Computational Geometry 48 (2015) 732–742

Contents lists available at ScienceDirect

Computational Geometry: Theory and Applications

www.elsevier.com/locate/comgeo

Shortest path planning for a tethered robot ☆

Peter Brass [a], Ivo Vigan [b], Ning Xu [b,*]

[a] The City College of New York, New York, NY 10031, USA
[b] The Graduate Center, The City University of New York, NY 10016, USA

ARTICLE INFO

Article history:
Received 1 December 2013
Accepted 19 June 2015
Available online 23 June 2015

Keywords:
Shortest path planning
Tethered robots

ABSTRACT

We consider the problem of finding the shortest path for a tethered robot in a planar environment with polygonal obstacles of n total vertices. The robot is attached to an anchor point by a tether of finite length. The robot can cross the tether; i.e., the tether can be self-intersecting. Neither the robot nor the tether may enter the interior of any obstacle. The initial tether configuration is given as a polyline of k vertices.

If the tether is automatically retracted and kept taut, we present an $O(kn^2 \log n)$ time algorithm to find the shortest path between the source and the destination point. This improves the previous $O(lkn^3)$ time algorithm [24], where l is the number of loops in the initial tether configuration. If the tether can only be retracted while the robot backtracks along the tether, we present an algorithm to find the shortest path in $O((n + \log k) \log n)$ time.

1. Introduction

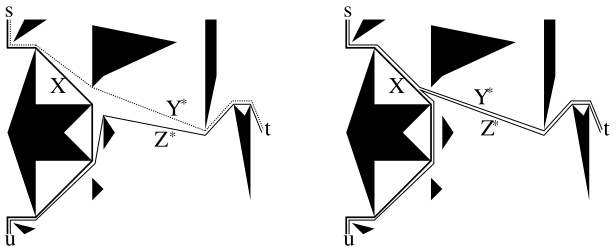
In recent decades, mobile robots have been used in many fields, such as in monitoring a building and exploring an unknown environment. In general, a robot carries batteries and communicates with humans and/or other robots over wireless connections. However, some robots (such as vacuum cleaner robots or "ROSIE" [1], a robot for decontamination and dismantlement operations) have high power demands which cannot be met by batteries. Moreover, for robots in a closed space, wireless communication may be impossible due to obstacle interference. If, on the other hand, a robot is attached to a tether, it can obtain sufficient power and stable communication through the tether.

There is a vast amount of literature focusing on the shortest path planning problem for untethered robots (see Halperin et al. [10] and Mitchell [22] for surveys). On the other hand, the same problem has received little attention for tethered robots. Path planning for tethered robots is much harder due to its temporal dependence, i.e., the shortest path is dependent on previous tether configurations.

The shortest path planning problem for a tethered robot can be described as follows. Let E be a known planar environment which consists of disjoint polygonal obstacles of n total vertices. Let s be the source point and t be the destination point of the robot in the environment. Suppose a robot, modeled as a point, is attached to an anchor point u by a tether of maximum length L. The initial configuration of the tether is given by a polyline X of k total vertices from s to u. The goal is to find the shortest path from s to t subject to the tether length constraint.

☆ This research is funded by NSF grant CCF-1017539.
* Corresponding author.
E-mail address: nxu@gradcenter.cuny.edu (N. Xu).

http://dx.doi.org/10.1016/j.comgeo.2015.06.004
0925-7721/



(a) The retraction model (b) The backtracking model

Fig. 1. Instances of the shortest path planning problem for a tethered robot: (a) the retraction model; and (b) the backtracking model. The paths and the tether configurations are drawn with a slight offset from the obstacles in order to be seen more easily.

In this paper, we assume that

- The tether is ideally flexible; i.e., there are no curvature limitation.
- Neither the robot nor the tether may enter the interior of any obstacle.
- The robot may cross the tether; i.e., the tether can be self-intersecting.
- The environment E can be modeled as a polygon with holes. Note that if the environment is unbounded, we can find a sufficiently large box containing the robot, the destination point, the tether and all obstacles.
- In the input, the path X is represented as an array of vertices. Random access and binary search on the array are supported.

We study the shortest path planning problem in two different models. In the first model, the tether is automatically retracted and kept taut; i.e., the tether is the shortest path in its homotopy equivalence class. This model is called the *retraction model*, and was considered by Xavier [24].

In the second model, we assume that the tether can only be retracted while the robot backtracks along the tether. This model is called the *backtracking model*. To our best knowledge, the backtracking model has never been studied before. We introduce this model by considering the case in which the tether cannot be dragged by the robot. For example, (1) the tether is too heavy to be dragged; (2) the tether may be damaged by debris on the ground; or (3) objects on the ground may fall over while the tether is dragged.

Fig. 1 illustrates an instance of each of the two models, where Y^* denotes the shortest path and Z^* denotes the final tether configuration after the robot reaches t.

Our contributions are described in the following theorems.

Theorem 1. *In the retraction model, the shortest path from the source point to the destination point can be computed in* $O(kn^2 \log n)$ *time.*

Theorem 2. *In the backtracking model, the shortest path from the source point to the destination point can be computed in* $O((n + \log k) \log n)$ *time.*

Note that our algorithms run on a machine which can compare sums of square roots and compute cosine functions.

This paper is organized as follows. Section 2 surveys the related works. Section 3 and Section 4 presents and analyzes the algorithm for the retraction model and the backtracking model respectively. Section 5 gives the conclusion and discusses some open problems.

2. Related problems

Studies on motion planning problems have been surveyed for untethered robots in Halperin et al. [10] and Latombe [19] and shortest path planning problems are surveyed in Mitchell [22].

