1-1-2013

Robot Motion Planning Under Topological Constraints

Soonkyum Kim
University of Pennsylvania, soonkyum@seas.upenn.edu

Robot Motion Planning Under Topological Constraints

Abstract
My thesis addresses the the problem of manipulation using multiple robots with cables. I study how robots with cables can tow objects in the plane, on the ground and on water, and how they can carry suspended payloads in the air. Specifically, I focus on planning optimal trajectories for robots.

Path planning or trajectory generation for robotic systems is an active area of research in robotics. Many algorithms have been developed to generate path or trajectory for different robotic systems. One can classify planning algorithms into two broad categories. The first one is graph-search based motion planning over discretized configuration spaces. These algorithms are complete and quite efficient for finding optimal paths in cluttered 2-D and 3-D environments and are widely used [48]. The other class of algorithms are optimal control based methods. In most cases, the optimal control problem to generate optimal trajectories can be framed as a nonlinear and non convex optimization problem which is hard to solve. Recent work has attempted to overcome these shortcomings [68]. Advances in computational power and more sophisticated optimization algorithms have allowed us to solve more complex problems faster. However, our main interest is incorporating topological constraints. Topological constraints naturally arise when cables are used to wrap around objects. They are also important when robots have to move one way around the obstacles rather than the other way around. Thus I consider the optimal trajectory generation problem under topological constraints, and pursue problems that can be solved in finite-time, guaranteeing global optimal solutions.

In my thesis, I first consider the problem of planning optimal trajectories around obstacles using optimal control methodologies. I then present the mathematical framework and algorithms for multi-robot topological exploration of unknown environments in which the main goal is to identify the different topological classes of paths. Finally, I address the manipulation and transportation of multiple objects with cables. Here I consider teams of two or three ground robots towing objects on the ground, two or three aerial robots carrying a suspended payload, and two boats towing a boom with applications to oil skimming and clean up. In all these problems, it is important to consider the topological constraints on the cable configurations as well as those on the paths of robot. I present solutions to the trajectory generation problem for all of these problems.

Degree Type
Dissertation

Degree Name
Doctor of Philosophy (PhD)

Graduate Group
Mechanical Engineering & Applied Mechanics

First Advisor
Vijay Kumar

Keywords
Motion Planning, Robot, Topology

Subject Categories
Mechanical Engineering | Robotics

ROBOT MOTION PLANNING UNDER TOPOLOGICAL CONSTRAINTS

Soonkyum Kim

A DISSERTATION

in

Mechanical Engineering and Applied Mechanics

Presented to the Faculties of the University of Pennsylvania

in

Partial Fulfillment of the Requirements for the

Degree of Doctor of Philosophy

2013

Supervisor of Dissertation

_____

Vijay Kumar, Professor
Department of Mechanical Engineering and Applied Mechanics

Graduate Group Chairperson

_____

Prashant K. Purohit, Associate Professor
Department of Mechanical Engineering and Applied Mechanics

Dissertation Committee
Mark Yim, Professor, Department of Mechanical Engineering and Applied Mechanics
Vijay Kumar, Professor, Department of Mechanical Engineering and Applied Mechanics
Robert Ghrist, Professor, Department of Mathematics
Maxim Likhachev, Assistant Research Professor, Robotics Institute, Carnegie Mellon University

ROBOT MOTION PLANNING UNDER TOPOLOGICAL CONSTRAINTS

Acknowledgments

First, I would like to express my sincere gratitude toward my advisor, Prof. Vijay Kumar for his ceaseless, dedicated and invaluable advise and guidance during the course of my graduate studies at GRASP Laboritory in the University of Pennsylvania. I would also like to heartily thank Prof. Mark Yim and Prof. Maxim Likhachev for their valuable time and effort in serving as my dissertation committee members. I also like to thank Prof. Robert Ghrist, not only for being on my dissertation committee, but also for collaborating on the topological exploration problem. I would also like to thank Dr. Subhrajit Bhattacharya for valuable discussions and for collaborating on many of the problems presented in this thesis. I would like to express my gratitude toward Prof. Frank C. Park who introduced me to the field of robotics.

My sincere appreciation goes to Dr. Koushil Sreenath for his collaboration on the problem of optimal trajectory generation under topological constraints. I would also like to thank Dr. Nathan Michael for collaborating on the aerial manipulation problem, and Dr. Peng Cheng for collaborating on the cooperative towing problem. My sincere thanks goes to Dr. Jornathan Fink for his collaboration on both the aforesaid problems. I would like to thank Prof. Gaurav Sukhatme and Hordur Heidarsson of USC for their collaboration on the field experiments in the problem related to manipulation of a set of objects.

Finally, I would like to thank my family. I would like to thank my brother Sanggyum, with whom I have enjoyed endless discussions about various control problems. I thank my sister-in-law, Christina Kang-Yi, and her husband, John Chanu Yi for their warm support and concern. I sincerely appreciate my parents' devotion and sacrifice. I would like to express my gratitude toward my parents-in-law for their devotion and for their adorable daughter. And, I would like to epxress my love to my little angels, David and Rachael, and my wife, Minki.

ABSTRACT

ROBOT MOTION PLANNING UNDER TOPOLOGICAL CONSTRAINTS

Soonkyum Kim

Vijay Kumar

My thesis addresses the the problem of manipulation using multiple robots with cables. I study how robots with cables can tow objects in the plane, on the ground and on water, and how they can carry suspended payloads in the air. Specifically, I focus on planning optimal trajectories for robots.

Path planning or trajectory generation for robotic systems is an active area of research in robotics. Many algorithms have been developed to generate path or trajectory for different robotic systems. One can classify planning algorithms into two broad categories. The first one is graph-search based motion planning over discretized configuration spaces. These algorithms are complete and quite efficient for finding optimal paths in cluttered 2-D and 3-D environments and are widely used [48]. The other class of algorithms are optimal control based methods. In most cases, the optimal control problem to generate optimal trajectories can be framed as a nonlinear and non convex optimization problem which is hard to solve. Recent work has attempted to overcome these shortcomings [68]. Advances in computational power and more sophisticated optimization algorithms have allowed us to solve more complex problems faster. However, our main interest is incorporating topological constraints. Topological constraints naturally arise when cables are used to wrap around objects. They are also important when robots have to move one way around the obstacles rather than the other way around. Thus I consider the optimal trajectory generation problem under topological constraints, and pursue problems that can be solved in finite-time, guaranteeing global optimal solutions.

In my thesis, I first consider the problem of planning optimal trajectories around obstacles using optimal control methodologies. I then present the mathematical framework and algorithms for multi-robot topological exploration of unknown environments in which the main goal is to identify the different topological

classes of paths. Finally, I address the manipulation and transportation of multiple objects with cables. Here I consider teams of two or three ground robots towing objects on the ground, two or three aerial robots carrying a suspended payload, and two boats towing a boom with applications to oil skimming and clean up. In all these problems, it is important to consider the topological constraints on the cable configurations as well as those on the paths of robot. I present solutions to the trajectory generation problem for all of these problems.

Contents

List of Figures

Chapter 1

Introduction

1.1 Introduction

Path planning or trajectory generation for robotic systems is an active area of research in robotics. Many algorithms have been developed to generate path or trajectory for different robotic systems. One can classify planning algorithms into two broad categories. The first one is graph-search based motion planning over discretized configuration spaces. These algorithms are complete and quite efficient for finding optimal paths in cluttered 2-D and 3-D environments and are widely used [48]. The other class of algorithms are optimal control based methods. In most cases, the optimal control problem to generate optimal trajectories can be framed as a nonlinear and non convex optimization problem which is hard to solve. Recent work has attempted to overcome these shortcomings [68]. Advances in computational power and more sophisticated optimization algorithms have allowed us to solve more complex problems faster. However, our main interest is incorporating topological constraints. Topological constraints naturally arise when cables are used to wrap around objects. They are also important when robots have to move one way around the obstacles rather than the other way around. Thus I consider the optimal trajectory generation problem under topological constraints, and pursue problems that can be solved in finite-time, guaranteeing global optimal solutions.

Early works on path planning or trajectory generation problem discussed the algorithms for mobile robots on the plane. Time optimal trajectories of differential drive mobile robots, which can rotate in position, can be computed by following sequence of primitive motions of rotating in position and straight moves [5]. If the vehicle has minimum turning radius, then the shortest path will consist of a sequence of arcs and straight lines [79]. Also, in [37] the authors find smooth shortest path with restriction on average curvature. Elastic bands introduced the algorithm to deform the path mobile robots in dynamic environments [78], which has been extended to mobile manipulation problem [22, 103]. Of course, finding smooth optimal trajectory is still one of active research topic. Recently, such problem has been extended to 3D path planning of UAVs [64].

However, the development of communication and sensing allows us to control multiple robots to accomplish complicated tasks which are too hard or take too long for a single robot [8]. Search-and-rescue or exploration problem is one of suitable examples to show the necessity of utilizing multiple robots [11, 17, 87]. The most popular technique to efficiently deploy the groups of robots is frontier-based planning [102]. In frontier-based algorithm, two paths or trajectories are different if they reach different frontiers. However, the frontier is sensitive with sensor noise or resolution of the map. We are interested in robust algorithm to

distinguish the class of paths or trajectories. Also, we want to find optimal paths or trajectories in *different* classes.

Cables are widely used in mechanical systems to transfer actuator powers. However, humans utilize cables or ropes to carry or manipulate various objects and there has been active research on utilizing cables in robot planning and control. We can transport a payload by towing with cables. Usually, this method requires one or multiple robot to carry a single payload. We can manipulate a larger number of small objects by skimming via cable. For efficiently manipulate objects, it is necessary to consider the configuration of the cable in planning and control.

The topology can give us the answer of these problem. The paths or trajectories are different if they are in different topology classes. Also, the objects separation problem can be formulated to path planning problem with topological constraints of the cable configuration and robot paths.

In my thesis, I first consider the problem of planning optimal trajectories around obstacles using optimal control methodologies. I then present the mathematical framework and algorithms for multi-robot topological exploration of unknown environments in which the main goal is to identify the different topological classes of paths. Finally, I address the manipulation and transportation of multiple objects with cables. Here I consider teams of two or three ground robots towing objects on the ground, two or three aerial robots carrying a suspended payload, and two boats towing a boom with applications to oil skimming and clean up. I present solutions to the trajectory generation problem for all of these problems.

1.2 Literature review

Trajectory generation problem for robotic systems is one of the most active areas in robotics research. Some literatures focus on finding optimal trajectories in convex or unbounded spaces [6, 19]. However, the development of computational capacities allows algorithms for generating trajectories in cluttered, non-convex environments with kinematic and dynamic constraints in the form of constraints on communication, coverage, environment, time, etc (see kinodynamic planners [35], RRT trees [61], LQR trees [94], Elastic Roadmaps [103] and references within). Most of the algorithms are developed to find optimal trajectories satisfying feasibility constraints. However, there have also been considerable amount of research interest in algorithms for generating trajectories for multi-agent problems [47, 28, 105]. In such problems it is often required that each robot follows different trajectories to cover or sense the whole work space as in search-and-rescue or surveillance problems. This brings forth the necessity of finding trajectories in topologically different classes. This requires that we impose constraints on the homotopy classes of the trajectories accordingly. However, in many practical robotic problems, homology class constraints act a suitable and convenient substitute for homotopy class constraints [14].

Early attempts at classifying homotopy classes in two dimensions include geometric methods [50, 46], homotopy preserving probabilistic road-map constructions [83], and triangulation-based path planning [27]. Two trajectories are said to be homotopic if one can be continuously deformed to another without intersecting any obstacle. Each set of trajectories that are homotopic forms an equivalence class, called a homotopy class. Considering laser beams as reference lines, topology of the path can be used to localize agents, compare homotopy class of paths and find winding number of path [98], which can be extended to localization of multiple robots [96]. However, in many practical robotic problems, homology class constraints act as suitable and convenient substitutes for homotopy class constraints [15].

Exploration and mapping have been treated quite extensively in the robotics literature. The general problem can be formulated as finding the next best view or pose [77] to acquire information required to build a map of the environment [88]. In most settings, the spatial representation of the map is based on metric information. Indeed approaches like metric-based multi-robot coordinated exploration have been studied widely in the past [11, 17, 87]. In decision-theoretic approaches to exploration, mutual information and entropy are often used [87, 89, 85, 95] to guide robots to perform efficient exploration. Simpler approaches involving the identification of frontiers and segmentation representing the boundaries between unexplored and explored regions have also been widely used for deployment of robots in exploration and mapping of unknown or partially known environments [102, 41, 100].

The advantages of using ropes with robots for manipulation were demonstrated by Donald *et al* [34]. An interesting problem that arises in these settings is the modeling of the shape of the cable and the motion planning for the robots to control the position and shape of the cable. Motion planning for manipulation of rope-like flexible objects is discussed in [82]. The problem of entangling and disentangling knots and the motion planning for this problem has been addressed in [60].

From the standpoint of robotics, towing is an important manipulation process [63]. The kinematics and dynamics of cable-actuated, parallel manipulators, which have been studied extensively [75, 20, 92, 99]. However, this body of literature primarily addresses the control of the cable extensions or forces in order to manipulate the payload. In contrast, towing involves cables of fixed length where manipulation is accomplished by controlling the motions of the "pivot points" in the parallel manipulators. While manipulation using cables has been studied in the context of distributed manipulation [33, 31, 32], these papers do not address the mechanics or control of the cooperative manipulation task.

The use of robots to tow objects using cables is discussed in [53, 23]. In [23], Cheng *et al* establish the quasi-static towing problem with n cables has a unique solution under certain conditions. In other words, if the robot motions are known, there is instantaneously a unique object motion. An extension of these ideas leads to using a cable with its ends tied to robots to cage and tow objects. Indeed this method is widely used in skimming operations on water surfaces [81, 54]. A description of the dynamics of such systems and an analysis of the problem of cooperative skimming are provided in [12, 4]. However, this work does not explicitly address the manipulation of objects.

In [66], manipulation and transportation with three aerial robots permits full six-dimensional pose control of a cable-actuated payload in three-dimensions despite the fact that the system is underactuated and limited by unilateral tension constraints for specific robots-cables-platform configurations. Aerial towing, the manipulation of a payload suspended by a cable from a moving aerial robot, has been studied in [70]. It is quite clear that the control of all six degrees of freedom requires more than one aerial robot and multiple cables. The underlying mechanics in such systems is closely related to the mechanics of cable-actuated parallel manipulators in three dimensions. In both cases, the position and orientation of the suspended payload can only be obtained by solving the kinematic equations and the equations of static equilibrium. The equilibrium solutions are configurations in which the gravity wrench is equilibrated by the wrenches exerted by the n cables. This means that the lines of action of the n cables can only belong to certain subspaces which are linear complexes (for $n = 5$), linear congruences (for $n = 4$) or reguli (for $n = 3$) [51, 76]. For $n < 3$, it is not possible to achieve an arbitrary position and orientation.

The cooperative aerial manipulation problem is more related to cable-actuated parallel manipulators in three dimensions, where in the former the platform pose is affected by robot positions and in the latter pose

control is accomplished by varying the lengths of multiple cable attachments. These systems offer similar workspace [92, 99], control [73, 74], and analysis [20].

The problem of finding a hypersurface separating two types of objects is studied as part of statistical classification problems [18, 93]. However such methods are susceptible to finding curves that can have disjoint components, do not have guarantees on optimality, and are statistical in nature.

1.3 Contribution

The first contribution of this thesis is generating an optimal trajectory that minimizes an integral cost functional (which depends on the trajectory), while also respecting kinematic constraints of the system, avoiding obstacles, and constraining the trajectory to a particular topology class. Although several of these subproblems have been solved separately (see [35, 61, 94, 97, 13, 14]), there is no literature, to our knowledge, that addresses the combined problem described above. We suggest the trajectory generation problem under topology class constraints which can be formulated as a MIQP or QP. This method can be used for trajectory generation for differentially-flat systems [72] with a two-dimensional flat output space, such as a kinematic car [71], or a tricycle robot [7], which not only produces a trajectory respecting the homology constraint, but also provides the nominal feedforward forces (due to the differential-flatness property,) for use in feedback control for trajectory tracking.

The second contribution of this thesis is to present the mathematical framework and algorithms for multi-robot topological exploration of unknown environments in which the main goal is to identify the different topological classes of paths. We consider two-dimensional configuration spaces. At any point in time, the robot's map consists of known, partially-mapped obstacles. The unknown, yet-to-be-explored area is mapped to a single point, thus giving us a quotient space. The topological classes on the quotient space allows us to define topological classes of paths connecting a robot pose to the unknown region in the original configuration space. Robots explore this configuration space choosing different homology classes when confronted by obstacles or walls.

The last contribution of this thesis is to manipulate multiple objects with only two robots connected with a cable. We will demonstrate how to control two robots efficiently to manipulate selected objects in the presence of other objects. The first key contribution is a topological description of the problem of separating two sets of objects and the algebraic formulation of the separation problem. The second contribution is a complete motion planning algorithm that relies on graph search [25] to drive the robots in order to achieve separation and then transport the objects to specified destinations. We also derive a decoupled algorithm that has the advantage of only requiring to plan in the individual robot's configuration space instead of the joint state-space. To find proper configuration of the cable, we will restrict the topology class of the cable connecting two robots. Also we expand this problem to manipulate a large number of objects with multiple pairs of cable-robot teams.

Chapter 2

Preliminaries

In this chapter, we will briefly review topology of paths/trajectories of robots and cables, which can be considered as a curve in the workspace.

2.1 Curves in $(W - \mathcal{O})$

Let $W \subset \mathbb{R}^2$ be a 2-dimensional simply connected and bounded region. Suppose it contains a set of objects, $\mathcal{O} = O_1 \cup O_2 \cup \cdots \cup O_n \subseteq W$, where $O_1, O_2, \cdots, O_n$ are n counts of objects. Each object, O_j is assumed to be connected.

Both robot paths/trajectories and cable configurations are 1-dimensional curves in $(W - \mathcal{O})$. They can thus be defined as continuous maps from the line segment $[0, 1]$ to $(W - \mathcal{O})$. We say a curve, $\gamma : [0, 1] \to (W - \mathcal{O})$, is embedded [69] if $\gamma(t) \neq \gamma(t'), \forall t \neq t'$ (*i.e.* the curve does not intersect itself).

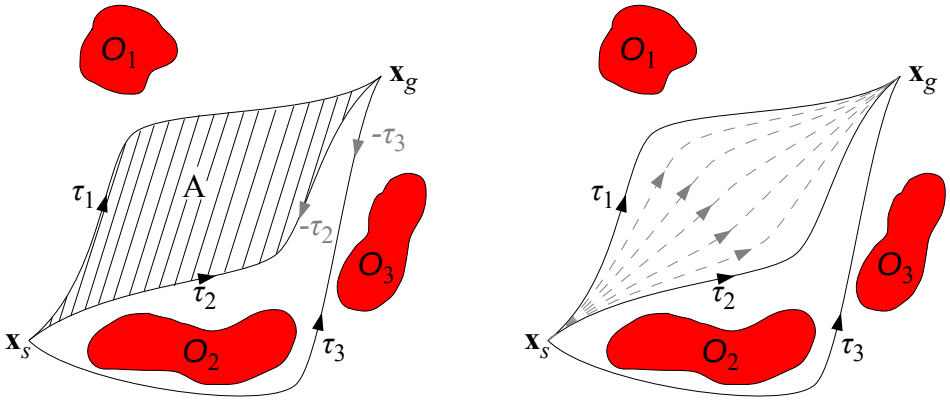
For a given curve, γ, we define $-\gamma : t \mapsto \gamma(1 - t)$. That is, $-\gamma$ is the same curve as γ, but with opposite orientation. The line integral of a differential 1-form, $\omega = f \, \mathrm{d}x + g \, \mathrm{d}y$, over γ is defined as $\int_\gamma \omega := \int_0^1 (f\dot{\gamma}_x + g\dot{\gamma}_y) \, \mathrm{d}t$.

