2015 IEEE International Conference on Robotics and Automation (ICRA)
Washington State Convention Center
Seattle, Washington, May 26-30, 2015

Optimal motion planning for a tethered robot: Efficient preprocessing for fast shortest paths queries

Oren Salzman* and Dan Halperin*

Abstract— **We study the problem of planning the shortest path for a polygonal robot anchored to a fixed base point by a finite tether translating among polygonal obstacles in the plane. Specifically, we preprocess the workspace to efficiently answer queries of the following type: Given a source location of the robot and an initial configuration of the tether, compute the shortest path to reach a target location while avoiding obstacles and adhering to the tether's constraints. Our work is an extension of the recent work by Kim et al. [1] who considered the problem for a point robot. Their algorithm relies on a discretization of the workspace and is optimal with respect to this discretization. We first replace their grid-based approach with a visibility-graph based approach. This allows to improve the running time of their algorithm by several orders of magnitude. Specifically, testing on a scenario similar to one presented by Kim et al., the running time is improved by a factor of more than 500. Moreover, our approach, which plans *optimal* paths, is applicable to polygonal (translating) robots and can be used to plan a shortest path while ensuring a predefined clearance from the obstacles. We report on our experimental results on a variety of scenarios. In all cases the preprocessing time is less than one second on a standard-commodity laptop, and a typical query takes several tens of miliseconds.**

I. Introduction

We consider a planar polygonal robot translating amidst polygonal obstacles while being anchored by a tether to a given base point p_b. The robot may drive over the cable, which is a flexible and stretchable elastic band remaining taut at all times. We study the problem of constructing a data structure that allows to efficiently compute the shortest path of the robot between any two given points p_s, p_t while satisfying the constraint that the tether can extend to length at most L from the base.

Tethers are often used to provide energy to robots or as a communication link for tele-operation. Examples of scenarios where such tethered robots are used can be found in underwater or disaster recovery missions [2], [3]. The finite length of the tether, as well as its position, need to be accounted for when planning a path for the robot.

This work builds on the recent work by Kim et al. [1]. They study the problem of planning the shortest path for a *point* tethered robot. In their work, they suggest to use the notion of *homotopy classes* of curves to address this problem. They overlay the workspace with a grid and compute all possible homotopy classes that could be used to reach any grid vertex from the base point. This expensive preprocessing stage allows for an efficient query stage: given a query source grid-vertex with the homotopy class of the cable, they use the A* algorithm [4] to traverse a precomputed graph. This allows to find a path that reaches the target grid vertex while satisfying the tether's constraints. Their approach obtains a solution that is optimal with respect to the grid size and to several simplifying assumptions.

We replace the approximate grid approach with an optimal visibility-graph based solution, which is at the same time significantly faster. We first consider the case of a point robot and discuss several additional issues such as the computational complexity of the data structure that is obtained and how to efficiently compute and use a data structure that describes the homotopy invariant of curves. We then extend the algorithm for the case of a tethered polygonal robot translating in the plane. This extension also allows to ensure a minimal clearance between the robot and the obstacles along the path traversed by the robot.

A. Related work

The problem of planning for a tethered robot has been studied for almost two decades. Hert and Lumelsky provide an algorithm for motion planning of multiple tethered robots that cannot cross each other's tether in a common planar and spatial environment that is free of obstacles [5], [6].

Planning the shortest path given a single source and a single target was considered by Xavier [7] and later by Xu et al. [8]. Similar to the work presented in this paper, their work uses the notion of *visibility graphs* (see Section II).

The work by Kim et al. builds upon the early work by Igarashi and Stilman who study a similar problem where the tether is not stretchable. They use an augmented graph to efficiently encode paths in the same homotopy class [9]. In addition, Kim et al. use the topological approach to motion planning taken by Bhattacharya et al. [10] and later, in another paper by Kim et al. [11] to formally define the data structure they use.

Planning for tethered robots has also been studied in the context of coverage. Shnaps and Rimon [12] give an online algorithm in which a tethered robot has to cover an unknown planar environment. Using similar techniques, they consider on-line coverage by a battery-powered robot [13]. As they represent the battery capacity by the path length L that the robot can travel under a full battery charge, their solution shares many characteristics of planning for a tethered robot.

Finally, Abad-Manterola et al. [14] study the problem of planning a path for a tethered axel rover on steep terrains. Their work was later extended to an online algorithm [15].

* Blavatnik School of Computer Science, Tel-Aviv University, Israel

This work has been supported in part by the Israel Science Foundation (grant no. 1102/11), by the German-Israeli Foundation (grant no. 1150-82.6/2011), and by the Hermann Minkowski–Minerva Center for Geometry at Tel Aviv University.

B. Paper organization

In Section II we formally describe the setting and provide the necessary algorithmic background. We continue in Section III to describe our visibility-graph based algorithm for the case of a point robot. Our algorithm is straightforwardly extended in Section IV for the case of a polygonal translating robot. Simulations are presented in Section V.