When the polygon is simple, i.e., without holes, some algorithms for shortest path planning problems have been studied. Lee and Preparata [20] gave an algorithm to compute the shortest path between two points in $O(n)$ time, where n is the number of vertices of the polygon. Guibas et al. [9] showed the length of the shortest path from a fixed point to a query point can be computed in $O(\log n)$ time, using $O(n)$ preprocessing time and space. Guibas and Hershberger [8] presented an algorithm to query the length of the shortest path between any two points in $O(\log n)$ time, with $O(n)$ preprocessing time and space.

Chiang and Tamassia [4] considered the shortest path between two convex regions as holes within a simple polygon. Let n be the number of vertices of the simple polygon and let m be the number of total vertices of two convex regions. They developed an algorithm to query the length of the shortest path in $O(\log m + \log n)$ time with $O(n)$ preprocessing time and

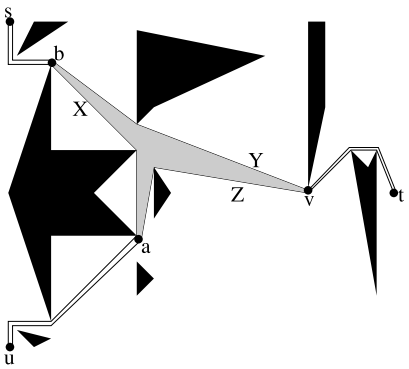


Fig. 2. The three paths X, Y, Z, and the three terminals a, b and v. The paths are drawn with a slight offset from the obstacles and from each other in order to be seen more easily.

space. In the same paper, they also considered the dynamic case. In the dynamic case, their algorithm queries the length of the shortest path in $O(\log m + \log^2 n)$ time with $O(n)$ preprocessing time and space and with $O(\log^2 n)$ update time.

Scholars also considered the shortest path planning problem on polygons with holes. Let n be the number of vertices of the polygon, and let h be the number of holes. For any two points in the polygon, the shortest path between them can be computed in $O(n \log n)$ time using $O(n \log n)$ space (Hershberger and Suri [12]), in $O(n^{1.5+\epsilon})$ time using $O(n)$ space (Mitchell [21]), or in $O(n + h^2 \log n)$ time using $O(n)$ space (Kapoor, Maheshwari and Mitchell [18]). Chiang and Mitchell [3] presented an algorithm to query the length of the shortest path between any two points in $O(\log n)$ time using $O(n^{11})$ preprocessing time and space, or in $O(\log^2 n)$ time using $O(n^{10} \log n)$ preprocessing time and space.

For tethered robots, Xavier [24] presented a shortest path planning algorithm with $O(lkn^3)$ running time. Sinden [23] gave an algorithm for scheduling multiple robots moving simultaneously to avoid tether tangling. Recently, Igarashi and Stilman [17] developed an $O(n)$ time algorithm to compute the shortest path for tethered robots on grid graphs without tether tangling.

For the case where a tethered robot cannot cross the tether but is allowed to push it, Hert and Lumelsky [13] considered the problem of planning trajectories for such tethered robots. They developed an algorithm to compute a set of trajectories to a target tether configuration for robots which depart from their anchor point. Later, they presented an algorithm for planning trajectories for tethered robots when each robot needs to visit its own target point [14], and generalized their algorithm into three dimensions [15].

The shortest path planning problem for tethered robots is similar to the problem of determining the path of a serpentine robot's head. The length, shape and curvature of the serpentine robot correspond to a tether. Chirikjian and Burdick [5] presented a method for obstacle avoidance by generating a configuration-space path for the robot from its path in working space. Several years later, Choset and Burdick [6,7] generalized Voronoi diagram constructions which are used for motion planning problems for serpentine robots.

3. The retraction model

In this section, we discuss the retraction model, in which the tether is automatically retracted and kept taut. This implies that the tether always coincides with its shortest homotopic path.

3.1. Notation

Let P be any arbitrary path within the environment E. Since we are only interested in shortest paths, we consider the case in which P is a directed polyline within E. We denote the shortest homotopic path of P by $H(P)$, and denote the Euclidean length of P by $|P|$. If P and Q are two paths in which Q starts from where P ends, we denote the path concatenated by P and Q by $P \circ Q$.

Because path P is a directed polyline, all points on P can be ordered from the source point of P to the destination point of P. For two points a and b on P, a is *before* b and b is *after* a, denoted by $a \prec b$, if a is closer to the source point than b along P. Suppose that $a \prec b$. We denote the subpath of P from a to b by $P_{a,b}$, and denote the reverse path of $P_{a,b}$ by $P_{b,a}$.

Let c and d be two points within E. We denote the shortest path from c to d within E by $SP(c, d)$.

3.2. Observations

Let Y be a path from s to t. Path Y is *admissible* if the robot can move to t along Y while subject to the tether length constraint. Let X be the initial tether configuration, and let Z be the final tether configuration after the robot arrives at t along Y, as illustrated in Fig. 2.

Lemma 1. *$X = H(X)$, $Y = H(Y)$ and $Z = H(Z)$.*

Proof. First, because the tether is kept taut, $X = H(X)$ and $Z = H(Z)$. Second, if $Y \neq H(Y)$, the robot can move to t along $H(Y)$ which is shorter than Y. If Y is admissible, so is $H(Y)$. Thus, $Y = H(Y)$. □

Because $X = H(X)$, the path X bends only at obstacle points. Similarly, Y and Z also bend only at obstacle points. We refer to all obstacle points and the endpoints of these three paths (u, s and t) as *terminals*.

Because Y and Z bend only at terminals, there exists a terminal v so that Y and Z merge at v and coincide from v to t (i.e., $Y_{v,t} = Z_{v,t}$). Similarly, we can find a terminal a so that X and Z coincide from u to a (i.e., $X_{u,a} = Z_{u,a}$) and separate at a. In addition, we can also find a terminal b so that Y and the reverse path of X coincide from s to b (i.e., $Y_{s,b} = X_{s,b}$) and separate at b. Fig. 2 illustrates the three paths and the corresponding terminals a, b and v.

