2020 IEEE/RSJ International Conference on Intelligent Robots and Systems (IROS)
October 25-29, 2020, Las Vegas, NV, USA (Virtual)

Exploration of unknown environments with a tethered mobile robot

Danylo Shapovalov and Guilherme A. S. Pereira

***Abstract*— This paper presents a tangle-free frontier based exploration algorithm for planar mobile robots equipped with limited length and anchored tethers. After planning a path to the closest point in the frontier between free and unknown space, the robot computes an estimate of the future length of its tether and decides, by comparing the anticipated length with the minimum possible tether length, whether the path should be followed or not. If the anticipated tether is longer than the minimum tether by a function of the expected radius of the obstacles, a path planner with homotopic constraints is used to plan a path that brings the robot tether to the same homotopy class of the shortest tether. This behavior will not only limit the tether length but also will prevent tether entangling on the obstacles of the environment. We evaluate our method in different simulated environments and illustrate the approach with an actual tethered robot.**

I. Introduction

Exploration of unknown environments is one the fundamental problems in mobile robotics. The main goal in this task is to quickly cover the environment with the "footprint" of the robot's sensors, which may be cameras, sonars, or LiDARs. With the data obtained during the exploration, the robot is then able to construct a map of the environment, usually using simultaneous localization and mapping (SLAM) approaches. Although several exploration solutions for untethered robots exist [1], no solutions have be found for tangle-free exploration with tethered vehicles. In this paper we solve the exploration motion planning problem with a single tethered robot. We assume a planar robot equipped with a retractile and limited tether. This problem is inspired by space [2] or underwater [3] exploration missions where small exploratory vehicles are connected to a fixed base through a power and/or communication cable.

Path planning for tethered planar robots with finite-length and anchored tethers was first considered in [4]. The author presents a provable correct algorithm that uses a visibility graph of a previously known polygonal environment to find the shortest path between two points that respects the tether length constraint. The tether is assumed to be very flexible and to be always in a taut condition. More recently, the authors of [5] presented another efficient solution for the problem. They build a graph that encodes the possible homotopy classes of the tether and search for the shortest path on this graph. By the definition of homotopy, two tether configurations are considered to be in the same homotopy class if they can be continuously deformed into the other without intersecting any obstacle. An interesting characteristic of the graph proposed in [5] is that it differentiates situations where the robot is in the same exact location but have different tether configurations.

This research was made possible by the NASA Established Program to Stimulate Competitive Research, Grant #80NSSC19M0054, and West Virginia University Statler College.

The authors are with the Department of Mechanical and Aerospace Engineering, West Virginia University, Morgantown, WV 26505. (e-mails: ds0144@mix.wvu.edu, guilherme.pereira@mail.wvu.edu)

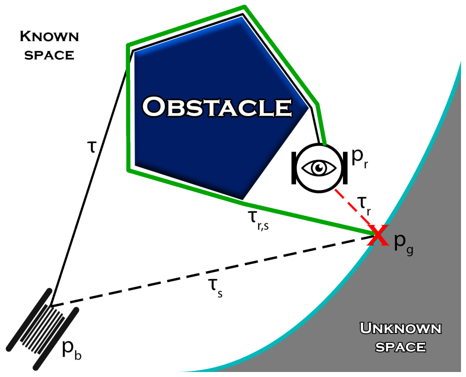

Fig. 1. Visual representation of our approach — the tethered robot in its current position p_r picks a path $\tau_{r,s}$ to reach a position p_g in the frontier between known and unknown space to minimize the tether length and avoid future tangling. Notice that $\tau + \tau_{r,s}$ is homopotic to the shortest path τ_s between the base, p_b, and p_g.

The authors of [6] solved the same problem proposed in [5]. They used a very similar idea but, instead of using only the distance from the anchor point to identify homotopy, which may fail in a few situations detected by the authors, they propose an elegant and simple homotopy invariant to construct a graph called homotopy augmented graph [7]. In this graph, each vertex has information of both its workspace position and the homotopy class of the path that starts at the anchor of the tether and passes through the vertex. Once the graph is constructed, it can be used to plan tangling-free paths between any two points in the workspace.

It is important to mention that all the literature cited above presented deliberative planners that assume a complete knowledge of the environment. Thus, although the tools developed in these papers are very useful for the general area of motion planning for tethered robots, the proposed algorithms cannot be directly used for exploration, which is usually performed in a reactive or greedy fashion [1]. In exploration, the robot usually plans only a few steps ahead, by determining the motion that will provide more information about the environment. A classic way to do exploration is based on frontiers [8]. In this approach a robot increases its knowledge about the environment by moving to points in the frontiers, which represent the boundaries between free space and unexplored space.