2.2 Homology and Homotopy Invariants

2.2.1 Homology of curves and Homology Invariants

Definition 2.2.1 (Homology classes of curves)**.** Two curves $\gamma_1, \gamma_2 : [0, 1] \to (W - \mathcal{O})$ connecting the same start and end points, are homologous (or belong to the same *homology class*) iff γ_1 together with γ_2 (the latter with opposite orientation) forms the complete boundary of a 2-dimensional manifold embedded in $(W - \mathcal{O})$ (not containing/intersecting any of the objects/obstacles) as shown in Figure 2.1(a) [15, 49].

A *homology invariant* is a function, H, from the space of all curves in $(W - \mathcal{O})$ (with fixed end points) to another much smaller space (in this case, a vector space), such that $H(\gamma_1) = H(\gamma_2)$ iff γ_1 is homologous to γ_2. In [15] a homology class invariant (called the H-signature) was proposed, which is based on simple



(a) τ_1 is homologous to τ_2 since $\tau_1 \sqcup -\tau_2$ forms A, the complete boundary of a 2-dimensional manifold embedded in $(W - \mathcal{O})$. τ_3 belongs to a different homology class since $\tau_1 \sqcup -\tau_3$ or $\tau_2 \sqcup -\tau_3$ encloses O_2.

(b) τ_1 is homotopic to τ_2 since there is a continuous sequence of trajectories representing deformation of one into the other. τ_3 belongs to a different homotopy class since it cannot be continuously deformed into any of the other two.

Figure 2.1: Illustration of homology class and homotopy class or curves.

results from complex analysis. In particular,

$$H(\gamma) = \frac{1}{2\pi i} \int_\gamma \begin{bmatrix} \frac{1}{z-\zeta_1}, \\ \frac{1}{z-\zeta_2}, \\ \vdots \\ \frac{1}{z-\zeta_n} \end{bmatrix} \mathrm{d}z \tag{2.2.1}$$

where, $z = x + iy$ is the complex representation of $(x, y) \in W - \mathcal{O}$, and $\zeta_j = \zeta_{j,x} + i\zeta_{j,y}$ are the complex representations of *representative points* inside the objects with respect to which we compute the H-signature ans shown in Figure 2.2(a). Thus, the function, H, is computed as the integration of the vector of differential 1-forms,

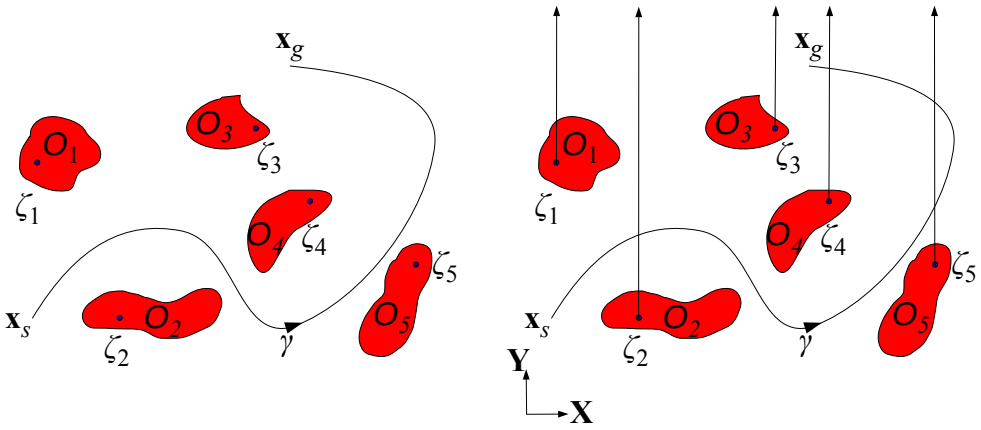
$$\omega_j = \frac{1}{2\pi i} \frac{\mathrm{d}z}{z - \zeta_j}, \tag{2.2.2}$$

over the curve γ.

However, the possible choice of such invariants has been broadened in [16], where the choice of the vector of differential 1-forms, which needs to be integrated over γ to obtain the invariant, has been proven to be any complete set of generators of the *de Rham cohomology group*, $H^1_{dR}(W - \mathcal{O})$. In particular, the *bump 1-forms* [21],

$$\omega_j = \upsilon(y - \zeta_{j,y})\delta(x - \zeta_{j,x})\, \mathrm{d}x, \tag{2.2.3}$$

(where δ is the *Dirac delta function*, and its integral, υ, is the *heaviside step function* – that is, informally speaking, ω_j are analogous to a *Dirac delta* distribution over rays emanating from ζ_j along positive Y axis) is a choice that has the simple interpretation of counting the number of times the curve, γ, crosses rays



(a) H-signature based on simple results from complex analysis.

(b) H-signature based on *bump* 1*-forms*.

Figure 2.2: Examples of possible H-signature functions.

emanating from ζ_j (Figure 2.2(b)). In particular, define, $\#_j\gamma := \int_\gamma \omega_j =$ (Number of times γ crosses the ray emanating from ζ_j from left to right) $-$ (Number of times γ crosses the ray emanating from ζ_j from right to left). Then,

$$H(\gamma) = \begin{bmatrix} \#_1\gamma, \\ \#_2\gamma, \\ \vdots, \\ \#_n\gamma \end{bmatrix}. \tag{2.2.4}$$

For example, curve γ in Figure 2.2(b) has the H-signature of $H(\gamma) = [0, 1, 0, -1, 0]^T$. For closed loops the value of H-signature won't depend on the choice of the differential 1-forms, as long as they form a generating set of the first *de Rham cohomology group*, $H^1_{dR}(W - \mathcal{O})$ [21], and will compute the *winding numbers* about ζ_j.

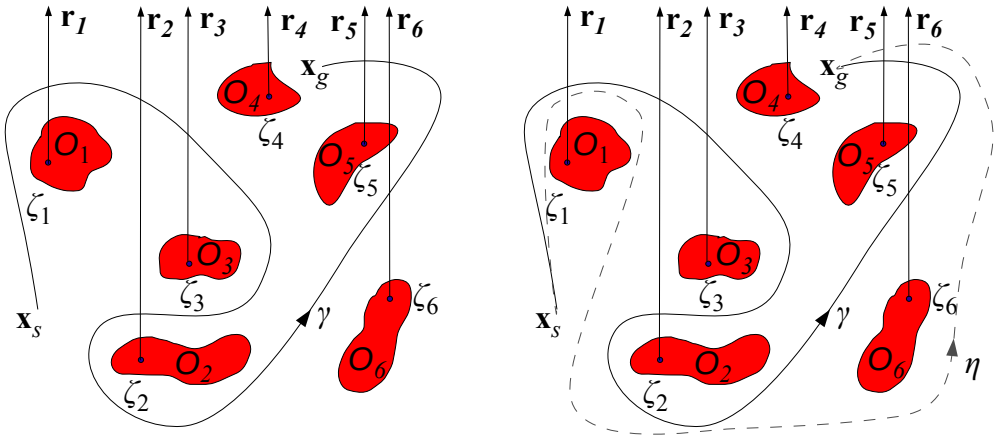
And we can find more generalized form of the H-signature of the given curve, $\gamma : [0, 1] \to (W - \mathcal{O})$, with arbitrary reference lines.

$$\#_j\gamma = \int_0^1 \upsilon(\mathbf{d_j} \cdot (\gamma(t) - \zeta_j))\delta(\tilde{\mathbf{d}}_\mathbf{j} \cdot (\gamma(t) - \zeta_j))\, \mathrm{d}t \tag{2.2.5}$$

where $\mathbf{d_j} = [d_{j,x}, d_{j,y}]^T$ is the direction of the reference line and $\tilde{\mathbf{d}}_\mathbf{j} = [d_{j,y}, -d_{j,x}]^T$ is the normal vector to the reference line. So, it is obvious that (2.2.3) is a special case of $\mathbf{d_j} = [d_{j,x}, d_{j,y}]^T = [0, 1]^T$. As a result, we can choose arbitrary reference lines to calculate H-signature of the given curves and environments.

2.2.2 Homotopy of curves and Homotopy Invariants

Definition 2.2.2 (Homotopy classes of curves)**.** Two curves $\gamma_1, \gamma_2 : [0, 1] \to (W - \mathcal{O})$ connecting the same start and end points, are homotopic (or belong to the same *homotopy class*) iff one can be continuously



(a) h-signature based on reference rays. $h(\gamma) =$ "$r_1^+ r_2^+ r_3^+ r_2^- r_5^-$". (b) h-signature of a closed loop. $h(\gamma \sqcup -\eta) =$ "$r_2^+ r_3^+ r_2^- r_6^+$".

Figure 2.3: Examples of homotopy class invariant function on 2-dimensional plane.

deformed into the other without intersecting any obstacle as sown in Figure 2.3(a).

Formally, if $\gamma_1 : [0,1] \rightarrow (W - \mathcal{O})$ and $\gamma_2 : [0,1] \rightarrow (W - \mathcal{O})$ represent the two trajectories (with $\gamma_1(0) = \gamma_2(0) = \mathbf{x}_s$ and $\gamma_1(1) = \gamma_2(1) = \mathbf{x}_g$), then γ_1 is homotopic to γ_2 iff there exists a continuous map $\eta : [0,1] \times [0,1] \rightarrow (W - \mathcal{O})$ such that $\eta(\alpha, 0) = \gamma_1(\alpha)\ \forall \alpha \in [0,1]$, $\eta(\beta, 1) = \gamma_2(\beta)\ \forall \beta \in [0,1]$, and $\eta(0, \gamma) = \mathbf{x}_s, \eta(1, \mu) = \mathbf{x}_s\ \forall \mu \in [0,1]$ [15, 49].

Homotopy invariants, in general, are much more difficult to design and compute. *Homotopy groups*, unlike *homology groups*, do not have the natural structure of a vector space [49]. However, for curves in 2-dimensional plane with punctures (*i.e.* obstacles/objects), there is a relatively simple representation of the homotopy group and a way of computing the homotopy class of a given curve [46, 50, 98, 49, 9]: We consider *representative points*, ζ_i as before, and *parallel non-intersecting* rays, $r_1, r_2, \cdots, r_n$, emanating from the objects respectively (Figure 2.3(a)). We form a *word* by tracing γ, and consecutively placing the letters of the rays that it crosses, with a superscript of '$+1$' (assumed implicitly) if the crossing is from left to right, and '-1' if the crossing is from right to left. Thus, for example, the word for γ in Figure 2.3(a) will be "$r_1^+ r_2^+ r_3^+ r_2^- r_6^+ r_6^- r_5^-$". We can *reduce* this word by canceling the same letters that appear consecutively but with opposite superscript signs. Thus, the word for γ in Figure 2.3(a) can be reduced to "$r_1^+ r_2^+ r_3^+ r_2^- r_5^-$". This reduced word representation is a *homotopy invariant* for open curves (with fixed end points), γ, and we will write this as $h(\gamma)$ and call it the "h-signature of γ". However, it is important to note that we cannot exchange position for arbitrary pairs of letters in the word (*i.e.* the juxtaposition of letters is *non-commutative*). Unlike the *homology invariant*, this is not a vector, but an element of the *non-abelian* group *freely generated* [86, 49] by $\{r_1, r_2, \cdots, r_n\}$. Thus, although *words* can't be added in the sense of vectors, they can be concatenated under the non-commutative *group operation*, '$\diamond$'. Also, the inverse of a word, $\mathfrak{w}$, written as $\mathfrak{w}^{-1}$, is the h-signature of the same curve but with opposite orientation (*i.e.* $h(-\gamma) = (h(\gamma))^{-1}$), and is a word where the order of the letters are reversed, and the exponent of each letter is flipped (so that $\mathfrak{w} \diamond \mathfrak{w}^{-1} =$" ", the identity element). Thus, $(\mathfrak{w}_1 \diamond \mathfrak{w}_2)^{-1} = {\mathfrak{w}_2}^{-1} \diamond {\mathfrak{w}_1}^{-1}$. As an example, ("$r_1^+ r_3^+ r_2^-$")$^{-1}$ ="$r_2^+ r_3^- r_1^-$".

However, if the curve is a closed loop (*e.g.* $(\gamma \sqcup -\eta)$ in Figure 2.3(b)), there is no preferred starting

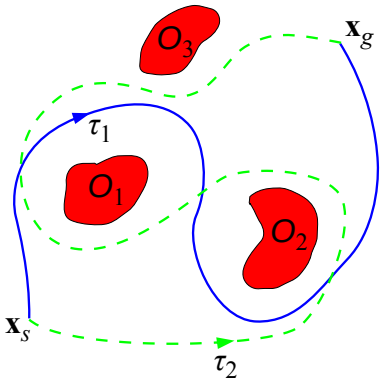


Figure 2.4: Examples where curves (τ_1 and τ_2) are homologous, but not homotopic.

point from where we should start tracing the curve and write the word. Thus, for such curves we need to consider the *cyclic permutations* of the letters in the reduced words to be equivalent. That is, a word, "$abcde$" will be considered to be the same as "$cdeab$". Thus, when reducing a word, we need to consider the cyclic permutations, and thus cancel a letter at the beginning of the word that appears at the end as well, but with opposite superscript signs. For example, in Figure 2.3(b), if we trace the curve, $\gamma \sqcup -\eta$, starting at the point $\mathbf{e}$, we get,

$$\begin{aligned} h(\gamma \sqcup -\eta) =& h(\gamma) \diamond h(-\eta) = h(\gamma) \diamond h(\eta)^{-1} \\ =& \text{“}r_1^+ r_2^+ r_3^+ r_2^- r_5^- \text{”} \diamond \left(\text{“}r_1^+ r_6^- r_5^- \text{”}\right)^{-1} = \text{“}r_1^+ r_2^+ r_3^+ r_2^- r_5^- \text{”} \diamond \text{“}r_5^+ r_6^+ r_1^- \text{”} \\ =& \text{“}r_1^+ r_2^+ r_3^+ r_2^- r_5^- r_5^+ r_6^+ r_1^- \text{”} = \text{“}r_1^+ r_2^+ r_3^+ r_2^- r_6^+ r_1^- \text{”} \\ =& \text{“}r_2^+ r_3^+ r_2^- r_6^+ \text{”} \textit{(after canceling the letters at the start \& the end)} \end{aligned} \tag{2.2.6}$$

which is the completely reduced word.

The *homotopy invariant* of a curve, γ, is the *reduced word* constructed in the described way, with cyclic permutations of a word being considered equivalent when γ is closed. It is easy to note that for closed curves, the value of the *homology invariant* described earlier as integral over the bump 1-forms, does not depend on the choice of the direction of the rays emanating from ζ_i. But the *homotopy invariant* word is highly dependent on the choice of the direction of the rays.

2.2.3 The Hurewicz map

While one may be tempted to think that the concepts of *homology* and *homotopy* are one and the same, that is in fact not true. While trajectories that are homotopic are also homologous, the converse is not necessarily true [15, 49]. For example, the two curves τ_1 and τ_2 are homologous, but not homotopic in Figure 2.2.3. This is due to existence of a *homomorphisms*, called the *Hurewicz maps* [49] from the homotopy groups to the homology groups, which are not necessarily isomorphisms. The Hurewicz map between the first homotopy group, $\pi_1(X)$ [49], and the first homology groups for any topological space, X, can be written explicitly as

the abelianization map (a group quotient map), $\mathsf{h}_* : \pi_1(X) \to \pi_1(X)/[\pi_1(X), \pi_1(X)]$, where $[\cdot, \cdot]$ is the *commutator subgroup* [45] of $\pi_1(X)$.

The Hurewicz map can be used to compute the H-signature (the *homology* invariant) from a given a h-signature (*homotopy* invariant) of a closed curve when the reference rays of H-signature and h-signature are the same for each object. To do this, we simply allow the letters in the given word to commute, thus reducing the word until each letter appear at most once with an exponent (which will be ± 1 for embedded curves). We then place the exponent of each letter in the corresponding position of the H-signature vector. Equivalently, we start with a zero-vector for the H-signature, and then for each letter we add 1 to the corresponding component of the vector if the letter appears with a '$+1$' exponent, and subtract 1 if it appears with a '-1' exponent. We will also write h_* to denote this map from the space of h-signatures (words) to the space of H-signatures (a vector space).

Thus, from the earlier example of Figure 2.3(a), we had $h(\gamma) =$ "$r_1^+ r_2^+ r_3^+ r_2^- r_5^-$". Since the 6 components of the H-signature vector corresponds to the objects $O_1, O_2, \ldots, O_6$ respectively, starting with $[0, 0, 0, 0, 0, 0]^T$, for 'r_1^+' we add 1 to the 1^{st} component, for 'r_2^+' we add 1 to the 2^{nd} component, for r_3^+ we add 1 to the 3^{rd} component, for 'r_2^-' we subtract 1 from the 2^{nd} component, and for 'r_5^-' we subtract 1 from the 5^{th} component. Thus, we end up having $H(\gamma) = \mathsf{h}_*(\text{"}r_1^+ r_2^+ r_3^+ r_2^- r_5^-\text{"}) = [1, 0, 1, 0, -1, 0]^T$.

2.2.4 Augmented Graph

Through this thesis, we will use an augmented graph in our graph-search planner. We will define an augmented graph in which a vertex includes additional information or component about topology of the path until this vertex. For example, if we build a graph of a mobile robot the vertex in the graph should be $v = (x, y)$ and two vertices, $v_1 = (x_1, y_1)$ and $v_2 = (x_2, y_2)$, are the same if $x_1 = x_2$ and $y_1 = y_2$. However, the vertices in the augmented graph include topology class information to reach each vertex. then the vertex in the augmented graph will be $v = (x, y, g)$ where g can be the H-signature for homology class or h-signature for homotopy class. In the augmented graph, two vertices, $v_1 = (x_1, y_1, g_1)$ and $v_2 = (x_2, y_2, g_2)$, are the same if $x_1 = x_2$, $y_1 = y_2$ and $g_1 = g_2$. By adapting this augmented graph, we can find optimal paths to the same point for vertex in the original graph in different topology classes.

The Figure 2.2.4 shows an example with a single obstacle in a four-way-connected graph. Two paths, τ_1 and τ_2, have the same initial vertex, v_s. In the original graph, the goal vertex, v_g of the two paths are the same because they have the same coordinates. However, in the h-signature augmented graph, the goal vertex of τ_1 would be $v_g^1 = (x_g, y_g, \text{"}r_1^+\text{"})$. While, the goal vertex of τ_2 would be $v_g^2 = (x_g, y_g, \text{" "})$. So we have $v_g^1 \neq v_g^2$. In the same manner, the goal vertices of H-signature (using the bump form in (2.2.3)) graph will be $v_g^1 = [x_g, y_g, [1])$ and $v_g^2 = [x_g, y_g, [0])$ to result in $v_g^1 \neq v_g^2$.