II. Setting and algorithmic background

A. Setting

Let $\mathcal{W} \subset \mathbb{R}^2$ be the workspace where the robot moves and let the obstacles $\mathcal{O} = \{\mathcal{O}_1, \ldots, \mathcal{O}_m\}$ be a collection of m pairwise interior-disjoint simple polygons having n vertices in total. Polygons are considered as open sets—this allows for motions where the robot translates while "grazing" the obstacles. Given two points p, q we denote by pq the straight-line segment connecting p to q, and by $\|pq\|$ the Euclidean distance between p and q. A graph $G = (V, E)$ is *embedded* in the workspace if (i) each vertex is associated with a point in the workspace and (ii) each edge of the graph is associated with the straight-line segment connecting the two points associated with its vertices. When referring to embedded graphs we use the vertices of the graphs and their associated points interchangeably. Finally, we say that a graph is embedded in the free space if it is embedded in $\mathcal{W} \setminus \mathcal{O}$, namely if (i) $\forall v \in V,\ v \notin \mathcal{O}$ and if (ii) $\forall (u, v) \in E,\ uv \cap \mathcal{O} = \emptyset$.

B. Visibility graph

To efficiently compute the shortest path for a tethered robot we make use of *visibility graphs*. The visibility graph $\mathcal{G}^{\text{vis}} = (\mathcal{V}^{\text{vis}}, \mathcal{E}^{\text{vis}})$ of $\mathcal{O}$ is an undirected graph embedded in the free space defined on the set of the polygon vertices. Its set of edges consists of those pairs of vertices that are mutually visible. Two vertices are *mutually visible* if the straight-line segment connecting them does not intersect the interior of any of the polygons in $\mathcal{O}$. In this case, we call this segment a *visibility edge*.

The visibility graph (consisting of the vertices and all the visibility edges) can be used to compute shortest paths amidst polygonal obstacles, where the polygons are considered as open sets [16, C. 15]. Each edge is given a weight equal to the Euclidean distance between its two end-vertices. To find a shortest path between a source s and a target t, one simply needs to add s and t as vertices in $\mathcal{G}^{\text{vis}}$ together with all visibility edges incident to s or t. The path may be obtained by executing Dijkstra's algorithm starting from s.

In fact, to compute shortest paths, it is sufficient to consider only the edges that are bitangent to the polygons they connect, namely edges that can be infinitesimally extended in both directions without penetrating any polygon. Such bitangent edges are called *reduced* visibility edges (see Fig. 1). We refer to the set of reduced visibility edges as $\tilde{\mathcal{E}}^{\text{vis}} \subseteq \mathcal{E}^{\text{vis}}$ and to the reduced visibility graph as $\tilde{\mathcal{G}}^{\text{vis}} = (\mathcal{V}^{\text{vis}}, \tilde{\mathcal{E}}^{\text{vis}})$. The path constructed using $\tilde{\mathcal{G}}^{\text{vis}}$ is the shortest path between s and t. The (reduced) visibility graph can be computed in $O(|\mathcal{E}^{\text{vis}}| + n \log n)$ time [17].

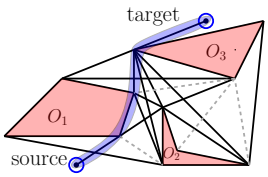


Fig. 1. Visibility graph defined over a set of three obstacles. The reduced visibility edges are depicted in black, the visibility edges which are not reduced are depicted in dashed gray. A shortest path computed using the visibility graph is highlighted in blue.

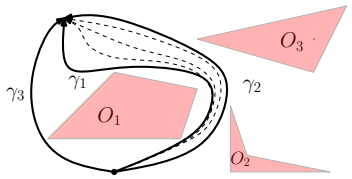


Fig. 2. The two curves γ_1, γ_2 are homotopic as there is a continuous deformation between the two that does not intersect any obstacle. γ_3 is not homotopic to γ_1, γ_2 as any deformation will intersect an obstacle.

As we will see, to allow for a minimal clearance for the robot, we will need to construct the visibility graph of a set of obstacles that have circular arcs on their boundary. In this case, $\mathcal{G}^{\text{vis}}$ consists of every visible bi-tangent of two circular arcs of the boundary of $\mathcal{O}$, every two mutually visible vertices and every visibility-edge between a vertex and a point tangent to a circular arc. Constructing the visibility graph of obstacles that include circular arcs can be done in $O(n^2 \log n)$ time [18].

C. Minkowski sums and translating polygons

To extend our algorithm from point robots to polygonal robots we will use *Minkowski sums*. Given two sets $P, Q \in \mathbb{R}^2$, their Minkowski sum, denoted $P \oplus Q$, is their pointwise vector sum, namely the set $P \oplus Q = \{p + q | p \in P, q \in Q\}$. The complexity of the Minkowski sum of two planar polygons P, Q with n_1 and n_2 vertices respectively is $\Theta(n_1 + n_2)$ if both P and Q are convex, $\Theta(n_1 n_2)$ in the case one of the two is convex [19] and $\Theta(n_1^2 n_2^2)$ in the general case [20]. Let R be a polygonal robot such that the origin lies in R and O be a polygonal obstacle. The robot intersects the obstacle if the origin lies inside the Minkowski sum of the obstacle and a reflection of R through the origin, namely in $M = -R \oplus O$. Thus, given a set of obstacles $\mathcal{O}$, the set of placements of the robot that will cause a collision with an obstacle is simply $\bigcup_{O \in \mathcal{O}} O \oplus -R$ (see, e.g, [21, C. 4]).