Although, by the authors knowledge, there is no previous literature that solves the motion planning exploration problem with tethered robots, at least two works present on-line coverage algorithms for tethered robots in unknown environments [9], [10]. In this sense, coverage is the problem of determining a path that passes over all points of the workspace. When the environment is unknown, this problem is closely related to exploration and can be viewed as an exploration problem with a sensor of very small field-of-view. Therefore, one could directly use coverage algorithms [9], [10] to explore an environment with a tethered robot. The only drawback of doing this would be time, since the robot path necessary to completely cover an environment is, for most environments, much longer than the one necessary to explore one. Thus, there is still a need for strategies that efficiently explore an environment with tethered robots, as the one presented in this paper.

The main contribution of this paper is an exploration algorithm to be used by tethered mobile robots with guaranteed tangle-free global paths. Our algorithm uses frontier-based exploration to select the next best view for the robot. To account for the tether, the algorithm keeps track of the tether configuration, including its homotopy. With this information, the robot can decide on-line what is the best motion to take to both comply with the length of the tether and avoid tangling around the objects of the environment. An illustration of our approach is shown in Fig. 1. The presented algorithm leverages on the homotopy invariant proposed in [6] and on path planning algorithms with homotopy constraints, such as [11], [12]. Those algorithms enforce that the planned trajectory belongs to a specified homotopy class. In the experiments of this paper we use a simple implementation of RRT with homotopy constraints.

The rest of this papers is divided as follows. We formally specify our problem in the next section. Section III introduces the concepts of homotopy and homotopy invariant, used in our exploration algorithm. This algorithm is presented and analysed in Sect. IV. Simulations in sparse and dense environments, along with real-robot experiments are shown in Sect. V. Conclusions and proposals of continuity are presented in Sect. VI.

II. Problem Definition

Consider a tethered planar robot in an environment $\mathcal{W} \subset \mathbb{R}^2$ populated with unknown obstacles. The robot is equipped with a sensor whose field-of-view (footprint) is a circle of radius r centered at the robot position. The tether is anchored to a fixed base at position $p_b = (x_b, y_b)$, where the robot will also start its motion. The tether is retractable, so it is taut most of the time, and has a variable length L, which is limited to a maximum length L_{max}. The task of the robot is to explore the environment, covering it as fast as possible with its sensor's footprint and building a map with the obstacles found. Notice that, because the robot tether is limited in length, regions of the environment that are relatively far from the base cannot and are not supposed to be explored. Assuming that a path is the continuous function $\tau : [0, 1] \rightarrow \mathbb{R}^2$, our problem is to find a short global non-entangling path for the robot so that, given the tether length constraint, its environment is explored. A non-entangling path is defined as follows.

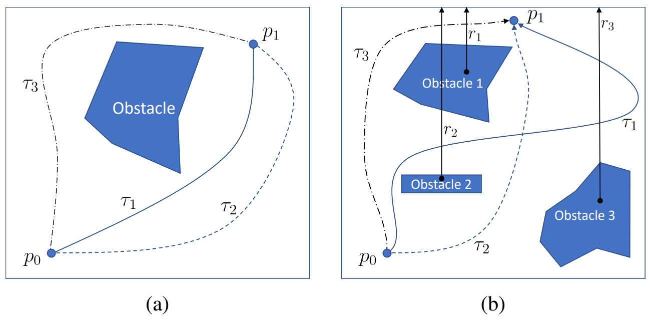

Fig. 2. (a) Path homotopy: paths τ_1 and τ_2 between p_0 and p_1 are homotopic while path τ_3 belongs to a different homotopy class. (b) h-signature: The homotopy class of a path can be uniquely identified by a word formed by the label of the rays emanating from each obstacle it crosses. In this way, the homotopy class of path τ_3 is identified by $h(\tau_3) = r_2 r_1$; the class of τ_2 is $h(\tau_2) = \emptyset$; and $h(\tau_1) = r_2$. In the case of τ_1, although the original invariant is "$r_2 r_3 r_3^{-1}$", because the path crosses r_3 twice and consecutively in different directions, it can be reduced to "r_2".

