robotics

Article

Tension-Aware Motion Planning for Tethered Robots

Rogério R. Lima and Guilherme A. S. Pereira *

Department of Mechanical, Materials and Aerospace Engineering, Benjamin M. Statler College of Engineering and Mineral Resources, West Virginia University, Morgantown, WV 26506, USA; rlima.rogerio@gmail.com
* Correspondence: guilherme.pereira@mail.wvu.edu

Abstract: This paper presents a path-planning approach for tethered robots. The proposed planner finds paths that minimize the tether tension due to tether–obstacle and tether–floor interaction. The method assumes that the tether is managed externally by a tether management system and pulled by the robot. The planner is initially formulated for ground robots in a 2D environment and then extended for 3D scenarios, where it can be applied to tethered aerial and underwater vehicles. The proposed approach assumes a taut tether between two consecutive contact points and knowledge of the coefficient of friction of the obstacles present in the environment. The method first computes the visibility graph of the environment, in which each node represents a vertex of an obstacle. Then, a second graph, named the tension-aware graph, is built so that the tether–environment interaction, formulated in terms of tension, is computed and used as the cost of the edges. A graph search algorithm (e.g., Dijkstra) is then used to compute a path with minimum tension, which can help the tethered robot reach longer distances by minimizing the tension required to drag the tether along the way. This paper presents simulations and a real-world experiment that illustrate the characteristics of the method.

Keywords: tethered robots; path planning; minimum tension

check for updates

Academic Editor: John Hedley

Received: 23 December 2024
Revised: 20 January 2025
Accepted: 25 January 2025
Published: 28 January 2025

Citation: Lima, R.R.; Pereira, G.A.S. Tension-Aware Motion Planning for Tethered Robots. *Robotics* **2025**, *14*, 11. https://doi.org/10.3390/robotics14020011

1. Introduction

One of the most challenging aspects of tethered robots is their reduced mobility compared to untethered vehicles. This is a concern because the robot's workspace typically contains obstacles that can restrict the robot's movement and reach, as the tether may become tangled or caught in an obstacle [1–3]. Consequently, efforts to prevent entanglements are well justified, as presented in [4], in which a framework of non-binary entanglement states was proposed to evaluate the likelihood of a tether becoming tangled. Another common approach to address this issue often involves finding paths to and from a specific location that are within the same homotopy class, which has been explored in various domains, including ground-based [5,6], underwater [7,8], and aerial [9] vehicles. A homotopy class comprises a set of paths in which each path can be continuously deformed into another without crossing any obstacles [10]. This concept is important for round-trip paths, as discussed in [11]. While homotopy-based tether problems form a critical component of path planning for cabled robots, they represent only one aspect of the broader challenges in this domain since practical applications, such as those discussed in [12,13], may present different tether-related issues. Multi-robot systems involving heterogeneous platforms connected by a tether, such as those in [14–16], for example, may require strict control of the tension and length of the tether to permit collaboration between the agents. Minimizing the length of the tether, as investigated in [17,18], is, in fact, the focus of several systems. While in real-world scenarios the tether length is inherently finite and reducing the length may help avoid entanglements [6], there are instances where obstacles in the environment can introduce additional challenges to a tether-based robotic mission, where even short tethers may create important constraints for the robot's motion.

The interaction between the tether and the obstacles can have significant implications and can sometimes be used for the robot's benefit. In some cases, the tether acts as both a constraint and an enabler for robots. For example, in [19], the tether–environment interaction was leveraged to enable tethered robots to traverse extreme terrains, such as steep slopes and vertical walls. The approach incorporates interaction points (anchors) between the tether and the terrain, dynamically predicting anchor attachments and detachments to ensure stability. Key aspects include on-demand traversability analysis that considers tether force, anchor history, and terrain–tether interactions, as well as a sampling-based planner that operates in a non-Markovian framework to generate safe and stable paths on 3D terrain.

A related application that exploits tether dynamics is the payload manipulation problem with tethered robots, as discussed in [20]. In this context, tethered agents use friction by allowing the cable to wrap around obstacles, thereby amplifying the holding force. Therefore, even small and weak tethered robots can leverage the capstan force to manipulate heavy loads. In this sense, the capstan effect refers to the tension in a rope necessary to maintain the equilibrium of a cable wrapped around a cylindrical object, preventing the rope from slipping or unwinding.

Using a similar idea to leverage the tether–environment interaction, the authors of [21] proposed a path-planning approach to generate cable tension via the capstan effect, allowing static equilibrium for the safe traversability of rappelling robots on extreme terrains. They used a $\mathcal{C}^1$-Tangent [22] graph for continuously differentiable ($\mathcal{C}^1$) path representation and employed a modified version of the Hybrid-state A* [23] algorithm to search for feasible paths. Friction is implicitly accounted for through the capstan effect (coefficient of friction is not explicitly used), which arises from the winding angle around obstacles. To ensure realistic operation, the algorithm includes a constraint on the winding angle, considering the finite tether length and a limited number of obstacles. The heuristic combines path length and winding angle in a convex cost function, weighted by a parameter. Depending on the parameter's value, the algorithm performs accordingly: when set to 0, it efficiently finds shorter paths (completeness not guaranteed); when set to 1, completeness is guaranteed. However, selecting an optimal value for this parameter can be challenging, as highlighted by the authors.

Friction, however, is undesirable for most tasks involving tethered robots, particularly those with an externally deployed tether (managed from the anchor point), which is the most common case. As the interaction with obstacles intensifies, so does the associated tension, which may lead to issues such as a higher force required from the robot to pull the cable and halting the vehicle if operation limits are exceeded. This is especially critical for flying robots with low thrust limits. Furthermore, for ground vehicles, an increased tether tension may lead to wheel slippage, which may damage their localization system. It is also important to note that higher levels of tether-obstacle friction correspond to increased wear of the tether. In such cases, minimizing cable–environment interaction is fundamental.

To minimize the effect of the environment on the cable, the authors of [24] proposed a tethering management system that mitigates potential snagging points resulting from tether–obstacle interactions. Their prototype was conceived to be agnostic, allowing seamless integration with any tethered ground vehicle without requiring specialized modifications. The system employs pulleys that envelop the tether and move in tandem to alleviate tension at points closer to the robot, where contact tension due to interactions tends to be most pronounced. Furthermore, the authors include a comprehensive analysis of tension resistance in real-world, non-idealized capstan objects, including those with circular and rounded square corners. A similar analysis of capstans in the environment, encompassing geologic (rock), living (trees), and built (fire hydrant) capstans, was presented in [20], based on a robotic experiment. Remarkably, the results indicate that the exponential relationship described by the capstan equation remains consistent for these objects, with variations primarily related to the coefficient of friction specific to each object. The findings of previous research provide insights for employing the capstan model when the tether bends around an obstacle with sufficient curvature in its corners. The problem resides in understanding the physical properties of the environment to quantify friction numerically. This information allows for modeling the environment based on the obstacle friction, enabling the identification of paths with the least amount of tension developed from tether–obstacle interaction. By finding low-tension paths (low friction levels), we can extend the robot's reachability to greater distances and safeguard the tether from premature abrasion.

Therefore, this paper proposes a new path-planning approach for tethered robots that aims to reduce tension resulting from the tether's interactions with obstacles when an external tether management system controls the tether pulled by the robot. The main contributions of this work include the creation of a tension-aware graph that incorporates a tension-based cost function, enabling the search for paths with minimal tension. This graph is built upon the visibility graph of the workspace by incorporating the edges of the visibility graph as the nodes of the tension-aware graph. This allows the use of both the bending angles of the tether when it wraps around obstacles and the cable segment between contact points to compute the two tension-based terms of the cost function. An overview of the method is shown in Figure 1. Although several previous works have proposed motion planners that deal with tether tangling, to the best of the authors' knowledge, the proposed algorithm is the first that explicitly minimizes tether tension. Initially designed for ground-based robots in a 2D environment, the method is later adapted for 3D environments. It operates under the assumption of a taut tether between consecutive contact points and requires knowledge of the coefficient of friction associated with the environment's obstacles and the ground.