D. Homotopy of curves and h-signature

Homotopy classes of curves are a fundamental notion used in the work of Kim et al. [1]. Let $\gamma_1, \gamma_2 \in \mathcal{W} \setminus \mathcal{O}$ be two curves sharing the same start and end points. They share the same *homotopy class* if there is a continuous deformation between γ_1 and γ_2 in $\mathcal{W} \setminus \mathcal{O}$ (see Fig. 2).

In order to identify if two curves that share the same endpoints are homotopic we use the notion of h-signature. This notion comes from early work on topology-based approaches to motion planning [10], [11], [22]. Take a point $p_k \in O_k$ in each obstacle and extend a vertical ray ρ_k towards $y = +\infty$

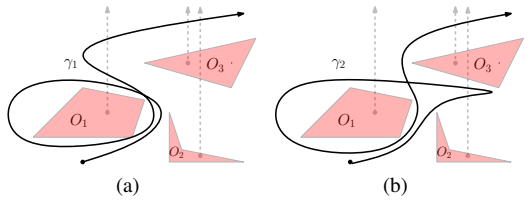


Fig. 3. Two homotopic curves γ_1, γ_2. (a) The signature of γ_1 is $s(\gamma_1) = \overline{t_1}t_1t_1t_3t_2$. The reduced signature of the curve γ_1 is $h(\gamma_1) = \overline{t_1}t_3t_2$. (b) The signature of γ_2 is $s(\gamma_2) = t_2\overline{t_2}t_1t_3t_2$. The reduced signature of the curve γ_2 is $h(\gamma_2) = t_1t_3t_2$.

from p_k. Given a curve γ, let $\rho_{k_1}, \rho_{k_2}, \ldots$ be the sequence of rays crossed when tracing γ. The *signature* of a curve γ, denoted by $s(\gamma)$, is defined as a sequence of letters, one for each such crossing. If γ intersects the ray ρ_k, extended from p_k, by crossing it from left to right then the letter t_k is added to the word. If γ intersects ρ_k by crossing it from right to left then the letter $\overline{t_k}$ is added. The reduced word, denoted by $r(s(\gamma))$, is constructed by eliminating every two consecutive letters in the form of $t_k\overline{t_k}$ or $\overline{t_k}t_k$. The reduced word $r(s(\gamma))$ is a *homotopy invariant* for curves with fixed endpoints. It will be denoted as $h(\gamma) = r(s(\gamma))$ and called the *h-signature* of γ (See Fig. 3).

The *h-signature* of a curve γ which is a concatenation of two curves $\gamma = \gamma_1 \cdot \gamma_2$ is the reduced signature of the concatenation of the two curves' signatures $h(\gamma) = r(s(\gamma_1) \cdot s(\gamma_2))$. The homotopy invariant of a curve, its h-signature, uniquely identifies the homotopy class of the curve. That is, two curves connecting the same points have identical reduced words iff they are homotopic. For more information, see [1].

E. Homotopy-augmented graph

We now review the *homotopy-augmented graph* G_h of a graph $G = (V, E)$ embedded in the free space. The graph G_h encodes for each vertex of G, all homotopy classes that can be used to reach the vertex using a tether of length L. It can then be used to efficiently answer queries solving the motion-planning problem for a point tethered robot.

Recall that p_b is the base point to which the tether is anchored and that the maximal length of the tether is L. The *homotopy-augmented graph* G_h is defined as follows: the vertices are tuples (v, w, c) where $v \in V$ is a vertex of G, w defines an h-signature to reach v from the base node p_b and c is the cost (Euclidean shortest path) to reach p from p_b in the homotopy class defined by w. Note that there could be several vertices in G_h sharing the same point p with different h-signatures and possibly different costs.

The homotopy-augmented graph can be constructed by running Dijkstra's shortest path algorithm from the base node which is $(p_b, \wedge, 0)$ i.e., using an h-signature that is the empty word (denoted by $\wedge$) and a cost of zero. The search is terminated when all nodes of cost less than L have been exposed. Let (v_i, w_i, c_i) be a vertex considered by Dijkstra's algorithm for some vertex $v_i \in V$ of G, h-signature w_i and some cost c_i representing the length of the shortest path to reach v_i in the homotopy class defined by w_i. For every $v_j \in V$ such that $(v_i, v_j) \in E$, the node (v_j, w_j, c_j) is added to the graph with h-signature $w_j = r(w_i \cdot h(v_iv_j))$ and cost $c_j = c_i + \|v_iv_j\|$ if (i) $c_j \leq L$ and (ii) there is no node (v_j, w_j, c'_j) already in the graph for some cost c'_j.