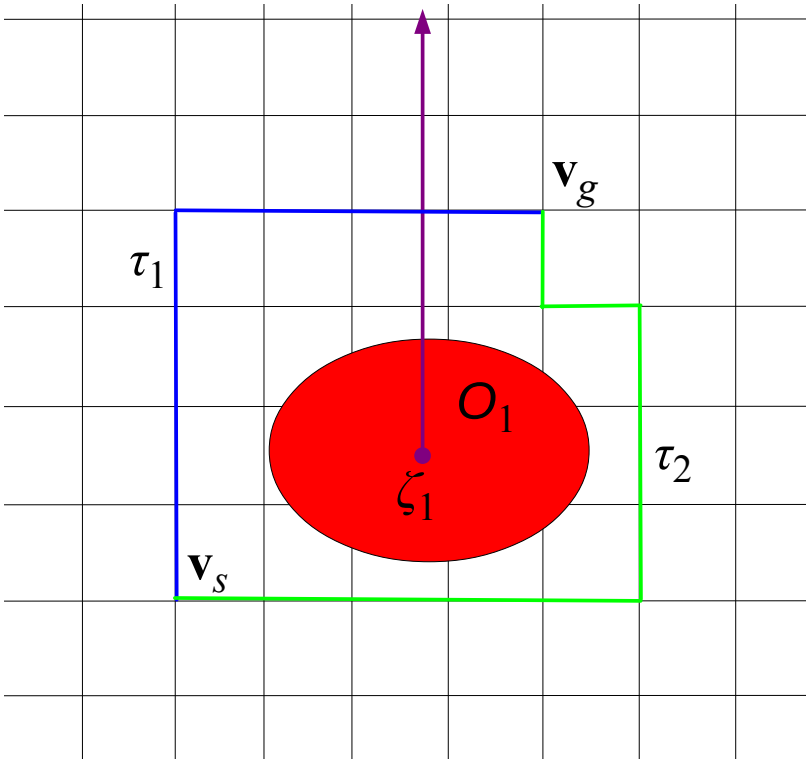


Figure 2.5: Example of augmented graph. The goal vertices of two different paths, τ_1 and τ_2, have the same coordinates but considerd to be different vertex in the augmented graph.

Chapter 3

Trajectory Generation under Topological Constraints

In this chapter, we present a method to generate an optimal trajectory restricted to a particular topology class. The optimality of the generated trajectory is achieved by formulating the trajectory generation problem as a Mixed-Integer Quadratic Program (MIQP) [84, 80]. As the H-signature is the homology class invariant function, we can find shortest paths with graph-search based planner [10]. But we cannot add H-signature constraints to optimal control because the gradient of H-signature is zero almost everywhere. So, we introduce binary variables that not only encode information about the satisfaction of geometric constraints, but also incorporate information about the topology class. We will cleverly consider topology class constraints so that the suggested trajectory generation problem under topology class constraints can still be formulated as a MIQP. We illustrate the method with examples of minimum acceleration trajectory generation under different topology class constraints with potential application to differentially-flat systems with a two-dimensional flat output space. The work in this chapter was performed in close collaboration with Dr. Koushil Sreenath and Dr. Subhrajit Bhattacharya. Much of the work in Section 3.2 and Section 3.3 were reported in [58] and [59], respectively.

3.1 Optimal Trajectory Generation

We consider trajectory planning in a compact subset $Q \subset \mathbb{R}^2$ of a plane. Let $O = \{o_1, o_2, \cdots, o_{n_o}\}$ be a set of convex, pair-wise disjoint *obstacles* in Q (The requirement of convexity of obstacles can be relaxed by considering a set of arbitrarily-shaped obstacles such that their convex hulls are pair-wise disjoint). Each obstacle $o_i \in O$ can be represented by a n_i-sided convex polygon, whose faces define hyperplanes that partition Q into two half-spaces. A binary variable is used to indicate whether a point is on the feasible side of the hyperplane, as described in [80]. So a point $q \in Q$ will be feasible and will avoid collision with an obstacle o_i if there is at least one face $f \in [1, ..., n_i]$ satisfying $n_{i,f} \cdot q \leq s_{i,f}$. Where $n_{i,f}$ is a normal vector to the f^{th} face of obstacle o_i pointing inward, and $s_{i,f} = n_{i,f} \cdot p$, for an arbitrarily chosen point p on the f^{th} face as shown in Figure 3.1. Similar to obstacle avoidance using binary variables $b_{i,f}$, as described in [80],

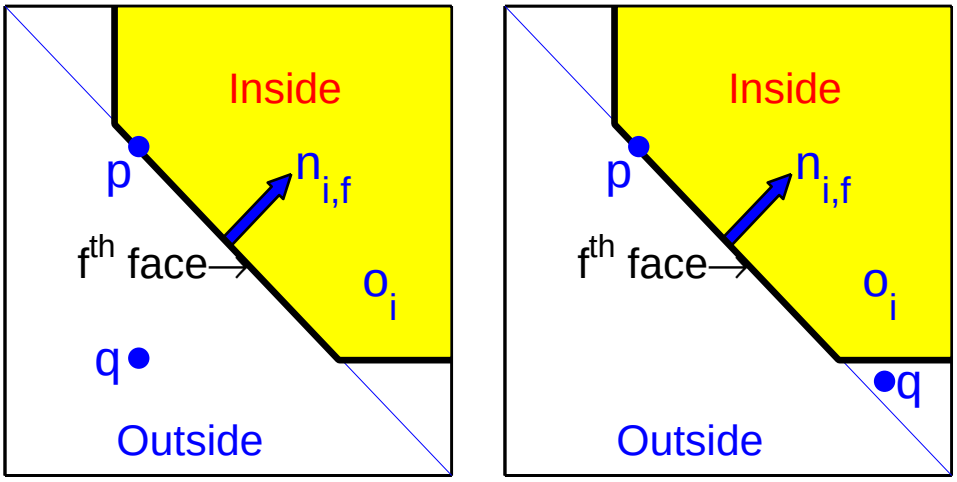


Figure 3.1: The normal vector, $n_{i,f}$, of the f^{th} face of obstacle o_i is pointing inward. p is an arbitrary point on the f^{th}face. (a) An example of $q \in Q$ when $b_{i,f} = 0$. (b) An example of $q \in Q$ when $b_{i,f} = 1$.

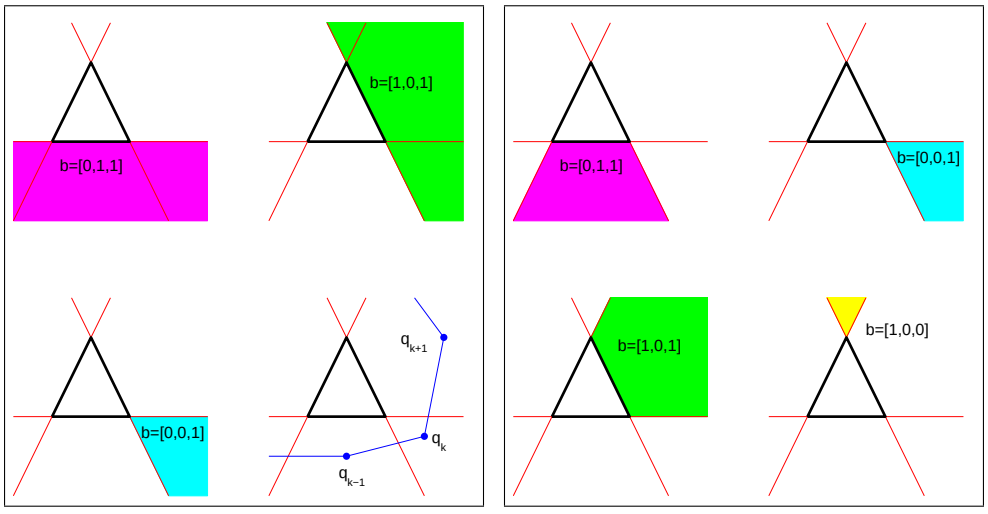


Figure 3.2: (a) Overlapping subsets divided by values of binary variables representing each face of triangular obstacle. (b) Disjointed cells divided by values of binary variables representing each face but considering additional constraint.

a given point is feasible with respect to obstacle o_i if

$$n_{i,f} \cdot q \leq s_{i,f} - \delta_r + M b_{i,f} \quad \text{for} \quad f = 1, ..., n_i \tag{3.1.1}$$
$$\sum_{f=1}^{n_i} b_{i,f} \leq n_i - 1,$$

where $b_{i,f} \in \{0, 1\}$ are binary variables (with $b_{i,f} = 0$ indicating that the point lies on the feasible side of the f^{th} face of the i^{th} obstacle as shown in Figure 3.1(a)), and $M > 0$ is a large positive number. $\delta_r \geq 0$ is the radius of the disk encircling the finite-sized robot, along with some safety-padding around it. The second inequality in (3.1.1) implies that the point q will be feasible with respect to at least one face, *i.e.,* for a given i, there exists at least one f such that $b_{i,f} = 0$. Although (3.1.1) is a sufficient condition for feasibility, this formulation breaks up Q into overlapping subsets as shown in Figure 3.2(a). The first three plots in Figure 3.2(a) illustrate that the subset corresponding to $b = [0, 0, 1]$ is the intersection of the two subsets corresponding to $b = [0, 1, 1]$ and $b = [1, 0, 1]$. Considering the segment of trajectory in the last plot of Figure 3.2(a), the binary variable vector b_k, corresponding to the point q_k, is not unique, but could be any of $b = [0, 1, 1]$, $b = [0, 0, 1]$ and $b = [1, 0, 1]$. As a result, we can have the same trajectory (represented by the points on it) described by different sets of binary variables. Such duplication increases the size of the feasible region in the space of binary variables, resulting in redundant searches, and larger computation times. To eliminate such cases, we introduce some additional inequality constraints to build disjointed cells like in Figure 3.2(b),

$$-n_{i,f} \cdot q \leq -s_{i,f} + \delta_r + M(1 - b_{i,f}) \text{ for } f = 1, ..., n_i. \tag{3.1.2}$$

The first inequality of (3.1.1) only guarantees that the point q is on the feasible side or outside of f^{th} face when $b_{i,f} = 0$. But the constraint (3.1.2) enforces that the point q be on the other side when $b_{i,f} = 1$. Thus, the feasible region, Q, is partitioned into disjoint cells, each of which is bounded by hyperplanes defined by the faces of the obstacles. (see Figure 3.2(b)). Moreover, each cell can be identified by a unique vector of binary variables, $b = [b_1^T, \ldots, b_{n_o}^T]^T$ where $b_i = [b_{i,1}, \ldots, b_{i,n_i}]^T \in \{0, 1\}^{n_i}$.

We parametrize the trajectory by splicing N_s segments of trajectories, each parametrized by linear combination of $N_p + 1$ basis functions,

$$q(t) = \sum_{k=0}^{N_p} c_{j,k}\, e_k(t - t_j) \quad \text{for} \quad t_j \leq t < t_{j+1}, \tag{3.1.3}$$

for $j \in [0, ..., N_s - 1]$, $0 = t_0 \leq t_1 \leq ... \leq t_{N_s} = t_f$. Where $e_k(t)$ is any basis function and $c_{j,k}$ are coefficients. So the whole trajectory is union of N_s subtrajectories The trajectory is restricted to be k_r-times differentiable at the junction of each of the segments of trajectories, $q(t_j)$, for $j \in [1, ..., N_s - 1]$. Further, obstacle avoidance is achieved by enforcing (3.1.1) at some equally distributed intermediate points on each segment of trajectories. we choose the cost function to be the integration of the square of the norm of r^{th}-derivative of the trajectory:

$$J(c) = \int_{t_0}^{t_f} \left\| \frac{d^r q(t)}{dt^r} \right\|^2 dt \;=\; c^T H c. \tag{3.1.4}$$

where $c = [c_0^T, ...c_{N_s-1}^T]^T$, and H depends only on the choice of the basis functions (note that we could choose a cost function that is a weighted sum of different order derivatives, and still keep it quadratic in c). The optimal trajectory generation problem can then be simplified as the following MIQP (Mixed-Integer Quadratic Program),

$$\begin{aligned} \min_{c,\, \widetilde{b}} \quad & c^T H c \\ \text{s.t.} \quad & A_f c + D_f \widetilde{b} \leq g_f \\ & A_b \widetilde{b} \leq g_b \\ & A_{eq} c = 0 \end{aligned} \tag{3.1.5}$$

where $\widetilde{b}$ is the vector formed by stacking all the binary vectors, b_k, corresponding to the intermediate points, q_k, of the trajectory and hence is a coarse representation of the continuous trajectory $q(t)$. The first inequality captures the feasibility constraints of (3.1.1) for the intermediate points, the second inequality captures the constraint on sum of binary variables in (3.1.1), and $A_{eq}c = 0$ imposes r^{th} order differentiability at the junction of the segments of trajectories and the boundary conditions of initial configuration, $q(0) = q_0$ and final configuration $q(t_f) = q_f$.

Now in the following sections, we will discuss how to add homology or homotopy class constraints on this trajectory generation problem while maintaining the MIQP formulation.

3.2 Optimal Trajectory with Homology Class Constraints

To find an optimal trajectory in a specific homology class, we can then add some topological constraints. If we add a constraint on the H-signature, which we described in the Chapter 2, such that the H-signature of the trajectory, $H(q)$, should be some desired H_d. However, all the form of H-signature (2.2.1), (2.2.3) and (2.2.5) are the nonlinear equations of the trajectory. So, the quadratic program (3.1.5) becomes a non-convex problem with this homology class constraints. Furthermore, the gradient of the new constraint, $H = H_d$, will be zero almost everywhere, because the value of the H-signature does not change within a particular homology class, (*i.e.* the range of the H-signature is a set of discrete variables). So, the resulting problem is a non-convex problem, which is numerically hard to solve based on gradients of cost and constraints. So, we need a different way to enforce topological constraints.

3.2.1 Algorithm Description

In this Section, we will describe our algorithm to generate the optimal trajectory with homology constraints while ensuring that the problem remains a MIQP.

Additional feasibility condition

We start by noting that the feasibility (with respect to obstacles) of each intermediate point on the trajectory does not guarantee the feasibility of the whole trajectory. Consider an example with only one parallelogram obstacle as shown in Figure 3.2.1. In Figure 3.2.1, two adjacent intermediate points, q_1 and q_2, are both feasible with respect to the given obstacle, o_1. The red curve in Figure 3.2.1 shows an infeasible curve

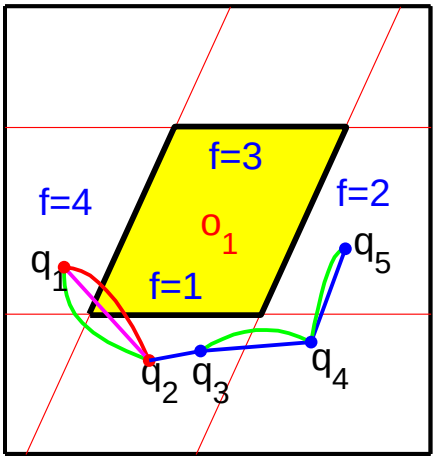


Figure 3.3: An example of parallelogram obstacle. f is the index of each face. Red and magenta curves are infeasible trajectories between two feasible configurations, q_1 and q_2. Adjacent intermediate points(q_3, q_4 and q_5) are satisfying additional constraint.

connecting two points. Of course, the optimal trajectory could be feasible like the green curve. However, the line segment connecting the two points (the magenta curve) is infeasible. To avoid such undesirable cases, we need additional constraint between adjacent intermediate points. The curves in Figure 3.2.1 illustrates this additional constraint: Considering the corresponding binary variables of each point, q_3 is only feasible with respect to face $f = 1$ and the next intermediate point, q_4 is also feasible with respect to the same face. So, the line segment, connecting these two points is also feasible with respect to face $f = 1$. Moreover, both q_5 and its previous point, q_4, are feasible with respect to face $f = 2$. So the line segment joining them is also feasible. In contrast, consider the case of q_1 and q_2 in Figure 3.2.1. These two adjacent intermediate points do not share feasibility with respect to a common face – q_1 is feasible with respect to only face $f = 4$ and q_2 is feasible with respect to only face $f = 1$. So we cannot guarantee the feasibility of the line segment connecting these two points.

The above discussion suggests an additional constraint that two consecutive intermediate points should share a common hyperplane with respect to which they are feasible, and this should hold true for each obstacle. In other words, the binary variables corresponding to the adjacent intermediate points should either be the same or differ by only one component, and this condition should be satisfied with respect to all obstacles. This constraint then guarantees the feasibility of a straight line segment connecting the two intermediate points. We write $b_{(o,k)}$ to describe the vector of binary variables for the k^{th} intermediate point formed by stacking together the binary variables for the different faces of the o^{th} obstacle (thus, it is a n_i-sized sub-vector of $\widetilde{b}$). Thus the constraint involving the k^{th} and $k + 1^{th}$ intermediate points with respect to o^{th} obstacle can be describe as

$$\begin{aligned} \left\| b_{(o,k)} - b_{(o,k+1)} \right\|_2^2 &= \sum b_{(o,k)} \cdot b_{(o,k)} + b_{(o,k+1)} \cdot b_{(o,k+1)} - 2b_{(o,k)} \cdot b_{(o,k+1)} \\ &= \sum b_{(o,k)} + \sum b_{(o,k+1)} - 2b_{(o,k)} \cdot b_{(o,k+1)} \\ &\leq 1 \end{aligned} \tag{3.2.1}$$

where, $\sum b$ denotes the sum of the elements of a binary vector, and the last equality holds since $b \cdot b = \sum b$ for a vector of binary variables, $b \in \{0,1\}^n$. This additional constraint on the gradual change of the binary variables along the trajectory plays an important role in formulation of a new h-signature based on binary variables, as described in the next section. However, this constraint is quadratic in the binary variables, and we will discuss how we can reduce this constraint to a linear one in Section 3.2.1.

Define H-signature

To find an optimal trajectory contained in a specific homology class, the H-signature defined in Chapter 2 can be used. However, these function are homology class invariant whose gradients are zero almost everywhere. So, the homology class constraints based on H-signature are not proper for gradient-based numerical solvers.

However, we choose H-signature of (2.2.5) for this work. We can choose arbitrary reference ray of each obstacle but for convenience of calculation and notation, we choose the reference ray as the extension of face $f = 1$ in the direction of the last face $f = n_f$ as shown in Figure 3.2.1. Also, for the consistency of sign of winding number, the faces are numbered in counterclockwise direction like Figure 3.2.1. So, for a given i^t obstacle, the H-signature will be

$$H_i\left(q(t)\right) = \int_{t_0}^{t_f} \upsilon(\mathbf{d_i} \cdot (\gamma(t) - \zeta_i))\delta(\mathbf{n_{i,1}} \cdot (\gamma(t) - \zeta_i))\, \mathrm{d}t \tag{3.2.2}$$

where $\mathbf{n_{i,1}}$ is the normal vector the of the 1^{st} face of the i^{th} obstacle and $\mathbf{d_i} = R\left(\frac{\pi}{2}\right)\mathbf{n_{i,1}}$ is the direction of reference ray, which is rotating the normal vector by $\frac{\pi}{2}$ and ζ_i is an arbitrary point on the 1^{st} face. However, the gradient of this integration will be zero almost everywhere and is not proper constraint. Here, we need to focus on the fact that geometric meaning of this integration is counting the number of times the trajectory crosses the given reference ray. Moreover, this reference ray is one the the boundary or hyper plane that splits the feasible space into cells. So the change of binary variable corresponding to the first face $b_{i,1}$ will give us alternative way to integrate (3.2.2).