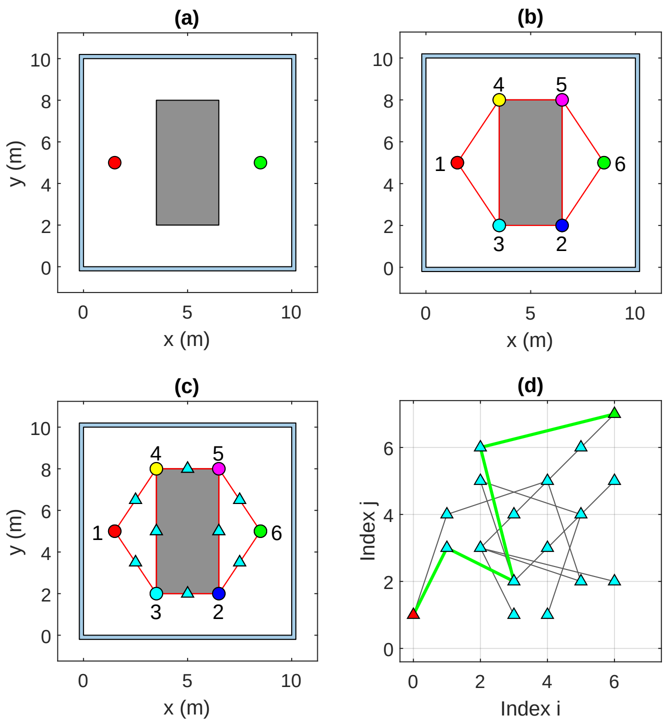


Figure 1. Example of the tension-aware path-planning pipeline: (**a**) 2D environment with an obstacle (■), start (●), and goal (●) positions. (**b**) Visibility graph, G, where edges are represented by red lines

and nodes are represented by circles numbered from 1 to 6. Here, v_1 and v_6 are the start and goal positions, while v_i with $i = 2, \dots, 5$ represents the obstacle vertices with known coefficients of friction $\mu_2 = 0.91$ (●), $\mu_3 = 0.13$ (●), $\mu_4 = 0.91$ (●), and $\mu_5 = 0.63$ (●). (**c**) Edges of G marked with triangles (▲) correspond to nodes of the tension-aware graph G_t in (**d**). (**d**) Besides the edges of G, two nodes named *virtual start* (▲) and *virtual goal* (▲) are added to G_t so that it has unique start and goal nodes. A path with minimum tension (—) was found in G_t using Dijkstra through nodes $(\textit{virtual start}) \rightarrow (1,3) \rightarrow (3,2) \rightarrow (2,6) \rightarrow (\textit{virtual goal})$, which corresponds to the lowest tension route via nodes $1 \rightarrow 3 \rightarrow 2 \rightarrow 6$ in G.

The rest of this paper is organized as follows. The problem statement is presented in Section 2. In Section 3, the proposed motion planner is formulated. Section 4 describes our experiments and discusses the most relevant results. Finally, Section 5 concludes this paper and proposes future work.

2. Problem Statement

The problem we want to solve is to find paths with minimal tension for tethered robots. In this scenario, a flexible and free tether is deployed externally by a fixed-position tethering system located at $x_\text{start} \in \mathbb{R}^d$, where $d = 2$ or 3, which serves as the initial position of the robot. It is assumed that at x_start, the tether is tensioned by a force T_0. The robot, modeled as a point mass, is tasked with navigating through a cluttered environment containing a set of n_o polygonal obstacles, denoted as O_i. The robot's goal is to reach the destination, identified as $x_\text{goal} \in \mathbb{R}^d$. The tether is permitted to make contact with the vertices of the obstacles along its path from the starting point to the goal. Additionally, it is assumed that there is knowledge of the static friction coefficient for each vertex and that the obstacle vertices are sufficiently rounded to accommodate the bending angle of the tether. Under these assumptions, the capstan equation, tailored for cylindrical-shaped objects, can be used along the bending angle and friction coefficients when the tether comes into contact with a corner. In a 2D environment ($d = 2$), where the tether makes contact with the floor, knowledge of the friction coefficient of the floor, denoted as μ_floor, is assumed. Also, the cable density is assumed to be constant and known.

3. Method

This section details the proposed methodology. An overview of the method is given next.

3.1. Overview

We initially assume a 2D environment containing polygonal obstacles. Figure 1a illustrates a scenario with a single rectangular obstacle, along with the start and goal positions. In the first step, our method computes the visibility graph (G). Nodes are numbered from 1 to n_V, as shown in Figure 1b, where $n_V = 6$, and visible nodes are connected by edges (Figure 1c).

Using a mapping function, Φ, each edge of G is then transformed into a node in the tension-aware graph, G_t. This transformation is necessary due to the nature of the problem, where two consecutive edges of G (sharing a common node) encode the bending angle formed by their respective line segments. In the context of the tension-aware graph, G_t , a single node represents an edge of G. Since two consecutive edges in G define an angle, the cost associated with two connected nodes in G_t is determined by a function of the wrapping angle (as used in the capstan model) and (half) the length of the two corresponding edges in G. The tension-aware graph G_t is established through Algorithm 1, detailed in Section 3.6.

In our example, the edge-to-node mapping is represented in the visibility graph in Figure 1c by triangle symbols placed at the midpoint of each edge of G. For example, three consecutive nodes $\{v_1, v_3, v_2\}$ in G form two consecutive edges: edge e_{13}, connecting nodes v_1 and v_3, and edge e_{32}, connecting nodes v_3 and v_2. This mapping is also displayed in the tension-aware graph in Figure 1d, where the triangle symbols in G_t represent nodes. It can be observed in this figure that the nodes $\hat{v}_{13}$ and $\hat{v}_{32}$ in G_t are derived from the two previous edges e_{13} and e_{32} in G. The 'hat' notation refers to the variables of the tension-aware graph. Additionally, notice that the edge connecting $\hat{v}_{13}$ and $\hat{v}_{32}$ in G_t corresponds to two consecutive edges e_{13} and e_{32} in G, which forms the angle $\angle v_1 v_3 v_2$. This angle, along with the lengths of the two edges, e_{13} and e_{32}, is then used to compute the edge cost in the tension-aware graph, G_t. In Figure 1d, the horizontal axis represents the index i of the parent node v_i, while the vertical axis represents the index j of the child node v_j, both referenced from the visibility graph. The red and green triangles in Figure 1d represent the start ($\hat{v}_\text{start}$) and goal ($\hat{v}_\text{goal}$) nodes in G_t.

As a final step, the Dijkstra algorithm searches on G_t to find the path with the minimum tension, which is then mapped back to the initial graph, G. This step creates a path with the least tension through the nodes $\{v_1, v_3, v_2, v_6\}$. In cases where such a path does not exist, the method indicates a failure, thus indicating the *completeness* of the path-planning method. A detailed breakdown of each step of the method is provided in the following sections.

3.2. Visibility Graph

The visibility graph of a 2D environment [25] is used as the starting point for the proposed method. Here, it is denoted as $G = (V, E)$, where V and E represent the sets of nodes and edges of the graph, respectively. V includes the start (x_start) and goal (x_goal) positions and encompasses all vertices of the polygonal obstacles in the set of obstacle vertices. The set of vertices, V, is ordered such that the first and last nodes of V represent the start and goal coordinates, respectively. Using a uniform notation, this set can be represented as $V = \{v_1, \ldots, v_{n_V}\}$, where v_1 and v_{n_V} correspond to the nodes associated with the coordinates x_start and x_goal, respectively.

3.3. Tension-Aware Graph

The tension-aware graph, denoted as $G_\text{t} = (V_\text{t}, E_\text{t})$, is generated from the visibility graph using a *bijective* function $\Phi : \mathbb{N}^2 \to \mathbb{N}$, mapping the edges $e_{ij} \in E$ in G to the nodes $\hat{v}_m \in V_\text{t}$. The 'hat' notation represents entities in the tension-aware graph. Note that there is a one-to-one correspondence between the edges and nodes in the graphs.

To address potential multiple start and goal nodes in G_t due to mapping Φ, two *virtual nodes*, $\{v_\text{vstart}, v_\text{vgoal}\} \notin V$ are introduced. These nodes are called virtual because they do not exist in G but are only used to compose the first and last nodes in G_t uniquely. The node *virtual start*, v_vstart, composes the edge $e_{v_\text{vstart}, v_\text{start}}$, which yields a single start node through the mapping function $\Phi(e_{v_\text{vstart}, v_\text{start}}) \to \hat{v}_\text{start}$. The node *virtual goal*, v_vgoal, composes the edge $e_{v_\text{goal}, v_\text{vgoal}}$, which gives a unique goal node via the mapping function $\Phi(e_{v_\text{goal}, v_\text{vgoal}}) \to \hat{v}_\text{goal}$. Note that while the nodes $\{v_\text{vstart}, v_\text{vgoal}\} \notin V$ and edges $\{(v_\text{vstart}, v_\text{start}), (v_\text{goal}, v_\text{vgoal})\} \notin E$, they are required to build the nodes $\{\hat{v}_\text{start}, \hat{v}_\text{goal}\} \in V_\text{t}$. All edges $\hat{e} \in E_\text{t}$ are weighted by a tension-based cost function, which is presented in detail in Section 3.5.

3.4. Assessing Feasible Free-Tether Configurations

To create the tension-aware graph, which contains only edges that correspond to feasible configurations of a free tether, a geometric analysis needs to be conducted. By "free" we mean a tether that does not become stuck or snagged in any part of the environment. Under this assumption, the cable can only bend toward an obstacle upon contact.

From the set of visibility graph nodes V, an ordered triplet of distinct nodes, A, B, C, is used to assess whether the tether configuration is feasible. The nodes are chosen so that the line segments $\overline{AB}$ and $\overline{BC}$ are adjacent at point B. Figure 2a shows the case where a cable touches three distinct obstacles through vertices, A, B, and C. The obstacle containing the vertex B, O_B, is used to check the intersection with the triangle ABC. An intersection between ABC and O_B ($ABC \cap O_B \neq \emptyset$) means that the tether will bend toward that obstacle, which represents a *feasible free-tether configuration*, as shown in Figure 2b. On the other hand, Figure 2c shows an unfeasible free-tether configuration, checked in the same way as before. As illustrated in Figure 2d, $ABC' \cap O_B = \emptyset$, which means that a free-tether configuration goes from A to C' directly through the line segment $\overline{AC'}$. Therefore, the path through the adjacent line segments $\overline{AB}$ and $\overline{BC'}$ is an infeasible free-tether configuration.