Definition 1: Considering a planar robot equipped with a retractable tether in an environment with finite size obstacles, a path $\tau(c)$, $0 \leq c \leq 1$, which crosses itself at positions $p_k \in \mathbb{R}^2$, $k = 1, 2, \ldots, n$, i.e. $\tau(c_i) = \tau(c_j) = p_k$ for $0 \leq c_i \leq 1$, $0 \leq c_j \leq 1$ and $c_i \neq c_j$, is a non-entangling path if and only if $L(c_i) = L(c_j)$, where $L(c)$ is the length of the tether in function of the path parameterization variable c.

By the previous definition, a robot performs a non-entangling or tangle-free path if, for example, it starts and finishes its motion at the same position without tangling its tether in any obstacle of the environment. Next section presents a background on techniques that are useful to understand the proposed methodology, presented in Sect. IV.

III. Background

Path planning for tethered robots is considered to be a topological path planning problem. A key notion in topological planning is homotopy. In this sense, two paths starting and finishing in the same workspace position belong to the same homotopy class if one can be transformed into the other through a continuous set of deformations without intercepting any obstacle in the environment. Two paths that belong to the same homotopy class are said to be homotopic. Figure 2(a) shows examples of homotopic and non-homotopic paths.

Invariants, called h-signatures, that can be used as a unique identification of a homotopy class of a path or curve are proposed by the authors of [6]. Using the notion of h-signature of paths, we can say that path τ_i is homotopic to τ_j if $h(\tau_i) = h(\tau_j)$, where $h(\cdot)$ is a function that computes the h-signature of a path. Figure 2(b) presents illustrative examples of h-signatures. Basically, if we assume that a series of parallel rays emanate from the obstacles as show in Fig. 2(b), a word is constructed by concatenating the labels of the rays in the order they are crossed by the path. If the ray is crossed from right to left, its label is added to the word with superscript "-1", which can be considered to be the "inverse" operator. After the word is complete, it can be reduced by cancelling consecutive labels with different superscripts. In Fig. 2(b), for example, the original word for $h(\tau_1)$ would be "$r_2 r_3 r_3^{-1}$". After reduction it becomes "r_2", which is invariant for all paths homotopic to τ_1.

Important properties of the h-signature are concatenation and inverse. Consider, for example, the existence of two paths, τ_i and τ_j with h-signatures $h(\tau_i)$ and $h(\tau_j)$, respectively. If the end of τ_i coincides with the beginning of τ_j, i.e. $\tau_i(1) = \tau_j(0)$, so the h-signature of the composition of both paths, $\tau_i + \tau_j$, is given by the concatenation of $h(\tau_i)$ and $h(\tau_j)$, i.e. $h(\tau_i + \tau_j) = h(\tau_i) \Diamond h(\tau_j)$, where "$\Diamond$" is the concatenation operator. On the other hand, the h-signature of a path followed in the opposite direction is the inverse of the original h-signature, i.e. $h(-\tau_i) = h(\tau_i)^{-1}$. It is important to mention that the inverse of h-signature "$r_2 r_3$", for example, is given by "$(r_2 r_3)^{-1} = r_3^{-1} r_2^{-1}$". More details on the h-signature algebra can be found in [6].

IV. Methodology

A. Basic Idea

The proposed algorithm for exploration with tethered robots uses traditional frontier-based exploration [8], which works by sequentially moving towards the closest point at the edge between known and unknown space (frontier) until there are no more frontiers to explore. In our case, the algorithm also stops if the tether length prevents the robot to reach any frontier. To avoid tangling while exploring, the algorithm keeps track of the size and homotopy of the tether. At each step, after the closest frontier to move is selected using simple Euclidean distance as a metric, two paths are computed: a shortest path from the base (τ_s), which would represent the shortest tether; and the shortest path from the robot's current position to the goal (τ_r). The second path is concatenated to the current robot path (τ), which represent the current tether approximation, and is optimized to generate a prediction of the tether the robot would have if it had took that path. The length of both tethers (shortest and predicted tethers) are then compared. If the second one is longer by more than a tolerance value (ΔL) or it exceeds the tether's maximum physical length ($L_{\max}$), the algorithm computes a new path for the robot (τ_r) that is the shortest path in the homotopy class of the conservative path that, before exploring a region, returns to the base to avoid tangling. When the robot follows this path, its tether will assume the form of the shortest path from the base. This would make the retractable tether to have the smallest possible length.