From this fact, it is obvious that we need to accumulate the value of $b_{i,1,k+1} - b_{i,1,k}$ for $\forall k$ (where by $b_{i,j,k}$ we mean the binary variable for the k^{th} intermediate point corresponding to the j^{th} face of the i^{th} obstacle). However, to avoid counting the number of intersection with the the other ray obtained by extending the face $f = 1$ in the other direction (the green line in Figure 3.2.1), we need to count the case when the two adjacent intermediate points are infeasible with respect to the second face $f = 2$, *i.e.* $b_{i,2,k+1} = b_{i,2,k} = 1$. So, the H-signature with respect to an obstacle, o_i, will be

$$\begin{aligned} H_i(\widetilde{b}) &= \sum_k \frac{b_{i,2,k+1} + b_{i,2,k}}{2}\left(b_{i,1,k+1} - b_{i,1,k}\right). \\ &= \sum_k b_{i,2,k}\left(b_{i,1,k+1} - b_{i,1,k}\right) \end{aligned} \tag{3.2.3}$$

The second equality of above equation holds because $b_{i,2,k+1} = b_{i,2,k}$ when $b_{i,1,k+1} \neq b_{i,1,k}$ due to the constraint we defined in (3.2.1). The H-signature with respect to all obstacles will be $H = [h_1, ..., h_{n_o}]^T$, where n_o is the number of obstacles. However, this new H-signature is also quadratic in binary variables. We will discuss how we can reduce this quadratic equation to a linear one in the next section.

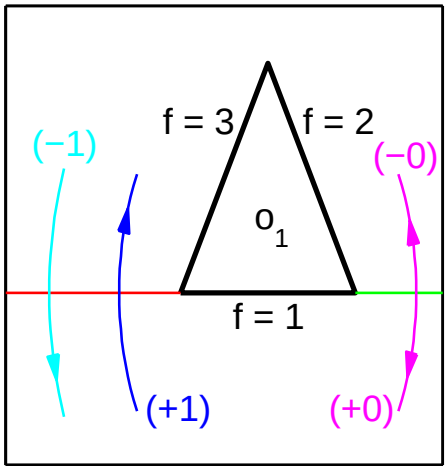


Figure 3.4: An example of calculating the h-signature with respect to a triangular obstacle.

Substitution binary variables

As the new constraint in (3.2.1) and the H-signature in (3.2.3) are quadratic with respect to the binary variables, we introduce some *substitution binary variables* that represent the product of two binary variables. For example, consider the product of two binary variables, $b_i \cdot b_j$, for $b_i, b_j \in \{0, 1\}$. Then, we substitute $b_i \cdot b_j$ with a new binary variable $d_{ij} \in \{0, 1\}$, on which we impose the following three inequalities,

$$d_{ij} \leq b_i \ , \quad d_{ij} \leq b_j \ , \quad -2 + \delta + b_i + b_j \leq d_{ij} \tag{3.2.4}$$

where $0 < \delta < 1$ is a design parameter. The first two inequalities in (3.2.4) enforce $d_{ij} = 0$ when $b_i = 0$ or $b_j = 0$, respectively. And the last inequality enforces $d_{ij} = 1$ when $b_i = b_j = 1$, because $0 < \delta \leq d_{ij}$. So the above three constraints let us perform the substitution $d_{ij} = b_i \cdot b_j$. Let d be the vector of substitution variables with which we need to replace all the quadratic terms in (3.2.1) and (3.2.3). Then we can rewrite the feasibility conditions of substitution binary variables, (3.2.4), as

$$A_{f,d}\widetilde{b} + B_{f,d}d \leq b_f. \tag{3.2.5}$$

Then we can rewrite the quadratic constraint of (3.2.1) for the whole trajectory as

$$A_{o,k}\widetilde{b} + B_{o,k}d \leq b_{o,k} \tag{3.2.6}$$

for all o and k. And the h-signature calculation of (3.2.3) becomes the following linear equation

$$H_i = A_{i,h}d. \tag{3.2.7}$$

So, we can reduce all equations containing quadratic terms in the binary variables to linear ones using the substitution binary variables.

Finding Optimal Trajectory in a given Homology Class

Since our goal is to design optimal trajectory with homology constraint, we can impose the new constraints of (3.2.5), (3.2.6), and (3.2.7) to the optimal trajectory generation problem (3.1.5) to formulate a new MIQP as follows

$$\begin{aligned} \min_{c,\, \widetilde{b},\, d} \quad & c^T H c \\ \text{s.t.} \quad & A_f c + D_f \widetilde{b} \leq g_f \\ & A_b \widetilde{b} \leq g_b \\ & A_d \widetilde{b} + B_d d \leq b_f \\ & A_o \widetilde{b} + B_o d \leq b_o \\ & A_{eq} c = 0 \\ & A_h d = H_d \end{aligned} \tag{3.2.8}$$

where $\widetilde{b}$ and d are vectors of binary variables as described earlier. The third inequality is the condition of substitution variables (3.2.5), the forth inequality is for additional feasibility constraint for continuous change of binary variables (3.2.6), and the last equality is for the homology constraint with respect to all obstacles (3.2.7). As the resulting problem is MIQP, we can get an anytime solution to this problem through numerical solvers like CPLEX [52]. However, we need enough number of segments of trajectories (N_s) and basis function (N_p) to be able to obtain a feasible trajectory in the given homology class.

Proposition 3.2.1 (Completeness Guarantee)**.** *Suppose there exists an arbitrary trajectory τ (dark blue curve in Figure 3.5(a)), not touching any of the obstacles, in the homology class represented by the H-signature of H_d, that crosses the cell boundaries (*i.e. *the hyperplanes) m or less number of times (for avoiding ambiguity we assume τ is generic and that it does not pass through the intersection of 2 or more hyperplanes). With the choice of basis functions $e_k(t) = t^k$ in* (3.1.3)*, and with $N_p > r$, it is then sufficient to choose $N_s = 2^r(m-1)+1$ in order to guarantee existence of a solution for the problem in* (3.2.8) *(*i.e. *all the conditions being satisfied, and with finite cost).*

Sketch of Proof.

Consider the $m-1$ consecutive cells that τ passes through. We choose $m-1$ points, $q_1^0, q_2^0, \cdots, q_{m-1}^0$, respectively in the interior of each of these cells. Now, two such consecutive cells together form a convex region (bounded by the hyperplanes the cells are individually bounded by, except for the one hyperplane that separates them). Thus, the piece-wise linear curve formed by joining these consecutive points (call this q^0) give a trajectory consisting of m segments (cyan curve in Figure 3.5(a)), connecting the initial and final points, not intersecting any of the obstacles, and is continuous (*i.e.* 0^{th} order differentiable). The affine segments are permitted by the choice of the basis functions (the parametrization may be chosen arbitrarily), thus giving values of coefficients, $c_{j,k}$, in (3.1.3) that describe this trajectory. Those, along with the binary vectors corresponding to each of these points, satisfy all the conditions in (3.2.8), except for the differentiability condition $A_{eq}c = 0$.

The main idea behind the proof of this proposition is that we can now replace each of the points q_j^0 by two points lying arbitrarily close to it, and thus "smoothen" the curve (Figure 3.5(b)). This smoothening is

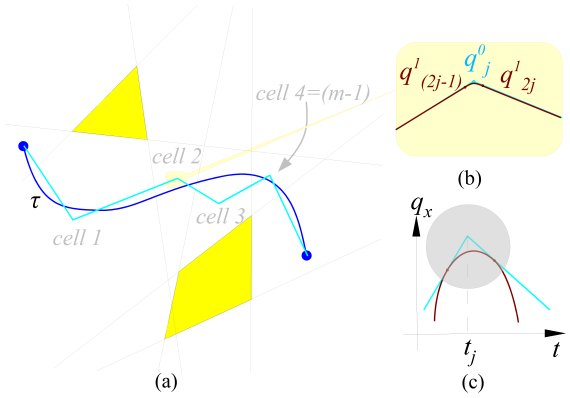


Figure 3.5: Starting from a piece-wise linear curve (cyan), we can progressively add points, to make the trajectory smoother by increasing the order of differentiability by one at each step.

possible to achieve with just an unit increase in the degree of the basis functions (which is evident by looking at the individual components $q^0_x(t)$ and $q^0_y(t)$, as illustrated in Figure 3.5(c)) – in this case, going from linear to quadratic (it is always possible to find a parabola that has two given lines with bounded slope as tangents, and then scale it down such that the contact points with the tangents lie within a small ball around the point of intersection of the lines).

Thus, now we have a new trajectory (call this q^1), that is smooth everywhere, but not twice differentiable (red trajectory in Figure 3.5(b)). However, we can continue the same process of smoothening the derivatives of $q(t)$ by adding points in a small neighborhood of the original t_j's, doubling the number of intermediate points at every step. The choice of this neighborhood can be arbitrarily small to ensure that the added points remain in the interior of the same cell. Continuing this until we have r^{th} order differentiability requires $2^r(m-1)$ intermediate points. In this way, we can construct a trajectory that satisfies all the conditions of (3.2.8). ■

Computational Complexity

The resulting optimal trajectory generation problem is a MIQP, which can be solved by an anytime solver like CPLEX. Thus, if there exists a feasible solution, it will be found by CPLEX. Moreover, with additional time available for computation, a lower cost solution can be found. However, the computation time will increase with the complexity of the given MIQP. So, in this section, we will discuss the computational complexity of the trajectory generation problem (3.2.8). The number of continuous variable in the problem is

$$n_c = 2(N_p + 1)N_s \tag{3.2.9}$$

where there are N_s segments of trajectories of N_p+1 basis function for each x and y. However, some equality constraints to satisfy initial and final configuration and the continuity between segments of trajectories will reduce the actual number of continuous variables by searching the null space of A_{eq} of (3.2.8). Again, the number of binary variables to describe feasibility with respect to each face of obstacle is

$$n_b = N_c \cdot N_f \tag{3.2.10}$$

where N_c is the number of intermediate points on the whole trajectory and $N_f = \sum_{i=1}^{n_o} n_i$ is the total number of faces of all obstacles. Then the number of substitution binary variables is

$$n_d = (N_c - 1)N_f + 2(N_c - 1) \tag{3.2.11}$$

where each term is related to the quadratic terms in (3.2.1) and (3.2.3), respectively. So the total number of binary variables will be $n_b + n_d$.

The number of constraint is also an important factor in computational complexity. The number of equality constraint will be

$$n_{eq} = 2(k_r + 1)(N_s - 1) + 2 \times 4 + n_o \tag{3.2.12}$$

where the first term represent the continuity between segments of trajectories. The second term represents the equality constraint of initial and final configuration; position and velocity of x and y. The last term is related to the H-signature constraint, which is the same as the number of obstacles. However, this equality constraint will disappear because we search in the null space of this equality constraint while reducing the number of continuous variables in the same manner. Most of the constraints are inequality constraints and we have

$$\begin{aligned} n_{ineq} =& 2N_c \cdot N_f + N_c \cdot n_o + N_c \cdot n_o + 3n_d \\ =& 5N_c \cdot N_f + 2N_c \cdot n_o + 6N_c - 3N_f - 6 \end{aligned} \tag{3.2.13}$$

where the first term represents on which side of each face the intermediate point is located – the first equation of (3.1.1) and equation of (3.1.2). The second term represents the second equation of (3.1.1) and the third term represents (3.2.1). The last term presents the condition of substitution binary variables (3.2.4). So the number of inequality constraint is bilinear with respect to the number of intermediate points and faces of obstacles.

3.2.2 Simulation Results

To illustrate how the suggested algorithm works, we performed some simulations to generate optimal trajectories of a point robot, $\delta_r = 0$, in various homology classes of a given environment. For all simulations, we use polynomial basis functions, $e_k(t) = t^k$ and minimize the integration of the norm of acceleration of trajectories, *i.e.* we choose $r = 2$ in (3.1.4). In the first simulation, we find optimal trajectories with two obstacles under homology constraint. As mentioned before, the planned trajectory could be for a differentially-flat dynamical robot system such as a kinematic car [71], or a tricycle robot [7].

Figure 3.6 shows optimal trajectories with the same initial and final configurations but with different desired h-signatures, and consequently different homology classes. For each homology class, the CPLEX solver finds the optimal trajectory. Comparing the computation time and cost of trajectories of each homology class, it can be observed that the optimal trajectory in the homology class corresponding to lower cost takes less time to compute. This is an expected phenomenon since in a branch-and-bound algorithm the tree of fixed binary variables tends to expand to minimize the cost.

In searching for the optimal trajectory in a particular homology class (*e.g.* the one in Figure 3.6(c)),

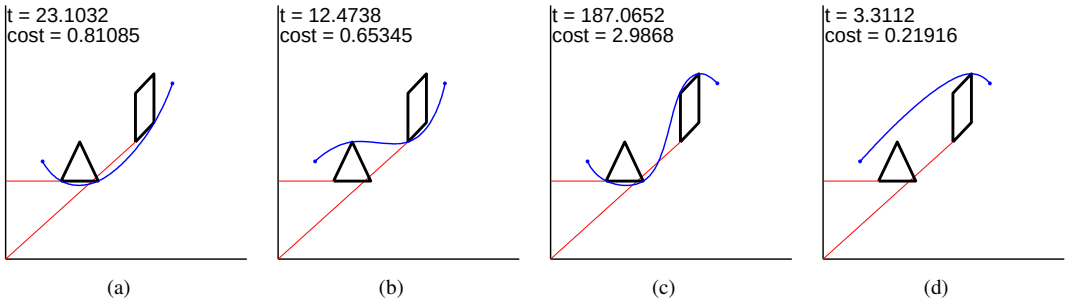


Figure 3.6: Simulation result of trajectory generation in four different homology classes with the same initial configuration (left bottom point) and final configurations(right upper point). The first obstacle is parallelogram and the second obstacle is triangle. The actual computation time(sec) and optimal costs are specified on the upper left corners of plots. (a) $H_d = [-1, -1]^T$. (b) $H_d = [-1, 0]^T$. (c) $H_d = [0, -1]^T$. (d) $H_d = [0, 0]^T$.

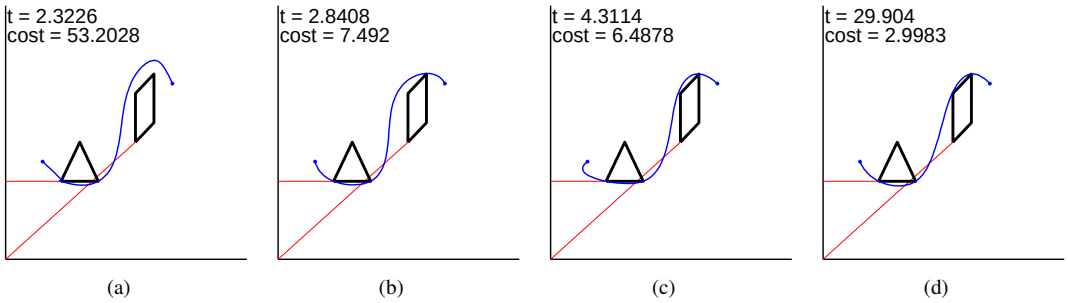


Figure 3.7: Simulation result with anytime solutions. The computation time(sec) and optimal costs are specified on the upper left corners of each plot.

the algorithm expands the nodes in the tree in such a way that feasible solutions corresponding to other homology classes, but with lower costs (*e.g.* the class in Figure 3.6(d)), are also obtained in the process.

If we want to find trajectory in a certain homology class like one in Figure 3.6(c), the tree is expanded to minimize the cost and finds the trajectories with less cost but in other homology classes like one in Figure 3.6(d). Even though, it finds this optimal solution, it moves on to check other nodes that could have less cost, a process that ends up finding trajectories in other homology classes.

To show the anytime performance of the suggested algorithm, we performed some simulations with the same environment as earlier, and with the homology constraint of Figure 3.6(c). The CPLEX solver was terminated at different times to compare the resulting trajectories. As shown in Figure 3.7, the cost decreases as we allow more computation time. And as illustrated in the previous example in Figure 3.6, we will get the global optimal trajectory with enough computation time.

Next we present a series of four examples with increasing number of obstacles, and subsequently increasing complexity (see Figure 3.8). For all examples, we choose six segments of trajectories with nine basis functions, such that the number of continuous variables for optimization, as given by (3.2.9), is $n_c = 108$. The optimizer is given a maximum time of one hour to search for a feasible trajectory. The resulting found trajectory will respect the homology constraint and may be either suboptimal or optimal. Figure 3.8(a)

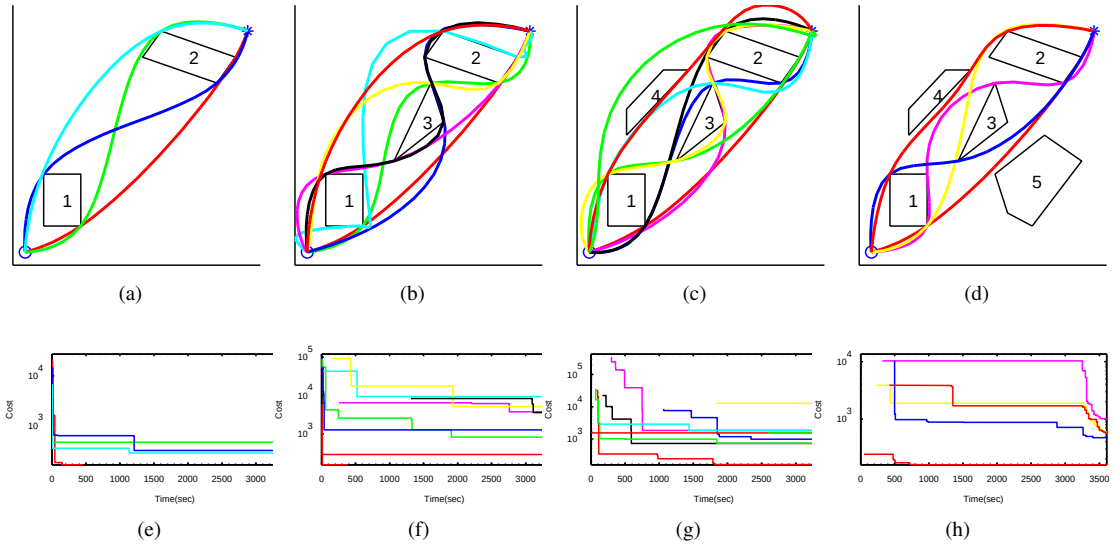


Figure 3.8: (a)-(d) Final trajectories in four different homology classes with two, three, four and five obstacles, respectively. (e)-(h) Cost of the trajectories along with computation time with two, three, four and five obstacles, respectively. The plots shows the change in cost with time plotted in log scale.

illustrates results for the two obstacle case, showing trajectories in all four different homology classes. Figure 3.8(e) shows how the cost of each generated trajectory changes as we keep searching – the corresponding trajectories have the same color as in Figure 3.8(a). It is obvious that the trajectory corresponding to the red curves in Figures 3.8(a), 3.8(e) is the global optimal one without topological constraints. So it terminates searching solution before the time limit. Note that we could not find global optimal trajectories for all the homology classes within the time limit. Figure 3.8(e) however shows that a feasible solution was found relatively quickly. With additional computation time, we expect the optimizer to either find the global optimal trajectories in each homology class or guarantee that the current solution is the global optimum.