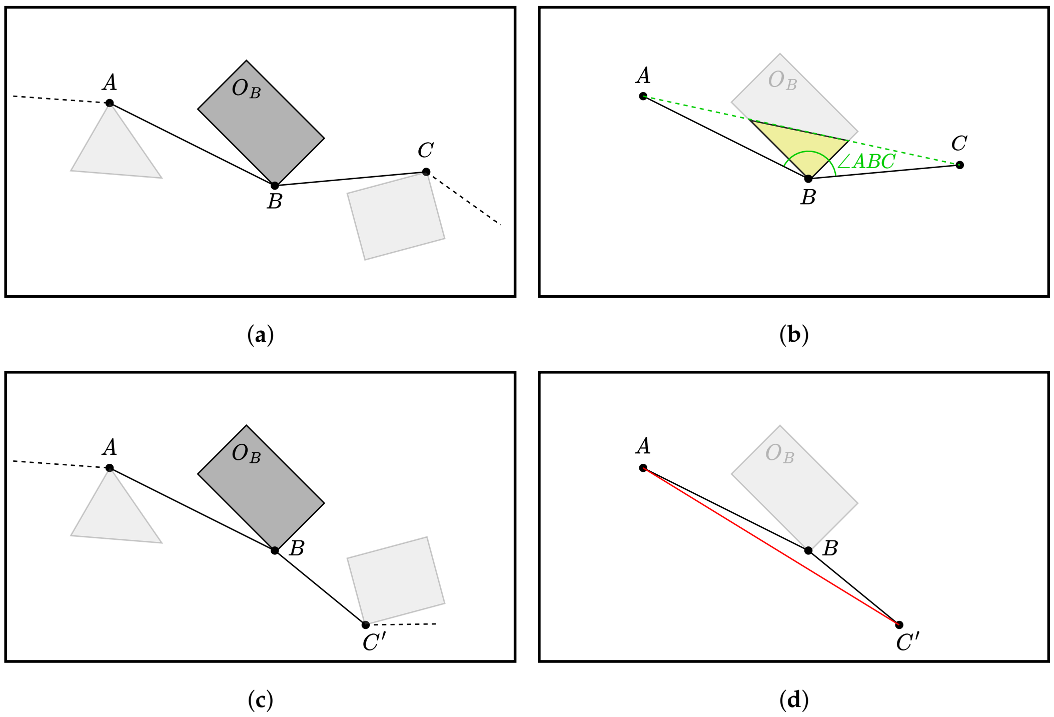


Figure 2. Tether configuration feasibility check. (**a**) Case where a tether touches three obstacles (gray polygons) at their vertices A, B, and C. The obstacle containing vertex B is highlighted and labeled O_B. (**b**) *Feasible* free-tether configuration, checked by assessing $ABC \cap O_B \neq \emptyset$. (**c**) *Unfeasible* free-tether configuration for vertices A, B, and C'. (**d**) Triangle ABC' does not intersect O_B ($ABC' \cap O_B = \emptyset$), indicating a non-feasible configuration.

3.5. Cost Function

The cost of each edge in G_t is based on two components. One term depends on the angles formed by the tether upon contact with obstacles and their friction coefficient. This term, denoted as τ_{capstan}, is modeled by the capstan equation [26]. The second component is derived from segments of the tether that also contribute to developing additional tension when it contacts the ground. Because this term manifests between pairs of nodes in the visibility graph, it is called the *intra-node* term, denoted as τ_{intra}. Thus, the combined cost function can be generically expressed as $\tau = \tau_{\text{capstan}} + \tau_{\text{intra}}$. Both terms in the function are discussed next.

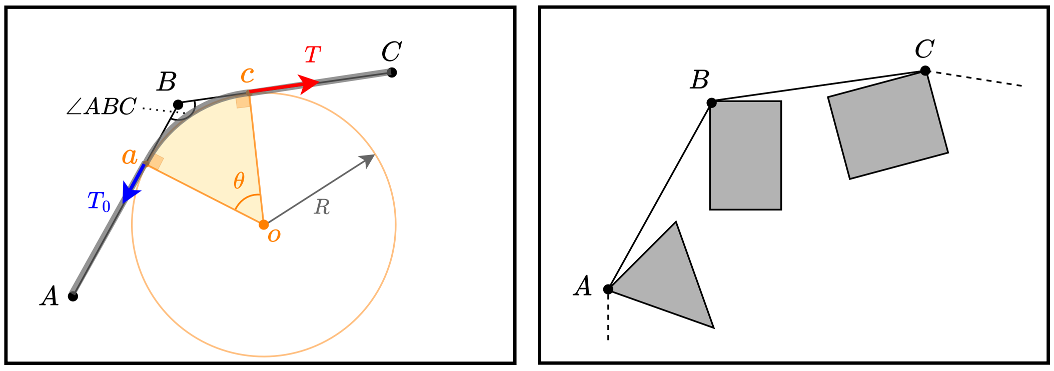
3.5.1. Capstan Term

The physical interaction between the tether and an obstacle vertex can be represented by the capstan model, as illustrated in Figure 3a. According to this principle, when a cable makes contact with a fixed cylindrical object that has a rough surface, the tension on each side of the cable varies due to the friction between the cable and the object, which counteracts the motion. As depicted in Figure 3a, the holding tension T_0 is less than the tension T when the cable is wrapped around an angle θ on a surface with a friction coefficient of μ. The capstan equation, which governs the relationship between these variables, is given by $T = T_0 e^{\mu\theta}$, which can be expressed as a ratio between the tensions as

$$\frac{T}{T_0} = e^{\mu\theta}\,. \tag{1}$$

It is important to note that, as shown in (1), the capstan model depends solely on the friction coefficient and the wrapping angle around a circular object. This implies that the friction is either derived from a surface with uniform roughness or represented by an average value for surfaces with nonuniform roughness. While this assumption simplifies the model, it does not account for variations in friction based on the wrapping angle or specific locations along the surface.

Given the friction coefficient at each vertex of an obstacle, μ, our task is to find the wrapping angle θ using the proposed formulation. As shown in Figure 2b, the tether bending angle can be found by examining three consecutive points A, B, and C, where at point B, the tether bends at an angle $\angle ABC$ in a feasible free-tether configuration. To derive θ, the scheme represented in Figure 3a is adapted for conditions similar to those depicted in Figure 2b, thus allowing us to express θ in terms of $\angle ABC$.



(**a**) (**b**)

Figure 3. (**a**) Transforming angle $\angle ABC$ into a capstan angle θ by establishing the angle relationship $\theta = 180^\circ - \angle ABC$ through a geometric scheme that relates the capstan model with the representation of polygonal obstacles. (**b**) Illustration of a viable free-tether configuration, wherein it interfaces with three obstacles (gray polygons) at vertices A, B, and C.

The process starts by introducing a circle with a radius R tangent to both line segments $\overline{AB}$ and $\overline{BC}$. The choice of the radius R is arbitrary, as it does not affect the capstan equation. The points of tangency, denoted as a and b along with the circle's center o, form a sector with an angle of θ. As depicted in Figure 3a, at the intersection points a and c, the tensions T_0 and T are also tangent to the circle, in accordance with the capstan model illustrated in the same figure. By considering the polygon $Baoc$ in Figure 3a, the sum of the internal angles gives the relationship between θ and $\angle ABC$ as $\theta = 180^\circ - \angle ABC\,$.

Without loss of generality, let us consider the situation represented in Figure 3a, where the tension at A is T_0 and at C is T, with $T \geq T_0$. Using the capstan Equation (1) and expressing the tension at C as $T = T_0 + \Delta T$, where ΔT is the tension developed when the tether interacts with the obstacle at B, we have $\tau_{\text{capstan}} = \Delta T / T_0 = e^{\mu\theta} - 1\,$, which is the normalized value of the tension created from the tether–obstacle interaction. To obtain a cost function with a physical unit, the normalized tension is multiplied by T_0, which gives the capstan term of the cost function:

$$\tau_{\text{capstan}} = T_0 (e^{\mu\theta} - 1)\,. \tag{2}$$

3.5.2. Intra-Node Term

The second term in the cost function, denoted as τ_{intra}, accounts for the increase in tension as the tether is paid out. This term takes into consideration the weight of a cable of length L_{e} given by linear distances lying intra-pair of consecutive nodes. Employing the notation adopted in the formulation, the tether length is given by the sum of line segments $\overline{AB}$ and $\overline{BC}$, as illustrated in Figure 3b. Consequently, the weight of a cable, characterized by linear density ρ and length $L_{\text{e}} = \overline{AB} + \overline{BC}$, is calculated using $W_{\text{e}} = \rho a_{\text{g}} L_{\text{e}}$, where a_{g} represents gravitational acceleration.

