A Topological Approach to Workspace and Motion Planning for a Cable-controlled Robot in Cluttered Environments

Xiaolong Wang[1] and Subhrajit Bhattacharya[1]

***Abstract*—There is a rising demand for multiple-cable controlled robots in stadiums or warehouses due to its low cost, longer operation time, and higher safety standards. In a cluttered environment, the cables can wrap around obstacles, but careful choice needs to be made for the initial cable configurations to ensure that the workspace of the robot is optimized. The presence of cables makes it imperative to consider the homotopy classes of the cables both in the design and motion planning problems. In this paper, we study the problem of workspace planning for multiple-cable controlled robots in an environment with polygonal obstacles. This paper's goal is to establish a relationship between the boundary of the workspace and cable configurations of such robots and solve related optimization and motion planning problems. We first analyze the conditions under which a configuration of a cable-controlled robot can be considered valid, discuss the relationship between cable configuration, the robot's workspace and its motion state, and using graph search based motion planning in h-augmented graph perform workspace optimization and compute optimal paths for the robot. We demonstrated the algorithms in simulations.**

***Index Terms*—Motion and Path Planning, Industrial Robots, Collision Avoidance.**

I. Introduction

A. Literature Review

DESPITE the advances in mobile and aerial robotics, there are various applications in which cable-controlled robots are better suited. A robotic system controlled by varying-length cables anchored to fixed *control points* provide greater reliability (less prone to environmental noise such as wind gusts [1] since the robot is tethered), has less onboard power consumption (since the actuation is done by the external cables [2]) and does not rely on onboard sensors for localization and control (thus works in GPS-denied and featureless environments).

Robots attached to passive cables for power supply and communication have been extensively used for many real-world applications [3], [4]. For such robots, the main challenge is to avoid entanglement of the cable with obstacles and to ensure that the tether does not violate any geometric constraints [5]–[8]. The use of cables to manipulate objects in an environment has also been studied extensively [9]–[11].

Active control of robots using cables, on the other hand, has gained relatively less attention in robotics literature. The typical controllers for such robots are designed for obstacle-free environments where the inverse-kinematics problem can be solved in a closed form [12]. However, the problem of negotiating obstacles for a cable-controlled robot requires significant additional consideration. With the recent advent of topological path planning techniques [6], [10], [13], it has become possible to compute optimal solutions to motion planning problems for systems involving flexible cables by reasoning about topological classes (homotopy and homology classes) of paths and cables in a cluttered configuration space. This paper uses these recent developments in the field of topological path planning to design algorithms for cable-controlled robots in environments with polygonal obstacles.

B. Problem Statement and Motivation

We consider a planar environment cluttered by polygonal obstacles. This is a model for robots that can be used to transport goods in a warehouse or to move overhead cameras in a stadium (Fig. 1) – attached and controlled by cables that are driven by motors at the boundaries of the environment (roof or walls). The obstacles which cables cannot penetrate would inevitably make some of the regions inaccessible to the robot. The initial cable configuration of the system influences the shape and size of the workspace (see the difference between Fig. 2a and Fig. 2b). It is thus important to choose the best cable configurations of optimizing workspace's area and ensuring that the robot is able to reach the desired locations. We need a method to search for a boundary of robot's workspace corresponding to its initial cable configuration. A related application is that of sea farming, where a net needs to be anchored at certain points on its boundary, ensuring that the net does not get tangled with obstacles, such as boats or buoys, while maximizing the area covered by the net (which is used for farming of marine species).

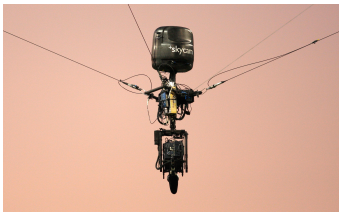


Fig. 1: A wire camera (Skycam, source: Wikimedia Commons)

C. Outline of the Paper

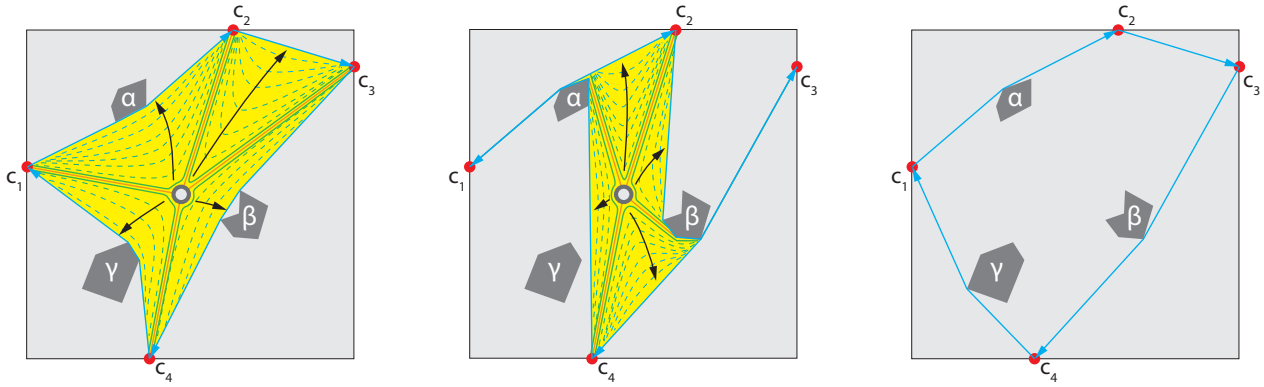
In this paper, we start by presenting some of the preliminary backgrounds including visibility graph, homotopy/homology class and h-augmented graph. Following that we analyze the properties of the workspace of a multiple-cable controlled robot and its boundary, and propose an algorithm for computing the boundary for which workspace's area is optimized

Manuscript received: December, 2, 2017; Revised: February, 12, 2018; Accepted: March, 5, 2018.

This paper was recommended for publication by Editor Wan Kyun Chung upon evaluation of the Associate Editor and Reviewers' comments.

[1]Both authors are with the Department of Mechanical Engineering and Mechanics, Lehigh University, Bethlehem, Pennsylvania, U.S.A [xiw716, sub216]@lehigh.edu

Digital Object Identifier (DOI): see top of this page.



(a) Original cables (orange solid) can be seen as a whole (green solid) which can deforming continuously into a closed boundary (blue solid).

(b) Another initial configuration of cables. The cables can deform continuously into a valid boundary of its workspace as well.

(c) An arbitrary combination of shortest paths between neighboring control points do not necessarily enclose a valid workspace.

Fig. 2: Same environment, different workspaces enclosed by shortest paths in some homotopy classes.

or certain specific points fall within the workspace. Finally, we describe the algorithms for robot motion planning and cable velocity control and apply these to several example applications.

II. Preliminaries

A. Construction of Visibility Graph

Consider a rectangular planar environment with polygonal obstacles. We establish a visibility graph of the environment with vertices, $\mathbf{v} \in \mathcal{V}$, consisting of the vertices of the polygonal obstacles, the control points (points on the environment boundary at which the cables are attached to the robot), robot point as well as start and end vertices of a trajectory, and the edges, $\{\mathbf{v} \rightarrow \mathbf{v}'\} \in \mathcal{E}$, consisting of line segments that connect vertices with direct line of sight. When dealing with the concave polygon, we select shortcuts between corners.