We are interested in polygon R, which is formed by the three subpaths: $X_{a,b}$, $Y_{b,v}$ and $Z_{a,v}$. In Fig. 2, the polygon R is drawn as the gray region.

Lemma 2. *(Derived from Lemma 3.2 in [24].) The polygon R does not contain any terminal in its interior. Furthermore, R is a pseudo-triangle $\triangle abv$, i.e., a simple polygon with exactly three convex vertices a, b and v.*

For completeness, we review the proof in [24] as follows to prove Lemma 2.

Proof. Because the tether can be retracted from $X \circ Y$ to Z, the polygon R does not contain any terminal in its interior.

We claim that the subpath $X_{a,b}$ is not self-intersecting. If it is self-intersecting, because R does not contain any terminal in its interior, one can find a path shorter than X but homotopically equivalent to X. This contradicts $X = H(X)$. Thus, $X_{a,b}$ is not self-intersecting. Similarly, the subpaths $Y_{b,v}$ and $Z_{a,v}$ are also not self-intersecting.

Suppose that q is a convex vertex on $X_{a,b}$ other than a and b. Let p and r be the vertices of R incident to q. Because R contains no terminal in its interior, if one replaces (p, q) and (q, r) by their shortcut (p, r), by the triangle inequality, the resulting path is homotopically equivalent to X but shorter than X. This contradicts $H = H(X)$. Thus, there is no convex vertex of R on $X_{a,b}$ except a and b. Similarly, R does not have any convex vertex on $Y_{b,v}$ and $Z_{a,v}$ except a, b and v. Therefore, R is a pseudotriangle. □

Lemma 2 implies the following lemma.

Lemma 3. *Y is admissible if and only if $|Z| \leq L$.*

Proof. If Y is admissible, by the definition, the robot can move to t along Y subject to the tether configuration. When the robot reaches t, we have $|Z| \leq L$.

Suppose that $|Z| \leq L$. Consider a point p on the path Y. When the robot moves to p along Y, because the tether is automatically retracted and kept taut, the current tether configuration is $H(X \circ Y_{s,p})$. We need to prove that such a tether configuration is subject to the length constraint; i.e., $|H(X \circ Y_{s,p})| \leq L$.

If p lies between s and b, $|H(X \circ Y_{s,p})| = |H(X_{u,p})| \leq L$. If p lies between v and t, $|H(X \circ Y_{s,p})| = |Z_{u,p}| \leq L$. If p lies between b and v, by Lemma 2, $\triangle abv$ is a pseudotriangle and $Y_{b,v}$ is a convex chain with respect to the exterior of $\triangle abv$. Thus, $H(X \circ Y_{s,p})$ is not longer than both of $X_{u,b}$ and $Z_{u,v}$; i.e., $|H(X \circ Y_{s,p})| \leq L$.

Therefore, by definition, if $|Z| \leq L$, Y is admissible. □

Lemma 4. *$Y_{v,t} = SP(v, t)$.*

Proof. Suppose that $Y_{v,t} \neq SP(v, t)$. If Y is admissible, by Lemma 3, $|Z| \leq L$. Since $SP(v, t)$ is the shortest path between v and t, $|Z_{u,v} \circ SP(v, t)| \leq L$. By Lemma 3, the path $Y_{s,v} \circ SP(v, t)$ is also admissible. Because $|Y_{s,v} \circ SP(v, t)| \leq |Y|$, the robot can choose the path $Y_{s,v} \circ SP(v, t)$ instead of Y. Therefore, $Y_{v,t} = SP(v, t)$. □

A point c on X is an *event point* with respect to a terminal v if v becomes visible at c while one walks along X from u to s. Because $\triangle abv$ is a pseudotriangle, Y and Z bend away from each other at v. This implies that there exists an event point c on $X_{a,b}$ with respect to v, as illustrated in Fig. 3.

As illustrated in Fig. 3, we have:

$$Y = H(X_{s,c} \circ (c, v) \circ SP(v, t))$$

$$Z = H(X_{u,c} \circ (c, v) \circ SP(v, t))$$

This leads to an intuitive algorithm presented by Xavier [24], described as Algorithm 1.

In this paper, we simplify Xavier's algorithm in two ways. First, we use binary search to reduce the number of event points computed, as discussed in Section 3.3. Second, we preprocess X to speed up shortest homotopic path computations, as discussed in Section 3.5.

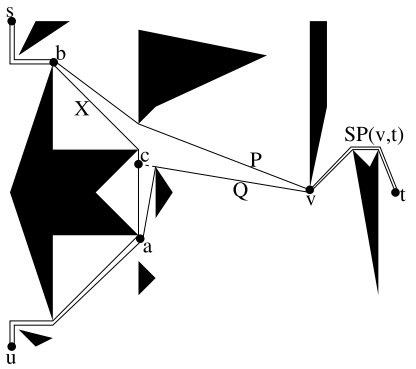


Fig. 3. The event point c, from where the terminal v becomes visible.

Algorithm 1: Xavier's algorithm [24].

Input: The environment E as a polygon, the initial tether configuration X from u to s, the destination point t, and the maximum tether length L.
Output: The shortest path from s to t subject to the tether length constraint.
1. Find all terminals;
2. foreach *terminal* v **do**
  (a) Find all event points with respect to v;
  (b) foreach *event point* c **do**
    (i) Compute the path $H(X_{u,c} \circ (c, v) \circ SP(v, t))$;
    (ii) If $|H(X_{u,c} \circ (c, v) \circ SP(v, t))| \leq L$, compute the path $H(X_{s,c} \circ (c, v) \circ SP(v, t))$, and consider such a path as a candidate path;
  end
end
3. Report the shortest candidate path.;

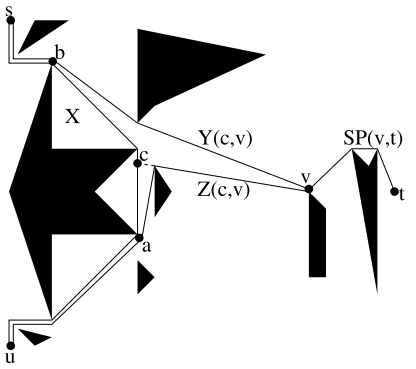


Fig. 4. The two paths $Y(c, v)$ and $Z(c, v)$.

