, every vertex of $BTFS_i$, the starting point s, and the target point t are visited by the output motion m and the optimal motion m^* at most once.*

The following theorem states our main result.

THEOREM 3 *The output motion m of the algorithm between the initial placement $S = (s, \theta_s)$ and the final placement $T = (t, \theta_t)$ satisfies the inequality*

$$d_1(m) \leq (1 + \epsilon)d_1(m^*),$$

where m^* is the optimal motion. Moreover, the running time of the algorithm is bounded by $O(k^3L^2n^7/\lg^2(1 + \epsilon_1))$.

Proof Let m (m^*) pass through $\sigma_1, \ldots, \sigma_q$ ($\sigma'_1, \ldots, \sigma'_{q'}$). More precisely, the motion m (m^*) is a sequence $u_1, w_1, \ldots, u_q, w_q$ ($u'_1, w'_1, \ldots, u'_{q'}, w'_{q'}$) where $u_h, w_h \in \sigma_h$ ($u'_h, w'_h \in \sigma'_h$) and w_h and u_{h+1} (w'_h and u'_{h+1}) are visible to each other and $u_1 = w_1 = s$ ($u'_1 = w'_1 = s$) and $u_q = w_q = t$ ($u'_{q'} = w'_{q'} = t$). Let $m_1, \ldots, m_q$ ($m'_1, \ldots, m'_{q'}$) be the midpoints of $\sigma_1, \ldots, \sigma_q$ ($\sigma'_1, \ldots, \sigma'_{q'}$). The following inequality can be derived from steps 6 and 7 of the algorithm:

$$|\overline{m_1m_2}| + \cdots + |\overline{m_{q-1}m_q}| \leq |\overline{m'_1m'_2}| + \cdots + |\overline{m'_{q'-1}m'_{q'}}|.$$

It is easy to see that the following inequalities always hold:

$$d_1(m) \leq |\overline{m_1m_2}| + \cdots + |\overline{m_{q-1}m_q}| + 2(|\sigma_1| + \cdots + |\sigma_q|)$$
$$|\overline{m'_1m'_2}| + \cdots + |\overline{m'_{q'-1}m'_{q'}}| \leq d_1(m^*) + 2(|\sigma'_1| + \cdots + |\sigma'_{q'}|).$$

We also know from lemma 6 that

$$|\sigma| \leq \epsilon_1 D(s, \sigma) \leq \epsilon_1 d_1(m^*).$$

Consequently we have $d_1(m) \leq d_1(m^*) + 4(q + q')d_1(m^*)$. According to lemma 7, m and m^* pass through each edge of $BRFS_i^j$, each vertex of $BTFS_i$, s, and t at most once. This implies that q and q' must be at most μ. On the other hand, we know that $\epsilon_1 = \epsilon/8\mu$. Therefore $d_1(m) \leq (1 + \epsilon)d_1(m^*)$.

Computing $BTFS_i$ ($i = 1, \ldots, k$) costs $O(kn \log^2 n)$ time (lemma 2) and computing $BRFS_i^j$ ($i, j = 1, \ldots, k$) costs $O(k^2n^2 \log n)$ time (lemma 4). Steps 2 and 3 can be done with time complexities of $O(k^2n^2)$ and $O(k^2n^3)$, respectively. The required time to determine whether σ_1 and σ_2 are visible to each other is $O(n^3)$. Since there are $O(\mu L/k \log(1 + \epsilon_1))$ segments in each plane, and $\mu = O(k^2n^2)$ and there are k planes, the required time for constructing the visibility graph is $O(k^3L^2n^7/\log^2(1 + \epsilon_1))$. Finding the shortest path can be done in

$$O\left(\frac{\mu L}{\log(1 + \epsilon_1)} \log\left(\frac{\mu L}{\log(1 + \epsilon_1)}\right) + k\frac{\mu^2L^2}{k^2\log^2(1 + \epsilon_1)}\right)$$

time by Dijkstra's algorithm. In total, the running time of the algorithm is $O(k^3L^2n^7/\log^2(1 + \epsilon_1))$. ■

5. Conclusion

We have considered the problem of characterizing and computing the optimal motions for a translating and rotating rod robot which can only rotate under a pre-specified fixed number of angles and around a fixed but arbitrary point on the rod. The latter condition is a new restriction for the known d_1-optimal problem. We characterized the d_1-optimal motion of the robot under the given conditions, and then presented a $(1 + \epsilon)$–approximation algorithm for finding the optimal motion of the rod. The running time of the algorithm was shown to be bounded by a polynomial in terms of problem parameters.