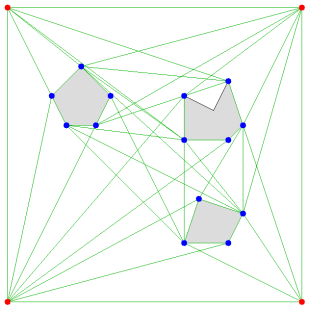


Fig. 3: A visibility graph (green line segments are edges, blue/red dots are vertices)

B. Brief Introduction of Homotopy and Homology Class [14]

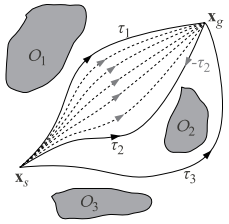
Definition 1: (Homotopy classes [13]) Two trajectories τ_1 and τ_2 connecting the same start and end points, $\mathbf{v}_s$ and $\mathbf{v}_g$ respectively, are homotopic or belong to the same homotopy class iff one can be continuously deformed into the other without intersecting any obstacle.

Definition 2: (Homology classes [13]) Two trajectories τ_1 and τ_2 connecting the same start and end points, $\mathbf{v}_s$ and $\mathbf{v}_g$ respectively, are homologous or belong to the same homology class iff τ_1 together with τ_2 (the later with opposite orientation) forms the complete boundary of a 2-dimensional manifold not containing/intersecting any of the obstacles.

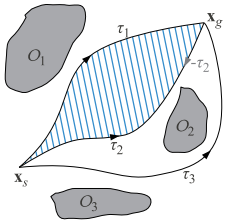
C. h-signature and H-signature

Assuming that all obstacles and ends of trajectories are fixed, h-signature and H-signature are respectively homotopy and homology invariants of trajectories – two trajectories connecting same start and end points have the same h- (or H-) signatures iff they are in the same homotopy (or homology) class [15]. Function $h(\cdot)$ is for denoting h-signature of a trajectory. We use representative points (obstacles), ζ_i, and the non-intersecting rays r_i emanating from the representative points for constructing h-signature. We form a word by tracing τ, and consecutively placing the letters of the rays that it crosses, with a superscript of +1 if the crossing is from right to left, and -1 if the crossing is from left to right. For example, in Fig. 5a, if τ first crosses r_b right to left then r_a from right to left as well, the h-signature is 'ba'(short for '$b^{+1}a^{+1}$'); if τ first crosses r_a left to right then r_b from left to right too, the h-signature is '$a^{-1}b^{-1}$'. If in an h-signature 'a^{-1}' appears next to 'a', indicating that the trajectory crosses r_a followed by crossing back, these two letters can cancel each other, like the trajectory never crosses r_a. We use simplified h-signatures. For instance, in Fig. 5a the empty h-signature of the upper curve was simplified from the initial '$baa^{-1}b^{-1}$' to 'bb^{-1}', then from 'bb^{-1}' to ' '(empty). The h-signature is internally non-commutative. The direction of a curve is important, for the same path in reverse direction will result in an inverse h-signature where both the order of letters and superscripts of letters are opposite [10]. If a curve is a closed loop and it encloses no representative points (obstacles), it is null homotopic with an empty h-signature. More examples are shown in Fig. 5a.

Likewise, $H(\cdot)$ is a function for denoting H-signature of a trajectory. H-signature is a vector, the i^{th} element of which has the simple interpretation of counting the number of times the curve, τ, intersects the ray emanating from ζ_i (see Fig. 5b). In particular, define $\#_i\tau =$ (Number of times τ crosses

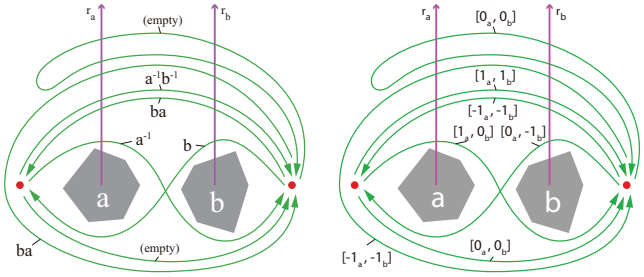


(a) τ_1 is homotopic to τ_2 since there is a continuous deformation from one to the other, but not τ_3.



(b) τ_1 is homologous to τ_2 since they bound an area, A, but τ_3 belongs to a different homology class.

Fig. 4: Illustration of homotopy and homology equivalences. In this example τ_1 and τ_2 are both homotopic and homologous



(a) h-signature (b) H-signature

Fig. 5: h- and H-signatures of different trajectories connecting two same points

the ray r_i emanating from ζ_i from left to right) − (Number of times τ crosses the ray r_i emanating from ζ_i from right to left). Then, $H(\tau) = [\#_1\tau, \#_2\tau, \ldots, \#_n\tau]^\intercal$ [10]. For instance, 3 obstacles in the environment, if the H-signature of trajectory τ is $H(\tau) = [1, 0, 1]^\intercal$, it shows that after all τ cross ray r_1 once and r_3 once from left to right, not crossing r_2. If τ is a closed loop, $H(\tau) = [1, 0, 1]^\intercal$ shows that ζ_1 and ζ_3 are inside the loop and ζ_2 is outside.

D. The h-signature Augmented Graph [13]

In order to keep track of the homotopy invariants, we define an h-augmented graph, $\mathcal{G}_h = (\mathcal{V}_h, \mathcal{E}_h)$, based on a visibility graph, $\mathcal{G} = (\mathcal{V}, \mathcal{E})$, such that a vertex in $\mathcal{G}_h$ contains the additional information of the h-signature of the trajectory leading from a start vertex $\mathbf{v}_s$ up to this vertex besides its coordinate. A transition from vertex $(\mathbf{v}, \mathfrak{h})$ to vertex $(\mathbf{v}', \mathfrak{h}')$ means that the h-signature, $\mathfrak{h}'$, is a concatenation of $\mathfrak{h}$ and the h-signature of trajectory from $\mathbf{v}$ to $\mathbf{v}'$.

1) $$\mathcal{V}_h = \left\{ (\mathbf{v}, \mathfrak{h}) \;\middle|\; \begin{array}{l} \mathbf{v} \in \mathcal{V}, \text{ and,} \\ \mathfrak{h} = h(\widetilde{\mathbf{v}_s\mathbf{v}}) \text{ for some trajectories from} \\ \quad \text{the start vertex } \mathbf{v}_s \text{ to this vertex } \mathbf{v} \end{array} \right\}$$

2) An edge $\{(\mathbf{v}, \mathfrak{h}) \to (\mathbf{v}', \mathfrak{h}')\}$ is in $\mathcal{E}_h$ for $(\mathbf{v}, \mathfrak{h}) \in \mathcal{V}_h$ and $(\mathbf{v}', \mathfrak{h}') \in \mathcal{V}_h$, iff $(\mathbf{v} \to \mathbf{v}') \in \mathcal{E}$, and, $\mathfrak{h}' = \mathfrak{h} \circ h(\mathbf{v} \to \mathbf{v}')$, where, "$\circ$" is a concatenate operator.