3.3. Binary search

Xavier's algorithm computes two paths: $H(X_{u,c} \circ (c, v) \circ SP(v, t))$, and $H(X_{s,c} \circ (c, v) \circ SP(v, t))$. We define two paths as follows.

$$Y(c, v) = H(X_{s,c} \circ (c, v))$$

$$Z(c, v) = H(X_{u,c} \circ (c, v))$$

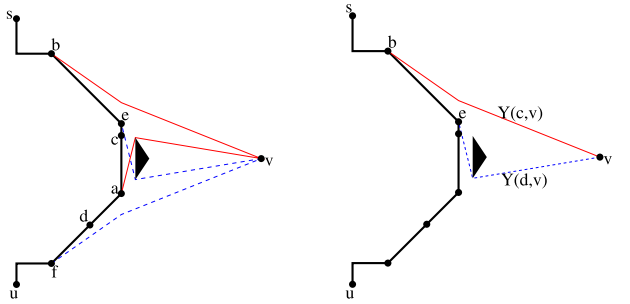
First, we modify Xavier's algorithm as follows. In step 2(b)(i), we compute the path $Z(c, v)$. In step 2(b)(ii), if $|Z(c, v) \circ SP(v, t)| \leq L$, we compute the path $Y(c, v) \circ SP(v, t)$, and call such a path a *candidate path*. The new algorithm is described as Algorithm 2, shown on the next page.

Fig. 4 illustrates the two paths. Note that $Y(c, v)$ and $Z(c, v)$ may not be the shortest homotopic path.

Lemma 5. *Algorithm 2 finds the shortest admissible path.*

Algorithm 2

Input: The environment E as a polygon, the initial tether configuration X from u to s, the destination point t, and the maximum tether length L.
Output: The shortest path from s to t subject to the tether length constraint.
1. Find all terminals;
2. foreach *terminal* v **do**
  (a) Find all event points with respect to v;
  (b) **foreach** *event point* c **do**
    (i) Compute the path $Z(c, v)$;
    (ii) If $|Z(c, v) \circ SP(v, t)| \leq L$, compute the path $Y(c, v) \circ SP(v, t)$, and consider such a path as a candidate path;
  end
end
3. Report the shortest candidate path;



(a) Two pseudotriangles $\triangle abv$ and $\triangle efv$.

(b) The path $Y(c, v)$ and $Y(d, v)$.

Fig. 5. The paths and vertices described in the proof of Lemma 6.

Proof. Suppose that Y^* is the shortest admissible path and Z^* is the corresponding final tether configuration. Let v^* be the terminal from where Y^* and Z^* separate, and let c^* be the corresponding event point. Because Y^* and Z^* separate at v^*, we have $Y^* = Y(c^*, v^*) \circ SP(v^*, t)$ and $Z^* = Z(c^*, v^*) \circ SP(v^*, t)$.

In step 2(b)(ii), if $|Z(c, v) \circ SP(v, t)| \leq L$, we have $|H(Z(c, v) \circ SP(v, t))| \leq L$. By Lemma 3, $H(Y(c, v) \circ SP(v, t))$ is an admissible path. Note that this step may discard some admissible paths, but will never discard the shortest admissible path.

If $|Z(c, v) \circ SP(v, t)| \leq L$, the algorithm computes the path $Y(c, v) \circ SP(v, t)$ as a candidate path. This path is at least as long as its shortest homotopic path. Therefore, the shortest admissible path is always the shortest candidate path reported by Algorithm 2. □

Next, we will show that both the length of $Y(c, v)$ and $Z(c, v)$ are monotone functions on event points in a subpath of X for the terminal v. Thus, we can use binary search to reduce the number of event points computed.

Let v be a terminal. Suppose p and q are two points on X with $p \prec q$ so that the subpath $X_{p,q}$ is concave with respect to v.

Lemma 6. *Let c and d be two event points on the subpath $X_{p,q}$ and $d \prec c$. It holds that $|Y(c, v)| \leq |Y(d, v)|$ and $|Z(d, v)| \leq |Z(c, v)|$.*

Proof. Let $\triangle abv$ be the pseudotriangle corresponding to c, and let $\triangle efv$ be the pseudotriangle corresponding to d, as illustrated in Fig. 5(a).

Because the two pseudotriangles do not contain any obstacle point in their interior, the paths $Y(c, v)$ and $Y(d, v)$ intersect only at v after they separate. Otherwise, they cannot be the shortest path in their homotopy equivalence class.

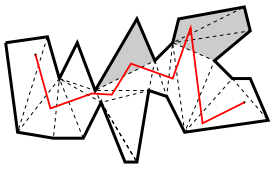
Furthermore, we claim that $b \neq e$. Assume $b \prec e$, because $d \prec c$, $Y(c, v)$ and $Y(d, v)$ must intersect at some point other than v after they separate. Thus, $Y(c, v)$ and $Y(d, v)$ separate at b, and they have a common prefix $X_{s,b}$.

Consider two paths $Y(c, v)$ and $Y(d, v)$, as illustrated in Fig. 5(b). They have the same endpoints and do not intersect in their interior. They lie on the same side of the line segment (b, v), but $Y(d, v)$ is farther to (b, v) than $Y(c, v)$. Because $Y(c, v)$ is convex with respect to (b, v), $|Y(c, v)| \leq |Y(d, v)|$.