Figure 3.8(b) shows the result of simulation with three obstacles. We find suboptimal trajectories in all the eight homology classes. As shown in Figure 3.8(f), an initial feasible trajectory was found relatively quickly for all but one homology class. For the homology class corresponding to the black curve, the optimizer took over 1000 sec to find a feasible trajectory.

For the four obstacle case shown in Figure 3.8(c), we found trajectories in nine homology classes, and could not find trajectories in other seven homology classes within the time limit. The missing seven trajectories should pass obstacle 4 on left like one of the green plots in Figure 3.8(c).

Similarly, for the five obstacle case shown in Figure 3.8(d), we found trajectories in only five homology classes among $2^5 = 32$ possible homology classes. As we found all eight trajectories passing between obstacle 4 and 5, like the trajectories in Figure 3.8(b), there are feasible trajectories in these homology classes which can be represented with our parametrization and can be found.

As illustrated by these simulations, although the optimizer quickly found initial feasible suboptimal trajectories in various homology classes, it could either not find corresponding optimal solutions for all classes or not find the trajectories in all the possible homology classes within the provided time. A few reasons for this are (a) insufficient computation time for the optimizer, (b) insufficient continuous-time variables for

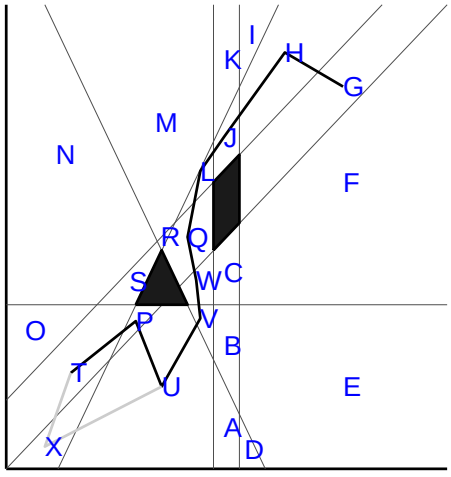


Figure 3.9: An example of a trajectory corresponding to the word TPUVWQLJHG.

parametrization of longer and winding trajectories in certain homology classes, and (c) fundamental limitation of using a general purpose solver such as CPLEX for this particular scenario. To address the issue of computation time, the algorithm can easily be run longer, but more importantly, the computation time can be improved significantly by providing an initial guess for the optimization. Further, increasing the number of continuous-time variables will also help since Proposition 1 guarantees that there exists a feasible trajectory with a sufficiently large number of segments of trajectories.

3.3 Optimal Trajectory with Homotopy Class Constraints

In this section, we present a method to generate an optimal trajectory restricted to a particular homotopy class, which is specified by a given representative trajectory. We partition the configuration space into nonoverlapping cells and model each cell in the partition with integer variables and inequality constraints. We associate with any sequence of integer variables a *word*, so that each trajectory can be mapped to a word. We then construct a set of all words that are homotopically equivalent to a given word. For each word, we fix the integer variables of the MIQP to find the optimal time distribution in each cell, by solving a QP for each iteration, to obtain the locally optimal trajectory in the specified homotopy class. We illustrate an example of minimum acceleration trajectory generation on a plane with different homotopy class constraints.

3.3.1 Algorithm Description

We have broken the problem of optimal trajectory generation into two parts. First we find a *word* that is a coarse representation of the trajectory and use this to restrict the homotopy class of the trajectory, and next find an optimal trajectory with this restriction. The following sections present the algorithm in more detail.

Cell and word

With feasibility constraints (3.1.1) and additional constraints (3.1.2), we can divide the work space with hyperplanes of obstacles by value of binary variables.

As a result, a set of connected *cells* is built, whose union is the feasible space, Q, and the intersection is only the extended lines of faces of the obstacles (see Figure 3.3.1). Each cell can be identified by a unique *letter*, representing the vector of binary variables with one binary variable for each face of each obstacle. Every point in a particular cell will have the same letter representation. It must be noted that not all binary vectors define valid cells, and hence letters. The collection of all possible valid letters is defined as an *alphabet*.

Determining homotopy class of a trajectory is non-trivial. However, we use location information of intermediate key point, each represented by a letter in the alphabet. Assembling the sequence of letters corresponding to each key point of the trajectory and removing trivial repetitions will results in a *word*, which is a coarse representation of the trajectory. For example, the path shown in Figure 3.3.1 can be represented by the word $TPUVWQLJHG$. This can then be used to restrict trajectories to a homotopy class as will be seen in the following Section.

Finding Words in the same Homotopy Class

To find an optimal trajectory satisfying a given homotopy class constraint, we first construct W_h, the set of words of the same homotopy class with the required one. We construct W_h by starting with the word for the given initial trajectory; $W_h = \{w_0\}$. Then we choose a word $w_c \in W_h$ and expand the chosen word as follows. For example let $w_c = TPUVWQLJHG$ as in Figure 3.3.1. We choose two letters, say T and U. If there is an alternative path, like TXU(the gray plot in Figure3.3.1), for the path TPU, we construct the closed loop by reversing the new path, and obtain $TPUXT$ after removing duplicating letters. If the length of the closed loop is less than six, no obstacle lies in the closed loop (since we need to visit at least six cells to encircle a triangle). So we replace the path between the two chosen letters with the new path, and an expanded word representing the same homotopy class is achieved, $w_1 = TXUVWQLJHG$. The new word is added into W_h. We repeat this expansion until there are no more new words.

Finding the Optimal Trajectory

For a given word, $w_c \in W_h$, we parameterize the trajectory with N_s subtrajectories, where N_s is same as the length of w_c. Each subtrajectory is restricted to be in a particular cell specified by the corresponding letter in the word. Thus, all the binary variables, b_c, of the trajectory generation problem of (3.1.5) are fixed by the given word w_c, to reduce the optimization problem to

$$\begin{aligned} \min_{c} \quad & c^T H c \\ \text{s.t.} \quad & A_f c \leq \tilde{g}_f, \quad A_{eq} c = 0, \end{aligned} \tag{3.3.1}$$

which is obtained by substituting b_c in (3.1.5) and $\tilde{g}_f = g_f - D_f b_c$. As the resulting problem (3.3.1) is a quadratic program, we can find the global optimal trajectory for all words in W_h, which are in the given homotopy class.

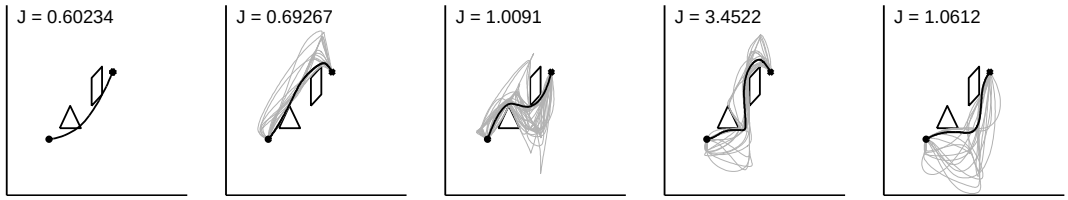


Figure 3.10: (a) Optimal trajectory without homotopy constraints (b)-(e) Trajectories with four different homotopy class constraints. The thick black curve is the optimal trajectory in each homotopy class and thin gray curves are the suboptimal trajectories for each word. The cost (J) for each case is specified on the upper left corners of plots.

However, it is not trivial to find the spending time in each cell to minimize the cost of the whole trajectory. To refine the trajectory further, we can adjust the time spent in each cell. With the final time, t_f, fixed, we can find an optimal time distribution by solving

$$\begin{aligned} \min_{t_j} \quad & \min_{c} \quad c^T H c \\ & \text{s.t.} \quad A_f c \leq \tilde{g}_f \\ & \qquad A_{eq} c = 0 \\ \text{s.t.} \quad & t_j \leq t_{j+1} \quad \text{for} \quad j = [0, ..., N_s - 1], \\ & t_0 = 0 \\ & t_{N_s} = t_f. \end{aligned} \tag{3.3.2}$$

As this problem is a nonlinear program, we cannot guarantee the global minimum. However, the trajectory is iteratively refined by starting with $\Delta t_j = t_{j+1} - t_j = \frac{t_f}{n_w}$ for $j \in [0, ..., N_s - 1]$ and solving (3.3.2) by an interior-point method. Although we can find an initial solution without iteration, a better trajectory can be obtained by iterating the time distribution. Moreover, since the optimization cost reduces with more iterations, this method can be considered as an anytime algorithm that produces better solutions with more time.

3.3.2 Simulation Results

To illustrate how the suggested algorithm works, we performed some simulations to generate optimal trajectories with various homotopy classes. In this simulation, we fix the final time $t_f = 10$ and find optimal trajectories for four homotopy classes. To reduce the computation time, we limit the maximum length of word to twelve.

The plot of Figure 3.10(a) shows the result of solving (3.1.5) without homotopy class constraints, resulting in an optimal cost of 0.60234. The plots of Figure 3.10(b)-3.10(e) show the result of solving (3.3.2) with four different homotopy class constraints, resulting in optimal costs that are greater than the global optimal one. When we search for trajectories with the same homotopy as the optimal trajectory achieved without homotopy class constraints (Figure 3.10(a)), the obtained optimal trajectory (Figure 3.10(e)) is a local optimal

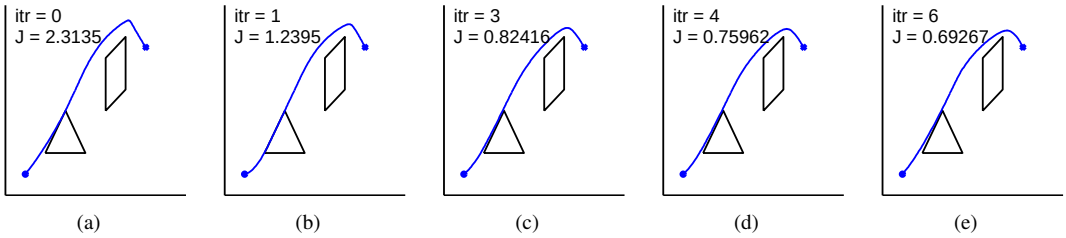


Figure 3.11: (a)-(e) Effect of varying the time distribution in each cell through iterations of the optimization (3.3.2). The number of iterations (itr) and cost are also specified on the upper left corner of each plot. Note that the cost converges to the local optimal cost of the case of Figure 3.10(b) in 6 iterations.

one with a larger cost. This disparity occurs due to restricting the trajectory to pass through certain cells and the fact that it is hard to find global optimal time distribution in each cell. The most optimal trajectory with homotopy class constraints lies in a different homotopy class from the global optimal one (Figure 3.10(b)). However, this is due to the symmetric arrangement of initial/final location of the trajectory and arrangement of obstacles.

With a fixed time distribution for each cell, the optimization reduces to a quadratic program for each word, which can be solved efficiently. To see the effect of optimizing the time distribution, we begin with a trajectory in the particular homotopy class of Figure 3.10(b) with equal time distribution over all the cells and iteratively optimize time distribution. The plots of Figure 3.11(a)-3.11(e) illustrate the changes in the trajectory and the corresponding cost with each iteration. Although this nested optimization is computationally expensive, with each iteration we get closer to the local optimal solution, resulting in an algorithm with *anytime* properties.

3.4 Conclusion

In this chapter, we have presented a method to find a smooth optimal trajectory subject to geometric and kinematic constraints, and restricted to a specific topology class. We used a MIQP to achieve global optimality. The homology constraint is considered by calculating the H-signature of the trajectory from its binary variables, which are a coarse representation of trajectory. The calculation of H-signature is quadratic in the binary variables but is reduced to a linear equation by introducing substitution binary variables. Then the resulting problem becomes a MIQP, which can be solved using an anytime numerical solver like CPLEX [52]. To find an optimal trajectory restricted to a specific homotopy class, we suitably modified a MIQP to partition the configuration space and by constructing a coarser representation of the trajectory in the form of a word to represent the homotopy class. The set of all words representing the same homotopy class is constructed, and a nested optimization is carried out to find a locally optimal trajectory restricted to a homotopy class.

Clearly reducing the computation time is the most important issue for this problem for practical applications. One direction is to reduce the computational complexity of the method by reducing the number of binary variables or choosing proper number of intermediate points or number of subtrajectories. The other direction is to develop proper solver to solve this problem efficiently. General MIQP solver like CPLEX does not consider the structure of the problem and relax the binary variables to real number on $[0, 1]$. However, we can divide the MIQP into two parts. First, we find the possible set of binary variables and solve QP

for the continuous variables. Or we can use numerical solvers with proper initial guess which is found by graph-search-based planner.

Chapter 4

Topological Exploration

In this chapter, we will present the mathematical framework and algorithms for multi-robot topological exploration of unknown environments in which the main goal is to identify the different topological classes of paths and thus efficiently distribute the task of exploration among different groups of robots. We consider two-dimensional configuration spaces. At any point in time, the robots' map consists of known, partially-mapped obstacles. The unknown, yet-to-be-explored area is mapped to a single point, thus giving us a *quotient space*. The topological classes on the quotient space allows us to define topological classes of paths connecting a robot pose to the unknown region in the original configuration space. Robots explore this configuration space choosing different homology classes when confronted by obstacles or walls. We illustrate the basic idea with simulations of small teams of robots. Experiments with a single robot illustrate the applicability of the method to robots that have small sensor footprints and limited computational resources. We also provide comparisons with a standard frontier-based algorithm. The work in this chapter was performed in close collaboration with Prof. Robert Ghrist and Dr. Subhrajit Bhattacharya. Much of this work was reported in [55].

4.1 Motivation

To motivate the approach in this chapter, consider the simple scenario in Figure 4.1(a) in which there is a group of robots at locations close to $\mathbf{p}$ equipped with sensors with a limited field of view mapping an unknown environment. In the figure, the current map consists of the three obstacles (marked in black) and the free space colored in pale blue. The region, L, in pale yellow is not visible to any of the sensors and hence is unknown. An information gain maximization based approach as in [85] or [89] will essentially give an unique gradient descent direction at the location of the robots (if all the robots are roughly the same location, the control inputs will also be very similar) and make the robots move together. However, clearly there are three distinct topological classes in this environment that can lead the robots to the unknown region (indicated by the blue dashed arrows in the figure). We are interested in methods that will maximize the collection of *information* by naturally assigning robots to different topological classes of paths.

A frontier-based exploration as in [41] would find three distinct paths or assignments in two steps [100]:

i. Identify the boundary between the known and the unknown regions and segment it to obtain its connected components (using an edge detection algorithm for example).

ii. Find optimal paths to each of the connected components (using Dijkstra's search).

While this method would perform satisfactorily in the example of Figure 4.1(a), remarkable outperformance of a topology-based method as ours is observed in scenarios like that in Figures 4.1(b) or 4.1(c) – when the known region is not simply-connected. Moreover, in a frontier-based algorithm like in [100], the additional step of identification of the connected components of the frontier may be expensive. When there are multiple robots, the standard frontier-based approach makes robot-frontier assignment by taking into consideration the *size* of the frontier [100]. In an indoor environment with lots of corridors and passages, possibly leading up to large open areas, the size of the frontier may not be the best indicator of information gain. Furthermore, when there are multiple groups of robots in different locations, performing distributed cooperative exploration, it is unclear how the groups can have a consistent way of referring to a particular frontier when communicating (without communicating the complete description of the frontier). Our topological approach, on the other hand, is completely free from the task of frontier identification or representation, and instead uses single pass of search/planning to discover paths in different topological classes, each of which is represented by a topological invariant (H-signature) that is consistent over the different groups (Figure 4.3(b)).

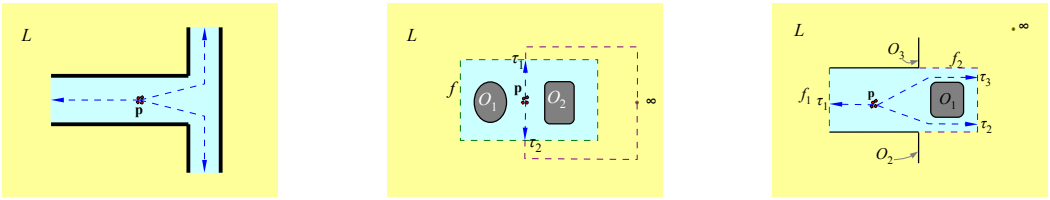
Consider the scenario illustrated in Figure 4.1(b), where a group of robots are provided with a partial map of the environment. There is a single frontier, f. A frontier-based approach would find a single shortest path to the frontier (either τ_1 or τ_2). However, there are two topological classes of paths in the plane punctured by the obstacle, O, that connect $\mathbf{p}$ to all points in L. As shown, τ_1 and τ_2 are two paths in different topological classes. Thus clearly, the number of frontiers do not correspond to the number of distinct non-looping topological classes when the explored/known free region is not simply connected. A similar example is shown in Figure 4.1(c), where $\mathcal{O}$ is the set of all obstacles that have been discovered. In particular, the group of robots have explored the perimeter of obstacle O_1, thus resulting in a map that is not simply connected. In this case although there are two connected components of the frontier, there are three topological classes of paths and therefore three directions for exploration. Clearly, deployment of groups of robots in each of the distinct topological classes will result in more efficient exploration.

Because our interest is in topological mapping, we will not concern ourselves with questions of localization, detection or mapping of obstacles or control. We will assume each robot is able to localize either using lasers and cameras or by using GPS, detect obstacles with a laser scanner, communicate with other robots, and avoid collisions with the environment. We implement our algorithm with nonholonomic robots in ROS, and demonstrate the multi robot exploration in simulation, along with comparisons with a frontier-based exploration algorithm. We also present experimental results with a single robot with a small field-of-view laser and odometry to illustrate the basic ideas in a real world setting.

4.2 The Quotient Space and H-signature

As mentioned in Chapter 2, the H-signature of paths can be used for finding different homology classes of paths connecting two points. However, for exploration we are interested in the topological classes of paths that emanate from a start coordinate, with the goal being not a single point but rather a set L. To adapt to this situation, we collapse the set L to a single point via the construction of a *quotient space* [69].

To adapt the definition of the H-signature in this context, we will collapse the entire unknown region to



(a) A group of robots, using their laser range sensors, finds 3 topological classes of paths leading to the unknown region, L. There are also 3 frontiers in this scenario.

(b) In this partially known environment there is a single frontier (green dashed curve), but 2 topological classes connecting the location of the robots and frontier, f. Such scenarios are natural when the known free region, $\mathbb{R}^2 - \mathcal{O} - L$, is not *simply-connected*.

(c) In this scenario the known region is not simply-connected and there are 2 frontiers. But on $\mathbb{R}^2 - \mathcal{O}$ as well as on $(\mathbb{R}^2 - \mathcal{O})/L$ there are 3 non-looping topological classes.

Figure 4.1: Partially explored environments. The group of robots (red dots) need to be split and deployed for exploration of the unknown regions (pale yellow region marked as L). The figures illustrate the distinction between frontier-based and topology-based deployments.