An observation about the actual path length and its calculation within the cost function can be addressed. The length of the path computed from the start point, v_{start}, to the goal point, v_{goal}, in a piecewise linear path, as shown in Figure 4, is given by

$$L = v_1v_2 + v_2v_3 + \ldots + v_{n-2}v_{n-1} + v_{n-1}v_n \,. \tag{3}$$

When constructing the tension-aware graph, three consecutive points (nodes in the visibility graph) are considered in each iteration. The process starts from the virtual start node, v_{vstart}, and goes toward the virtual goal node, v_{vgoal}. Since this process involves three points and only one point moves toward the goal in each iteration, a line segment is included twice in the computation. The following relationship can be established using Equation (3), recalling that the lines connecting to the virtual nodes have length zero, i.e., $v_{\text{vstart}}v_{\text{start}} = v_{\text{goal}}v_{\text{vgoal}} = 0$, or, in terms of numeric indices, $v_0v_1 = v_nv_{n+1} = 0$:

$$\begin{aligned}
\sum L_{\text{e}} &= (v_0v_1 + v_1v_2) + (v_1v_2 + v_2v_3) + \\
&\quad (v_2v_3 + \ldots + v_{n-2}) + (v_{n-2}v_{n-1} + v_{n-1}v_n) + (v_{n-1}v_n + v_nv_{n+1}) \\
\sum L_{\text{e}} &= \underbrace{v_0v_1}_{0} + 2(v_1v_2 + v_2v_3 + \ldots + v_{n-2}v_{n-1} + v_{n-1}v_n) + \underbrace{v_nv_{n+1}}_{0} \\
\sum L_{\text{e}} &= 2L\,.
\end{aligned} \tag{4}$$

This relationship shows that the computed length $L_{\text{e}} = \overline{AB} + \overline{BC}$, used in the following cost function, must be divided by a factor of 2.

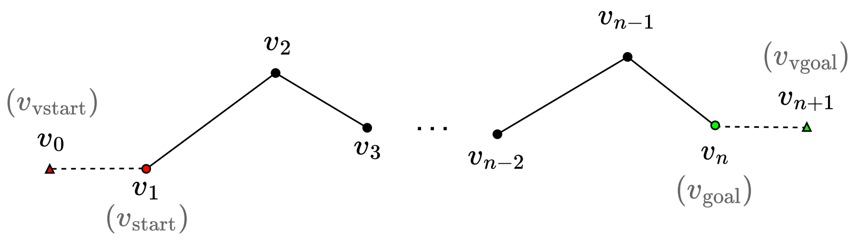


Figure 4. Computation of the path length through the edges in the visibility graph G. The path is represented as a sequence of nodes (●) in G, starting at the initial node v_1 (🔴) and ending at the goal node v_n (🟢). The edges connecting consecutive nodes (—) encode the path length based on Euclidean distance. The virtual nodes v_{vstart} (▲) and v_{vgoal} (▲), which connect to the start (v_{start}) and goal (v_{goal}) nodes, form zero-length edges represented by dashed lines (– –) and do not contribute to the total path length. Thus, the total path length is given by $L = v_1v_2 + v_2v_3 + \ldots + v_{n-2}v_{n-1} + v_{n-1}v_n$.

Finally, when the cable is in contact with the floor in a static equilibrium state, the normal force on the cable is equal to its weight, i.e., $N = W_{\text{e}}$. Given the friction coefficient of the floor, the tension resulting from its interaction with the cable is expressed as

$$\tau_{\text{intra}} = c \cdot \left(\rho a_{\text{g}} \sum L_{\text{e}}/2\right), \tag{5}$$

where $c = \mu_{\text{floor}}$ is the friction coefficient of the floor.

3.6. Algorithms

To determine the path with the minimum tension, two algorithms are proposed. Algorithm 1 builds the tension-aware graph, in which the edges are weighted by a tension-based cost function. Algorithm 2 performs a graph search on the tension-aware graph using the Dijkstra algorithm to find the path with the lowest cost. It also verifies the existence of the path and checks whether it exceeds the maximum tether length.

Algorithm 1: Building a tension-aware graph

```
Input  : $G = (V, E)$: Visibility graph of the map
         $\Phi$: Mapping function
         $O$: A set of disjoint polygonal obstacles
         $\mathcal{F}$: A structure with params $\mu$, and $c$
         $\mathcal{T}$: A structure with tether params $\rho$ and $T_0$
Output : $G_t = (V_t, E_t)$: Tension-aware graph
1  $(v_{start}, v_{goal}) \leftarrow$ Start and goal nodes from $G$
2  $v_{vstart} \leftarrow v_{start} - 1$
3  $v_{vgoal} \leftarrow v_{goal} + 1$
4  $V_t \leftarrow \{\Phi((v_{vstart}, v_{start}))\} \cup \Phi(E) \cup \{\Phi((v_{goal}, v_{vgoal}))\}$
5  $E_t \leftarrow \emptyset$
6  for all $\hat{v} \in V_t$ do
7      $e_{ij} = \Phi^{-1}(\hat{v})$
8      $V_t^j \leftarrow$ All edges connecting to $e_{ij}$ by node $v_j$
9      for all $\hat{v}_j \in V_t^j$ do
10         $e_{jk} = \Phi^{-1}(\hat{v}_j)$
11         $v_i, v_j \leftarrow$ Nodes from edge $e_{ij}$
12         $v_j, v_k \leftarrow$ Nodes from edge $e_{jk}$
13         $A, B, C \leftarrow$ Coordinates of nodes $v_i, v_j, v_k$
14         if $v_i == v_{vstart}$ then
15             $E_t \leftarrow E_t \cup \{(e_{ij}, e_{jk})\}$
16             Set cost of edge $\{(e_{ij}, e_{jk})\}$ to $c \cdot (\rho a_g \overline{BC}/2)$
17         else if $v_k == v_{vgoal}$ then
18             $E_t \leftarrow E_t \cup \{(e_{ij}, e_{jk})\}$
19             Set cost of edge $\{(e_{ij}, e_{jk})\}$ to $c \cdot (\rho a_g \overline{AB}/2)$
20         else
21             $ABC \leftarrow$ Triangle from coordinates $A$, $B$, and $C$
22             $O_B \leftarrow$ Obstacle that contains the point $B$ from $O$
23             if $ABC \cap O_B \neq \emptyset$ then
24                 $\theta \leftarrow \pi - \angle ABC$
25                 $L_e \leftarrow \overline{AB} + \overline{BC}$
26                 $E_t \leftarrow E_t \cup \{(e_{ij}, e_{jk})\}$
27                 Set cost of edge $\{(e_{ij}, e_{jk})\}$ to $\tau = T_0(e^{\mu\theta} - 1) + c \cdot (\rho a_g L_e/2)$
28             end
29         end
30     end
31 end
32 End Function
```

3.6.1. Tension-Aware Graph Creation

Algorithm 1 takes five inputs and produces an output, which is the tension-aware graph. In line 1, the algorithm begins by retrieving the start (v_{start}) and goal (v_{goal}) nodes from the visibility graph, G. The node-index definition for the *virtual nodes* takes place in lines 2 and 3, where the virtual node v_{vstart} precedes v_{start}, and v_{vgoal} succeeds v_{goal}. The initialization of the tension-aware graph, $G_{\mathrm{t}} = (V_{\mathrm{t}}, E_{\mathrm{t}})$, occurs in lines 4 and 5. Line 4 performs the mapping function Φ to map the edges in G to the nodes in G_{t}. Two additional edges, which connect the virtual nodes to the start and goal nodes, are added to G_{t} as follows: $\{\Phi((v_{\mathrm{vstart}}, v_{\mathrm{start}}))\} \cup \Phi(E) \cup \{\Phi((v_{\mathrm{goal}}, v_{\mathrm{vgoal}}))\}$. The number of nodes of the tension-aware graph is $n_E + 2$, where n_E is the number of edges in the visibility graph.

Line 6 iterates over all nodes in G_{t}, and in line 7, the inverse mapping function Φ^{-1} retrieves the edge e_{ij} from the visibility graph G, which is linked to the current node $\hat{v}$ in the tension-aware graph G_{t}. Line 8 creates a list of all outbound edges of the visibility graph that are connected to e_{ij} at v_j. Line 9 iterates over all elements in V_{t}^{j}. Line 10 is used to retrieve the edge e_{jk} that is connected to the previous edge e_{ij}. Then, lines 11 and 12 retrieve nodes v_i, v_j, v_k from edges e_{ij} and e_{jk}. Line 13 extracts the coordinates A, B, and C from nodes v_i, v_j, and v_k, respectively. These coordinates are used to compute the tether length and check feasible configurations of the tether.

Lines 14 to 16 check whether the virtual start node v_{vstart} has been found. If found, a new edge $\{(e_{ij}, e_{jk})\}$ is appended to the tension-aware graph with a cost of $\tau_{\mathrm{intra}} = c \cdot (\rho g \overline{BC}/2)$. The same rule applies to lines 17 to 19 when the virtual goal node v_{vgoal} has been reached.

The strategy for constructing the tension-aware graph relies on a geometric approach to check feasible free-tether configurations and compute the tension-based cost function. In line 21, the coordinates A, B, and C are used to form a triangle ABC. Additionally, in line 22, the obstacle containing vertex B, O_B, is selected, as shown in Figure 2a. To find feasible tether shapes when contacting obstacle corners, an intersection between the triangle ABC and O_B is checked. When the intersection exists, lines 24 to 27 are executed, where the corresponding edge is added to the tension-aware graph (G_{t}), the wrapping angle θ is computed, and the tether length associated with this segment is also calculated. The friction coefficient (μ) at B is retrieved from the data structure $\mathcal{F}$, along with the parameter c, which defines the intra-node cost mode. The force on the cable T_0 and the linear density ρ are retrieved from the data structure $\mathcal{T}$. Finally, the cost function $\tau = T_0(e^{\mu\theta} - 1) + c \cdot (\rho a_{\mathrm{g}} L_{\mathrm{e}}/2)$ is computed to set the cost of the newly created edge in the tension-aware graph.