3) The cost associated with an edge $\{(\mathbf{v}, \mathfrak{h}) \to (\mathbf{v}', \mathfrak{h}')\} \in \mathcal{E}_h$ is the same as that associated with edge $\{\mathbf{v} \to \mathbf{v}'\} \in \mathcal{E}$.

The h-augmented graph is unbounded. Its vertices are generated on-the-fly and as required during the execution of *Dijkstra*'s/*A** search on the graph, which we describe later.

III. Algorithm Design for Workspace Computation

In this section, we describe some specific features, properties and algorithms related to the cable configurations and the workspace of the cable-driven robot. These include the topological properties of workspace's boundary and algorithms for obtaining all valid workspaces from given obstacle configurations.

A. Definition of Boundary of a Workspace

The configurations of a cable-robot system in which the robot is capable of moving in any direction are called an *interior point* (Fig. 6a). On the contrary, a *boundary point* is a configuration where the robot can move only in some specific directions. These directions can constitute a half plane (Fig. 6b) or more generally union of cones (Fig. 6d). All the boundary points constitute a *boundary* of the workspace.

B. Force Analysis

The physical constraint of the cable is that there can only be tensions but no pressure acting on the cable and that the net force on the robot must be zero to make it stay at a certain position (see Fig. 6a), denoted as $\mathbf{0} = \sum_i F_i \hat{\mathbf{e}}_i, i = 1, 2, \ldots, n$, where $F_i \geq 0$ is a non-negative scalar of tension, $\hat{\mathbf{e}}_i$ is a unit vector of the direction in which the i^{th} cable points at the location of the robot and n is the number of cables. When the robot is maneuverable, not all of the cables can be slack, that is, $\sum_i F_i \neq 0$. To make the net force zero with some taut cables, there must exist at least two of them and they must lie in different half-planes to counteract each other. If all cables are pointing in the same half-plane, the robot cannot go any further towards the other half-plane (Fig. 6b and 6c). We call this case the boundary state in the open area.

C. Boundary State

There are two kinds of boundary states.

1) In the Open Area: If $\hat{\mathbf{e}}_i$ of all taut cables span a one-dimensional line, it is a boundary state and the robot is located at the boundary of the workspace (see Fig. 6b and 6c).

2) Touching an Obstacle: When the robot driven by cables touches an obstacle, the robot is in touching-obstacle boundary state and the obstacle's edge or/and corner that is touched is a part of boundary of the workspace, where the net force of cables may not be zero, shown in Fig. 6d.

D. Shortest Paths and Boundary

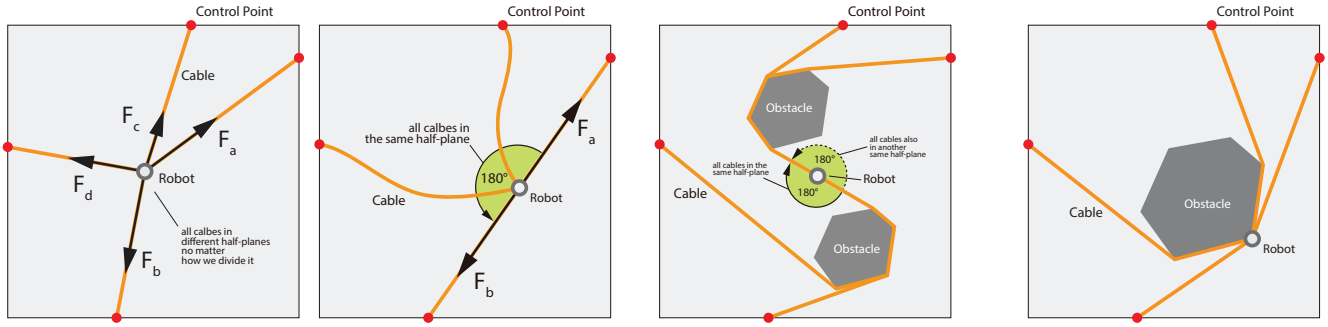
Lemma: The workspace boundary curve connecting a pair of neighboring control points is the shortest path in the same homotopy class connecting that pair (by "neighboring" we refer to *adjacent* control points encountered as we trace the boundary of the environment).

Sketch of Proof: When in boundary state, if we remove all slack cables, the robot's position and taut cables will keep stable. We can regard the pair of taut cables as a whole which is also taut, going from one control point to a neighboring one though the robot which can be regarded as a mass point on that whole cable. This whole taut cable is the shortest path in current homotopy class connecting those two control points, shown as blue curves in Fig. 2a and 2b.

E. Boundary's Features

When the robot is at an interior point, we can move the robot along arbitrary trajectories inside the boundary. As the robot is moving toward a part of boundary, a pair of neighboring cables can deform continuously into that part of boundary, without interfering with any obstacle, shown in Fig. 2a and 2b. This deformation from the initial configuration into a part of boundary holds for any pair, which indicates that all cables pairs can deform into a complete and closed-loop boundary. Since no obstacles are crossed during this deformation, there must be no obstacles inside the boundary. Just for comparison, Fig. 2c shows an invalid boundary, even if it is made of shortest paths.

Proposition 1: The closed-loop formed by the boundary of a workspace is null homotopic (i.e. its h-signature is 'empty' word), stated in Section II-C.



(a) Interior state (b) Boundary state in open area (c) A special boundary state in open area (d) Boundary state touching an obstacle

Fig. 6: Robot states and cable configuration

F. Shortest Paths Searching

After constructing the visibility graph, we use *Dijkstra*'s search in the h-augmented graph to get the shortest paths with various h-signatures. We construct n threads for multi-threading search, $\mathbf{T}_i$, where $i = 1, 2, \ldots, n$ and n is the number of control points. Each thread contains a *Dijkstra* search returning paths connecting a pair of neighboring control points which are indexed along clockwise or anticlockwise direction. For example, $\mathbf{T}_1$ is for paths connecting control point $\mathbf{c}_1$ and $\mathbf{c}_2$, namely $\mathbf{T}_2$ for $\mathbf{c}_2$ and $\mathbf{c}_3$, ..., $\mathbf{T}_n$ for $\mathbf{c}_n$ and $\mathbf{c}_1$, as shown in Fig. 2a. We insert the output τ into n corresponding sets of shortest paths, $\mathcal{P}_i = \{\tau_{i1}, \tau_{i2}, \ldots\}$. These n threads keep searching till the length of boundary (consist of n paths, one from each set) must exceed a limit $\mathcal{L}$ we properly preset.

Algorithm 1 Shortest path searching in the i^{th} thread $\mathbf{T}_i$

Input: The h-augmented graph, $\mathcal{G}_h = \{\mathcal{V}_h, \mathcal{E}_h\}$; the set of control points for start vertices and goal vertices, $\mathcal{C} = \{\mathbf{c}_1, \mathbf{c}_2, \ldots, \mathbf{c}_n\}$; a limit of boundary cost $\mathcal{L}$.

Output: Set of shortest paths in different homotopy classes, $\mathcal{P}_i = \{\tau_{i1}, \tau_{i2}, \ldots\}$; set of h-signatures of paths, $\mathcal{H}_i = \{\mathfrak{h}_{i1}, \mathfrak{h}_{i2}, \ldots\}$.