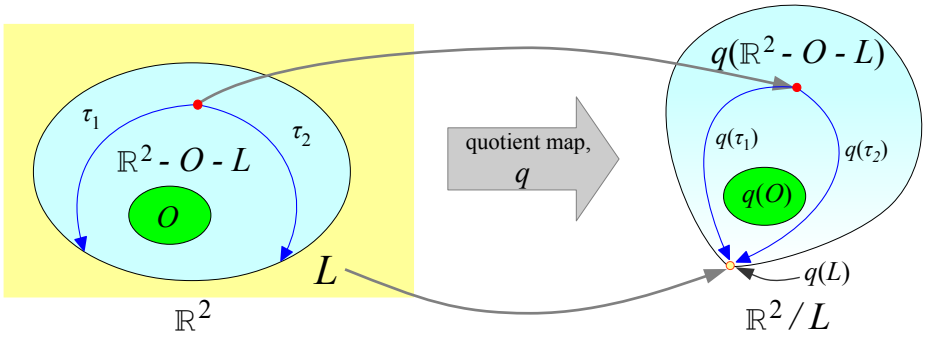


Figure 4.2: A simple illustration of a quotient map. The set L is collapsed to a point, $q(L)$. Here we consider the Euclidean plane, $\mathbb{R}^2$, with its subset L being the entire region outside a small disk on the plane. Collapsing L to a single point gives us the topological 2-sphere. All non-trivial 1-cycles (or closed loops) that completely lie in L become trivial in the quotient space under the quotient map, q.

a single abstract point via a *quotient map*, q, so that the space under consideration becomes $(\mathbb{R}^2 - \mathcal{O})/L$ (where $\mathcal{O}$ is the set of obstacles in the known region). The image of L under the quotient map, q, thus being a single point lets us use the notion of homology classes of paths connecting to this point from the image of the start coordinate on the quotient space (Figure 4.2). For a formal definition of quotient map see [69, 49].

The following proposition extends the homology class invariant function of H-signature to the quotient space $(\mathbb{R}^2 - \mathcal{O})/L$.

Proposition 4.2.1 (Homology invariant in quotient space [16])**.** *Let $\mathcal{O}$ be the collection of obstacles in $\mathbb{R}^2$ with respect to which we compute the H-signature as described in Section 2, and let $L \subset \mathbb{R}^2 - \mathcal{O}$. Let Q be the set of H-signatures of all closed loops (1-cycles) in $(\mathbb{R}^2 - \mathcal{O})$ contained entirely in L. Let τ_1 and τ_2 be two paths connecting two points, $\mathbf{s}, \mathbf{g} \in (\mathbb{R}^2 - \mathcal{O})$. Now consider the quotient map $q : (\mathbb{R}^2 - \mathcal{O}) \to (\mathbb{R}^2 - \mathcal{O})/L$. The images of the paths τ_1 and τ_2 under the action of q are homologous in $(\mathbb{R}^2 - \mathcal{O})/L$ iff $\mathcal{H}(\tau_1) - \mathcal{H}(\tau_2) \in Q$.*

Sketch of proof. First, note that the set Q is a countable set. Each element of Q corresponds to an element of the homology group $H_1(L; \mathbb{Z})$ [49]. The proof follows from the observation that by identifying L to a point under the quotient map, we essentially *trivialize* every closed loop (1-cycle) in L. This implies that the

loops that were non-trivial in L before applying the quotient map (*i.e.* whose H-signatures were not zero), need to be set to zero when we compute and compare the H-signatures in the quotient space. Thus, before applying the quotient map we would say that $\tau_1 \approx \tau_2$ (*i.e.* belong to same homology class) iff $\mathcal{H}(\tau_1) - \mathcal{H}(\tau_2) = \mathbf{0}$. However, after applying the quotient map, each element of Q, containing the H-signatures of non-trivial loops in L, are to be considered equivalent to $\mathbf{0}$. Thus the new criteria becomes $q(\tau_1) \approx q(\tau_2)$ (*i.e.* the images of the paths belong to same homology class in the quotient space) iff $\mathcal{H}(\tau_1) - \mathcal{H}(\tau_2) \in Q$.

For a more formal algebraic proof and an illustration demonstrating the concept behind the proof, see Section 7 of [16]. □

4.3 The Algorithm

In this section we provide a complete description of the algorithm for topological exploration with multiple robots while respecting the present constraints on the available space. We assume that the reader is familiar with the construction of the H-augmented graph [15] and the process of performing search (using Dijkstra's or A* algorithm) in it [25].

4.3.1 Representation

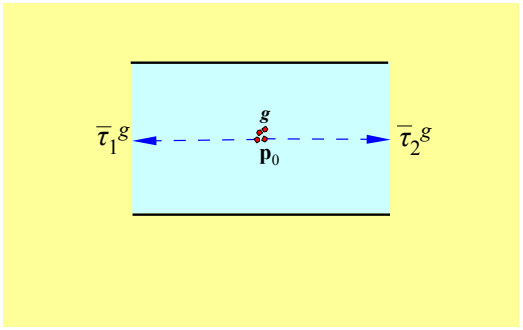
We discretize the environment (the subset of $\mathbb{R}^2$ that is of interest) into a uniform square grid and create a graph, $\mathcal{G}$, by placing a vertex in each square cell and connecting a cell with its neighbors using directed edges. More complex forms of discretization (triangulation, unstructured or adaptive discretization) can also be used. But to focus on the main contribution of this thesis, we choose the simplest discretization scheme. We maintain a probability map by associating an occupancy probability with each cell. The initial probability for each cell in a completely unknown environment is set to 0.5, and the *state* of each cell is designated as 'unknown'. As the laser sensor data are received, the probability map is updated. If the probability of a cell goes above a high threshold, T_{obs}, we designate the cell as an 'obstacle'. Otherwise if it goes below T_{free}, we designate it as a 'free' cell. This, at any instant of time, gives us an obstacle map (see Line 3 of Algorithm 1: *ToplogicalExplore*).

A *candidate point* (an arbitrarily chosen point) is placed inside each connected component of the unknown region (a point is chosen near the boundary of the region, and shifted, if possible, to create a padding). Like *representative point*, the exact location of a candidate point is not of significance as long as it falls inside the desired region.

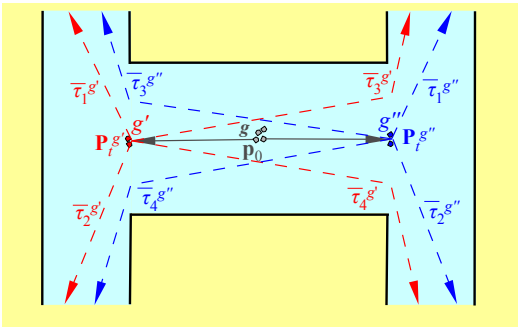
4.3.2 Multi-robot Exploration Algorithm

Suppose we start with N robots at a location, say $\mathbf{p}_0$, in the environment. At the beginning we have a single group of robots. The basic idea behind our algorithm is to split the group of robots based on the number of homology classes of paths discovered and deploy each newly-formed smaller group along those paths, and repeat this process for each subsequently formed group (Figure 4.3).

Discrete time is represented by t. The re-planning for paths does not happen in every time step, and instead happens at time steps $t_0, t_1, \cdots$. The values at the subscript of these time steps are the *planning cycle* numbers, and are denoted by the variable, $pl = 0, 1, 2, \cdots$.



(a) At $t = t_0 = 0$ a planning cycle starts. A single group of N robots starts at $\mathbf{p}_0$. Thus $G_{t_0} = \{\{1, 2, \cdots, N\}\} =: \{g\}$. The group finds 2 topological classes of paths: $\{\{\overline{\tau}_1^g, \overline{h}_1^g\}, \{\overline{\tau}_2^g, \overline{h}_2^g\}\} = FindPaths(\mathbf{p}_0, \emptyset)$. Thus the group splits into two groups, each containing $\sim N/2$ robots. The new groups are $g' = \{1, 2, \cdots, \lfloor N/2 \rfloor\}$ and $g'' = \{\lfloor N/2 \rfloor + 1, \cdots, N\}$ (see figure on the right), and they follow paths $\tau_1^{g'} := \overline{\tau}_1^g$ and $\tau_1^{g''} := \overline{\tau}_2^g$. At the end of this planning cycle, we set $t_1 = t$.



(b) At the beginning of the next planning cycle there are two groups: $G_{t_1} = \{g', g''\}$, when the condition in Line 4 of Algorithm *ToplogicalExplore* returns true. Thus, in this cycle of planning the groups obtain the following paths respectively:
$FindPaths(\mathbf{P}_{t_1}^{g'}, \tau_1^{g'})$
$= \{\{\overline{\tau}_1^{g'}, h_a\}, \{\overline{\tau}_2^{g'}, h_b\}, \{\overline{\tau}_3^{g'}, h_c\}, \{\overline{\tau}_4^{g'}, h_d\}\}$;
$FindPaths(\mathbf{P}_{t_1}^{g''}, \tau_1^{g''})$
$= \{\{\overline{\tau}_1^{g''}, h_c\}, \{\overline{\tau}_2^{g''}, h_d\}, \{\overline{\tau}_3^{g''}, h_b\}, \{\overline{\tau}_4^{g''}, h_a\}\}$.
Note the correspondence between the values of the H-signatures.

Figure 4.3: Illustration of algorithm *ToplogicalExplore*.

At any instant, the groups formed by the robots are represented by a partition of the set of robot indices, $\{1, 2, 3, \cdots, N\}$. We represent that partition (created after *planning cycle*, pl) by the ordered set $G_{pl} = \left\{\{r_{pl}^{1,1}, r_{pl}^{1,2}, \cdots\}, \{r_{pl}^{2,1}, \cdots\}, \cdots\right\}$. A group, g, is simply a partition element $g \in G_{pl}$, and variables giving attributes to the groups are indexed by g (*e.g.*, τ_{pl}^g). $|G_{pl}|$ denotes the number of groups.

The planning cycle, pl, creates a set of paths, $\tau_{pl}^g,\ g \in G_{pl}$ (with H-signature, h_{pl}^g, w.r.t. base-point $\mathbf{p}_0$ – see Section 4.3.2), that the groups need to follow. We will unambiguously (and without going into implementational details) refer to two obvious components of each such path: *traversed part* and the *untraversed part*.

Each group of robots, during their coordinated travel together as a group, has a representative location (a point in configuration space), with respect to which all computations of paths are performed. This point, representing the position of the group $g \in G_{pl}$ at time t (with $t_{pl} \leq t < t_{pl+1}$), is denoted as $\mathbf{P}_t^g$. On the contrary, the positions of individual robots are denoted by $\mathbf{p}_t^r,\ r \in \{1, 2, \cdots, N\}$ (and thus at the individual level of robot r, the control objective will be to reach $\mathbf{P}_t^g$, where $r \in g$). We represent the path history of the g^{th} group at the time instant t by $\mathbf{P}_{0:t}^g$.

At $t = 0, pl = 0$, we start with a single group, $G_0 = \{\{1, 2, \cdots, N\}\}$. After obtaining the first few sets of laser sensor data and building the occupancy map in the neighborhood of the robot group, the algorithm *ToplogicalExplore* (Algorithm 1) is used to direct the exploration task. Figure 4.3 illustrates the working of the algorithm.

We use '$*$' in place of a index of a variable to denote the entire set of variables over all the possible indices (*e.g.*, $\tau_{pl}^* = \{\tau_{pl}^g \mid g \in G_{pl}\}$). An *overline* over a variable is used to emphasize that it is a temporary variable.

Algorithm 1**:** Pseudocode for *ToplogicalExplore*:

1. $|t = 0; pl = 0; t_{pl} = 0; G_{pl} = \{\{1, 2, \cdots, N\}\}$

2. |**while** TRUE
3. |*i.* Update probability map based on laser sensor data.
   |*ii.* Threshold probability map to generate obstacle map.
4. |**if** $t == 0$ **OR** map has changed significantly
   | **OR** a group has reached its immediate goal
5. |*i.* Place representative points on newly discovered obstacles,
   |*ii.* Place candidate points in connected components of
   | unexplored regions.
6. |**for each** $g \in G_{pl}$ // Plan new paths
7. |$\overline{\gamma}^g = \{\{\overline{\tau}_1^g, \overline{h}_1^g\}, \{\overline{\tau}_2^g, \overline{h}_2^g\}, \cdots\} =$
   | $FindPaths(\mathbf{P}_t^g, \mathbf{P}_{0:t}^g)$
8. |**end** for each
9. |**if** $\overline{\gamma}^g = \emptyset,\ \forall g \in G_{pl}$ // No path found. All explored!
10. |**break** while loop
11. |**end** if
12. |Set $G_{pl+1} = G_{pl}$ // Copy groupings from previous plan cycle.
13. |$\{\overline{H}^g \mid g \in G_{pl+1}\} =$
    | $AssignHomologyClassesToGroups(c(\overline{\tau}_*^*), \overline{h}_*^*, h_{pl}^*)$
14. |$\{G_{pl+1}, \overline{\gamma}^*\} =$
    | $CheckNearbyGroupsForRedistribution(\mathbf{P}_t^*, G_{pl+1}, \overline{H}^*)$
15. |**for each** $g \in G_{pl+1}$
16. |**if** $|\overline{H}^g| == 0$ // Group not assigned any homology class.
17. |$\{G_{pl+1}, \overline{H}^*, \overline{\gamma}^*\} = RejoinWithClosestGroup(g)$
18. |**end** if
19. |**end** for each
20. |**for each** $g \in G_{pl+1}$
21. |**if** $|\overline{H}^g| > 0$ // Group assigned multiple homology classes.
22. |$\{G_{pl+1}, \overline{H}^*, \overline{\gamma}^*\} = SplitGroup(g)$
23. |**end** if
24. |**end** for each
    | // At this point each $\overline{H}^g,\ g \in G_{pl+1}$ contains one H-signature.
    | // The new group structure is present in G_{pl+1}.
25. |**for each** $g \in G_{pl+1}$
26. |$\tau_{pl+1}^g = \overline{\tau}_k^g,\ h_{pl+1}^g = \overline{h}_k^g,$ k is such that $\overline{h}_k^g \in \overline{H}^g$
27. |**end** for each
28. |$t_{pl+1} = t;\ pl++$
29. |**end** if
30. |**for each** $g \in G_{pl}$
31. |Choose the next point ($(t - t_{pl})^{th}$ point in τ_{pl}^g), $\mathbf{P}_{t+1}^g \in \tau_{pl}^g$.
32. |$\mathbf{P}_{0:t+1}^g = \mathbf{P}_{0:t}^g \sqcup \mathbf{P}_{t+1}^g$

33. |**for each** $r \in g$

34. |Move robot r towards $\mathbf{P}^{g}_{t+1}$ via the shortest path in the map.

| // Controller for making robot follow planned path.

35. |**end** for each

36. **end** for each

37. $|t++$

38. **end** while

In Line 4 of the above algorithm, the condition for checking whether the '*map has changed significantly*' consists of two checks:

i. If any of the most recently planned paths (*i.e.*, $\tau^{i}_{pl},\ i \in G_t$) has become invalid (blocked by newly discovered obstacles).

ii. The number of cells in the environment that have changed state (*i.e.* from 'unknown' to 'free' or 'obstacle') is greater than a threshold.

Below are brief descriptions of each of the remaining subroutines used in the algorithm.

FindPaths$(\mathbf{P}, \tau)$

(Refer to Figure 4.3) This subroutine is used to find all paths emanating from $\mathbf{P}$ in the different topological classes. The subroutine also returns the H-signature of the planned path appended with the already traversed path, τ. This requires searching in the H-augmented graph, $\mathcal{G}_H$, as described in [15]. However, in the search algorithm we *initiate* the *open set* with the vertex $\{\mathbf{P}, \mathcal{H}(\tau)\}$ (*i.e.*, instead of using $\mathbf{0}$ as the H-signature of the start vertex, we use $\mathcal{H}(\tau)$ – the H-signature of the traversed path, τ). Consequently we expand the vertices in $\mathcal{G}_H$ as usual. This ensures that we consider $\mathbf{p}_0$ as the *base point* of the space so that the value of the H-signature remains consistent over the different groups and over time (see Figure 4.3(b)). Vertices that lie in the explored region are expanded, and a path is stored every time a vertex connected to the unknown region is reached via a new homology class (identified by the sum of the H-signature of the expanded vertex and the H-signature of a path connecting that vertex with the candidate point in unknown region).

Note that according to Proposition 4.2.1, the way we determine whether H-signatures h and h' represent the same homology class in the quotient space is to check if the elements of the difference, $h - h'$ (which, recall from the definition of H-signature, is a vector of complex numbers), are either ***i.*** all equal when the unknown region is not simply connected (*i.e.*, the unknown region that extends to the boundary of the environment), or, ***ii.*** all zero when the unknown region is simply connected (for all other unknown regions). If none of these is true, they represent different homology classes. Using a method similar to [15], we do not allow path that loop around obstacles. Moreover we do not place *representative points* on obstacles smaller than a threshold radius, thus avoiding multiplicity of topological classes merely due to sensor noise.

AssignHomologyClassesToGroups

The number of paths returned by the '*FindPaths*' procedure will be the same for each of the groups $g \in G_{pl}$ (see Figure 4.3(b)). Since we used the same base-point, $\mathbf{P}_0$, for the searches for each group, we will obtain the same set of H-signatures for each group from the search in Line 7, although the paths will of course be different.

The purpose of this subroutine is to make the assignment of each of the homology classes to the different groups of robots based on the cost of the planned paths, $c(\overline{\tau}^*_*)$, their H-signatures, $\overline{h}^*_*$, and the H-signature of the paths assigned in the last plan cycle, h^*_{pl}. The basic strategy for doing this is as follows:

i. If, for a group g, the H-signature of the last planned path, h^g_{pl}, that it has been following, is found in the result returned by *FindPaths*, that homology class is assigned to the group g (the H-signature comparison being made with respect to obstacles that are common to the time instants when the last plan was made and the current time). This ensures that a group (or one of its subgroups) keep following the homology class that it has been following.

ii. Whichever homology class remain unassigned after this is assigned to group for which the path corresponding to the class is shortest.

The H-signatures of the homology classes assigned to group $g \in G_{pl+1}$ is fixed in $\overline{H}^g$ (*i.e.*, it is a set of H-signatures, $\overline{H}^g = \{\overline{\eta}^g_1, \overline{\eta}^g_2, \cdots\}$).

CheckNearbyGroupsForRedistribution

If a group has been assigned homology classes more than the number of robots available in that group (*i.e.*, $|\overline{H}^g| > |g|$), then it is checked if there is another *nearby* group, g', such that $c|\overline{H}^{g'}| < |g'|$, $\|\mathbf{P}^{g'}_t - \mathbf{P}^g_t\| < R$ ($c > 1, R > 0$ are parameters). If so, a re-shuffling of the groups is performed (with dome robots from g' being transferred to g) and the new group arrangement is returned to G_{pl+1}. Since the content of each group gets changed, the indices of $\overline{\gamma}^*$ are updated accordingly.

RejoinWithClosestGroup

This subroutine gets triggered when a group is not assigned any homology class. The reason for this is typically two-fold:

i. Sometimes a spurious homology class may be observed because of incorrect laser readings, which would soon turn out to be blocked as new sensor data arrives, thus resulting in some of the recently created group to be assigned no paths.

ii. A group can reach a dead-end in the environment (*e.g.*, end of a corridor).

This requires that we rejoin those groups with other groups so that they don't remain idle. We first look for closest "cousin" groups (groups having common distant parent – group at an earlier plan cycle from which the current groups originated – see $SplitGroup$ next) that are not more than D generations apart. This requires a traversal of D levels of the family tree (the sets $G_{pl}, G_{pl-1}, G_{pl-2}, \cdots, G_{pl-D}$ contain all the information required for this) and identification of the closest *cousin*. If such a cousin cannot be found, the group is joined with the distance-wise closest group in the environment. The subroutine returns the new grouping (*i.e.* partition of the set $\{1, 2, \cdots, N\}$) and the corresponding re-ordering that is required in $\overline{H}^*$. Since $\overline{\gamma}^*$ and $\overline{H}^*$ are indexed by the groups, an update of their indices is also required (and removal of the elements corresponding to the joined, hence no-more existing, group).