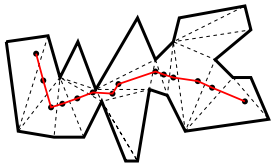
Similarly, we can show that $|Z(d, v)| \leq |Z(c, v)|$. □

The monotonicity on $Y(c, v)$ and $Z(c, v)$ allows us to use binary search to compute fewer event points. For a terminal v, we can partition the initial tether configuration X into subpaths so that every subpath is concave to v.

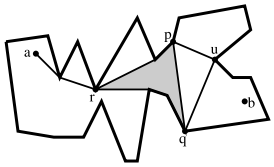
For each subpath, by Lemma 6, the length of $Z(c, v)$ monotonously increases when the event point c moves along X, while the length of $Y(c, v)$ monotonously decreases. Thus, we do not need to compute all event points on the subpath,



(a) An example of a path in a simple polygon.

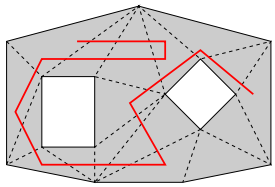


(b) The sleeve is the sequence of triangles visited by the red path.

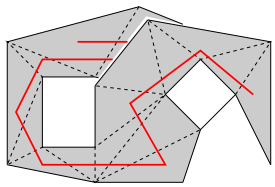


(c) A funnel before processing $\triangle pqu$ on the sleeve.

Fig. 6. The funnel algorithm. (For interpretation of the colors in this figure, the reader is referred to the web version of this article.)



(a) The polygon with holes. A path in it is drawn as the red polyline.



(b) "Lift" the polygon to be simply connected.

Fig. 7. The "lifting" technique. (For interpretation of the colors in this figure, the reader is referred to the web version of this article.)

as stated in step 2(b) in Algorithm 2. We can use binary search to find the event point c with longest $Z(c, v)$ subject to $|Z(c, v) \circ SP(v, t)| \leq L$. Such an event point corresponds to the shortest $Y(c, v) \circ SP(v, t)$ subject to the tether length constraint because of the monotonicity.

3.4. The funnel algorithm

In Algorithm 2, we compute two paths, $Y(c, v)$ and $Z(c, v)$. By their definition, these two paths are both the shortest path in their homotopy equivalence class.

The general idea for computing a shortest homotopic path is to use the funnel algorithm presented by Hershberger and Snoeyink [11]. We briefly summarize their algorithm.

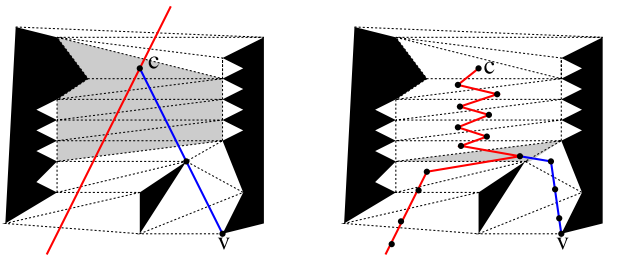
Let T be any triangulation of a simple polygon and note that the dual graph of T is a tree. If a path enters a triangle from an edge and immediately leaves the triangle through the same edge, such a triangle is *redundant*. Removing such a triangle does not change the homotopy class of the path.

A *sleeve* is a sequence of triangles whose dual graph is a simple path. The funnel algorithm first removes redundant triangles until it finds a sleeve, and processes the triangles on the sleeve in order. A data structure, called *funnel*, is maintained while processing the sleeve. A funnel is a collection of shortest homotopic path corresponding to the same sleeve. Fig. 6 illustrates an example of the funnel algorithm. In Fig. 6(a), a path is drawn as the red polyline, and redundant triangles are drawn as light gray regions. After removing redundant triangles, the corresponding sleeve is drawn in Fig. 6(b), as triangles visited by the red path. Fig. 6(c) illustrates the funnel when the algorithm traverses the sleeve before it reaches the triangle $\triangle pqu$.

When the environment is a polygon with holes, one can use the same algorithm by "lifting" the polygon to be simply connected, as illustrated in Fig. 7 (for details, see Hershberger and Snoeyink [11]).

3.5. Compute shortest homotopic paths faster

Recall that we need to compute $Y(c, v)$ and $Z(c, v)$. Because $Z(c, v)$ has the same prefix as X, we can compute $H(X)$ and store the funnels associated with each triangle on the sleeve of X. After this preprocessing, when we compute $Z(c, v)$, we do not need to compute funnels from the beginning, but from the triangle at which the sleeve of X and the sleeve of



(a) X is drawn as the red line. c is the event point at which the terminal v becomes visible. Gray regions are redundant triangles.

(b) After removing redundant edges, we backtrack the sleeve to the triangle $\triangle_s$ drawn as the gray triangle. The old sleeve to c and the new sleeve to v separate from $\triangle_s$.

Fig. 8. Remove redundant triangle, and compute the sleeve. (For interpretation of the colors in this figure, the reader is referred to the web version of this article.)

$Z(c, v)$ separate. We can use the funnel stored in this triangle and continue the funnel algorithm on the sleeve of $Z(c, v)$. This method can also be applied to compute $Y(c, v)$ because $Y(c, v)$ has the same prefix as the reverse path of X.

Lemma 7. *Given a terminal v and an event point c with respect to v, the path $Y(c, v)$ and $Z(c, v)$ can be computed in $O(n)$ time after preprocessing.*

Proof. Suppose v is a terminal and c is an event point with respect to v. We first triangulate the environment and compute $H(X)$ using the funnel algorithm [11]. In the funnel algorithm we process the triangles on the sleeve of X in order. For every processed triangle we store the current funnel associated with it.

We would like to mention that Bespamyatnikh has presented an algorithm to compute the shortest homotopic path [2] in sub-linear time. However, we do not use Bespamyatnikh's algorithm in this paper. Because we need to store funnels for every triangle on the sleeve of X, which needs $O(n)$ time. Bespamyatnikh's algorithm cannot reduce the time of preprocessing, but is more complicated than the funnel algorithm in [11].