3.6.2. Tension-Aware Graph Search

In line 1, Algorithm 2 starts by using the Dijkstra algorithm to search for a path from $\hat{v}_{\mathrm{start}}$ to $\hat{v}_{\mathrm{goal}}$ in G_{t}, storing the result as $\mathcal{P}_{\mathrm{t}}$. Line 2 obtains the number of nodes in $\mathcal{P}_{\mathrm{t}}$. Lines 3 and 4 initialize the variables for the path length (L) and the path ($\mathcal{P}$).

Lines 5 to 9 build the path $\mathcal{P} \in G$ and calculate its cumulative tether length. Note that the iteration excludes the first ($j = 1$) and the last ($j = n$) nodes in $\mathcal{P}_{\mathrm{t}}$, which represent the virtual start and goal nodes, respectively. Line 6 decodes each node in $\mathcal{P}_{\mathrm{t}} \in G_{\mathrm{t}}$ using the inverse mapping function to retrieve the edges from the visibility graph G. Line 7 appends the corresponding edge to the path $\mathcal{P}$, while lines 8 and 9 determine the length of each edge and add it to the total length of the path.

Line 11 verifies the feasibility of the path by checking whether a path has been found during the search in line 1 and whether the tether length is shorter than the maximum allowable length L_{max}. If these conditions are met, the algorithm returns the path with the minimum tension; otherwise, it returns a failure, thus showing the *completeness* of the path-planning method.

Algorithm 2: Searching the tension-aware graph

```
Input  : $G_t = (V_t, E_t)$: Tension-aware graph
         $\Phi$: Mapping function
         $L_{max}$: Maximum length of the tether
         Start ($\hat{v}_{start}$) and goal ($\hat{v}_{goal}$) nodes of $G_t$
Output : A path from $v_{start}$ to $v_{goal}$ with minimum tension or failure
1  $\mathcal{P}_t \leftarrow$ Dijkstra($\hat{v}_{start}$, $\hat{v}_{goal}$, $G_t$)
2  $n \leftarrow$ Number of nodes in $\mathcal{P}_t$
3  $L \leftarrow 0$
4  $\mathcal{P} \leftarrow \emptyset$
5  for $j = 2$ to $n - 1$ do
6      $e = \Phi^{-1}(\mathcal{P}_t(j))$
7      $\mathcal{P} \leftarrow \mathcal{P} \cup e$
8      $L_e \leftarrow$ Euclidean length of edge $e$
9      $L \leftarrow L + L_e$
10 end
11 if $\mathcal{P} \neq \emptyset$ and $L \leq L_{max}$ then
12     return $\mathcal{P}$
13 else
14     return failure
15 end
16 End Function
```

3.7. Analysis

The proposed tension-aware path-planning algorithm is complete. Note that it is always possible to construct a tension-aware graph using Algorithm 1. If this graph connects the virtual start to the virtual goal, the Dijkstra algorithm will return a path that can always be related to the edges in the visibility graph. If the path in the visibility graph is shorter than the maximum tether length, Algorithm 2 succeeds. If the Dijkstra algorithm returns an empty path or the path found is longer than the tether length, the algorithm indicates that a path does not exist.

The time complexity of the method can also be analyzed. The creation of the visibility graph G, necessary for Algorithm 1, can be performed in $\mathcal{O}(n_V^2)$ in the worst-case scenario [27]. The complexities of Algorithms 1 and 2 are computed as described below.

3.7.1. Tension-Aware Graph Creation

Lines 1 to 5 of Algorithm 1 contribute a constant time complexity of $\mathcal{O}(1)$, with the exception of line 4, which has a complexity of $\mathcal{O}(\hat{n}_V)$, where $\hat{n}_V$ is the number of nodes in G_t. The two nested loops starting at lines 6 and 9 iterate over the nodes in G_t. Within these loops, most of the operations have a complexity of $\mathcal{O}(1)$. Checking for obstacles associated with point B, if stored beforehand in an appropriate data structure, has a complexity of O(n_o), where n_o is the number of obstacles in the environment. The 2D polygon intersection check in line 23, if implemented with an optimal algorithm [28], can be performed in $\mathcal{O}(n_s \log n_s)$, where n_s is the total number of sides of the obstacle O_B. Lines 24 to 27 have a time complexity of $\mathcal{O}(1)$ for each operation. Therefore, the complexity of Algorithm 1 arises from the two nested loops (lines 6 and 9), which iterate over the nodes in G_t and contribute to a worst-case complexity of $\mathcal{O}(\hat{n}_V^2)$ or $\mathcal{O}(n_V^4)$, since every edge in G is mapped to a node in G_t plus two virtual nodes, i.e., $\hat{n}_V = n_E + 2$, and the relationship between the nodes and edges in a directed graph in a worst-case scenario is given by $n_E = n_V(n_V - 1)$. Considering the complexities of the inner operations within these loops, the overall complexity becomes $\mathcal{O}(n_V^4(n_o + n_s \log n_s))$.

3.7.2. Tension-Aware Graph Search

The search for the minimum tension path begins at line 1 by seeking a path from $\hat{v}_{\text{start}}$ to $\hat{v}_{\text{goal}}$ in G_{t} using the Dijkstra algorithm [27]. Considering the number of nodes $(\hat{n}_V)$ and edges $(\hat{n}_E)$ in the tension-aware graph, the overall time complexity of this operation is $\mathcal{O}(\hat{n}_V \log(\hat{n}_V + \hat{n}_E))$. Lines 2 to 4 exhibit a complexity of $\mathcal{O}(1)$. The loop in line 5 encompasses lines 6 to 9, which also have a complexity of $\mathcal{O}(1)$. As the loop in line 5 iterates over the number of nodes n in the path $\mathcal{P}_{\text{t}}$, it has a complexity of $\mathcal{O}(n)$, which, in the worst-case scenario, is $\mathcal{O}(\hat{n}_V)$. The remaining lines of the algorithm also have complexity $\mathcal{O}(1)$. Therefore, by noticing that $\hat{n}_V = \mathcal{O}(n_V^2)$ and $\hat{n}_E = \mathcal{O}(n_V^4)$ in the worst-case scenario, the overall time complexity of this algorithm is $\mathcal{O}(n_V^2 \log(n_V))$.

Thus, by combining the complexities of creating and searching the graph, the overall complexity of the method can be expressed as $\mathcal{O}(n_V^4(n_o + n_s \log n_s))$, which is mainly due to the creation of the tension-aware graph.

3.8. Extension to 3D

This section presents the extension of the method to 3D scenarios. To do so, we assume that the environment is known a priori and that the 3D obstacles have a special prismatic shape created from 2D polygonal obstacles by extending their shapes up in a vertical straight path to a certain height. Despite the simplicity of such shapes, they can be considered good approximations for trees in a forest or pillars in an indoor building. The heights of the obstacles may vary. The method assumes a taut tether between pairs of consecutive contact points, including the start and goal positions.

Let the start and goal positions be defined as $\{x_{\text{start}}, x_{\text{goal}} \in \mathbb{R}^3\}$, which are located in free space and connected through a tether. The start and goal positions may not be in the line of sight of each other due to the presence of obstacles. Let three consecutive tether–obstacle contact points be defined as A, B, and C. These points are used to define a plane $\mathcal{H}$ such that $\{A, B, C\} \subset \mathcal{H}$. The contact points for prism-shaped obstacles, such as those in Figure 5, can be computed in advance using the start and goal positions. This involves two steps: (1) determining the horizontal coordinates (x, y) in the xy-plane; and (2) calculating the z-coordinate based on the horizontal distance and the cable's elevation angle. Assuming a constant elevation angle between x_{start} and x_{goal}, the height (z) of each contact point can be derived from the known obstacle edges and the horizontal distances. A plane defined by these contact points slices through the obstacles that contain these points, thereby creating a 2D representation of the space and allowing the use of the method originally developed for the 2D case.

To use this approach, the cost function is adjusted to account for the cable's weight, which contributes to changes in tension as the tether length varies. In this scenario, only the weight of the cable corresponding to the length L_{e} is considered. Equation (5) is used, but the parameter c is set to 1.0 instead of the value of the friction coefficient of the floor, $c = \mu_{\text{floor}}$, used in the 2D scenario. Hence, the intra-node term of the cost function can be generically expressed as

$$\tau_{\text{intra}} = c \cdot \left(\rho a_{\text{g}} \sum L_{\text{e}}/2\right),$$

where

$$c = \begin{cases} \mu_{\text{floor}} & : \quad \text{2D Case (Tether on the floor)} \\ 1.0 & : \quad \text{3D Case (Tether suspended)} \end{cases}. \tag{6}$$

Figure 5 illustrates a 3D environment populated with four prism-shaped obstacles numbered from 1 to 4. In this scenario, a tether connects the start and goal positions through the contact points A (obstacle 2) and B (obstacle 3). Based on tether tautness, three consecutive points, $\{A, B, x_{goal}\}$, define a plane, exemplified by the plane $\mathcal{H}$. Within this plane, the method developed for 2D cases through Algorithms 1 and 2 can be extended to the 3D environment.