1: $k \leftarrow 1$
2: **loop**
3: $\tau_{ik} \leftarrow$ a shortest path, in the k^{th} homotopy class connecting $\{\mathbf{c}_i, \text{`'}\}$ and $\{\mathbf{c}_{i+1}, \mathfrak{h}\}$ for some $\mathfrak{h}$
4: $l_i \leftarrow C(\tau_{ik}) + \sum_j C(\tau_{j1}), j = 1, 2, \ldots, n, j \neq i$
5: **if** $(l_i > \mathcal{L})$ **then**
6: **break loop**
7: **end if**
8: Insert τ_{ik} into $\mathcal{P}_i$
9: $\mathfrak{h}_{ik} \leftarrow h(\tau_{ik})$
10: Insert $\mathfrak{h}_{ik}$ into $\mathcal{H}_i$
11: $k \leftarrow k + 1$
12: **end loop**

Since there are only n control points, the indices of control points in Algorithm 1 can run from 1 to n, thus in Line 3, if $i = n$, then by $i + 1^{\text{th}}$ we refer to 1^{st} control point (i.e.: we take the modulo with respect to n with shift of 1). Hereafter whenever we refer to $i+1$ for a control point index, we assume this convention. The paths returned from *Dijkstra*'s search are in an order from least cost to higher cost. Thus the 1^{st} path, τ_{i1}, in every thread's outcome is of the least cost. In Line 4-7, length l is the sum of the latest path in the current thread and the 1^{st} paths from other threads, a possible least boundary cost for this latest outcome. If l_i is greater than $\mathcal{L}$, indicating all subsequent outcomes of the current thread must form a boundary that has a length greater than $\mathcal{L}$, thus we stop this thread. Function C is for obtaining the cost of path. In Line 10, h-signatures of all shortest paths in different homotopy classes are stored in the set $\mathcal{H}_i$ for later boundary validation. Here we introduce a function P for later use to get a part of the boundary such that $P(\mathbf{v}_s, \mathbf{v}_g, \mathfrak{h}_g^i)$ returns the shortest path from $\{\mathbf{v}_s, \text{`'}\}$ to $\{\mathbf{v}_g, \mathfrak{h}_g^i\}$.

G. Valid Boundary Construction

After we finished searching in all threads, we need to find out all proper combinations that have an empty concatenation of h-signatures. Algorithm 2 retrieves one path's h-signature at a time from each h-signature set $\mathcal{H}_i$, n h-signatures in total, to check whether their full concatenation is empty in Line 6 and 7. If so, we store the whole boundary, ω, into the set of all boundaries, $\mathcal{W}$, shown in Line 11. Although the complexity of the algorithm rises exponentially with the number of control points or the size of set $\mathcal{H}_i$, we have limited number of control points and the upper bound of the length of cables during the search hence make it computationally feasible.

H. Area Computation

Every boundary is made up of a few vertices, $\omega = \{\mathbf{v}_1, \mathbf{v}_2, \ldots, \mathbf{v}_n\}$, where vertex, $\mathbf{v}_i = \{x_i, y_i\}$, is either a control point or a vertex of the polygon obstacles. We use the following formula to compute the area of the workspace: $A(\omega) = \frac{1}{2}\left|(x_1y_2 - y_1x_2) + (x_2y_3 - y_2x_3) + \cdots + (x_ny_1 - y_nx_1)\right|$.

IV. Workspace Optimization

The methods described in this paper was implemented in C++ and Discrete Optimal Search Library (DOSL) [16]. In the sections below we mostly use 200 $\times$ 200 environment with two convex and one concave polygon as obstacles. Four control points are placed at each corner of the environment.

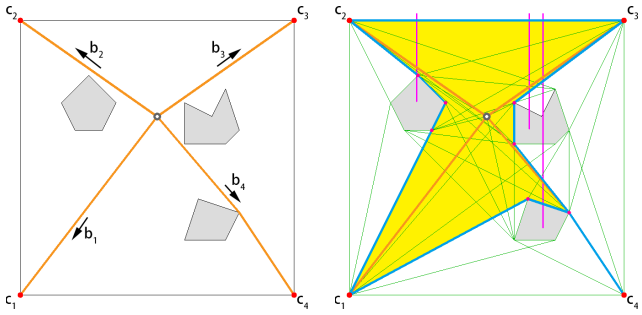
A. Computing Workspace's Boundary and Area from Initial Cable Configuration

If we have the initial cable configuration, we are able to compute the corresponding workspace. The cable b_i is given in form of vertices in visibility graph, going from the robot

Algorithm 2 Getting valid closed boundary

Input: Set of shortest paths in different homotopy classes, $\mathcal{P}_i = \{\tau_{i1}, \tau_{i2}, \dots\}$; set of h-signatures, $\mathcal{H}_i = \{\mathfrak{h}_{i1}, \mathfrak{h}_{i2}, \dots\}$, $\mathfrak{h}_{ij} = h(\tau_{ij})$;
Output: Set of valid closed boundary, $\mathcal{W}$;

$m \leftarrow 1$
for all $\mathfrak{h}_{1i}$ in $\mathcal{H}_1$ **do**
  for all $\mathfrak{h}_{2j}$ in $\mathcal{H}_2$ **do**
    ...
    for all $\mathfrak{h}_{nk}$ in $\mathcal{H}_n$ **do**
      $q \leftarrow \mathfrak{h}_{1i} \circ \mathfrak{h}_{2j} \circ \cdots \circ \mathfrak{h}_{nk}$
      if $q =$ ' ' **then**
        if (any of $\tau_{1i}, \tau_{2j}, \dots, \tau_{nk}$ self-tangles) **then**
          continue loop
        end if
        Insert $\omega_m = \{\tau_{1i}, \tau_{2j}, \dots, \tau_{nk}\}$ into $\mathcal{W}$
        $m \leftarrow m + 1$
      end if
    end for
    ...
  end for
end for



(a) Input: initial cable configuration and robot, control points and obstacles (b) Output: closed boundary and the area: 16200

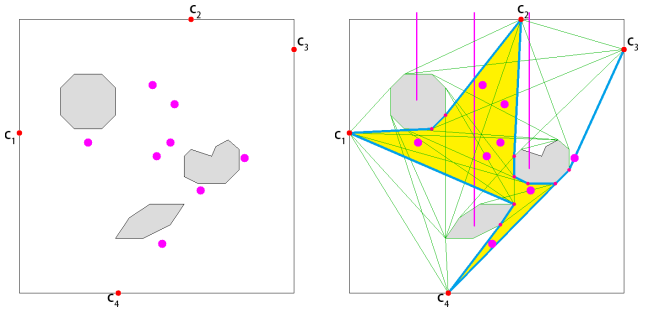
Fig. 7: The workspace from given cable configuration

to the i^{th} control point. We need goal h-signatures $\mathfrak{h}_g^i$ of pairs of cables for searching for corresponding boundaries. We concatenate the inverse of the i^{th} cable's h-signature with the $i+1^{\text{th}}$ cable's, $\mathfrak{h}_g^i = (h(b_i))^{-1} \circ h(b_{i+1})$, where superscript "$-1$" indicates inverse operation explained in Section II-C.