SplitGroup

If $\overline{H}^g$ contains more than one element (*i.e.* multiple homology classes assigned to a single group of robots), the group will be split into sub-groups of almost-equal sizes and at most one homology class will be assigned to each of the sub-groups. Thus, if there aren't enough robots in the group (*i.e.* $|\overline{H}^g| > |g|$), clearly a choice has to be made and some of the homology classes has to be left unattended for future exploration. Under such situations the unattended homology classes are removed from $\overline{H}^g$. As before, the indices of $\overline{\gamma}^*$ and $\overline{H}^*$ are updated.

4.3.3 Distributed Implementation

It is to be noted that the algorithm *ToplogicalExplore* can be implemented in a distributed manner where the i^{th} group performs its own computation for the robots in the group. In a distributed implementation the 'for each' loops starting at Lines 6, 15, 20, 25 and 30 would be replaced by computation for the respective group only in their respective threads. Each group would maintain its own probability map and update it based on the laser sensor readings. Each group also broadcasts the changes in its own map so that the other groups in the environment can update their maps (a communication protocol similar to that in [17]). Moreover, when one group decides that a re-planning of path is required (condition in Line 4 becomes true), all the groups are communicated the decision and they come to a consensus to re-plan. Since the procedure $AssignHomologyClassesToGroups$ requires a consensus, the groups communicate the cost of their respective planned paths as well.

4.4 Results

We implemented the *ToplogicalExplore* algorithm on ROS (Robot Operating System), that lets us accurately simulate robot dynamics, actuator noise and sensor noise. Although our current implementation is mostly centralized and runs on a single processor, the overall structure of the algorithm is perfectly suited for distributed implementation on multiple parallel processors as described in Section 4.3.3. Such an implementation is within the scope of future work.

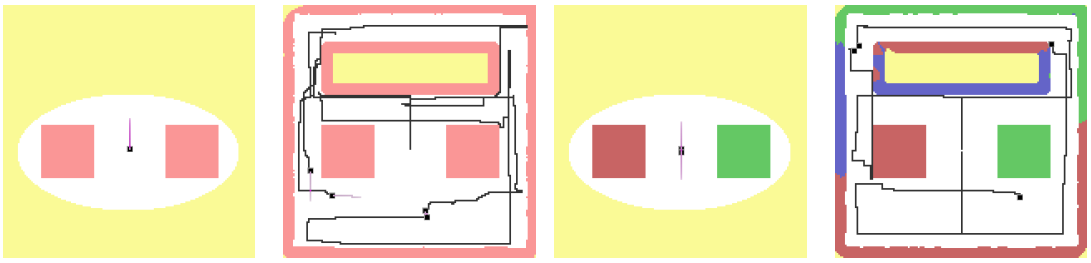
We also provide extensive comparison with the frontier-based algorithm described in [100] (the implementation of which was also made in ROS, with identical models of robot dynamics, sensor and actuator). Section 4.4.1 illustrates, using a simple environment, why our algorithm logically outperforms a frontier-based algorithm. Section 4.4.2 demonstrates similar performance comparison for a more complex indoor environment.

All simulations were run on a dual core machine with processor clock speed of 2.6GHz and 4GB memory. Note that the run times reported involve the complete dynamic simulation of the non-holonomic robots.

4.4.1 Partially Known Environment

We consider a simple partially known environment that is 30m $\times$ 30m in size, discretized by 0.1m $\times$ 0.1m cells, with 4 robots exploring it. The environment has 3 rectangular obstacles, of which two fall inside the initially known elliptical region as shown in Figures 4.4(a) and 4.4(c). The initial known region, as clearly seen, is not simply-connected. Consequently, the number of topological classes do not correspond

to the number of frontiers. Thus, using a frontier-based algorithm (as described in [100]) the entire group of 4 robots are driven towards the single frontier as shown in Figure 4.4(a). However, using our topological exploration, the initial group of robots discover two topological classes of paths and hence split up into two sub-groups as seen in Figure 4.4(c). This, without surprise, results in more efficient exploration of the environment. Our $TopologicalExplore$ algorithm explores the entire environment in 1045 iterations (and actual run time of $\sim$ 35mins), while the frontier-based algorithm took 2359 iterations (and run time of $\sim$ 78mins).



(a) $t = 3$: A *frontier-based exploration* algorithm initially finds a single frontier and plans a path to drive all the robots towards it.

(b) Using the *frontier-based algorithm* the robots explore the entire environment in $t = 2359$ iterations.

(c) $t = 3$: Our $TopologicalExplore$ algorithm finds 2 topological classes of paths and hence splits the group of robots into two.

(d) Using $TopologicalExplore$ algorithm the robots explore the entire environment in $t = 1045$ iterations.

Figure 4.4: Comparison between the frontier-based exploration algorithm (top row) of [100] and our $TopologicalExplore$ algorithm (bottom row) in a partially-known environment using 4 robots. The purple curves show parts of the planned paths, while black represents traversed paths. White is known/explored, while light yellow is the unknown region.

This example illustrates how a topological approach to exploration, as ours, visibly and structurally outperforms standard frontier-based approaches in cases when the known environment is not simply-connected.

4.4.2 Simulations of Multi-Robot Topological Exploration

Figure 4.6 shows an example with eight robots. The environment used is a part of the 4^{th} floor of the Levine hall at the University of Pennsylvania (a 21.3m $\times$ 34.2m environment, discretized by 0.1m $\times$ 0.1m cells). In Figure 4.6(a) the single group of robots discovers two topological classes, and hence splits into two groups, each consisting of four robots (Figure 4.6(b)). In Figure 4.6(c) each of those groups get assigned two topological classes to discover, thus each splitting further into groups of two robots (Figure 4.6(d)). Further splitting of three of those groups happen in Figure 4.6(e). Following which, as some of the groups end up exploring the homology classes assigned to them, they rejoin the other groups to help explore whatever remains.

Figure 4.6(i) shows comparison with the final result obtained using the frontier-based approach of [100]. Even in this case not only the number of iterations required using the frontier-based approach is higher, the actual time required for computation was also higher in case of the frontier-based exploration. The frontier-based approach took $\sim$ 100mins, while our algorithm took $\sim$ 69mins to completely explore this particular environment.

4.4.3 Experiment with a Single Robot

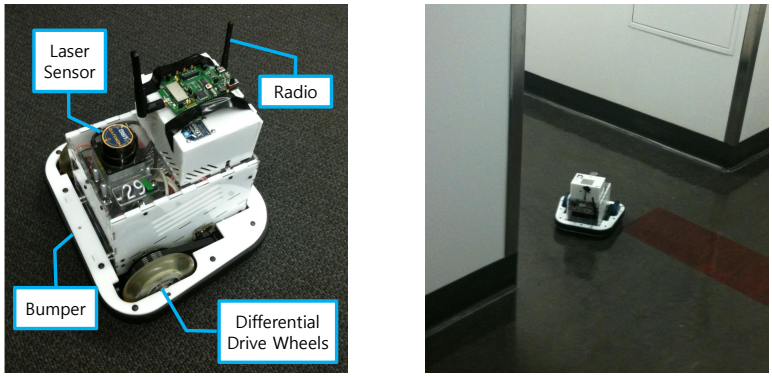


Figure 4.5: The SCARAB mobile robot platform [65]

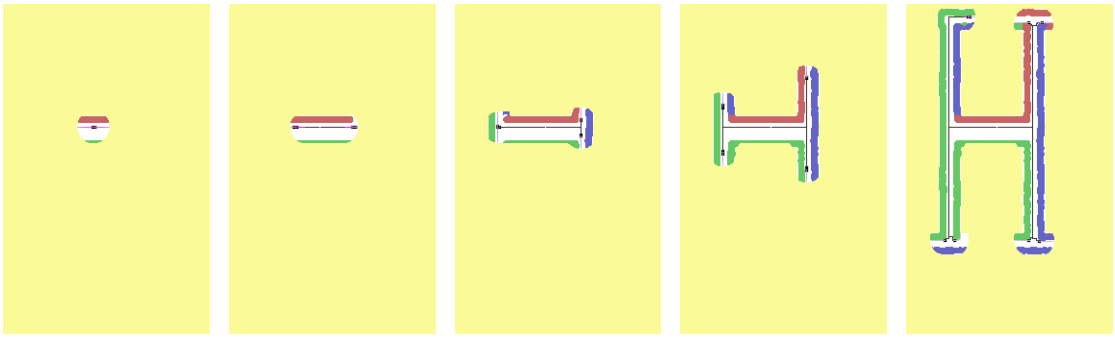
To demonstrate practical applicability, we implemented our algorithm on a mobile robot platform developed in the GRASP laboratory and known as the SCARAB [65]. Figure 4.5 illustrates the various components of the experimental platform and a snapshot of the robot in action. To localize the robot we currently use an adaptive Monte Carlo localization [44] module that relies on laser sensor data. Having multiple robots in the environment would not only require an additional local collision check layer, but also an additional complexity for localization.

The overall *ToplogicalExplore* algorithm, even when there is a single robot, remains the same. The key feature during the execution, however, is that we always have a single group of robots consisting of a single robot (*i.e.*, $G_{pl} = \{\{1\}\}$), and whenever the $SplitGroup$ subroutine is called, the group/robot has to choose one of the paths.

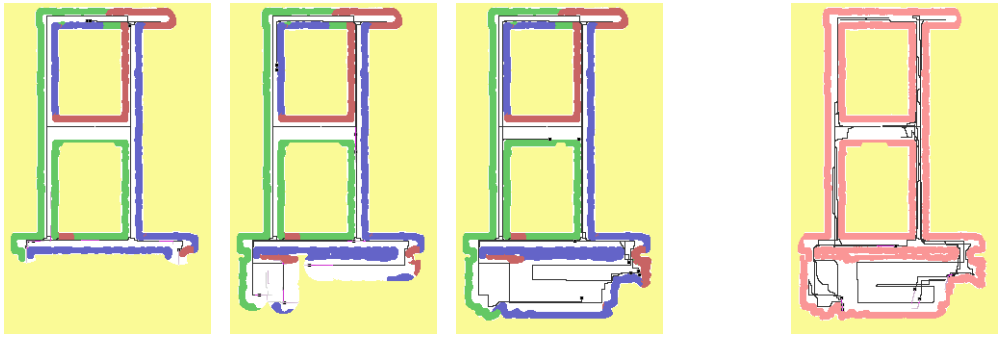
We performed the single-robot experiment in the same indoor environment (the blue-print of which we used to perform the multi-robot simulations). However, we sealed the two entrances at lower left and lower right leading to the larger room at the bottom. Figure 4.7 shows the result. In Figure 4.7(a), the robot starts from the bottom left corner and explores the environment to initially find three topological classes. The robot follows one of these paths that lead to the frontier 2 in Figure 4.7(a) (In the figures we number the frontiers for convenience of referencing. It should however be noted that at no point in our algorithm do we need to compute or identify the frontiers or its connected components). When the robot finds further branches, it keeps on following the path with the current H-signature (thus, for example, reaching frontier 2 in Figure 4.7(b)). When there are no more feasible paths with the current H-signature (*e.g.*, the frontier 2 disappears in Figure 4.7(c)), the robot starts following the shortest path with a new H-signature to a new frontier (*e.g.*, frontier 5 in Figure 4.7(c)). This process continues until there are no frontiers left, hence completing the process of building the map (Figure 4.7(e)).

4.5 Conclusion

In this chapter, we have presented an algorithm to explore an unknown or partially known environment by gradually building a topological description of the environment. Using the notion of quotient spaces,

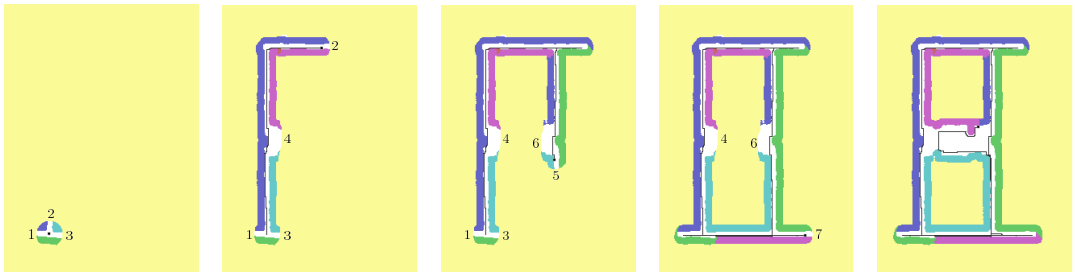


(a) $t = 4$. (b) $t = 136$. (c) $t = 186$. (d) $t = 275$. (e) $t = 580$.



(f) $t = 891$. (g) $t = 1180$. (h) $t = 1790$ (exploration complete). (i) *Comparison:* Final result ($t = 2638$ iterations) using frontier-based approach of [100].

Figure 4.6: (a)-(h): Simulation result with 8 robots exploring an indoor office-like environment. (i): Comparison of performance with frontier-based algorithm of [100] (in the same environment, with same number of robots and same initial configurations).



(a) $t = 1$. (b) $t = 310$. (c) $t = 540$. (d) $t = 900$. (e) $t = 1226$.

Figure 4.7: Experiment result with a single robot exploring an indoor office-like environment.

optimal paths in different topological classes leading up to the unknown region were found by searching in the H-augmented graph. Groups of robots are split into subgroups with each subgroup being assigned to a different homology class to enable efficient exploration of the environment. In contrast to previous work, the exploration is guided by topological and not metric information about the world and is ideally suited to obtaining a coarse topological map without detailed metric information. We demonstrated the performance of our algorithm in simulation using multiple robots, and in experiment using a single robot. We also provided a comparison of performance between our algorithm and a frontier-based approach.

Chapter 5

Manipulation with Cables

Cables are widely used to transport power or towing payload. Before we present the application how we can manipulate a number of obejcts with a single cable, we briefly discuss another problem to manipulate or transport a heavy object or payload with multiple small robots and cables in this chapter. The work in Section 5.1.1 was performed in close collaboration with Dr. Peng Cheng and Dr. Jonathan Fink. Much of this work was reported in [24]. The work in Section 5.2 was performed in close collaboration with Dr. Jonathan Fink and Dr. Nathan Michael. Much of this work was reported in [67, 40].

5.1 Cooperative Towing With Multiple Ground Robots

In this section, we address the cooperative towing of a payload by a mobile robot that moves in the plane. Robot pulls via cable attached to an object or a pallet carrying a payload and coordinates its motion to manipulate the payload through a planar, warehouse-like environment. We formulate a quasi-static model for manipulation and derive equations of motion that yield the motion of the payload for a prescribed motion of the robot in the presence of dry friction and tension constraints. We derive conditions of stable equilibrium for a robot towing the payload.

5.1.1 The Quasi-Static Model for Cooperative Towing

Our task is to control a robot so it can tow or carry an object subject to gravity and frictional forces from any initial part configuration to a desired goal part configuration. We want to achieve a specified degree of precision in positioning and orienting the object at a desired final position and orientation.

The different variables characterizing a robot towing a payload are shown in Figure 5.1. The position and orientation of the payload is given by (X, Y, θ) in the world frame. The velocity of the payload is a twist (in the plane) in the body-fixed frame, $x_b - y_b$, attached to the payload: $\xi = [\dot{x}, \dot{y}, \dot{\theta}]^T$. We will use this body-fixed frame representation for the twist throughout the chapter.

Let R_j denote the position of a reference point on robot j. There are m robots, each of which is attached via an inextensible cable that connects the point R_j on robot j and the point P_j on the payload. If $\left|\overrightarrow{P_j R_j}\right|$ equals the free length of the cable, then the dot product of the unit vector u_j with the relative velocity of the

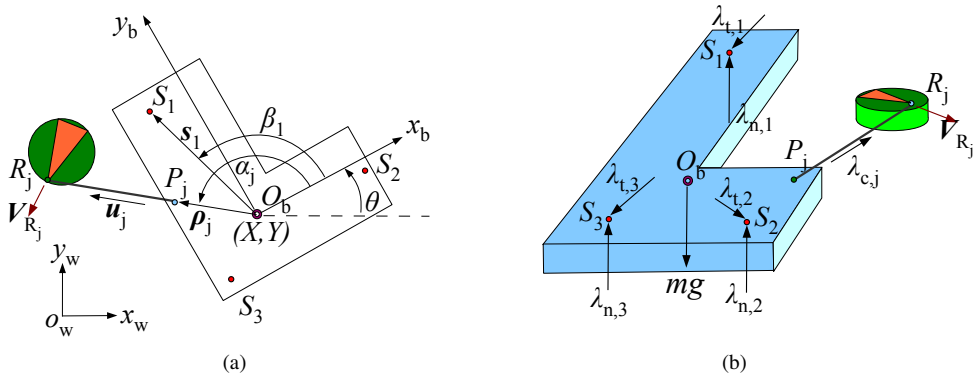


Figure 5.1: Quasi-static manipulation: The object is supported by three support points, S_i, with normal forces (out of the plane), $\lambda_{n,i}$ and tangential frictional forces, $\lambda_{t,i}$. It is pulled by m cables, each exerting a force $\lambda_{c,j}$. Note the robot R_j pulls by moving the object with a prescribed (given) velocity, V_{R_j}.
(This figure is taken from [24].)

point P_j to the point R_j is non negative, which can be written as unilateral kinematic constraints[1]:

$$A\xi \geq b \tag{5.1.1}$$

where

$$A = \begin{bmatrix} A_1 \\ \dots \\ A_m \end{bmatrix}, \; b = \begin{bmatrix} b_1 \\ \dots \\ b_m \end{bmatrix} \tag{5.1.2}$$

where A_j is a function of the unit vector u_j, showing the direction of the cable j, and the position vector $\rho_j = \overrightarrow{O_b P_j}$:

$$A_j^T = \begin{bmatrix} u_j \cdot \mathbf{i} \\ u_j \cdot \mathbf{j} \\ (\rho_j \times u_j) \cdot \mathbf{k} \end{bmatrix}, \tag{5.1.3}$$

where $\mathbf{i}, \mathbf{j}$ and $\mathbf{k}$ are respectively unit vectors along x_b, y_b, and z_b axis, and b_j is a function of u_j and the velocity $\mathbf{v}_{R,j}$ of the towing robot j:

$$b_j = u_j^T \mathbf{v}_{R,j}. \tag{5.1.4}$$

Note that $b_j \in [\, -v_R^{\max}, v_R^{\max} \,]$ because the robot velocity $\mathbf{v}_{R,j}$ is bounded by $v_R^{\max}$.
The set of *twists of freedom* [62] is defined with respect to the tuple (A, b) as follows:

$$\Sigma_{(A,b)} = \{\, \xi \,|\, A\xi \geq b \,\}. \tag{5.1.5}$$

Note that this set is determined by the towing configuration of the robots and the payload, specified by

[1] The vector inequality denotes that each element of the vector satisfies the inequality.

the matrix A, and the velocities of the robots, given by the vector b.