B. Algorithm

Our approach for tether-aware exploration is formally presented in Algorithm 1, which requires both the current position of the robot, p_r, and the position of its base (i.e. the tether anchor), p_b, as well as the maximum allowed tether length, $L_{\max}$, and the length tolerance, ΔL. Line 1 initiates frontier ($\mathcal{F}$), free space ($\mathcal{C}_f$), obstacles ($\mathcal{O}$), and the current

Algorithm 1: Tether-aware exploration algorithm.

```
   input: $p_r$, $p_b$, $L_{\max}$, $\Delta L$
1  $\mathcal{F} \longleftarrow \mathcal{C}_f \longleftarrow \mathcal{O} \longleftarrow \tau \longleftarrow \emptyset$;
2  $[\mathcal{F}, \mathcal{C}_f, \mathcal{O}] \longleftarrow$ LookAround($p_r$, $\mathcal{F}$, $\mathcal{C}_f$, $\mathcal{O}$);
3  while $\mathcal{F} \neq \emptyset$ do
4      do
5          $p_g \longleftarrow$ SelectNearbyFrontier($p_r$, $\mathcal{F}$);
6          if $p_g = \emptyset$ then
7              return $\mathcal{C}_f, \mathcal{O}$;
8          end
9          $\tau_s \longleftarrow$ ShortestPath($p_b$, $p_g$, $\mathcal{C}_f$);
10     while $L(\tau_s) > L_{max}$;
11     $\tau_r \longleftarrow$ ShortestPath($p_r$, $p_g$, $\mathcal{C}_f$);
12     if $(L(\tau + \tau_r) - L(\tau_s)) > \Delta L$ or $L(\tau + \tau_r) > L_{max}$ then
13         $h^* \longleftarrow h(\tau)^{-1} \Diamond h(\tau_s)$;
14         $\tau_r \longleftarrow$ ShortestHPath($p_r$, $p_g$, $h^*$, $\mathcal{C}_f$, $\mathcal{O}$);
15     end
16     if $\tau_r = \emptyset$ then
17         $\tau_r \longleftarrow [\tau(-1), \tau(-2)]$;
18     end
19     $p_r \longleftarrow$ FollowPath($\tau_r$);
20     $\tau \longleftarrow \tau + \tau_r$;
21     $[\mathcal{F}, \mathcal{C}_f, \mathcal{O}] \longleftarrow$ LookAround($p_r$, $\mathcal{F}$, $\mathcal{C}_f$, $\mathcal{O}$);
22 end
23 return $\mathcal{C}_f, \mathcal{O}$;
```

tether approximation (τ) as empty sets. The map related variables are then passed to the "LookAround" function in line 2, which initiates the first view around the robot before proceeding to the main loop. Function "LookAround" processes the data of the robot's sensors and updates the map with frontier, free space, and obstacles. It basically traces out rays from the robot position (p_r) up to the end of the sensor field of view, ending either in an obstacle, a frontier, or nothing (if going through an explored area).

The main loop (Lines 3 to 22) continues until either there are no more frontiers to explore, or until no more frontiers can physically be reached with a given $L_{\max}$. The latter condition is being defined in lines 6 through 10, with line 7 explicitly breaking the loop and returning current known free space and obstacles in case no more frontiers could be physically reached. Line 5 selects the nearest point at the frontier as the next goal for the robot (p_g) by computing the shortest Euclidean distance between the current robot position (p_r) and the set of frontiers. Line 9 computes a shortest path from the base, p_b, to p_g, and line 11 computes the shortest path from the robot's position, p_r, to p_g. Line 12 checks if either ΔL or $L_{\max}$ constraints would be violated. It is important to notice that function $L(\cdot)$, which computes the length of a path, should be intelligent enough to simplify the path (without changing its h-signature) before computing its length. If the shortest path (τ_s) and predicted path ($\tau + \tau_r$) lengths are too different or the predicted path is too long, the algorithm calculates in Line 14 a path with homotopy h^* that would return the robot to the shortest tether configuration. For this, h^* would be the h-signature that represents the homotopy class of a path that would retreat to the base by following the tether (path $-\tau$, which is the reverse of τ) and then take the shortest path τ_s to p_g. As shown in Line 13 of the algorithm, h^* is computed using the properties of the h-signature presented in Section III as:

$$h^* = h(\tau)^{-1} \Diamond h(\tau_s) \,. \tag{1}$$

Notice that by following a path with h-signature h^* the robot does not return to the base but, the tether, which is retractable, will assume the same configuration as it would have if the robot had returned to the base. Line 17 is a fail-safe that initiates a single-point backtrack in case the previous path computations failed. In this line $\tau(-1)$ and $\tau(-2)$ are respectively the last and the second last waypoints of the current tether approximation. Subsequent lines control the robot to follow the computed path (Line 19), add that path to the total path record so it is used as a current tether approximation (Line 20), and update the map with information gathered at the robot's new position (Line 21).