F. Planning using grid-based homotopy-augmented graphs

Kim et al. [1] addressed the motion-planning problem of a point tethered robot by overlaying a grid on $\mathcal{W} \setminus \mathcal{O}$. The grid induces an embedded graph where the vertices of the graph are the grid vertices. Each grid point has eight neighbors (grid points to the left, right, top, bottom and the four diagonals). An edge is added between two vertices if the line connecting the corresponding grid points is in $\mathcal{W} \setminus \mathcal{O}$. After computing this grid-based graph, they compute the homotopy-augmented graph induced by the grid. As they use a grid, the cost computed is an approximation of the Euclidean distance traversed by the tether. Thus, they add a shortcutting algorithm to reduce this effect.

The algorithm presented by Kim et al. relies on the grid resolution to be fine enough to accurately capture the free space $\mathcal{W} \setminus \mathcal{O}$. Even for simple scenarios with a few convex polygons, their homotopy-augmented graph consists of around one million vertices and takes between several seconds to several hundreds of seconds to compute.

III. Homotopy-augmented visibility graphs

In this section we describe how to replace the grid-based homotopy-augmented graph introduced by Kim et al. with a visibility-based homotopy-augmented graph $\mathcal{G}_h^{\text{vis}}$. We start by describing the data structure used to efficiently compute the h-signature. We then describe $\mathcal{G}_h^{\text{vis}}$ and explain how it is used to answer a query. Finally, we summarize the computational complexity of our algorithm.

A. Extending rays from the obstacles

Given the set $\mathcal{O}$ of m obstacles, we take a point $p_k \in O_k$ in each obstacle and extend a ray towards $y = +\infty$ from p_k. The points are picked such that no two points share the same x-value (this is easy to do assuming that each obstacle has non-zero area). We set $P = \{p_1 \ldots p_m\}$ to be all such points.

Our algorithm will use curves that are polylines, thus, we wish to efficiently compute the h-signature of a line segment. To do so, we need a structure that will answer queries of the form: Given a query line segment q in the plane, efficiently report all the points in P that lie in the x-range of q and are below q. More formally, if the endpoints of q are (x_1, y_1) and (x_2, y_2), then we seek to report all the points $p = (x, y) \in P$ such that (i) $x_1 \leq x \leq x_2$ and (ii) the vertical line through p intersects q in a point (x', y') with $y' \geq y$.

There is a variety of data structures that could be used here and we restrict our attention to those that require only near-linear space and preprocessing time of $|P|$.

We handle the case where the query segment q is vertical separately with a simple search structure on the x-coordinates of the points. From this point on we assume that q is not vertical. We construct a *mutli-level* data structure as follows. The first level of the data structure is a balanced

binary tree T on the points in P according to their x-coordinates. Each node v of T is assigned with a subset of points of P, which we denote by $v(P)$: Each leaf of T is assigned with the single point of P with the specific x-coordinate of that point. An internal node of T is assigned all the points in the leaves of its subtree. With each node v of T we associate a secondary search structure on the points of $v(P)$. This secondary structure is a partition tree [16] that answers half-plane range queries, namely which points of $v(P)$ lie below a given query line in the plane.

We query T with the x-range of q to obtain all the points in P that lie in this range, arranged in a logarithmic number of subsets. Now for each such subset we query the associated partition tree with the halfplane below the line supporting q.

The construction of the structure takes $\widetilde{O}(m)$ time and requires $\widetilde{O}(m)$ storage space[1]. The query takes $\widetilde{O}(\sqrt{m}+k)$ time, where k is the number of points in the output. One could further refine this construction to balance between the preprocessing time and the overall query time.

At the other end of the spectrum one can devise a structure with only logarithmic overhead in the query time. However, while theoretically these are the best known data structures, practically, these structures are rather wasteful in space because of the constants hidden in the big-Oh notation, and are difficult to program. We therefore resort to a simpler solution which is nevertheless efficient in practice.

Let $B(x_1^b, y_1^b, x_2^b, y_2^b)$ be an axis-aligned bounding box (AABB for short) of the entire workspace. We denote a two-dimensional AABB by its left-bottom and top-right points. We preprocess the points of P by constructing a two-dimensional kd-tree [16, C. 5], which we denote by KD(P), for answering orthogonal range queries.

Recall that q is the query segment and let (x_1, y_1) and (x_2, y_2) denote its endpoints. Set $x_1^q = \min(x_1, x_2)$, $x_2^q = \max(x_1, x_2)$, $y_1^q = \min(y_1, y_2)$ and $y_2^q = \max(y_1, y_2)$. We query KD(P) twice. First with the rectangle $B(x_1^q, y_1^q, x_2^q, y_2^q)$, and filter out output points that lie above q. To the remaining points we add the result of querying KD(P) with $B(x_1^q, y_1^b, x_2^q, y_1^q)$. See Fig. 4.