Use function $P(\mathbf{c}_i, \mathbf{c}_{i+1}, \mathfrak{h}_g^i)$ to get all corresponding shortest paths, then concatenate them into a closed boundary ω. For we have shown that cables can deform continuously into a closed boundary, no need to check the concatenation of goal h-signatures. In the end compute the area of this boundary, $A(\omega)$. An example is shown in Fig. 7.

B. Maximization of the Workspace Covering Multiple Task Points

If we expect the robot to perform multiple tasks at static points in the environment, we should choose an appropriate initial cable configuration which can generate a workspace covering all task points. For this kind of problem, we need to use H-signature (homology signature) to check if all the task points are inside the workspace. If the components in the H-signature of boundary corresponding to task points are non-zero, that vertex is inside the boundary. If we get an H-signature that does not have any zero component, all task



(a) Input: task points (in purple), control points (in red) and obstacles (b) Output: only one valid boundary when $\mathcal{L} = 900$

Fig. 8: Planning of the workspace that covers multiple task points

points are enclosed. After getting all workspaces that satisfy the criteria and the areas of them, the algorithm returns the one that has the largest area. A case is shown in Fig. 8.

C. Maximization of Expected Workspace's Area in an Environment with Moving Obstacle

In real-world scenarios, despite fixed control points, there could be moving obstacles. Although uncertain about the future locations of the obstacles, if we have prior knowledge of probabilities associated with different configurations of obstacles in the environment, we can choose a cable configuration with the max expectation of workspace's area.

1) Boundaries Change upon Obstacle Reconfiguration: If the current workspace's boundary is ω_m, the m^{th} one in set $\mathcal{W}$ we previously established, when the obstacles move from current configuration to the j^{th} potential configuration, ω_m deforms as well into a new boundary ω_m^j , where $\omega_m = \{\tau_{m1}, \tau_{m2}, \dots, \tau_{mn}\}$ and $\omega_m^j = \{\tau_{m1}^j, \tau_{m2}^j, \dots, \tau_{mn}^j\}$ (for example, the initial configuration of Fig. 10a deforms into Fig.s 10b, 10c and 10d when the obstacles move). Because control points do not move and cables do not intersect obstacles, τ_{mk} has the same start and end vertices as τ_{mk}^j and is homotopic to τ_{mk}^j, where $\tau_{mk} \in \omega_m$, $\tau_{mk}^j \in \omega_m^j$, and $k = 1, 2, \dots, n$. Based on the homotopy invariants, we are able to search for τ_{mk}^j,

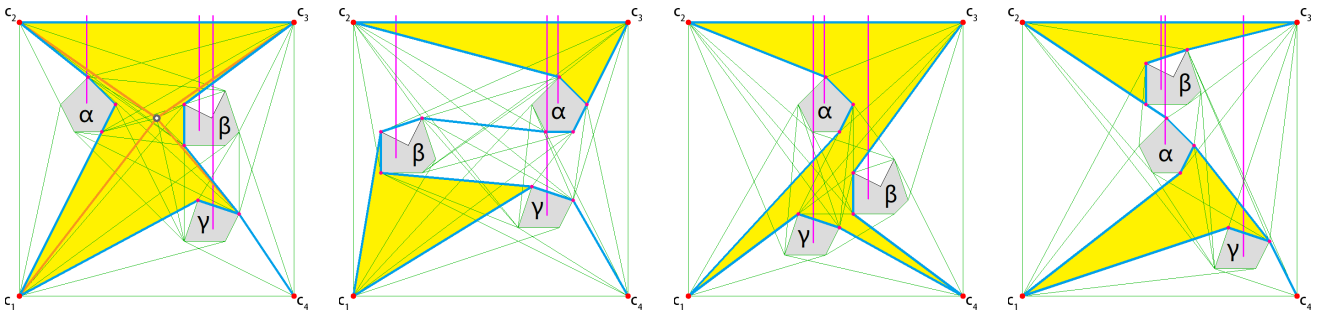
$$\tau_{mk}^j = P(\mathbf{c}_k, \mathbf{c}_{k+1}, R_j(h(\tau_{mk}))) \tag{1}$$

thus the new boundary in the j^{th} obstacle configuration is

$$\omega_m^j = \mathbf{P}_j(\omega_m) = \left\{ \tau \,\middle|\, \begin{array}{l} \tau = P(\mathbf{c}_k, \mathbf{c}_{k+1}, R_j(h(\tau_{mk}))), \\ \text{where } \tau_{mk} \in \omega_m, k = 1, 2, \dots, n \end{array} \right\} \tag{2}$$

where function $R_j(\cdot)$ is for revising h-signatures for the j^{th} potential configuration. We use $R_j(h(\tau_{mk}))$ instead of $h(\tau_{mk})$ because sometimes $h(\tau_{mk}^j) = h(\tau_{mk})$ may not hold (Fig. 9). The motion of obstacles could be so significant that the rays emanating from obstacles interchange of their coordinates. As a result, although τ_{mk}^j and τ_{mk} belong to the same homotopy class, the h-signatures of them in terms of crossing rays may be different. Hence we need to revise h-signatures.

2) Revision of h-signatures: We revise h-signatures depending on how the obstacles move. The revision applies in four steps: add/remove, interchange, insert and simplify. The following is an example of revision triggered by the representative point ζ_α crossing the ray of the representative point ζ_β from left to right, namely, the trajectory along which ζ_α moves has an h-signature of 'β^{-1}', shown in Fig. 9.



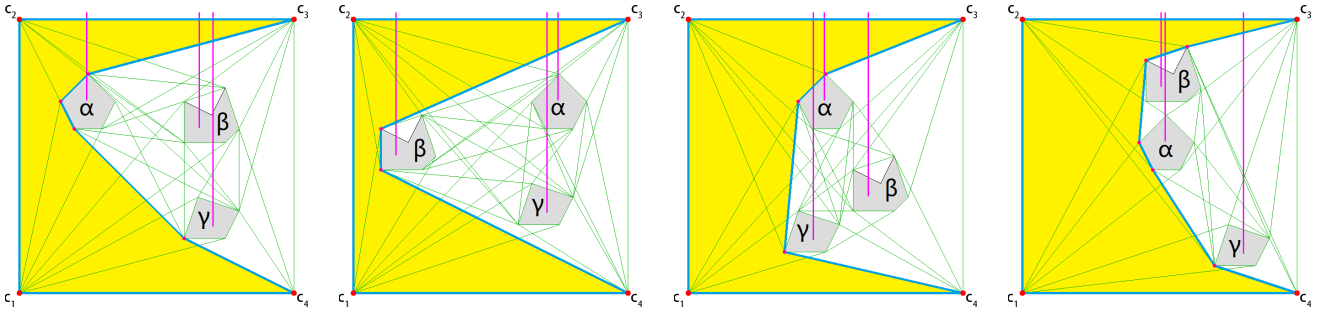
(a) Current configuration, area: 16200, max cable length sum: 1138.39

(b) Potential configuration 1, $\mathfrak{p}_1$ = 0.5, area: 10250, max cable length sum: 2155.56

(c) Potential configuration 2, $\mathfrak{p}_2$ = 0.3, area: 11400, max cable length sum: 1241.1

(d) Potential configuration 3, $\mathfrak{p}_3$ = 0.2, area: 9450, max cable length sum: 1364.59

Fig. 10: Boundary of Fig. 7b changes in potential configurations, expectation: 10435, max cable length sum: 2155.56