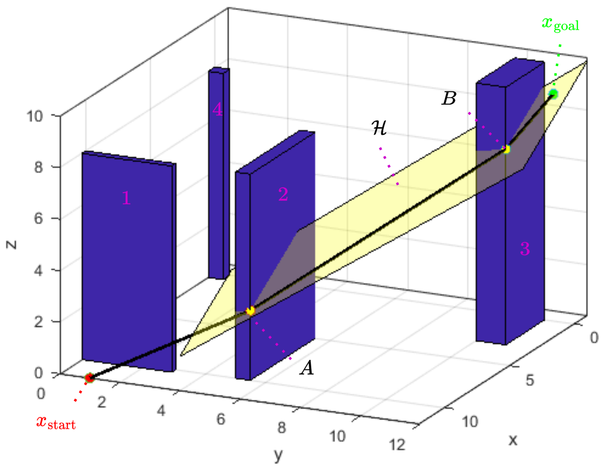


Figure 5. Representation of a 3D environment with four prism-shaped obstacles (■). A tether (—) connects the start (●) and goal (●) positions passing through the contact points A and B (●). A plane $\mathcal{H}$ (■) is defined by three consecutive path nodes (A, B, and x_{goal} in this case), assuming a taut tether (straight line) between pairs of consecutive nodes.

3.9. Relaxing Constraints

This section discusses some of the method's constraints and considerations for relaxing them, thus allowing it to be applied in several practical situations.

The proposed formulation considers polygonal obstacles regardless of their convexity, meaning that both convex and non-convex polygons can represent obstacles within the workspace. This flexibility is enabled by the feasible free-tether configuration feature present in the tension-aware algorithm. However, the method is still based on the presence of vertices (corners) on the obstacles. To use the method with generic-shaped objects, including circular or spherical obstacles, the user would need to create virtual vertices along the surfaces of the obstacle. This is a common approach when visibility graphs are used [29]. Since the capstan model is independent of the radius of a corner, a small number of virtual vertices could be created at points of high tether curvatures.

The method also assumes that the tether is either fully on the ground (2D) or in the air (3D). Consequently, the intra-node term defined in (5) is set based on the following situations: $c = \mu_{floor}$ if the tether is entirely on the floor, or $c = 1.0$ if the tether is completely in the air, as summarized in (6). In intermediate situations, where the tether could be partially touching the ground, c could be considered a weight and set to any value between μ_{floor} and 1. Note, however, that situations like this, in most cases, would also violate our assumption of a taut tether. In fact, the experienced reader should know that a completely taut tether is almost impossible to achieve due to the very high tensions required. Despite this, our method can be used, as long as the tether is reasonably straight between vertices. In 3D, this constraint is used to define a plane between three consecutive points. With a small slack, the cable will be slightly off the plane, resulting in a small deviation between the computed and actual tether lengths, which can usually be accepted.

4. Experiments

This section presents simulations and experiments conducted to evaluate the proposed method. The simulations assumed a bounded 2D space containing polygonal obstacles. The friction coefficient of the floor was assumed to be $\mu_{\text{floor}} = 0.6$. The linear density of the cable was $\rho = 0.01\,\text{kg/m}$ for both the simulations and real-world validations. The obstacles were assumed to have vertices with identical friction coefficients.

To evaluate our method, since no other tension-aware planner exists, we compared the paths found by the proposed tension-aware algorithm with those of the breadth-first search (BFS) and Dijkstra algorithms in the visibility graph G of the environment. The BFS algorithm found the path with the smallest number of nodes, while the Dijkstra algorithm found the one with the shortest distance. We chose the BFS algorithm because it was expected that the total contact friction would be reduced if the number of nodes was also reduced. On the other hand, if the length of the path was reduced, the friction with the floor was also reduced, justifying the comparison with the Dijkstra algorithm.

Figure 6 shows an environment of dimensions $10\,\text{m} \times 10\,\text{m}$ populated with nine obstacles. The start and goal positions are located at $x_{\text{start}} = (1.00, 9.00)$ and $x_{\text{goal}} = (9.00, 1.50)$. The friction coefficients for the obstacles are $\mu_1 = \mu_3 = \mu_5 = \mu_6 = \mu_8 = 0.40$, $\mu_2 = \mu_4 = 0.20$, and $\mu_7 = \mu_9 = 0.10$. The Dijkstra algorithm identified the shortest path with a length of $L = 12.84\,\text{m}$ and a tension cost of $\tau = 1.87$. Both the BFS and the tension-aware algorithms yielded paths with equivalent node counts. Although the BFS algorithm found a path with a shorter length of $L = 13.92\,\text{m}$ than the one found by the tension-aware algorithm with a length of $L = 14.32\,\text{m}$, the tension cost derived from the BFS algorithm was $\tau = 2.32$, which was more than twice as high as the tension cost of $\tau = 1.01$ derived from the tension-aware algorithm.

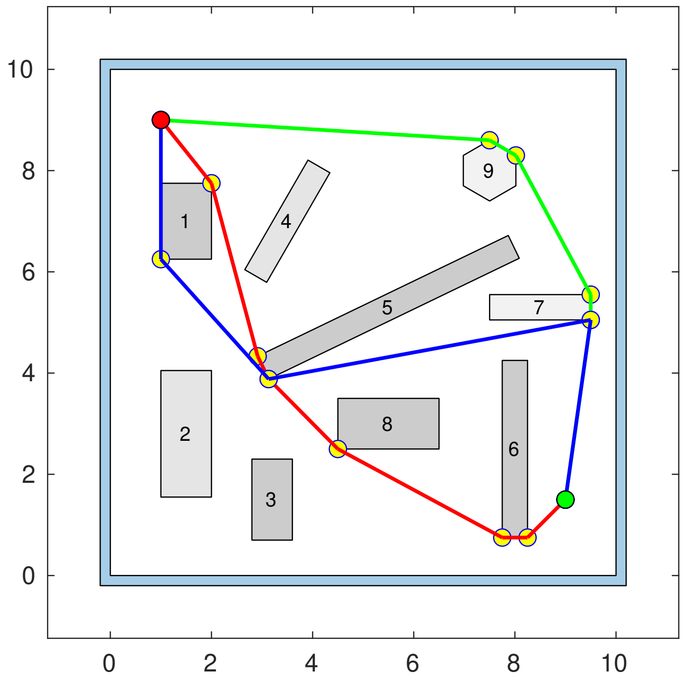


Figure 6. Path-planning results obtained by the BFS (—), Dijkstra (—), and tension-aware (—) algorithms in an environment containing 9 obstacles, numbered for reference in the figure. The paths connect the start (●) to the goal (●) positions along with all other vertices that contact the tether (●). The lighter the obstacle, the smaller its friction coefficient.

Figure 7 depicts a larger environment with dimensions of $25\,\text{m} \times 10\,\text{m}$ populated with 25 obstacles. The start position is located at $x_{\text{start}} = (0.40, 1.60)$ and the goal position is located at $x_{\text{goal}} = (19.60, 9.00)$. The friction coefficients for the obstacles are all chosen randomly within the range $0.0 \leq \mu \leq 1.0$. The shortest path (red) was found by the Dijkstra algorithm with a length of $L = 22.28$ m, but it came at the highest cost of $\tau = 3.67$. The path found by the BFS algorithm was the longest, with a length of $L = 25.90$ m and an intermediate cost of $\tau = 2.49$. This path also had the fewest nodes, with only 4 nodes, excluding the start and goal positions, in contrast to the 7 and 10 nodes in the paths found by the tension-aware and Dijkstra algorithms, respectively. Note that a path containing fewer contact points does not guarantee a minimum-tension path. Finally, the tension-aware algorithm found a path with an intermediate length of $L = 23.87$ m (approximately 7% longer than the shortest path) and the minimum cost of $\tau = 2.07$.

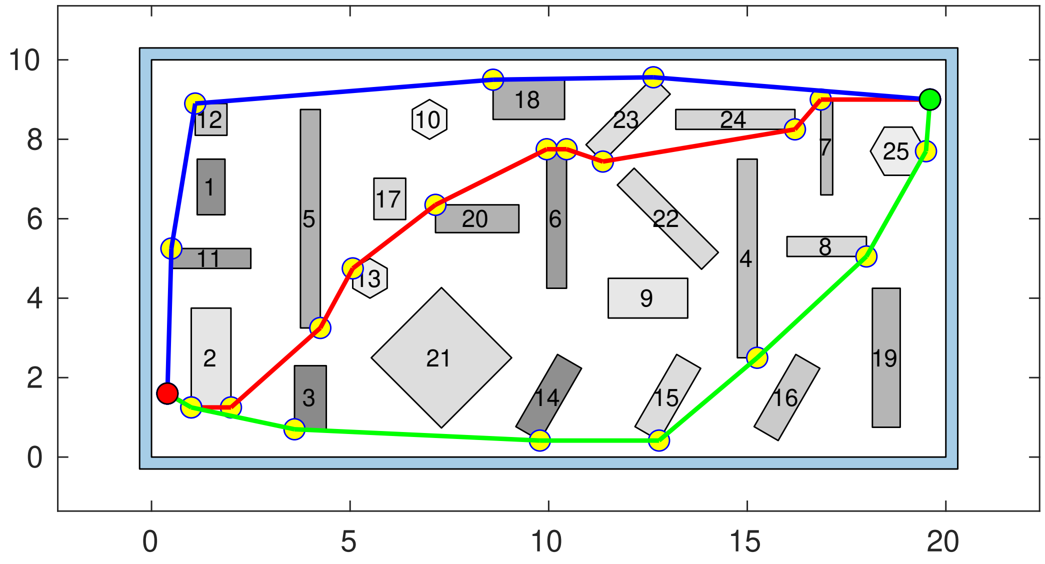


Figure 7. Path-planning results obtained by the BFS (—), Dijkstra (—), and tension-aware (—) algorithms in an environment containing 25 polygonal obstacles, numbered for reference in the figure. The paths connect the start (●) to the goal (●) positions along with all other vertices that contact the tether (●). The lighter the obstacle, the smaller its friction coefficient.