Constructing KD(P) takes $O(m \log m)$ time. Let $B_q = (x_1^q, y_1^b, x_2^q, y_2^q)$ be the rectangle which is the union of the two query rectangles above and let k_a and k_b be the number of points of P in B_q above and below the query segment, respectively. A query takes $O(\sqrt{m} + k_a + k_b)$ time. Thus, theoretically the weakness of our approach is that the query time includes k_a which is the size of the spurious part of the output. However, in practice the structure runs so quickly that this overhead is negligible.

B. Constructing the homotopy-augmented visibility graph

In order to construct the homotopy-augmented visibility graph $\mathcal{G}_h^{\text{vis}}$, we first construct the reduced visibility graph $\mathcal{G}^{\text{vis}} = (\mathcal{V}^{\text{vis}}, \tilde{\mathcal{E}}^{\text{vis}})$ over the set of obstacles while including the base point p_b as an additional vertex. Given the reduced visibility graph one can compute the homotopy-augmented visibility graph $\mathcal{G}_h^{\text{vis}}$ as described in Section II.

[1]The $\widetilde{O}$ notation absorbs polylog(m) and m^{ε} factors for a given $\varepsilon > 0$.

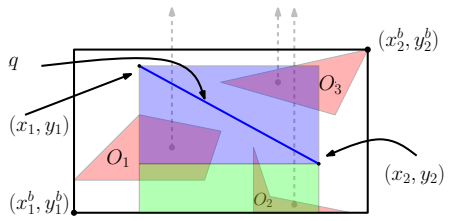


Fig. 4. Querying the kd-tree. Given a query segment q with endpoints (x_1, y_1) and (x_2, y_2), we find all points below q by using two range-queries over the set of points induced by the obstacles. The first range is defined over the bounding box of the endpoints of q (shaded in blue) while the second is the rectangle below the first (shaded in green). Spurious points need to be filtered out from the result of the first query.

C. Answering a query

A query is given in the form of two points p_s, p_t and the h-invariant w_s describing the tethered placement at p_s. We note that this step differs slightly from the query performed by Kim et al. as they assume that p_s and p_t are vertices of the grid-based homotopy-augmented graph.

In order to compute a path (if one exists) between p_s and p_t with an original tether placement defined by w_s, we need to traverse the homotopy-augmented graph $\mathcal{G}_h^{\text{vis}}$. To do so, we add a vertex to $\mathcal{G}_h^{\text{vis}}$ corresponding to the source node and a set of vertices corresponding to all the possible target nodes (as there may be many h-signatures at p_t). Additionally, we need to add edges from the newly introduced nodes to the vertices of the graph. We note that once the nodes are added to the graph, the cost encoded in each node of $\mathcal{G}_h^{\text{vis}}$ is irrelevant as it is only used to ensure that the tether length is bounded by L. Thus, after describing the way to compute each new node we will refer to all nodes as pairs (v, w_v) while omitting the cost.

Let $\tilde{E}_s^{vis}$, $\tilde{E}_t^{vis}$ be all the reduced visibility edges between p_s, p_t and $\mathcal{V}^{\text{vis}}$, respectively. We first add the vertex corresponding to the source, namely (p_s, w_s). To compute the vertices corresponding to the target, we compute H_t, the set of all the h-signatures that can be used to reach p_t. Namely, for every vertex $(v, w_v, c_v) \in \mathcal{V}_h^{\text{vis}}$ such that $(v, p_t) \in \tilde{E}_t^{vis}$, if $c_v + \|vp_t\| \leq L$ then $s(w_v \cdot h(v, p_t))$ is added to H_t. Finally, for each h-signature $w_t \in H_t$ we add the node (p_t, w_t).

The nodes newly introduced to $\mathcal{G}_h^{\text{vis}}$ induce new edges as well, similar to the ones described in Subsection II-E. Namely, for every node (p, w_p) such that $w_p = s(w_s \cdot h(p_s, p))$, we test if there exists a reduced visibility edge $(p_s, p) \in \tilde{E}_s^{vis}$. If so, an edge is added between (p_s, w_s) and (p, w_p). Similarly for every node (p, w_p), we test if there exists a reduced visibility edge $(p, p_t) \in \tilde{E}_t^{vis}$ and add an edge between (p, w_p) and (p_t, w_t) if $w_t \in H_t$ and $w_t = s(w_p \cdot h(p, p_t))$. Finally, if the source and target points p_s, p_t are mutually visible, and $s(w_s \cdot h(p_s p_t)) \in H_t$, we add an edge between (p_s, w_s) and $(p_t, s(w_s \cdot h(p_s p_t)))$.

Once nodes corresponding to the source and target points are added as vertices to $\mathcal{G}_h^{\text{vis}}$ together with their appropriate edges, we run an A* search from the source node and terminate the search once we reach a node corresponding to the target point. As an admissible heuristic for A* we use the Euclidean distance between the node and p_t.