C. Analysis

It is important to notice that Algorithm 1 is complete. In this sense, we consider that an exploration algorithm for tethered robots is complete if the robot not only explores all the area that it can physically reach with the tether, but also does so without tangling the tether around any obstacles. Because we are using the frontier-based exploration approach, the first requirement will always be satisfied by Algorithm 1 due to the nature of this approach — the robot will continue to pick goals on the frontier until it either runs out of frontiers to sample (all the area is successfully explored), or there will be no more frontiers it can physically reach with the tether (available area is successfully explored).

As for the second requirement, the path will be kept tangle-free provided that the length tolerance, ΔL, used as input of the algorithm, is correctly chosen. In the limit, as ΔL goes to zero, the algorithm would always conform to the shortest path to the goal from the base, leaving no room for tangling. This, however, is not a very efficient approach, and would require a lot of unnecessary backtracking towards the base. Otherwise, as ΔL goes to infinity, the robot would only be returning to the shortest path if the maximum tether constraint $L_{\max}$ would be violated. If $L_{\max}$ also approaches infinity, the robot would start ignoring the tether altogether. This is very efficient for an unconstrained robot, but with the tether it will highly increase the probability of tangling.

Thus, a good balance between these extremes is necessary for both efficient and tangle-free planning. If we assume a point robot and a circular obstacle of radius R, we can define the maximum value of ΔL to guarantee tangle-free path planning. Notice that the shortest possible length that will result in tangling would be larger than the full circumference around the obstacle, giving the maximum length tolerance for tangle-free paths to be $2\pi\mathrm{R}$. Any value less that this will still guarantee a tangle-free path, but very low values would trigger more often the necessity of moving backwards by following the paths in the homotopy class h^* (Lines 12 to 14 of our algorithm). Thus, we would want ΔL to be closer to $2\pi\mathrm{R}$, where R would be the radius of the smallest expected obstacle when this obstacle is represented in the robot's configuration space.

In fact, given the nature of this algorithm and frontier-based exploration, it could be argued that reducing the length tolerance all the way down to its theoretical guaranteed limit might be unnecessary, given that it's extremely unlikely for the sole path to be strictly around an obstacle — it will generally be far longer than that, triggering the shortest path conforming anyway.

Notice that Algorithm 1 does not rely on any specific path planner or map representations. Therefore, the computational complexity of the algorithm will vary and will depend on the choice of algorithm used. It is important to observe that the complexity of functions "ShortestPath", called at least two times, and "ShortestHPath", called eventually, will directly influence the complexity of the proposed approach. From our previous discussion, the larger ΔL is, the less times "ShortestHPath" will be called, yielding in a less complex algorithm. The map representation is also important to be discussed. It will not only affect the complexity of the path planning functions, but also will determine, in the loop of lines 4 to 10, how many times "ShortestPath" will be called to check if a point in the frontier is reachable by the tethered robot.

In the next section we present experiments that evaluate the proposed algorithm and illustrates some of the discussions of this section.

V. Experiments

This section presents simulations and real robot experiments that evaluate our approach. Algorithm 1 was implemented in Python. We used RRT for path planning (Lines 9 and 11 of Algorithm 1) and a modified version of RRT for planning with homotomic contraints (Line 14). In the latter, once we have a new random sample, we try to connect it to the current tree using only edges whose homotopy invariant symbols belong to the desired homotopy invariant, h^*. A similar idea was used in [11] for RRT*. It is important to mention that RRT is not an optimal planner and, therefore, does not necessarily compute the shortest path. So after the path is found we run a simple post-processing algorithm to reduce its size [13]. We noticed that this did not change the expected behavior of the proposed algorithm, although we plan to replace RRT by an optimal planner in future implementations. To map the environment, we used a grid where the cells could assume four states: obstacles, free space, frontier, and unexplored. For path planning, obstacle and unexplored cells are considered to be the same (obstacles) and all other cells are seen as free space. Notice that although the map is discrete, both versions of RRT work in the continuous free space.