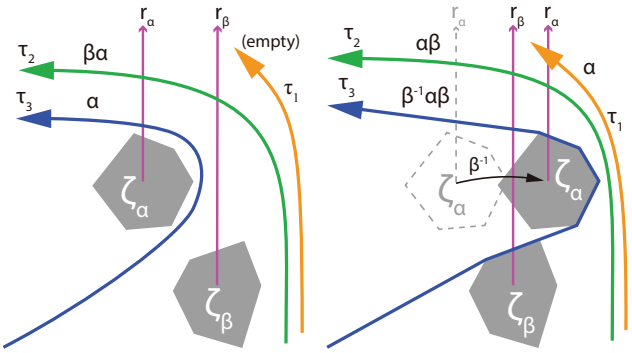
(a) Current configuration, area: 19300, max cable length sum: 1435.66

(b) Potential configuration 1, $\mathfrak{p}_1$ = 0.5, area: 19300, max cable length sum: 1488.13

(c) Potential configuration 2, $\mathfrak{p}_2$ = 0.3, area: 20100, max cable length sum: 1374.21

(d) Potential configuration 3, $\mathfrak{p}_3$ = 0.2, area: 22925, max cable length sum: 1283.95

Fig. 11: A boundary with max expectation (20265) in an environment with moving obstacles, max cable length sum: 1488.13



(a) Before significant motion of obstacle

(b) Obstacle ζ_α right crosses ζ_β's ray r_β along trajectory of 'β^{-1}'

Fig. 9: h-signature revision

a) Add/remove: When the representative point ζ_α's ray r_α moves from the left of the end vertex of one path to its right, we add a corresponding letter 'α' at the back of the path's h-signature, like τ_1 in Fig. 9. Likewise, if it is the start vertex that any representative point's ray crosses, add/remove that letter at the front of path's h-signature.

b) Interchange: In the h-signature of the path, when 'α' is next to 'β', we interchange positions of 'α' and 'β', like τ_2 in Fig. 9. Likewise, if 'α^{-1}' is next to 'β^{-1}', interchange positions of 'α^{-1}' and 'β^{-1}'.

c) Insert pair: If 'α' or 'α^{-1}' appears alone (its previous and next letters are not 'β' or 'β^{-1}'), insert 'β^{-1}' into its left and 'β' into its right, like τ_3 in Fig. 9.

d) Simplify: Check if a letter and its inverse appear side-by-side. If so, cancel both of them. Keep checking until there is no such a case.

On the contrary, when representative point ζ_α crossed r_β's right to left, going along 'β', do "add/remove", "interchange" and "simplify" in the same way described above; but when coming to "insert pair", if there is an lone 'α' or 'α^{-1}', we insert 'β' into its left and 'β^{-1}' into its right.

A representative point may cross multiple rays. Thus we have to decompose the crossing into several stages of coordinates interchanges, then revise h-signatures stage by stage till reaching the final configuration. E.g. in Fig. 11, we can split the transformation from Fig. 11a to 11b into two stages: firstly ζ_α going along a trajectory 'β^{-1}', secondly ζ_α going along 'γ^{-1}'; transformation from Fig. 11a to 11b in two stages: ζ_β going along 'γ^{-1}', then ζ_α going along 'γ^{-1}'; transformation from Fig. 11a to 11d in one stage: ζ_β going along 'α'. Here we use the boundary in Fig. 7b as a current configuration to illustrate how h-signatures are revised for potential configurations, shown in TABLE I, and how the boundary changes accordingly in Fig. 10.

TABLE I: Stage-by-stage h-signature revision for paths in Fig. 10

Cfg	Stg	$\widetilde{\mathbf{c}_1\mathbf{c}_2}$	$\widetilde{\mathbf{c}_2\mathbf{c}_3}$	$\widetilde{\mathbf{c}_3\mathbf{c}_4}$	$\widetilde{\mathbf{c}_4\mathbf{c}_1}$
Cur.	N/A	α	$\alpha^{-1}\beta^{-1}\gamma^{-1}$	$\gamma\beta\gamma^{-1}$	γ
P. 1	1	$\beta^{-1}\alpha\beta$	$\beta^{-1}\alpha^{-1}\gamma^{-1}$	$\gamma\beta\gamma^{-1}$	γ
	2	$\beta^{-1}\gamma^{-1}\alpha\gamma\beta$	$\beta^{-1}\gamma^{-1}\alpha^{-1}$	$\gamma\beta\gamma^{-1}$	γ
P. 2	1	α	$\alpha^{-1}\gamma^{-1}\beta^{-1}$	β^{*}	γ
	2	$\gamma^{-1}\alpha\gamma$	$\gamma^{-1}\alpha^{-1}\beta^{-1}$	β	γ
P. 3	1	α	$\beta^{-1}\alpha^{-1}\gamma^{-1}$	$\gamma\alpha\beta\alpha^{-1}\gamma^{-1}$	γ

* Initially it was '$\beta\gamma\gamma^{-1}$' before simplification.

3) Expectation Computation: The probability of the i^{th} potential configuration is denoted as $\mathfrak{p}_i, i = 1, 2, \ldots, \rho$,

where ρ is the number of potential configurations and $\sum_{i=1}^{\rho} \mathfrak{p}_i = 1$. The area expectation of the m^{th} valid boundary, $\omega_m = \{\tau_{m1}, \tau_{m2}, \ldots, \tau_{mn}\} \in \mathcal{W}$, is $E(\omega_m) = \sum_{j=1}^{\rho} A(\mathbf{P}_j(\omega_m))\mathfrak{p}_j$. Next we can choose a valid boundary with the max expectation from set $\mathcal{W}$.

4) Maximum Cable Length Computation: In order to ensure that cables are long enough for all potential configurations, we need to know the maximum length of each cable. Since the workspace is a polygon that must have convex vertices at control points, the max length from the i^{th} control point is used when it reaches one of the other control points. In a particular obstacle and workspace configuration, that is $\max\{C(\widetilde{\mathbf{c}_i\mathbf{c}_1}), C(\widetilde{\mathbf{c}_i\mathbf{c}_2}), ..., C(\widetilde{\mathbf{c}_i\mathbf{c}_{i-1}}), C(\widetilde{\mathbf{c}_i\mathbf{c}_{i+1}}), ..., C(\widetilde{\mathbf{c}_i\mathbf{c}_n})\}$, where $\widetilde{\mathbf{c}_i\mathbf{c}_j}$ is the shortest path between the i^{th} and the j^{th} control points inside the workspace. To get goal h-signatures for searching, we concatenate the h-signatures tracing the boundary from the i^{th} control point to the j^{th} in clockwise or counterclockwise direction. For instance, if $j > i$, the shortest path between $\mathbf{c}_i$ and $\mathbf{c}_j$ is $\widetilde{\mathbf{c}_i\mathbf{c}_j} = P(\mathbf{c}_i, \mathbf{c}_j, h(\tau_i) \circ h(\tau_{i+1}) \circ \cdots \circ h(\tau_{j-1}))$. In Fig. 10 and Fig. 11, the sum of four maximum cable lengths were calculated for each configuration.