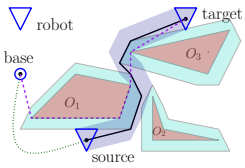


Fig. 5. Shortest path for a tethered triangular robot R. Obstacles are depicted in red, inflated obstacles ($\bigcup_{O\in\mathcal{O}} O \oplus -R$) are depicted in turquoise, initial and final placements of the tether are drawn in dotted green and dashed purple, respectively. Finally, the path traversed by the center of the robot is drawn in black while the area swept by the robot along this path is depicted in blue. Notice the difference in the paths traversed by the center of robot and by the tether.

D. Computational complexity

Recall that $\mathcal{O}$ is the set of m obstacles having a total of n vertices, P is the set of m points, one in each obstacle, from which we extend the set of rays. The preprocessing time consists of (i) building the visibility graph, which takes $O(|\mathcal{E}^{\text{vis}}| + n\log n)$ time, (ii) building the kd-tree KD(P), which takes $O(m\log m)$ time, and (iii) constructing the homotopy-augmented visibility graph $\mathcal{G}_h^{\text{vis}}$, which consists of running Dijkstra's algorithm and for each edge querying KD(P). This takes $O(|\mathcal{E}_h^{\text{vis}}| + |\mathcal{V}_h^{\text{vis}}|\log|\mathcal{V}_h^{\text{vis}}| + \kappa)$ where κ denotes the total time it takes to answer all the range search queries together. Note that in the worst case κ can be $O(|\mathcal{E}_h^{\text{vis}}|m)$. Thus, assuming that $|\mathcal{E}_h^{\text{vis}}| \geq \max(|\mathcal{E}^{\text{vis}}|, n\log n)$ the total construction time is $O(\kappa + |\mathcal{V}_h^{\text{vis}}|\log|\mathcal{V}_h^{\text{vis}}|) = O(|\mathcal{E}_h^{\text{vis}}|m + |\mathcal{V}_h^{\text{vis}}|\log|\mathcal{V}_h^{\text{vis}}|)$.

The query time is dominated by (i) the time required to add edges from the source and target vertices to $\mathcal{G}_h^{\text{vis}}$, which requires $O(|H_t|)$ queries to KD(P), and (ii) traversing $\mathcal{G}_h^{\text{vis}}$ using the A* algorithm.

Remark One may expect the computational complexity to be described only as a function of n, the complexity of the obstacles. However, the complexity of the problem is also directly related to the length L of the tether. The homotopy augmented graph $\mathcal{G}_h^{\text{vis}}$ encodes both the complexity of the obstacles and the length of the tether. This is why we describe the computational complexity as a function of $\mathcal{G}_h^{\text{vis}}$.

IV. Planning for a polygonal tethered robot

We use the homotopy-augmented visibility graph to efficiently plan a path for a translating polygonal tethered robot among polygonal obstacles. We describe how to plan the *shortest path* and continue to discuss a straightforward extension that allows to ensure a predefined clearance between the robot and the obstacles.

A. Shortest path for a polygonal tethered robot

It is important to notice that the shortest path traversed by a polygonal robot differs from the path traversed by the tether: While the shortest path that the robot takes lies in the reduced visibility graph of the *inflated obstacles*, the tether takes a path that lies in the reduced visibility graph of the obstacles (see Fig. 5).

Each vertex in the path traversed by the polygonal robot needs to be a valid placement for the robot (i.e., it should not collide with the obstacles) and the for the tether (i.e., its distance from the base should be less than L). Additionally, the tether may "graze" the obstacles (similar to the point-robot case). Thus, in order to compute the shortest path for the polygonal robot, we compute the Minkowski sum $M = \bigcup_{O\in\mathcal{O}} O\oplus -R$ of a reflection of R through the origin and the obstacles. We then compute the visibility graph defined over the obstacles together with the vertices of the the Minkowski sum M and continue to compute the homotopy-augmented visibility graph $\mathcal{G}_h^{\text{vis}}$ defined over this visibility graph.

In the query phase, only the vertices of M are used in the search from p_s to p_t on $\mathcal{G}_h^{\text{vis}}$ (see Section III).

B. Ensuring minimal clearance

Planning using the Minkowski sum $M = \bigcup_{O\in\mathcal{O}} O \oplus -R$ as described in Section IV-A causes the robot to "graze" the obstacles. This is often undesirable due to safety requirements and the planned path should have a certain *clearance*—a minimal distance between the robot and the obstacles. To ensure a clearance c between the robot and the obstacles at all times, one needs to inflate the robot by a disk of radius c and plan a path for the inflated robot.