Two other simulations, as shown in Figure 8a,b, aimed to highlight the impact of the intra-node term of the cost function, τ_{intra}, especially on longer paths. Figure 8a shows an environment with dimensions of 5×5 m and two obstacles. Obstacle O_1 has a friction coefficient of $\mu_1 = 0.5$, while O_2 has a friction coefficient of $\mu_2 = 0.1$. The start and goal positions are placed at $x_{\text{start}} = (0.20, 4.80)$ and $x_{\text{goal}} = (4.80, 0.20)$. The BFS algorithm in G found a path with a length of $L = 7.26$ m and a cost of $\tau = 1.06$. The Dijkstra algorithm found a path with a length of $L = 6.70$ m and a cost of $\tau = 0.68$, while for the tension-aware algorithm, the path had a length of $L = 8.71$ m and a cost of $\tau = 0.67$. It can be observed in Figure 8a that the Dijkstra algorithm found a path with smaller tether bending angles while touching the obstacles with higher friction coefficients. On the other hand, the tension-aware algorithm chose paths with higher tether bending angles with smoother surfaces (lower friction). Although the tether length in the tension-aware case was longer, its length was not enough to penalize the selected path through the linear term of the cost function, which depends on the tether–floor friction. In Figure 8b, we show a different environment with the same obstacles as the previous simulation. The differences between the two environments are as follows: (i) the environment in Figure 8b is stretched horizontally to have dimensions of 5×15 m and; (ii) the obstacles are positioned at different locations: obstacle O_1 is located in the middle of the horizontal axis, whereas O_2 is located at the northeast part of the environment. The start and goal positions are located at $x_{\text{start}} = (0.20, 4.80)$ and $x_{\text{goal}} = (14.80, 0.20)$. The BFS algorithm found a similar path as that observed in Figure 8a by passing in the lower portion of obstacle O_1. This path had a length of $L = 15.80$ and a cost of $\tau = 1.21$. Both the Dijkstra and tension-aware algorithms found the same path with a length of $L = 15.44$ and a cost of $\tau = 1.05$. Note in this case how the tether–floor interaction impacted the cost function on longer paths. In Figure 8a, given the set of friction coefficients (tether–obstacle and tether–floor interactions), the main contribution of the cost function comes from the capstan term, which, in turn, grows exponentially with its variables (bending angle and friction). When distances become longer, the intra-node term tends to have a more significant impact on the cost function, acting to minimize the length indirectly, as shown in Figure 8b, where the tension-aware method provides the same path as the Dijkstra algorithm.

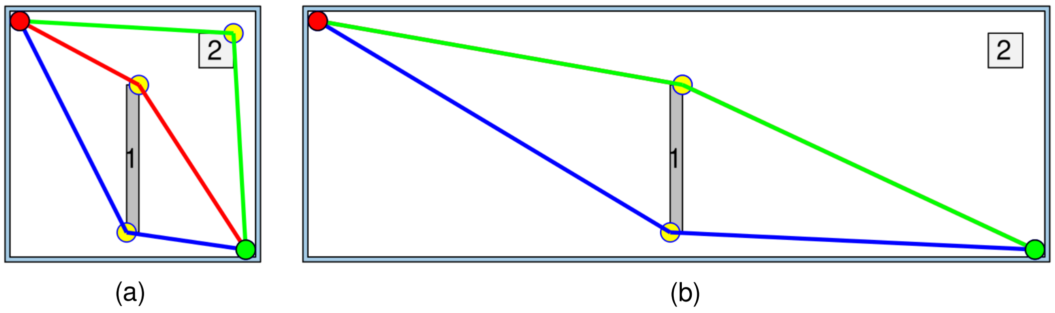


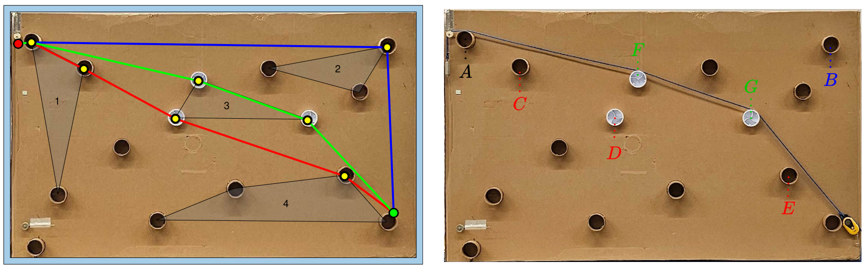
Figure 8. Path-planning results highlighting the effect of the "intra-node" term of the cost function on the tension-aware algorithm (—), the BFS algorithm (—), and the Dijkstra algorithm (—). The lighter the obstacle, the smaller its friction coefficient. Obstacles are numbered for reference in the figure (**a**) In a squared environment, each algorithm yields a distinct path. (**b**) In an environment three times wider, the BFS and Dijkstra algorithms find paths similar to that of their counterpart in (**a**), while the path of the tension-aware algorithm matches that of the Dijkstra algorithm, indicating that the minimum tension path is also the shortest. In this case, the "intra-node" term, which is proportional to the distance, becomes more important than the capstan term.

Our method was also assessed experimentally. Our setup employed a cardboard box measuring 1.24×2.04 m placed in a vertical orientation, as shown in Figure 9a. Fifteen cylindrical objects, each with a diameter of 8 cm, were attached to the box. These objects represent the vertices of obstacles within the workspace (cardboard box). Three of these objects (white circles) were made of plastic with a friction coefficient of $\mu_p \approx 0.19$. The remaining twelve objects were made of cardboard with a friction coefficient of $\mu_p \approx 0.32$. The friction coefficient for each type of corner used in the experimental setup was determined using the following method:

1. A cable was hung around the obstacle with a known wrapping angle;
2. A small bucket was attached to one side of the cable, and tiny metal balls were gradually added to the bucket to increase the weight until the cable overcame static friction and started to move;
3. The tension on the opposite side of the cable was measured using a digital hanging scale;
4. With the known wrapping angle, the added mass, and the measured tension, the friction coefficient was calculated using the capstan equation.

At the starting position, tension was generated by suspending an object with a known mass. Given that gravity was used to create tether tension at the starting position, the mass of the object was selected to be considerably larger than that of the cable in the experiments. Because of this, the intra-node term of the cost function in (2) was neglected by setting $c = 0$. Using the experimental setup as a reference, a virtual setup was recreated to closely replicate it. The paths generated in this virtual environment were subsequently recreated with an actual cable for tension assessment.

The start and goal positions at $x_{\text{start}} = (0.04, 1.08)$ and $x_{\text{goal}} = (1.93, 0.23)$ were connected with paths obtained by the BFS (blue line), Dijkstra (red line), and tension-aware (green line) algorithms, as shown in Figure 9a. To evaluate the actual tension on each path, we replicated the paths in the experimental setup by extending a cable through the same corners and measuring the tension using a scale at the goal position. Figure 9b shows the cable replicating the path obtained by the tension-aware algorithm. The cable touched vertices $A - F - G$, leading to a path length of $L = 2.39$ m and a cost of $\tau = 0.38 \pm 0.06$, mainly because this path explored vertices with lower friction coefficients (vertices F and G). The path obtained by the BFS algorithm (not shown in the figure) contacted obstacles through vertices A and B. The path length was $L = 2.95$ m and its cost was $\tau = 0.65 \pm 0.06$. The path obtained by the Dijkstra algorithm (also not shown) contacted obstacles through vertices $A - C - D - E$, and it had the shortest length of $L = 2.37$ m and cost of $\tau = 0.57 \pm 0.03$.



(**a**) (**b**)

Figure 9. (**a**) Paths obtained by the tension-aware (—), BFS (—), and Dijkstra (—) algorithms in a virtual environment with obstacles numbered 1–4 for reference, and overlaid on the experimental setup image. Start (●) and goal (●) positions along with all other vertices that contact the tether (●). (**b**) Path determined by the tension-aware algorithm and recreated with a tensioned cable passing by vertices $A - F - G$. A scale is used at the goal position to measure the tension.

5. Conclusions and Future Work

This paper presented a novel path-planning method for tethered robots. The proposed approach, which aims to minimize the tension on the tether, assumes an externally managed cable, with the robot pulling the tether from the anchor point. Initially designed for a 2D environment populated with polygonal obstacles, the method is extended to 3D scenarios with prism-shaped obstacles. Originally assuming tether tautness between pairs of contact points, the method remains valid in 3D scenarios, even for tethers with minimal slack (tensioned tether). In such cases, a triplet of consecutive contact points is used to form a plane. Although the cable may deviate slightly from this plane, such minor deviations have little impact on tether length and can be disregarded when applying the method. Finally, our approach was evaluated through simulations and real-world experiments for 2D environments, yielding paths with minimum tension. Nonetheless, the "intra-node" term penalizes longer paths and can indirectly minimize the path length over longer distances, as shown in the simulations. In the future, we plan to use the method to plan paths for tethered drones and ground robots in mining inspection tasks.