A. Simulations

We evaluated the performance of Algorithm 1 in a simulated $20 \times 20\,\mathrm{m}$ environment with random obstacles. The map of the environment used a grid resolution of $0.5\,\mathrm{m}$. We have two situations: 15 to 20 obstacles with radius up to $2\,\mathrm{m}$;

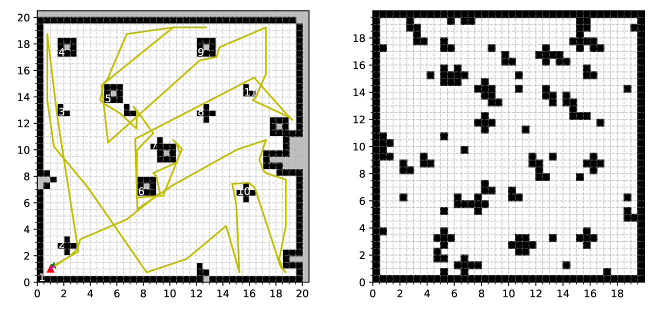

Fig. 3. Typical environments used in our simulations. (a) 15 to 20 random obstacles shown in black. The tangle-free path used by the robot to explore the environment using our method is shown in yellow. (b) 60 to 70 random obstacles shown in black.

TABLE I

SIMULATED RESULTS WITH THE ENVIRONMENT IN FIG. 3(A).

Tolerance ΔL **(m)**	**Tangle (%)**	**Time per iteration (s)**	**Max tether length (m)**	**Total path length (m)**
Backtrack	0	0.50 ± 0.09	26.48 ± 2.60	1181.4 ± 93.4
0	0	2.07 ± 0.56	33.18 ± 7.06	224.0 ± 60.2
2πR	0	2.04 ± 0.68	31.36 ± 4.45	216.6 ± 40.1
4πR	5 ± 2	2.13 ± 0.51	33.29 ± 3.96	196.4 ± 30.0
8πR	28 ± 4	2.32 ± 0.65	37.10 ± 2.50	199.3 ± 25.8
16πR	43 ± 5	2.32 ± 0.71	39.30 ± 2.01	215.1 ± 49.0
∞	73 ± 4	2.56 ± 0.77	39.77 ± 0.80	225.6 ± 53.5
∞ (no limit)	93 ± 3	2.17 ± 0.49	78.83 ± 14.85	129.3 ± 15.3

60-70 obstacles with radius up to 0.5 m. Typical examples of these situations are show in Fig. 3(a) and (b), respectively. All simulations were executed using an AMD Ryzen 9 3950x processor clocked at 4100 MHz on a computer running Windows 10.

Tables I and II show numerical values of simulations for both kinds of environments. In these simulations the tether length L_{max} was set to be 40 m. In the tables, we compare our method with a conservative exploration strategy that avoids tangling by always retreating along the tether back to the base before exploring a new region (Backtrack), and a standard frontier exploration that ignores the tether (∞, no limit). This case was simulated by making the length tolerance and tether length to be very large in our algorithm ($\Delta L = L_{\text{max}} = \infty$). We also evaluate the effect of parameter ΔL, which is varied in function of the minimum obstacle radius R. The results shown in the tables were obtained as average and standard deviation of 40 exploration runs per line.

Backtrack, besides guaranteeing tangle-free paths, resulted in the longest total path and required the smallest tether length, as expected. As discussed in Section IV-C, our method resulted in tangle free paths for $\Delta L \leq 2\pi$R, with a still small tether length and paths five times shorter than the ones obtained by backtracking. As ΔL increases, the maximum recorded tether length approaches L_{max}. This happens because the max tether trigger (Line 12 of the algorithm, second condition) generally overtakes the tolerance trigger (Line 12, first condition), also resulting in progressively more tangling. An interesting phenomenon could also be observed in the total path length — while it starts decreasing at first,

TABLE II

SIMULATED RESULTS WITH THE ENVIRONMENT IN FIG. 3(B).