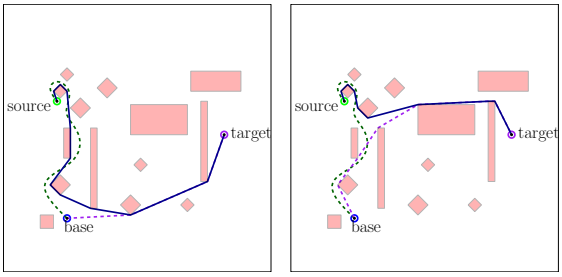
Thus, in order to plan a shortest path for a polygonal tethered robot with clearance c one needs to replace the robot R with an inflated robot $R \oplus D(c)$ where $D(c)$ is a disk of radius c. Note that the Minkowski sum $M(c) = \bigcup_{O\in\mathcal{O}} O \oplus -(R \oplus D(c))$ now has circular arcs on its boundary. This, in turn, should be accounted for when constructing of the visibility graph. As mentioned in Section II the visibility graph of $M(c)$ may be constructed in $O(n^2\log n)$ time. Once the visibility graph has been constructed, the rest of the algorithm remains the same.

V. Evaluation

The code for all experiments was written in C++ using CGAL [23] and run on a laptop with an Intel Core i7 processor clocked at 2.8 GHz equipped with 8GB of RAM.

We start by reporting on simulations for a point robot. Fig. 6 depicts a scenario similar to the one used by Kim et al. together with two possible solutions for two different tether lengths. The upper part of Table I reports on the size of $\mathcal{G}^{\text{vis}}$ and $\mathcal{G}_h^{\text{vis}}$, the preprocessing times and the time to perform one query. Note that although the scenario is not identical to, it is very similar to one used by Kim et al. For their scenario, they report on tens and hundreds of seconds for the preprocessing times of the short and long tether while the times used by our implementation are in the order of hundreds of milliseconds. This is not surprising as they report on expanding millions of vertices in the preprocessing stage while our algorithm uses less than 25,000 vertices even for the case of a long tether. We chose to compare our implementation with the results by Kim et al. because it is the only approach, to the extent of our knowledge, which performs a preprocessing stage to more efficiently solve problem instances at the query phase.

Moving to a polygonal robot, Fig. 7 depicts the same scenario as in Fig. 6 but planning for a diamond-shaped robot. Fig. 8 depicts two queries in a different scenario with a



(a) Short tether (320 units). (b) Long tether (420 units).

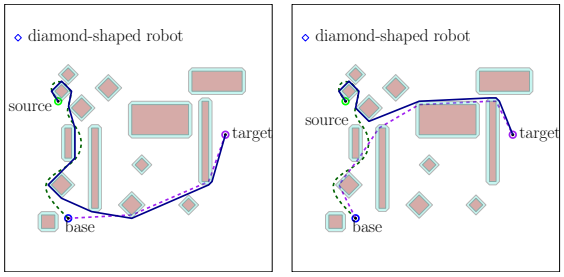
Fig. 6. Motion planning for a tethered point robot. Obstacles are depicted in red, initial and final placements of the tether are shown in dashed green and purple, respectively. The shortest path traversed by the robot is drawn in dark blue. Scenario based on [1, Fig. 9].

TABLE I

Statistics for scenarios. pnt and pgn denote a point and polygonal robot, respectively. Recall that $\mathcal{G}^{\text{vis}} = (\mathcal{V}^{\text{vis}}, \mathcal{E}^{\text{vis}})$, $\mathcal{G}_h^{\text{vis}} = (\mathcal{V}_h^{\text{vis}}, \mathcal{E}_h^{\text{vis}})$ denote the visibility graph and homotopy-augmented visibility graph, respectively. Finally, t_p and t_q denote the preprocessing time and time to perform one query (in milliseconds), respectively.

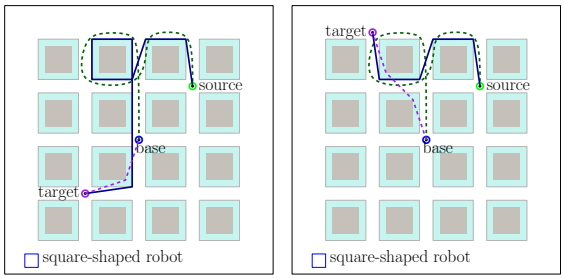
Setting	$\lvert\mathcal{G}^{\text{vis}}\rvert$		$\lvert\mathcal{G}_h^{\text{vis}}\rvert$		Time [msec]	
	$\mathcal{V}^{\text{vis}}$	$\mathcal{E}^{\text{vis}}$	$\mathcal{V}_h^{\text{vis}}$	$\mathcal{E}_h^{\text{vis}}$	t_p	t_q
Fig. 6a (pnt)	57	766	2,508	11,620	69	41
Fig. 6b (pnt)	57	766	23,849	113,362	712	56
Fig. 7a (pgn)	153	5,890	5,486	70,478	312	61
Fig. 7b (pgn)	153	5,890	50,031	656,468	4,463	240
Fig. 8a (pgn)	129	5,752	4,344	63,316	239	13
Fig. 8b (pgn)	129	5,752	4,344	63,316	239	12

square-shaped robot. The lower part of Table I reports on the size of the visibility graph, the visibility-based homotopy-augmented graph, the preprocessing time and the time to perform a query for both scenarios.



(a) Short tether (320 units). (b) Long tether (420 units).