Author Contributions: Conceptualization, R.R.L. and G.A.S.P.; methodology, R.R.L. and G.A.S.P.; software, R.R.L.; validation, R.R.L. and G.A.S.P.; formal analysis, R.R.L.; investigation, R.R.L.; resources, R.R.L. and G.A.S.P.; data curation, R.R.L.; writing—original draft preparation, R.R.L.; writing—review and editing, R.R.L. and G.A.S.P.; visualization, R.R.L.; supervision, G.A.S.P.; project administration, G.A.S.P.; funding acquisition, G.A.S.P. All authors have read and agreed to the published version of the manuscript.

Funding: This research was funded by the Alpha Foundation for the Improvement of Mine Safety and Health, Inc., under grant number AFCTG22r2-159. The views, opinions, and recommendations expressed herein are solely those of the authors and do not imply any endorsement by Alpha Foundation, its directors, and its staff.

Data Availability Statement: The original data and code presented in the study are openly available at https://bitbucket.org/wvufarolab/tension-aware-motion-planning/(accessed on 27 January 2025).

Acknowledgments: This work was supported by the Alpha Foundation for the Improvement of Mine Safety and Health, Inc.

Conflicts of Interest: The authors declare no conflicts of interest.

Abbreviations

The following abbreviations are used in this manuscript:

BFS	Breadth-First Search

References

1. Yang, X.; Voyles, R.M.; Li, K.; Povilus, S. Experimental comparison of robotics locomotion with passive tether and active tether. In Proceedings of the 2009 IEEE International Workshop on Safety, Security & Rescue Robotics (SSRR 2009), Denver, CO, USA, 3–6 November 2009; IEEE: Piscataway, NJ, USA, 2009; pp. 1–6.
2. Evan Ackerman. Unlucky Robot Gets Stranded Inside Fukushima Nuclear Reactor, Sends Back Critical Data. 2015. Available online: https://spectrum.ieee.org/robot-stranded-inside-fukushima-nuclear-reactor (accessed on 11 October 2023).
3. Yang, T.; Liu, J.; Wang, Y.; Xiong, R. Self-Entanglement-Free Tethered Path Planning for Non-Particle Differential-Driven Robot. In Proceedings of the 2023 IEEE International Conference on Robotics and Automation (ICRA), London, UK, 25 May–2 June 2023; IEEE: Piscataway, NJ, USA, 2023; pp. 7816–7822.
4. Battocletti, G.; Boskos, D.; Tolić, D.; Palunko, I.; De Schutter, B. Entanglement Definitions for Tethered Robots: Exploration and Analysis. *arXiv* **2024**, arXiv:2402.04909.
5. Kim, S.; Bhattacharya, S.; Kumar, V. Path planning for a tethered mobile robot. In Proceedings of the 2014 IEEE International Conference on Robotics and Automation (ICRA), Hong Kong, China, 31 May–7 June 2014; IEEE: Piscataway, NJ, USA, 2014; pp. 1132–1139.
6. Shapovalov, D.; Pereira, G.A.S. Tangle-free exploration with a tethered mobile robot. *Remote Sens.* **2020**, *12*, 3858.
7. McCammon, S.; Hollinger, G.A. Planning and executing optimal non-entangling paths for tethered underwater vehicles. In Proceedings of the 2017 IEEE International Conference on Robotics and Automation (ICRA), Singapore, 29 May–3 June 2017; IEEE: Singapore, 2017; pp. 3040–3046.
8. Hernández, E.; Carreras, M.; Antich, J.; Ridao, P.; Ortiz, A. A topologically guided path planner for an auv using homotopy classes. In Proceedings of the 2011 IEEE International Conference on Robotics and Automation, Shanghai, China, 9–13 May 2011; IEEE: Shanghai, China, 2011; pp. 2337–2343.
9. Cao, M.; Cao, K.; Yuan, S.; Liu, K.; Wong, Y.L.; Xie, L. Path Planning for Multiple Tethered Robots Using Topological Braids. *arXiv* **2023**, arXiv:2305.00271.
10. Hershberger, J.; Snoeyink, J. Computing minimum length paths of a given homotopy class. *Comput. Geom.* **1994**, *4*, 63–97.
11. Abad-Manterola, P.; Nesnas, I.A.D.; Burdick, J. Motion planning on steep terrain for the tethered axel rover. In Proceedings of the 2011 IEEE International Conference on Robotics and Automation, Shanghai, China, 9–13 May 2011; IEEE: Shanghai, China, 2011; pp. 4188–4195.
12. Kirchgeorg, S.; Aucone, E.; Wenk, F.; Mintchev, S. Design, Modeling and Control of AVOCADO: A Multimodal Aerial-Tethered Robot for Tree Canopy Exploration. *IEEE Trans. Robot.* **2023**, *40*, 592–605.
13. Kang, G.; Güneş, O.; Lee, S.; Azhari, M.B.; Shim, D.H. SPIBOT: A Drone-Tethered Mobile Gripper for Robust Aerial Object Retrieval in Dynamic Environments. *arXiv* **2024**, arXiv:2409.16181.
14. Martinez Rocamora Jr, B.; Lima, R.R.; Samarakoon, K.; Rathjen, J.; Gross, J.N.; Pereira, G.A. Oxpecker: A tethered uav for inspection of stone-mine pillars. *Drones* **2023**, *7*, 73.
15. Capitán, J.; Díaz-Báñez, J.M.; Pérez-Cutiño, M.A.; Rodríguez, F.; Ventura, I. An efficient strategy for path planning with a tethered marsupial robotics system. *arXiv* **2024**, arXiv:2408.02141.
16. Novák, F.; Báča, T.; Saska, M. Collaborative Object Manipulation on the Water Surface by a UAV-USV Team Using Tethers. *arXiv* **2024**, arXiv:2407.08580.

17. Salzman, O.; Halperin, D. Optimal motion planning for a tethered robot: Efficient preprocessing for fast shortest paths queries. In Proceedings of the 2015 IEEE International Conference on Robotics and Automation (ICRA), Seattle, WA, USA, 26–30 May 2015; IEEE: Piscataway, NJ, USA, 2015; pp. 4161–4166.
18. Petit, L.; Desbiens, A.L. TAPE: Tether-Aware Path Planning for Autonomous Exploration of Unknown 3D Cavities Using a Tangle-Compatible Tethered Aerial Robot. *IEEE Robot. Autom. Lett.* **2022**, *7*, 10550–10557.
19. Paton, M.; Strub, M.P.; Brown, T.; Greene, R.J.; Lizewski, J.; Patel, V.; Gammell, J.D.; Nesnas, I.A. Navigation on the line: Traversability analysis and path planning for extreme-terrain rappelling rovers. In Proceedings of the 2020 IEEE/RSJ International Conference on Intelligent Robots and Systems (IROS), Las Vegas, NV, USA, 24 October 2020–24 January 2021; IEEE: Piscataway, NJ, USA, 2020; pp. 7034–7041.
20. Page, J.J.; Treers, L.K.; Jorgensen, S.J.; Fearing, R.S.; Stuart, H.S. The Robustness of Tether Friction in Non-Idealized Terrains. *IEEE Robot. Autom. Lett.* **2022**, *8*, 424–431.
21. Chipade, V.S.; Kumar, R.; Yong, S.Z. WiTHy A*: Winding-Constrained Motion Planning for Tethered Robot using Hybrid A. In Proceedings of the 2024 IEEE International Conference on Robotics and Automation (ICRA), Yokohama, Japan, 13–17 May 2024; IEEE: Piscataway, NJ, USA, 2024; pp. 8771–8777.
22. Chipade, V.S.; Panagou, D. Approximate time-optimal trajectories for damped double integrator in 2d obstacle environments under bounded inputs. *arXiv* **2020**, arXiv:2007.05155.
23. Dolgov, D.; Thrun, S.; Montemerlo, M.; Diebel, J. Path planning for autonomous vehicles in unknown semi-structured environments. *Int. J. Robot. Res.* **2010**, *29*, 485–501.
24. Specian, A.; Yim, M. Friction binding study and remedy design for tethered search and rescue robots. In Proceedings of the 2015 IEEE International Symposium on Safety, Security, and Rescue Robotics (SSRR), West Lafayette, IN, USA, 18–20 October 2015; pp. 1–6.
25. De Berg, M. *Computational Geometry: Algorithms and Applications*; Springer: Berlin/Heidelberg, Germany, 2000.
26. Levin, E. Friction Experiments with a Capstan. *Am. J. Phys.* **1991**, *59*, 80–84.
27. Choset, H.; Lynch, K.M.; Hutchinson, S.; Kantor, G.A.; Burgard, W. *Principles of Robot Motion: Theory, Algorithms, and Implementations*; MIT Press: Cambridge, CA, USA, 2005.
28. Shamos, M.I.; Hoey, D. Geometric intersection problems. In Proceedings of the 17th Annual Symposium on Foundations of Computer Science (sfcs 1976), Houston, TX, USA , 25–27 October 1976; IEEE: Houston, TX, USA, 1976; pp. 208–215.
29. Schøler, F.; la Cour-Harbo, A.; Bisgaard, M. Generating configuration spaces and visibility graphs from a geometric workspace for uav path planning. In Proceedings of the AIAA Guidance, Navigation, and Control Conference, Portland, OR, USA, 8–11 August 2011; pp. 8–11.