Tolerance ΔL **(m)**	**Tangle (%)**	**Time per iteration (s)**	**Max tether length (m)**	**Total path length (m)**
Backtrack	0	0.56 ± 0.07	27.79 ± 4.89	1504.4 ± 267.9
0	0	2.60 ± 0.52	28.53 ± 3.21	350.4 ± 58.8
2πR	0	2.49 ± 0.52	31.07 ± 3.63	283.8 ± 42.8
4πR	28 ± 4	2.57 ± 0.63	33.13 ± 3.49	267.1 ± 37.0
8πR	80 ± 4	2.95 ± 0.62	38.46 ± 1.44	283.1 ± 40.6
16πR	80 ± 4	3.09 ± 0.59	39.76 ± 0.21	312.9 ± 60.8
∞	88 ± 3	3.29 ± 0.54	39.86 ± 0.14	329.9 ± 61.9
∞ (no limit)	100	3.26 ± 0.72	108.97 ± 34.89	159.9 ± 34.9

reaching its minimum at around $\Delta L = 4\pi$R, it then goes back up, reaching around the same value at $\Delta L = \infty$ as it did at $\Delta L = 0$. This is explained by increased length tolerance ΔL causing the robot to stray increasingly farther from the shortest path (possibly tangling in the process), requiring a more substantial backtrack in order to return to the minimum tether length situation. This also explains the correlation between increasing length tolerance ΔL and increasing computation time. A path that has strayed too far from the optimum generally has more homotopic constraints associated with the return path, causing the modified RRT implementation (Line 14 in the algorithm) to spend more time to find a path constrained to h^*. While the total path values in Table I are all still technically within the margin of error of each other, this phenomenon is a lot more prominent in Table II, obtained with the denser environment. The exception is the last entry, when the path length reduced dramatically because the tether length constraint was removed, which also resulted in a near-100% tangling rate.

B. Actual robot implementation

Our methodology was also tested with an iRobot's Create 2 robot using a VICON system for localization. The robot is controlled by a Netbook with Intel Core i3, 1.8GHz and 4 gigabytes of memory that runs Linux Ubuntu 18.04. A retractile clothesline was used to simulate the tether. For this demo, the robot sensor was simulated as a circle of 70 cm radius centered on the robot. The maximum tether length was set to be 4 m and the tolerance ΔL to be 0.8 m, which is smaller than the actual theoretical minimum of approximately 2 m and considers the radius of the robot (0.17 m) and of the smallest obstacle (0.15 m). Figure 4 shows a picture of the robot and Fig. 5 shows snapshots of the experiment while the robot explores a small 3×3 m environment with a circular and a rectangular object. Observe that the robot explores the entire environment and returns to the base by following a tangle-free global path. For the sake of comparison, Fig. 6 shows the final trajectory of the robot when it executes a standard frontier-based strategy ($\Delta L = L_{\text{max}} = \infty$). The environment is completely explored but the tether ended up tangled around the obstacles. A video with other demonstrations accompanies the paper and can be found at https://youtu.be/2lRHnf43T9g.

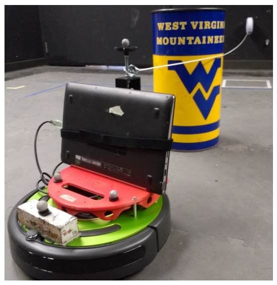

Fig. 4. Tethered robot used in the experiments of this paper.

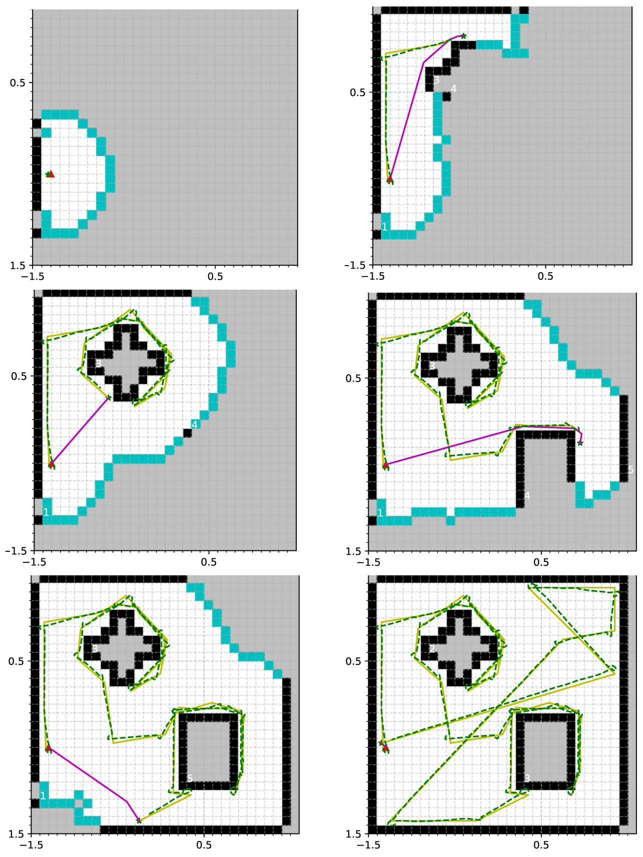

Fig. 5. Six snapshots of a real robot experiment. From left to right and top to bottom, the robot starts close to the base and maps the whole environment, returning to the base without tangling. The planned path is represented in yellow, the actual robot trajectory in green, and the tether estimate in purple. While the discovered free space is represented in white, light gray areas represent unknown space, cyan areas are the frontiers, and the obstacles are plotted in black.