Recall that $Z(c, v) = H(X_{u,c} \circ (c, v))$. Because the tether is kept taut, we have $X_{u,c} = H(X_{u,c})$. Thus, only the line segment (c, v) may introduce redundant triangles for finding the corresponding sleeve. Since (c, v) intersects $O(n)$ triangles, it introduces at most $O(n)$ redundant triangles. Thus, one can find the corresponding sleeve in $O(n)$ time.

The new sleeve corresponds to a path from u to v. Comparing this sleeve to the old sleeve corresponding to the path along X from u to c, these two sleeves have the same prefix and separate at some triangle $\triangle_s$, as illustrated in Fig. 8.

When we compute $Z_{c,v}$, we do not need to compute the funnels before $\triangle_s$ again, but use the funnel associated with $\triangle_s$, and continue the funnel algorithm on the new sleeve after $\triangle_s$. While we find the sleeve, we backtrack along the path X and find $\triangle_s$ at the same time. Because the number of triangles on the new sleeve after $\triangle_s$ is at most $O(n)$, by using the funnel algorithm, one can compute $Z_{c,v}$ in $O(n)$ time.

Similarly, we can compute $Y_{c,v}$ in $O(n)$ time. □

3.6. Algorithms

By applying binary search and maintaining a data structure which stores funnels associated with triangles, our algorithm to compute the shortest path from s to t in the retraction model is described as Algorithm 3.

The correctness of Algorithm 3 was discussed before. We now analyze the time complexity of Algorithm 3.

Step 1 can be computed in $O(n \log n)$ time using the triangulation method of Hertel and Mehlhorn [16].

For step 2, we use Hershberger and Snoeyink's algorithm [11] to find the shortest homotopic path of the path X and the reverse path of X. Because the path X has k vertices and there are $O(n)$ triangles in the triangulation, we need $O(kn)$ time to compute the two shortest homotopic paths. Furthermore, a funnel contains at most $O(n)$ triangles. Thus step 2 can be done in $O(kn^2)$ time.

In step 3, we use Hershberger and Suri's algorithm [12] to construct the Euclidean shortest path map of all terminals and compute the shortest path from t to every terminal. Since there are $O(n)$ terminals, this step can be done in $O(n \log n)$ time.

In step 4, because the X contains k vertices, we can traverse along X and partition X into subpaths concave to a terminal v in $O(k)$ time. Since there are $O(n)$ terminals in total, this step can be done in $O(kn)$ time.

In step 5(a), for each pair of terminal v and a subpath of X concave to v, we find all event points on the subpath from where v becomes visible by using a rotational sweep algorithm described as follows. We sort the terminals by their angular

Algorithm 3

Input: The environment E as a polygon, the initial tether configuration X from u to s, the destination point t, and the maximum tether length L.
Output: The shortest path from s to t subject to the tether length constraint.
1. Triangulate the environment E;
2. Compute the shortest homotopic path of the path X and the reversed path of X, and store the funnels associated with each triangle on their sleeve;
3. Construct the Euclidean shortest path map from t to every terminal;
4. For each terminal, partition X into subpaths so that every subpath is concave with respect to v, and find all event points with respect to this terminal;
5. foreach *pair of a terminal v and a subpath of X concave to v* **do**
  (a) Use binary search to find the event point c on the subpath with longest $Z(c, v)$ subject to $|Z(c, v) \circ SP(v, t)| \leq L$. Use funnels stored in step 2 in computing $Z(c, v)$;
  (b) Compute the path $Y(c, v) \circ SP(v, t)$, and consider such a path as a candidate path, also using funnels stored in step 2 to compute $Y(c, v)$;
end
6. Report the shortest candidate path;

order from v, rotationally sweep a ray from v and keep track of the terminals and the event points on the subpath. Since we need $O(n \log n)$ time to sort the vertices and $O(kn)$ time to compute the event points, we need $O(kn + n \log n)$ time for each terminal in the step 5(a).

For step 5(b), we use binary search in each subpath to find the event points, compute $Z_{c,v}$ for each event point accessed during binary search and compute $Y_{c,v}$ once for each subpath. As shown in Lemma 7, $Y_{c,v}$ and $Z_{c,v}$ can be computed in $O(n)$ time. Because there are $O(k)$ subpaths and each subpath contains $O(n)$ event points, we need $O(kn \log n)$ time for each terminal in the step 5(b).

Because there are at most $O(n)$ terminals, for step 5, we need $O(kn^2 \log n)$ time to process all terminals. This time cost also covers the time cost of all other steps. This proves Theorem 1.

4. The backtracking model

In this section, we discuss the backtracking model, in which the tether is retracted only while the robot backtracks along the tether. We will use the same notation as in Section 3.

4.1. Observations

The tether is retracted only while the robot backtracks along the tether. When the robot leaves from X at a point p, because the tether is not retracted any more after the robot leaves from p, the robot traverses the shortest path $SP(p, t)$. The final tether configuration will be $X_{u,p} \circ SP(p, t)$. We say that a point p on X is *feasible* if $|X_{u,p} \circ SP(p, t)| \leq L$, i.e., the robot's path is admissible. We also say that a point p is the *last feasible point* if there is no feasible point q with $p \prec q$.

Lemma 8. *For two points p and q on X with $p \prec q$, if q is feasible, so is p.*

Proof. Because q is a feasible point, $|X_{u,q} \circ SP(q, t)| \leq L$. Because $SP(p, t)$ is the shortest path from p to t, $|SP(p, t)| \leq |X_{p,q} \circ SP(q, t)|$. Since $X_{u,q} = X_{u,p} \circ X_{p,q}$, $|X_{u,p} \circ SP(p, t)| \leq L$, i.e., p is a feasible point. □

Lemma 9. *The shortest admissible path from s to t is the path $X_{s,r} \circ SP(r, t)$, where r is the last feasible point on X.*

Proof. Suppose p and q are two feasible points with $p \prec q$. If the robot leaves from X at p, the robot traverses along the path $X_{s,p} \circ SP(p, t)$. Similarly, if the robot leaves at q, it traverses along the path $X_{s,q} \circ SP(q, t)$.