V. Algorithm Design for Robot Path Planning within the Workspace

We move a robot from one point to another by controlling the motors to change the cables' lengths at each one's desired speed. The cable control algorithm has the following inputs: h-augmented graph of environment, coordinate of start and goal vertices, $\mathbf{v}_s$ and $\mathbf{v}_g$, coordinate of control points (x_c^i, y_c^i), initial cable configuration b_i, and desired robot speed v_r; and outputs: shortest trajectory from the start vertex to the end vertex, velocities of each cable v_c^i over time.

A. h-signature of Shortest Trajectory within Boundary

With a correct h-signature of the desired trajectory, we can ensure the search outcome is within the workspace. We chose a path constituting of parts of the boundary (see Fig. 12) that can be easily obtained from the said inputs and can be continuously deformed into (homotopic to) the desired shortest path connecting the start and goal points. If both start and goal vertices are at the boundary, we can directly construct the h-signature of a trajectory between two vertices along the boundary. If they are inside the workspace, we can use *alternative vertices* – points on the boundary adjacently above ($\mathbf{v}'_s$, $\mathbf{v}'_g$) or below ($\mathbf{v}''_s$, $\mathbf{v}''_g$) start and goal vertices, shown in Fig. 12. $\widetilde{\mathbf{v}_s\mathbf{v}_g}$ inside the workspace can continuously deform into $\widetilde{\mathbf{v}_s\mathbf{v}'_s\mathbf{v}'_g\mathbf{v}_g}$ partially at the boundary – they are in identical homotopy class: $h(\widetilde{\mathbf{v}_s\mathbf{v}_g}) = h(\widetilde{\mathbf{v}_s\mathbf{v}'_s\mathbf{v}'_g\mathbf{v}_g}) = h(\widetilde{\mathbf{v}_s\mathbf{v}'_s}) + h(\widetilde{\mathbf{v}'_s\mathbf{v}'_g}) + h(\widetilde{\mathbf{v}'_g\mathbf{v}_g})$. Since path $\widetilde{\mathbf{v}_s\mathbf{v}'_s}$ and $\widetilde{\mathbf{v}'_g\mathbf{v}_g}$ are vertical, they do not cross any rays, having empty h-signatures. Hence, $h(\widetilde{\mathbf{v}_s\mathbf{v}_g}) = h(\widetilde{\mathbf{v}'_s\mathbf{v}'_g})$. Also, we can use $\mathbf{v}''_s$ and $\mathbf{v}''_g$ (below $\mathbf{v}_s$ and $\mathbf{v}_g$, respectively) instead. See more in Fig. 12. In addition, we can apply this method for obtaining cables' new h-signatures in the next subsection while the robot is moving, where $\mathbf{v}_s = \mathbf{c}_i$ and $\mathbf{v}_g = \mathbf{v}_r$ (the robot's location). The shortest trajectory from start vertex $\mathbf{v}_s$ to goal vertex $\mathbf{v}_g$ is $\widetilde{\mathbf{v}_s\mathbf{v}_g} = P(\mathbf{v}_s, \mathbf{v}_g, h(\widetilde{\mathbf{v}'_s\mathbf{v}'_g}))$.

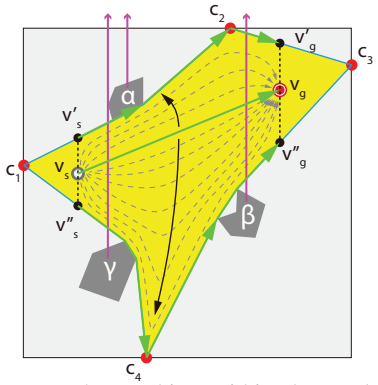
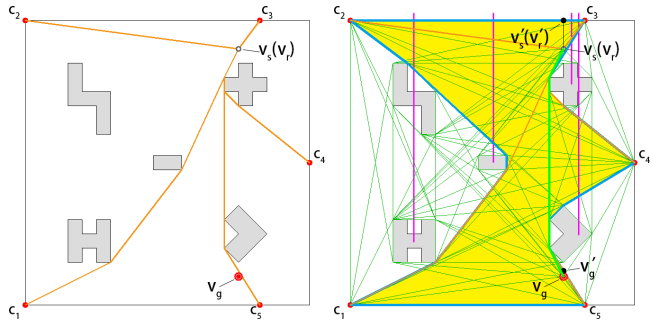


Fig. 12: Shortest path searching within the workspace by using alternative vertices ($\mathbf{v}'_s$, $\mathbf{v}''_s$, $\mathbf{v}'_g$, $\mathbf{v}''_g$) at the boundary instead of start and goal vertices ($\mathbf{v}_s$, $\mathbf{v}_g$) to get the goal h-signature.



(a) Input (b) Output

Fig. 13: Goal-directed path planning within the workspace. Path is colored in green. Alternative points at the boundary in black.

B. Cable Velocity

The cable velocity is the projection of the robot's velocity in the cable's direction. Denote coordinates of the i^{th} control point and robot as (x_c^i, y_c^i) and (x_r, y_r), and the speed of robot as (v_r^x, v_r^y). The velocity of cable attached to the i^{th} control point is

$$v_c^i = \frac{(x_c^i - x_r)v_r^x + (y_c^i - y_r)v_r^y}{\sqrt{(x_c^i - x_r)^2 + (y_c^i - y_r)^2}}.$$

where v_c^i is a scalar and its positive direction is from the robot to the i^{th} control point, and robot coordinates (x_r, y_r) are integrals of its velocity over time plus start coordinates. Sometimes the cable may go around an obstacle, which makes it no longer straight. Instead we use a temporary control point which is at the nearest turn of cables to the robot. Thus we search for a new cable configuration based on the robot's position. We construct h-signatures from robot's alternative vertex $\mathbf{v}'_r$ to each control point along the boundary in a same direction. In the test shown in Fig. 13, we got $h(\widetilde{\mathbf{v}'_r\mathbf{c}_3})$ first, and then got $h(\widetilde{\mathbf{v}'_r\mathbf{c}_4})$, $h(\widetilde{\mathbf{v}'_r\mathbf{c}_1})$ and $h(\widetilde{\mathbf{v}'_r\mathbf{c}_2})$.

VI. Path Planing for Multi-task Accomplishment

Suppose the cable-controlled robot needs to execute M unordered tasks, each described as static points in the workspace, before it arrives at the goal vertex. We need to solve the *traveling salesman problem* – find the shortest trajectory that goes through these static task points. One way to accomplish this is to construct a task indicator augmented graph and search in it [17]. A task indicator is a string of M binary digits, denoted by β, in which each bit is a flag or indicator of whether

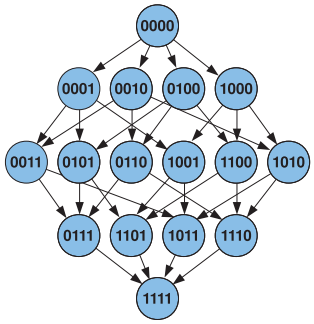
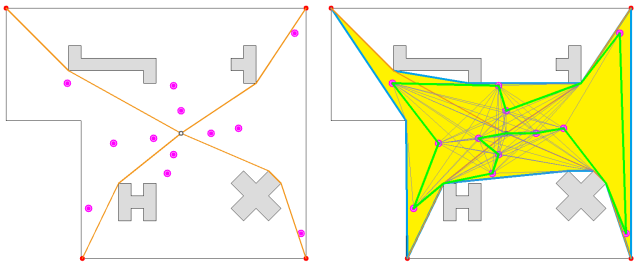


Fig. 14: The task graph $\mathcal{Y}$ showing the possible transitions of the task indicator, β, for 4 tasks.