VI. Conclusions

This paper presented a tether-aware exploration algorithm for tethered planar mobile robots. The algorithm is based on frontier based exploration and is guaranteed to provide tangle-free global paths. Our simulated results show that the length of the final robot path is only about 50% longer than the one provided by algorithms for robots without tether, and is approximately four to five times shorter than the trivial and conservative solution that always bring the robot to the base before it explores new regions of the workspace.

Future work will include the extension and implementation

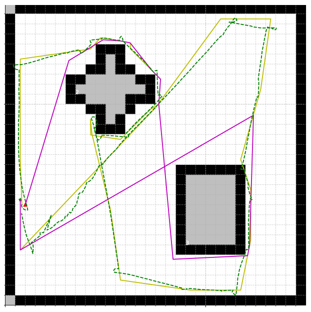

Fig. 6. Final configuration of a real robot experiment that uses standard frontier based exploration and ignores the presence of the tether. Notice that the tether estimate (represented in purple) circulates the obstacles when the robot returns to the base.

of the method to 3 dimensions, as our final goal is to work with tether-powered drones. Although the proposed algorithm itself can be directly used in 3D, homotopy invariants and path planning with homotopy constraints are not so simple in higher dimensions, specially when the obstacles are not known a priory. Those points will be considered for further investigation.

References

[1] D. Holz, N. Basilico, F. Amigoni, and S. Behnke, "Evaluating the efficiency of frontier-based exploration strategies," in *ISR 2010 (41st International Symposium on Robotics) and ROBOTIK 2010 (6th German Conference on Robotics)*, 2010, pp. 1–8.

[2] P. Abad-Manterola, I. A. D. Nesnas, and J. W. Burdick, "Motion planning on steep terrain for the tethered axel rover," in *IEEE International Conference on Robotics and Automation*, May 2011, pp. 4188–4195.

[3] J. Yuh, "Design and control of autonomous underwater robots: A survey," *Autonomous Robots*, vol. 8, no. 1, pp. 7–24, Jan 2000.

[4] P. G. Xavier, "Shortest path planning for a tethered robot or an anchored cable," in *IEEE International Conference on Robotics and Automation*, 1999, pp. 1011–1017.

[5] T. Igarashi and M. Stilman, "Homotopic path planning on manifolds for cabled mobile robots," in *Algorithmic Foundations of Robotics IX: Selected Contributions of the Ninth International Workshop on the Algorithmic Foundations of Robotics*, D. Hsu, V. Isler, J.-C. Latombe, and M. C. Lin, Eds. Springer Berlin Heidelberg, 2011, pp. 1–18.

[6] S. Kim, S. Bhattacharya, and V. Kumar, "Path planning for a tethered mobile robot," in *IEEE International Conference on Robotics and Automation*, May 2014, pp. 1132–1139.

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

[8] B. Yamauchi, "A frontier-based approach for autonomous exploration," in *IEEE International Symposium on Computational Intelligence in Robotics and Automation*, July 1997, pp. 146–151.

[9] 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, Aug 2014.

[10] L. S. R. Mechsy, M. U. B. Dias, W. Pragithmukar, and A. L. Kulasekera, "A novel offline coverage path planning algorithm for a tethered robot," in *International Conference on Control, Automation and Systems*, Oct 2017, pp. 218–223.

[11] D. Yi, M. A. Goodrich, and K. D. Seppi, "Homotopy-aware RRT*: Toward human-robot topological path-planning," in *2016 11th ACM/IEEE International Conference on Human-Robot Interaction*, March 2016, pp. 279–286.

[12] E. Hernandez, M. Carreras, and P. Ridao, "A comparison of homotopic path planning algorithms for robotic applications," *Robotics and Autonomous Systems*, vol. 64, pp. 44–58, 2015.

[13] H. M. Choset, S. Hutchinson, K. M. Lynch, G. Kantor, W. Burgard, L. E. Kavraki, and S. Thrun, *Principles of robot motion: theory, algorithms, and implementation*. The MIT Press, 2005.