Because $SP(q, t)$ is the shortest path from q to t, we have $|SP(q, t)| \leq |X_{q,p} \circ SP(p, t)|$. Because $X_{s,p} = X_{s,q} \circ X_{q,p}$, $|X_{s,q} \circ SP(q, t)| \leq |X_{s,p} \circ SP(p, t)|$. This means that if the robot leaves X at q instead of p, it follows a path shorter or of equal length.

By Lemma 8, every point before the last feasible point is a feasible point. Let r be the last feasible point on X. We have $|X_{s,r} \circ SP(r, t)| \leq |X_{s,p} \circ SP(p, t)|$ for any feasible point p other than r. Therefore, the path $X_{s,r} \circ SP(r, t)$ is the shortest admissible path; i.e., the robot should leave X from r. □

Lemma 10. *If the last feasible point is not the source point s, the length of the final tether configuration is exactly L.*

Proof. Let p be the last feasible point. Because p is a feasible point, the length of the final tether configuration, $X_{u,p} \circ SP(p, t)$, is $L - \alpha$ for some $\alpha \geq 0$.

If the length of the final tether configuration is not L, we have $\alpha > 0$. One can find a point q on X with $p \prec q$ and $X_{p,q}$ has length at most $\alpha/2$. Because $|X_{u,q} \circ SP(q, t)| \leq |X_{u,p} \circ X_{p,q} \circ X_{q,p} \circ SP(p, t)| \leq L$, the point q is also a feasible point, which contradicts that p is the last feasible point. Therefore, the length of the final tether configuration is exactly L. □

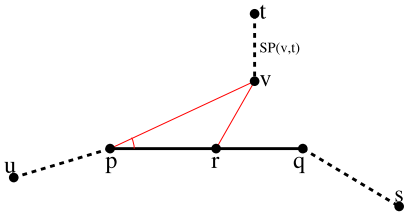


Fig. 9. An illustration of the last feasible point r and the triangle $\triangle vpr$.

Algorithm 4: Find the shortest path in the backtracking model.

Input: The environment E as a polygon, the initial tether configuration X from u to s, the destination point t, and the maximum tether length L.
Output: The shortest path from s to t subject to the tether length constraint.
1. Construct the Euclidean shortest path map from t to every obstacle point;
2. If s is a feasible point, directly return the path $SP(s,t)$ as the shortest path;
3. Use binary search to find the line segment $X_{p,q}$ that contains the last feasible point;
4. **foreach** *terminal* v **do**
  (a) Check whether there exists a feasible point c on $X_{p,q}$ such that the length of the path $X_{u,c} \circ SP(c,t)$ is exactly L;
  (b) If such a feasible point exists, compare it with the feasible points obtained before, and choose the last one among them;
end

Thus, we can check the vertices of X, and use binary search to find the line segment containing the last feasible point.

Suppose that r is the last feasible point and $r \neq s$. Let $X_{p,q}$ be the line segment containing r with $p \prec q$. Because $SP(r,t)$ is the shortest path from r to t, there exists a terminal v such that $SP(r,t) = (r,v) \circ SP(v,t)$ and v is visible from r. Fig. 9 illustrates the last feasible point r and the terminal v.

Because the length of the final tether configuration is exactly L, it holds that

$$|(p,r)| + |(r,v)| = L - |X_{u,p}| - |SP(v,t)|$$

Because

$$|(r,v)|^2 = |(p,r)|^2 + |(p,v)|^2 - 2|(p,r)| \cdot |(p,v)| \cdot \cos\angle vpr$$

we can compute the corresponding point c on X in $O(1)$ time for any given obstacle point v. We can then verify whether the point c lies on the line segment $X_{p,q}$ in $O(1)$ time and verify whether $|X_{u,c} \circ SP(c,t)| = L$ by computing the length of $SP(c,t)$ in $O(\log n)$ time [12]. If the point passes the verification, we consider the point to be a candidate point.

4.2. Algorithm

Algorithm 4 computes the shortest path from s to t in the backtracking model.

The correctness of Algorithm 4 has been discussed before. We now analyze its time complexity.

Since there are at most n obstacle points, step 1 can be done in $O(n \log n)$ time using the algorithm of Hershberger and Suri [12].

In step 2, we check whether the path $X \circ SP(s,t)$ is not longer than L. This can be done by querying $SP(s,t)$ in $O(\log n)$ time [12].

For step 3, we use binary search to find the line segment containing the last feasible point. Since the path X has k vertices, we need $O(\log k)$ queries to find the shortest path from a vertex on X to t. Each query can be done in $O(\log n)$ time [12]. Thus, step 3 can be done in $O(\log k \log n)$ time.

In step 4, we need $O(\log n)$ time to compute the candidate point for each obstacle point. Since there are n obstacle points in total, this step can be done in $O(n \log n)$ time.

Thus Algorithm 4 finds the shortest path from s to t in $O((n + \log k)\log n)$ time. This proves Theorem 2.

As stated in Section 1, we assume that the path X is represented by an array of vertices, which supports random access and binary search on such an array. If the path in X is represented by other data structures, e.g., a list, we have to establish such an array by traversing X, which costs an additional $O(k)$ time, leading to an algorithm with running time $O(k + (n + \log k)\log n)$.

5. Conclusion and open problems

In this paper, we discussed the shortest path planning problem for a tethered robot. If the tether is automatically retracted and kept taut, we present an algorithm to compute the shortest path in $O(kn^2 \log n)$ time. If the tether can only be retracted while the robot backtracks along the tether, we present an algorithm to compute the shortest path in $O((n + \log k)\log n)$ time.