A. Appendix: main notations

$\overrightarrow{AB}$	Rod robot
$\{\vec{\alpha_1}, \ldots, \vec{\alpha_k}\}$ and $\{\alpha_1, \ldots, \alpha_k\}$	Specified orientations and corresponding angles
$\overset{\frown}{B_i B_j}$	Arc (a part of a disc) between B_i and B_j
$BTFS_i$	Boundary of TFS_i
$C(\Re)$	Configuration space of $\Re$
$C_{forb}(\Re)$	Forbidden space of $\Re$
$C_{free}(\Re)$	Free space of $\Re$
$C_{obstacle}(\mathcal{O}_i)$	Forbidden space of obstacle $\mathcal{O}_i$
ϵ	Arbitrary positive number
F	Fixed point on $\Re$
L	Bound on the number of bits for each integer
m	Motion
$n = \sum_{i=1}^{\ell} n_i$	Total number of edges of all obstacles
n_i	Number of edges of $\mathcal{O}_i$
$\mathcal{O} = \{\mathcal{O}_1,\ dots, \mathcal{O}_\ell\}$	Set of obstacles
$q[(p, \alpha)]$	New point reached by rotating point q under α degrees around the origin and translating under $\vec{p}$
$\mathbb{R}$	Real numbers
$\Re$	Robot
$\Re_{\alpha_i}(x, y)$	Rod with its fixed point at point (x, y) and orientation α_i
$\mathbb{R}^2 * \{\alpha_1, \ldots, \alpha_k\}$	$C(\Re)$ for a rotating and translating $\Re$ with fixed rotations
RFS_i^j	Rotating forbidden space which is the forbidden space in $\mathcal{S}_i$ when rotating to α_j
$\mathcal{S}_i$	Set of all points (x, y) such that $(x, y, \alpha_i) \in C(\Re)$
$S = (s, \theta_s)$	Initial placement of the robot
TFS_i	Translating forbidden space in $\mathcal{S}_i$
$T = (t, \theta_t)$	Final placement of the robot
X_m	Orbit of the point X on $\Re$ under motion m
$\underline{x}$	Reflections of a point, arc, or a shape x about the origin

Acknowledgements

The authors wish to thank the anonymous referees for many useful comments.

References

[1] Reif, J., 1979, Complexity of the movers problem and generalizations. In *Proceedings of the 20th IEEE Symposium on Foundations of Computer Science*, pp. 420–427.

[2] Schwartz, J.T. and Sharir, M., 1983, On the piano movers' problem. I: The case of a rigid polygonal body moving amidst polygonal barriers. *Communications on Pure and Applied Mathematics*, **36**, 345–398.

[3] Schwartz, J.T. and Sharir, M., 1983, On the piano movers' problem. II: General techniques for computing topological properties of real algebraic manifolds. *Advances in Applied Mathematics*, **4**, 298–351.

[4] Schwartz, J.T. and Sharir, M., 1983, On the piano movers' problem. III: Coordinating the motion of several independent bodies: the special case of circular bodies moving amidst polygonal barriers. *Robotics Research*, **2**, 46–75.

[5] Schwartz, J.T. and Sharir, M., 1985, On the piano movers' problem. IV: Efficient motion planning algorithms in environments of bounded local complexity. *Technical Report* 164, Courant Institute, New York University.

[6] Leven, D. and Sharir, M., 1985, An efficient and simple motion planning algorithm for a ladder moving in two-dimensional space amidst polygonal barriers. In *Proceedings of the 1st Annual ACM Symposium on Computational Geometry*, pp. 221–227.

[7] Schwartz, J.T. and Sharir, M., 1984, On the piano movers' problem. V: The case of a rod moving in three-dimensional space amidst polyhedral obstacles. *Communications in Pure and Applied Mathematics*, **37**, 815–848.

[8] Sifrony, S. and Sharir, M., 1986, A new efficient motion planning algorithm for a rod in polygonal space. In *Proceedings of the 2nd Annual ACM Symposium on Computational Geometry*, pp. 178–185.

[9] Icking, C., Rote, G., Welzl, E. and Yap, C.K., 1993, Shortest paths for line segment. *Algorithmica*, **10**, 182–200.

[10] O'Rourke, J., 1987, Finding a shortest ladder path: a special case. IMA Preprint Series 353, Institute for Mathematic and its Applications, University of Minnesota.

[11] Asano, T., Kirkpatrick, D. and Yap, C.K. 1996, d_1 optimal motion for a rod. In *Proceedings of the 12th Annual ACM Symposium on Computational Geometry*, pp. 252–263.

[12] Asano, T., Kirkpatrick, D. and Yap, C.K., 2003, d_1 minimizing the endpoint trace length of the rod motions amidst polygonal obstacles is NP-hard. In *Proceedings of the 15th Canadian Conference on Computational Geometry*, pp. 10–13.

[13] Papadimitriou, C.H. and Silverberg, E.B., 1987, Optimal piecewise linear motion of an object among obstacles. *Algorithmica*, **2**, 523–539.

[14] Sharir, M., 1989, A note on the Papadimitriou–Silverberg algorithm for planning optimal piecewise-linear motion of a ladder. NYU Robotics Report, no. 188, Courant Institute, New York University.

[15] Asano, T., Kirkpatrick, D. and Yap, C.K., 2002, Pseudo approximation algorithms, with applications to optimal motion planning. In *Proceedings of the 18th Annual ACM Symposium on Computational Geometry*, pp. 170–178.

[16] De Berg, D., Van Kreveld, M., Overmars, M. and Schwarzkopf, O., 1997, *Computational Geometry: Algorithms and Applications*, Chapters 2, 13, 15 (Berlin: Springer).

[17] Choi, J., Sellen, J. and Yap, C.K., 1994, Approximate Euclidean shortest path in 3-space. In *Proceedings of the 10th Annual ACM Symposium on Computational Geometry*, pp. 41–48.