Fig. 7. Motion planning for a tethered diamond-shaped robot depicted in blue (top-left corner of the figure). Obstacles and inflated obstacles are depicted in red and turquoise, respectively. Initial and final placements of the tether are shown in dashed green and purple, respectively. The shortest path traversed by the robot is drawn in dark blue. Scenario based on [1, Fig. 9].



(a) (b)

Fig. 8. Two queries for the motion of a tethered square-shaped robot (depicted in blue in the bottom-left corner of the figure) with the same initial placement of the tether. Obstacles and inflated obstacles are depicted in red and turquoise, respectively. Initial and final placements of the tether are shown in dashed green and purple, respectively. The shortest path is drawn in dark blue.

References

[1] S. Kim, S. Bhattacharya, and V. Kumar, "Path planning for a tethered mobile robot," in *ICRA*, 2014, pp. 1132–1139.

[2] R. Christ and R. Wernli, *The ROV Manual: A User Guide for Remotely Operated Vehicles*. Elsevier Science, 2013.

[3] K. S. Pratt, R. R. Murphy, J. L. Burke, J. Craighead, C. Griffin, and S. Stover, "Use of tethered small unmanned aerial system at Berkman plaza II collapse," in *Safety, Security and Rescue Robotics*, 2008, pp. 134–139.

[4] J. Pearl, *Heuristics: Intelligent Search Strategies for Computer Problem Solving*. Addison-Wesley, 1984.

[5] S. Hert and V. J. Lumelsky, "The ties that bind: Motion planning for multiple tethered robots," *Robotics and Autonomous Systems*, vol. 17, no. 3, pp. 187–215, 1996.

[6] ——, "Motion planning in R^3 for multiple tethered robots," *IEEE T. Robotics and Automation*, vol. 15, no. 4, pp. 623–639, 1999.

[7] P. G. Xavier, "Shortest path planning for a tethered robot or an anchored cable," in *ICRA*, 1999, pp. 1011–1017.

[8] N. Xu, P. Brass, and I. Vigan, "An improved algorithm in shortest path planning for a tethered robot," in *22nd Annual Fall Workshop on Computational Geometry*, 2012.

[9] T. Igarashi and M. Stilman, "Homotopic path planning on manifolds for cabled mobile robots," in *WAFR*, 2010, vol. 68, pp. 1–18.

[10] S. Bhattacharya, M. Likhachev, and V. Kumar, "Topological constraints in search-based robot path planning," *Autonomous Robots*, vol. 33, no. 3, pp. 273–290, 2012.

[11] S. Kim, S. Bhattacharya, H. K. Heidarsson, G. Sukhatme, and V. Kumar, "A topological approach to using cables to separate and manipulate sets of objects," in *RSS*, 2013.

[12] I. Shnaps and E. Rimon, "Online coverage by a tethered autonomous mobile robot in planar unknown environments," *IEEE Transactions on Robotics*, vol. 30, no. 4, pp. 966–974, 2014.

[13] ——, "On-line coverage of planar environments by a battery powered autonomous mobile robot," in *WAFR*, 2014.

[14] P. Abad-Manterola, I. A. D. Nesnas, and J. W. Burdick, "Motion planning on steep terrain for the tethered axel rover," in *ICRA*, 2011, pp. 4188–4195.

[15] M. M. Tanner, J. W. Burdick, and I. A. D. Nesnas, "Online motion planning for tethered robots in extreme terrain," in *ICRA*, 2013, pp. 5557–5564.

[16] M. d. Berg, O. Cheong, M. v. Kreveld, and M. Overmars, *Computational Geometry: Algorithms and Applications*, 3rd ed. Springer, 2008.

[17] S. K. Ghosh and D. M. Mount, "An output-sensitive algorithm for computing visibility graphs," *SIAM Journal on Computing*, vol. 20, no. 5, pp. 888–910, 1991.

[18] R. Wein, J. van den Berg, and D. Halperin, "The visibility-Voronoi complex and its applications," *Computational Geometry: Theory and Applications*, vol. 36, no. 1, pp. 66–78, 2007.

[19] K. Kedem, R. Livne, J. Pach, and M. Sharir, "On the union of jordan regions and collision-free translational motion amidst polygonal obstacles," *Discrete Comput. Geom.*, vol. 1, pp. 59–71, April 1986.

[20] P. K. Agarwal, E. Flato, and D. Halperin, "Polygon decomposition for efficient construction of minkowski sums," *Comput. Geom.*, vol. 21, no. 1-2, pp. 39–61, 2002.

[21] S. M. LaValle, *Planning Algorithms*. Cambridge University Press, 2006.

[22] D. Grigoriev and A. Slissenko, "Polytime algorithm for the shortest path in a homotopy class amidst semi-algebraic obstacles in the plane," in *International Symposium on Symbolic and Algebraic Computation*, 1998, pp. 17–24.

[23] The CGAL Project, *CGAL User and Reference Manual*, 3.9 ed. CGAL Editorial Board, 2011, http //www.cgal.org/.