(a) Input: 12 tasks to be finished (b) Output: a shortest trajectory

Fig. 15: Using t-augmented graph for multi-task planning within the workspace in a non-convex environment. The robot returns to the start after finishing all tasks. A shortest path is colored in green.

the corresponding task has been completed. For example, if there are 4 tasks to be finished, '0101' means that the 1st and the 3rd task is finished while others are not. The robot must set off at the start vertex with $\beta = 0000$ and arrive at goal vertex with $\beta = 1111$. The task indicator augmented graph $\mathcal{G}_t = (\mathcal{V}_t, \mathcal{E}_t)$ is defined as

1) $\mathcal{V}_t = \{(\mathbf{v}, \beta) | \mathbf{v} \in \mathcal{V}, \beta \in \mathcal{Y}\}$
2) An edge $\{(\mathbf{v}, \beta) \rightarrow (\mathbf{v}', \beta')\} = P(\mathbf{v}, \mathbf{v}', h(\widetilde{(\mathbf{v})'(\mathbf{v}')'}))$ is in $\mathcal{E}_t$ for $(\mathbf{v}, \beta) \in \mathcal{V}_t$ and $(\mathbf{v}', \beta') \in \mathcal{V}_t$, $(\mathbf{v})'$ and $(\mathbf{v}')'$ being alternative vertices of $\mathbf{v}$ and $\mathbf{v}'$ respectively, iff one of the followings holds
   a) The edge $\{(\mathbf{v}, \beta) \rightarrow (\mathbf{v}', \beta')\} \in \mathcal{E}$, and $\mathbf{v}' \notin \{\tau_l |$ the l^{th} bit of β is $0\}$, and $\beta = \beta'$
   b) The edge $\{(\mathbf{v}, \beta) \rightarrow (\mathbf{v}', \beta')\} \in \mathcal{E}$, and $\mathbf{v}' \in \{\tau_l |$ the l^{th} bit of β is $0\}$, with $\mathbf{v}' = \tau_\lambda$, and $\beta \rightarrow \beta' \in \mathcal{Y}$ such that the λ^{th} bit of β' is 1.
3) The cost associated with an edge $\{(\mathbf{v}, \beta) \rightarrow (\mathbf{v}', \beta')\}$ is the same as that associated with edge $\{\mathbf{v} \rightarrow \mathbf{v}'\} \in \mathcal{E}$.

In the example of Fig. 15 we place 12 task points inside the workspace, the start vertex in the middle overlapping the end vertex. We find the workspace first then build the t-augmented graph to find the shortest path that visits all the task points.

VII. Conclusion

In this paper we have introduced a novel and efficient method for workspace planning for a cable-control robot in a cluttered environment, using the h-signature augmented graph. We have presented algorithms for workspace planning and trajectory planning based on it. Also, we have developed several applications and demonstrated those through simulations in both static and changing environment. More applications such as optimization of location of control points and extension to a full 3-D workspace are within the scope of future research.

References

[1] N. Sydney, B. Smyth, and D. A. Paley, "Dynamic control of autonomous quadrotor flight in an estimated wind field," in *52nd IEEE Conference on Decision and Control*, Dec 2013, pp. 3609–3616.

[2] Y. Mei, Y.-H. Lu, Y. C. Hu, and C. S. G. Lee, "A case study of mobile robot's energy consumption and conservation techniques," in *ICAR '05. Proceedings., 12th International Conference on Advanced Robotics, 2005.*, July 2005, pp. 492–497.

[3] K. Pratt, R. Murphy, J. Burke, J. Craighead, C. Griffin, and S. Stover, "Use of tethered small unmanned aerial system at berkman plaza ii collapse," in *Safety, Security and Rescue Robotics, 2008. SSRR 2008. IEEE International Workshop on*, 2008, pp. 134–139.

[4] N. Michael, S. Shen, K. Mohta, Y. Mulgaonkar, V. Kumar, K. Nagatani, Y. Okada, S. Kiribayashi, K. Otake, K. Yoshida, K. Ohno, E. Takeuchi, and S. Tadokoro, "Collaborative mapping of an earthquake-damaged building via ground and aerial robots," *Journal of Field Robotics*, vol. 29, no. 5, pp. 832–841, 2012.

[5] R. H.Teshnizi and D. A. Shell, "Planning motions for a planar robot attached to a stiff tether," in *2016 IEEE International Conference on Robotics and Automation (ICRA)*, May 2016, pp. 2759–2766.

[6] S. Kim, S. Bhattacharya, and V. Kumar, "Path planning for a tethered mobile robot," in *Proceedings of IEEE International Conference on Robotics and Automation (ICRA)*, Hong Kong, China, May 31 - June 7 2014.

[7] R. H. Teshnizi and D. A. Shell, "Computing cell-based decompositions dynamically for planning motions of tethered robots," in *2014 IEEE International Conference on Robotics and Automation (ICRA)*, May 2014, pp. 6130–6135.

[8] 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.

[9] B. R. Donald, L. Gariepy, and D. Rus, "Distributed manipulation of multiple objects using ropes," in *Proceedings of the 2000 IEEE International Conference on Robotics and Automation, ICRA*, San Francisco, CA, USA, April 24-28 2000, pp. 450–457.

[10] S. Bhattacharya, S. Kim, H. Heidarsson, G. S. Sukhatme, and V. Kumar, "A topological approach to using cables to separate and manipulate sets of objects," *The International Journal of Robotics Research*, vol. 34, no. 6, pp. 799–815, 2015.

[11] P. Cheng, J. Fink, V. Kumar, and J.-S. Pang, "Cooperative towing with multiple robots," *ASME Journal of Mechanisms and Robotics*, vol. 1, no. 1, pp. 011 008–011 008–8, 2008.

[12] D. Theodorakatos, E. Stump, and V. Kumar, "Kinematics and pose estimation for cable actuated parallel manipulators," in *Proceedings of the ASME International Design Engineering Technical Conferences and Computers and Information in Engineering Conference*, vol. 8, no. PART B, 2007, pp. 1053–1062.

[13] 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.

[14] R. W. Ghrist, *Elementary applied topology*. Createspace, 2014.

[15] S. Bhattacharya, D. Lipsky, R. Ghrist, and V. Kumar, "Invariants for homology classes with application to optimal search and planning problem in robotics," *Annals of Mathematics and Artificial Intelligence*, vol. 67, no. 3, pp. 251–281, Mar 2013.

[16] S. Bhattacharya, "Discrete optimal search library (dosl): A template-based c++ library for discrete optimal search," 2017, available at https://github.com/subh83/DOSL. [Online]. Available: https://github.com/subh83/DOSL

[17] S. Bhattacharya, M. Likhachev, and V. Kumar, "Multi-agent path planning with multiple tasks and distance constraints," in *Robotics and Automation (ICRA), 2010 IEEE International Conference on*. IEEE, 2010, pp. 953–959.