In the shortest path planning problem, the destination point may be too far to be reached from the anchor point. One possible solution is deploying multiple anchor points in the environment. Each anchor point has a finite length tether associated with it. The robot can be attached to a new tether and can be released from the old one when it reaches a new anchor point. A released tether is automatically retracted to zero length. The robot can reach a destination point by transferring between anchor points. In this setting, our algorithms can be applied to find the shortest path to the destination point. The key observation is that when the robot transfers at an anchor point, the tether length is zero, so the robot should follow the Euclidean shortest path to the destination point, or the Euclidean shortest path to the next transfer anchor point. One can solve this problem by constructing a reachability graph between the anchor points and computing the shortest path to the destination point and every anchor point without transfer.

In this paper, the tether has no curvature limitation. However, it may not hold in practice. An interesting open problem is: how to compute the shortest path from s to t if the tether has limitations to the curvature?

Another interesting problem is introduced by combining the tether robot problem and the watchman route problem: how to compute the shortest watchman route for a tethered robot?

References

[1] L.C. Bares, L.S. Conley, B.R. Thompson, Rosie: A Mobile Worksystem for Decontamination and Dismantlement Operations, 1995, pp. 231–238.
[2] S. Bespamyatnikh, Computing homotopic shortest paths in the plane, J. Algorithms 49 (2) (2003) 284–303.
[3] Y.-J. Chiang, J.S.B. Mitchell, Two-point Euclidean shortest path queries in the plane, in: R.E. Tarjan, T. Warnow (Eds.), SODA, ACM/SIAM, 1999, pp. 215–224.
[4] Y.-J. Chiang, R. Tamassia, Optimal shortest path and minimum-link path queries between two convex polygons inside a simple polygonal obstacle, Int. J. Comput. Geom. Appl. 7 (1) (1997) 85–121.
[5] G.S. Chirikjian, J.W. Burdick, An obstacle avoidance algorithm for hyper-redundant manipulators, in: Proceedings of IEEE International Conference on Robotics and Automation, vol. 1, 1990, pp. 625–631.
[6] H. Choset, J. Burdick, Sensor based planning. I. The generalized Voronoi graph, in: Proceedings of IEEE International Conference on Robotics and Automation, vol. 2, 1995, pp. 1649–1655.
[7] H. Choset, J. Burdick, Sensor based planning. II. Incremental construction of the generalized Voronoi graph, in: Proceedings of IEEE International Conference on Robotics and Automation, vol. 2, 1995, pp. 1643–1648.
[8] L.J. Guibas, J. Hershberger, Optimal shortest path queries in a simple polygon, J. Comput. Syst. Sci. 39 (2) (1989) 126–152.
[9] L.J. Guibas, J. Hershberger, D. Leven, M. Sharir, R.E. Tarjan, Linear-time algorithms for visibility and shortest path problems inside triangulated simple polygons, Algorithmica 2 (1987) 209–233.
[10] D. Halperin, L.E. Kavraki, J.-C. Latombe, Robotics, in: J.E. Goodman, J. O'Rourke (Eds.), Handbook of Discrete and Computational Geometry, CRC Press, Boca Raton, NY, 2004, pp. 1065–1094, chapter 48.
[11] J. Hershberger, J. Snoeyink, Computing minimum length paths of a given homotopy class, Comput. Geom. 4 (1994) 63–97.
[12] J. Hershberger, S. Suri, An optimal algorithm for Euclidean shortest paths in the plane, SIAM J. Comput. 28 (6) (1999) 2215–2256.
[13] S. Hert, V.J. Lumelsky, The ties that bind: motion planning for multiple tethered robots, in: ICRA, 1994, pp. 2734–2741.
[14] S. Hert, V.J. Lumelsky, Moving multiple tethered robots between arbitrary configurations, in: Proceedings of IEEE/RSJ International Conference on Intelligent Robots and Systems, vol. 2, 1995, pp. 280–285.
[15] S. Hert, V.J. Lumelsky, Motion planning in R^3 for multiple tethered robots, IEEE Trans. Robot. Autom. 15 (4) (1999) 623–639.
[16] S. Hertel, K. Mehlhorn, Fast triangulation of the plane with respect to simple polygons, Inf. Control 64 (1–3) (1985) 52–76.
[17] T. Igarashi, M. Stilman, Homotopic path planning on manifolds for cabled mobile robots, in: D. Hsu, V. Isler, J.-C. Latombe, M.C. Lin (Eds.), WAFR, in: Springer Tracts in Advanced Robotics, vol. 68, Springer, 2010, pp. 1–18.
[18] S. Kapoor, S.N. Maheshwari, J.S.B. Mitchell, An efficient algorithm for Euclidean shortest paths among polygonal obstacles in the plane, Discrete Comput. Geom. 18 (4) (1997) 377–383.
[19] J.-C. Latombe, Robot Motion Planning, Kluwer Academic Publishers, Norwell, MA, USA, 1991.
[20] D.-T. Lee, F.P. Preparata, Euclidean shortest paths in the presence of rectilinear barriers, Networks 14 (3) (1984) 393–410.
[21] J.S.B. Mitchell, Shortest paths among obstacles in the plane, Int. J. Comput. Geom. Appl. 6 (3) (1996) 309–332.
[22] J.S.B. Mitchell, Shortest paths and networks, in: J.E. Goodman, J. O'Rourke (Eds.), Handbook of Discrete and Computational Geometry, CRC Press, Boca Raton, NY, 2004, pp. 607–642, chapter 27.
[23] F.W. Sinden, The tethered robot problem, Int. J. Robot. Res. 9 (1) (1990) 122–133.
[24] P.G. Xavier, Shortest path planning for a tethered robot or an anchored cable, in: ICRA, 1999, pp. 1011–1017.