The kinematics-statics duality is evident in this problem. λ_c, the m-vector of cable tensions, is non negative and is non zero only when the equality in (5.1.1) is satisfied. Thus we write complementarity constraints:

$$0 \leq \lambda_c \perp A\xi - b \geq 0, \tag{5.1.6}$$

where "$\perp$" implies $\lambda_{c,j}(A\xi - b)_j = 0$. Since λ_c and $A\xi - b$ are non negative, $\lambda_c^T(A\xi - b) = 0$.

In order to model the dry friction between the object and the support surface, we assume that the object is supported by a finite number of frictional point contacts. For three non collinear support points, the support forces $\lambda_{n,i} \geq 0, i = 1, 2, 3$ can be uniquely obtained from the following equation

$$\begin{bmatrix} 1 & 1 & 1 \\ y_{s,1} & y_{s,2} & y_{s,3} \\ x_{s,1} & x_{s,2} & x_{s,3} \end{bmatrix} \begin{bmatrix} \lambda_{n,1} \\ \lambda_{n,2} \\ \lambda_{n,3} \end{bmatrix} = \begin{bmatrix} mg \\ 0 \\ 0 \end{bmatrix}, \tag{5.1.7}$$

where $(x_{s,i}, y_{s,i})$ for $i = 1, 2, 3$ are the coordinates of the support points in the body-fixed frame.

If the object undergoes quasi-static motion, the tensions associated with m cables and the frictional forces $(\lambda_{t,i,x}, \lambda_{t,i,y})$ at each of the three support points ($i = 1, 2, 3$) must add to zero. Thus, we have the equilibrium equations:

$$B^T\lambda_t + A^T\lambda_c = 0, \quad \lambda_c \geq 0 \tag{5.1.8}$$

where B is a full rank 6×3 matrix:

$$B^T = \begin{bmatrix} B_1^T & B_2^T & B_3^T \end{bmatrix} = \begin{bmatrix} 1 & 0 & 1 & 0 & 1 & 0 \\ 0 & 1 & 0 & 1 & 0 & 1 \\ -y_{s,1} & x_{s,1} & -y_{s,2} & x_{s,2} & -y_{s,3} & x_{s,3} \end{bmatrix} \tag{5.1.9}$$

with

$$B_i = \begin{bmatrix} 1 & 0 & -y_{s,i} \\ 0 & 1 & x_{s,i} \end{bmatrix}$$

and λ_t is an unknown 6-vector with components in the body-fixed frame:

$$\lambda_t = \begin{bmatrix} \lambda_{t,1}^T \\ \lambda_{t,2}^T \\ \lambda_{t,3}^T \end{bmatrix} \tag{5.1.10}$$

with

$$\lambda_{t,i} = \begin{bmatrix} \lambda_{t,i,x} \\ \lambda_{t,i,y} \end{bmatrix}. \tag{5.1.11}$$

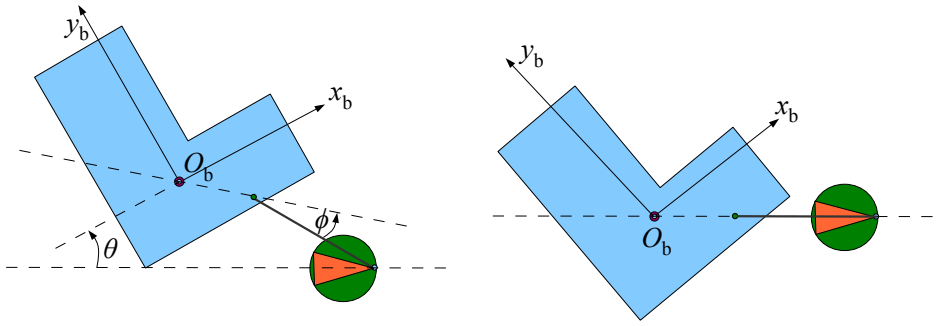


Figure 5.2: (Left) Arbitrary initial configuration. (Right) Stable equilibrium configuration. (This figure is taken from [24].)

We use $\mathcal{FC}_i$ to denote the friction cone at the ith support point defined by Coulomb friction:

$$0 \leq \|\lambda_{t,i}\|_2 = \sqrt{\lambda_{t,i,x}^2 + \lambda_{t,i,y}^2} \leq \mu\lambda_{n,i}. \tag{5.1.12}$$

Note that $\mathcal{FC}_i$ is the friction cone with a known $\lambda_{n,i}$.

For a given object twist, the velocity vector of the support point can be written in the body-fixed frame:

$$v_{t,i} = B_i\,\xi = \begin{bmatrix} 1 & 0 & -y_{s,i} \\ 0 & 1 & x_{s,i} \end{bmatrix} \xi. \tag{5.1.13}$$

From Coulomb's law, the friction forces are equal to $\mu\lambda_{n,i}$ and are opposite to the direction of slip, except if the slip is zero when the magnitude is indeterminate. This can be written explicitly as:

$$\lambda_{t,i}(\xi) \in \underset{\lambda_i \in \mathcal{FC}_i}{\operatorname{argmin}} \; v_{t,i}(\xi)^T \lambda_i,$$

or in aggregate form,

$$\lambda_t(\xi) \in \underset{\lambda_i \in \mathcal{FC}_i}{\operatorname{argmin}} \; \xi^T B^T \lambda. \tag{5.1.14}$$

It is not too hard to verify that this is equivalent to the Coulomb friction law [2].

The existence and uniqueness of solutions to (5.1.6), (5.1.8), and (5.1.14) have been prove in [23].

5.1.2 Equilibrium Analysis

In this section, we will study the equilibrium of the towing system, *i.e.*, the system has an invariant state, when one or two robots move along straight lines.

One-robot towing case

The single robot towing system will converge to an equilibrium in which the system exhibits pure translation and the cable, the robot, and the center of mass of the part will be aligned as shown in Figure 5.2. The result is stated in the following theorem.

Theorem 5.1.1. If a single robot tows the part with positive cable tension and moves in the invariant direction along a straight line, the angle ϕ will converge to zero.

Proof. Because the cable will have positive tension, the kinematic constraint will be an equality constraint and the part twist will be the result of the following optimization problem:

$$\xi^*(\phi) = \begin{bmatrix} \dot{x}(\phi) \\ \dot{y}(\phi) \\ \dot{\theta}(\phi) \end{bmatrix} = \begin{array}{ll} \text{argmin}_\xi & -\varphi(\xi) \\ \text{s. t.} & A(\phi) = b \end{array} . \tag{5.1.15}$$

We will prove this by showing that

1) $\dot{\theta}\phi < 0$.

2) $\dot{\theta} = 0$ if $\phi = 0$.

We will first prove that $\xi^*(\phi)$ (and therefore $\dot{\theta}(\phi)$) is a continuous function of ϕ in **Step 1**. Second, we show that $\dot{\theta} = 0$ if and only if $\phi = 0$, $-\pi$, or π in **Step 2**. Finally, we can infer that $\dot{\theta}$ is negative when $\phi \in (0, \pi)$ and positive when $\phi \in (-\pi, 0)$ as a property of the continuous function.

Step 1: We will first prove that $\xi^*(\phi)$ is well-defined, and then is continuous.

Because there is only one robot, the matrix $A(\phi)$ is full rank. It is easy to check that

$$\exists\, \lambda_c > 0,\, A^T \lambda_c = 0 \tag{5.1.16}$$

is not true because the unique solution to its equality is $\lambda_c = 0$ which does not satisfy the inequality. By Stiemke's lemma, (5.1.16) is false implies that

$$\exists\, \xi \,:\, A\,\xi > 0 \tag{5.1.17}$$

is true. Therefore there must exist twists that satisfy the constraint (5.1.17). For any given b, if elements $b_j \leq 0$, (5.1.17) implies the constraint $A_j \xi \geq b_j$ is satisfied. If there are elements $b_j > 0$, then we can always scale ξ with a positive scalar so that a feasible ξ that satisfies $A_j \xi \geq b_j$ can be found. Thus, $\Sigma_{(A,b)}$ is not empty. There always exists a unique part twist $\widehat{\xi}$ for (5.1.6), (5.1.8), and (5.1.14)[23]. Therefore, $\xi^*(\phi)$ is a well-defined function and continuous.

Step 2: When $\dot{\theta} = 0$, the part has pure translation and does not rotate around any of the three support points. Therefore, the objective function is differentiable at $\dot{\theta} = 0$ and we can compute the following necessary KKT conditions.

$$\sum_{i=1}^{3} \alpha_i(\dot{x} - y_{s,i}\dot{\theta}) - \lambda \cos\phi = 0 \tag{5.1.18}$$

$$\sum_{i=1}^{3} \alpha_i(\dot{y} + x_{s,i}\dot{\theta}) - \lambda \sin\phi = 0 \tag{5.1.19}$$

$$\sum_{i=1}^{3} \alpha_i(-y_{s,i}(\dot{x} - y_{s,i}\dot{\theta}) + x_{s,i}(\dot{y} + x_{s,i}\dot{\theta})) - \lambda\rho \sin\phi = 0 \tag{5.1.20}$$

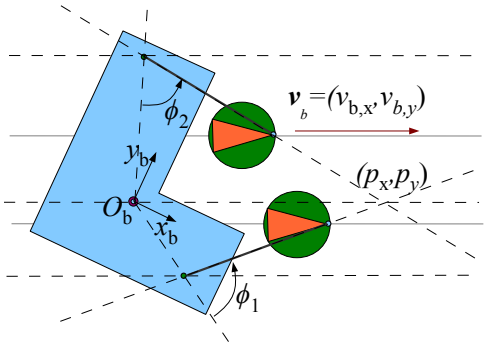


Figure 5.3: The equilibrium of two-robot towing. (This figure is taken from [24] and reproduced.)

in which

$$\alpha_i = \frac{\mu\lambda_{n,i}}{\sqrt{(\dot{x} - y_{s,i}\dot{\theta})^2 + (\dot{y} + x_{s,i}\dot{\theta})^2}}. \tag{5.1.21}$$

Solving these equations, we can see that $\dot{\theta} = 0$ if and only if $\phi = 0$, π, or $-\pi$

Step 3: It can also be checked that when $\phi > 0$ (for example, $\frac{\pi}{2}$) the resulting $\dot{\theta}$ is negative. Similarly, when $\phi < 0$ (for example, $-\frac{\pi}{2}$), the resulting $\dot{\theta}$ is positive. This complete the proof. □

Two-robot towing case

An equilibrium of two-robot towing is stated in the following theorem.

Theorem 5.1.2. When two robots tow the part by moving in the same direction and velocity (therefore maintaining fixed relative positions), the part can have zero angular velocity if the line passing through the center of mass and the intersection point of two cables is parallel to the robot moving direction as shown in Fig. 5.3.

Proof. The position vector $\rho_i = [\rho_i \cos(\alpha_i), \rho_i \sin(\alpha_i)]^T$ in the body fixed frame, in which ρ_i is the length of the vector ρ_i. Let ϕ_j to be the angle between the anchor point position vector and the cable direction. Then the direction of cable will be $u_j = [\cos(\alpha_j + \phi_j), \sin(\alpha_j + \phi_j)]^T$ in the body fixed frame.

Assuming both robots and the part have a pure translation with the velocity of $v_b = [v_{b,x}, v_{b,y}]^T$ in the body fixed frame, the wrench balance equation is

$$\begin{bmatrix} \cos(\alpha_1 + \phi_1) & \cos(\alpha_2 + \phi_2) \\ \sin(\alpha_1 + \phi_1) & \sin(\alpha_2 + \phi_2) \\ \rho_1 \sin(\phi_1) & \rho_2 \sin(\phi_2) \end{bmatrix} \begin{bmatrix} \lambda_{c,1} \\ \lambda_{c,2} \end{bmatrix} = \begin{bmatrix} \mu mg \frac{v_{b,x}}{\|v_b\|} \\ \mu mg \frac{v_{b,y}}{\|v_b\|} \\ 0 \end{bmatrix}, \tag{5.1.22}$$

which is true only when

$$\frac{\rho_2 \sin(\phi_2) \sin(\alpha_1 + \phi_1) - \rho_1 \sin(\phi_1) \sin(\alpha_2 + \phi_2)}{\rho_2 \sin(\phi_2) \cos(\alpha_1 + \phi_1) - \rho_1 \sin(\phi_1) \cos(\alpha_2 + \phi_2)} = \frac{v_{b,y}}{v_{b,x}}. \tag{5.1.23}$$

It can be easily checked that (5.1.23) is true when the line of action of the first cable, the line of action of

the second cable, and the line passing through the center of mass and parallel to the robot moving direction intersect at a single point. □

Exhaustive simulation and experimental results show that the two robot system converges to the equilibrium state shown in Figure 5.3, given by (5.1.22). However, we have not been able to prove this analytically. Thus the result of a stable equilibrium under two-robot towing remains a conjecture.

5.2 Kinematics and Statics of Cooperative Multi-Robot Aerial Manipulation with Cables

This section addresses the forward and inverse kinematics of payloads carried by aerial robots. We address the cases with one or two aerial robots and derive the kinematics and conditions for stable static equilibrium with non-negative cable tensions. We can establish the maximum number of equilibrium positions. We also present the conditions of aerial robot configurations to achieve unique stable configuration of payload. However, we can extend this result to the case of three aerial robots to find sufficient conditions.

5.2.1 Kinematics of Planar Manipulation Systems

Let n be the number of cables or aerial robots. For the planar case, we study the $n = 1$ and $n = 2$ cases. The $n \geq 3$ cases in the plane are special configurations where n cables with positive tensions completely constrain (or over constrain) the manipulated object.

Model: $\mathbf{n = 1}$

The planar case with a single cable or robot ($n = 1$) can be considered as a pendulum. An object is suspended by a massless cable with length l whose one end is fixed at the origin of the fixed frame and the other end is attach on the object. The configuration of the part is constrained by the length of cable, which is modeled as

$$(x - r\sin\theta)^2 + (y + r\cos\theta)^2 = l^2 \tag{5.2.1}$$

where $(x,\, y,\, \theta)$ is the position and orientation of the body frame in fixed frame and r is the distance from the center of mass to the attachment point.

Model: $\mathbf{n = 2}$

The planar case with two cables or robots ($n = 2$) is modeled as a four-bar-linkage (as shown in Figure 5.4), with the payload as the coupler. We assume the lengths of both cables are equal (l) and the center of mass is at the midpoint of the coupler (payload) whose length is R. The well-known constraint equations for a four-bar linkage are:

$$\begin{aligned} f(\theta,\, \phi) =& (l\cos\theta - l\cos\phi - a)^2 + (l\sin\theta - l\sin\phi - b)^2 - R^2 \\ =& 2l^2 + a^2 + b^2 - R^2 - 2l^2\cos\theta\cos\phi - 2l^2\sin\theta\sin\phi - 2al\cos\theta + 2al\cos\phi - 2bl\sin\theta + 2bl\sin\phi \\ =& 2l^2 + a^2 + b^2 - R^2 - 2l^2\cos(\theta - \phi) - 2al\cos\theta + 2al\cos\phi - 2bl\sin\theta + 2bl\sin\phi \\ =& 0 \end{aligned} \tag{5.2.2}$$

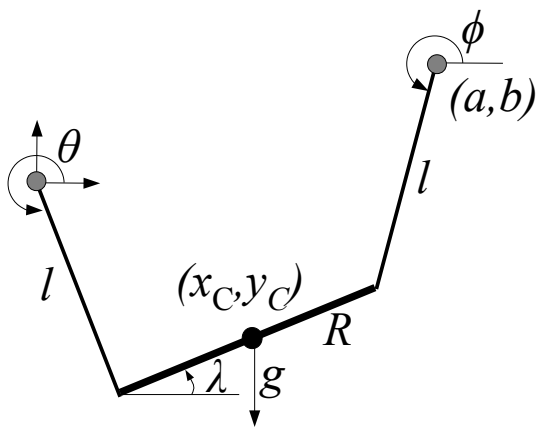


Figure 5.4: The planar system modeled as a four-bar-linkage. The suspended payload is the coupler with an assumed center of mass at the middle point of the coupler.

where a, b, θ, and ϕ are as shown in Figure 5.4.

Using the tangent half-angle formulation ($u = \tan\frac{\theta}{2}$, $v = \tan\frac{\phi}{2}$), the constraint equation may be reduced to a 4^{th} order polynomial of u and v,

$$\begin{aligned} f(u,\, v) =& \left((b^2+a^2-R^2)u^2 - 4blu + a^2 + b^2 - 4al + 4l^2 - R^2\right)v^2 + \left(4blu^2 - 8l^2u + 4bl\right)v \\ &+ (4l^2 + 4al - R^2 + a^2 + b^2)u^2 - 4blu + b^2 + a^2 - R^2 \\ =& 0. \end{aligned} \tag{5.2.3}$$

The center of mass, $(x_C,\, y_C)$, must trace a couple curve and is defined by the equation:

$$\begin{aligned} &4x^6 + 4y^6 + 12x^4y^2 + 12x^2y^4 - 12ax^5 - 12by^5 - 12bx^4y - 12axy^4 - 24ax^3y^2 - 24bx^2y^3 \\ &+ (13a^2 + 5b^2 + 8r^2 - 8l^2)x^4 + (8r^2 + 5a^2 + 13b^2 - 8l^2)y^4 + 16abx^3y \\ &+ (18b^2 + 16r^2 + 18a^2 - 16l^2)x^2y^2 + 16abxy^3 + (-16r^2a - 6a^3 + 16l^2a - 6ab^2)x^3 \\ &- (6a^2b + 6b^3 + 16br^2 - 16bl^2)y^3 - (6a^2b - 16bl^2 + 6b^3 + 16br^2)x^2y \\ &+ (-16r^2a - 6a^3 + 16l^2a - 6ab^2)xy^2 \\ &+ (2b^2r^2 + a^4 + b^4 + 4l^4 + 2a^2b^2 + 10a^2r^2 - 10a^2l^2 - 6b^2l^2 + 4r^4 - 8r^2l^2)x^2 \\ &+ (+b^4 - 10b^2l^2 + 4r^4 + 2a^2b^2 - 8r^2l^2 + a^4 + 4l^4 - 6a^2l^2 + 2a^2r^2 + 10b^2r^2)y^2 \\ &+ (-2b^2r^2a + 2b^2l^2a - 2a^3r^2 + 8r^2l^2a - 4r^4a - 4l^4a + 2a^3l^2)x \\ &+ (16abr^2 - 8abl^2)xy + (-2a^2br^2 - 4r^4b + 2a^2bl^2 + 8r^2bl^2 + 2b^3l^2 - 2b^3r^2 - 4l^4b)y \\ &+ b^2r^4 - 2b^2r^2l^2 + r^4a^2 + l^4a^2 - 2r^2a^2l^2 + b^2l^4 = 0. \end{aligned} \tag{5.2.4}$$

where a and b are shown in Figure 5.4 and we set $r = \frac{R}{2}$, $x = x_C$, and $y = y_C$ for simplification. It is well-

known that this curve is a tri-circular sextic with triple points at infinity, $x = \pm iy,\ w = 0$ with coordinates $(x,\ y,\ w)$ in projective space [51].

5.2.2 Direct Problem

In this section, we address the direct problem to find the configuration of payload from given configurations of the aerial robots. Considering the aerial robots as a fixed anchor points and the payload is hanging by cables connecting these point and anchor points on the payload. Then the sum of all the wrenches on the payload should be zero. And this condition is the First Order Necessary Condition of minimum gravity potential energy problem. In this section, let q_i be the position of the aerial robot and p_i be the position of the anchor point on the payload. And the center of mass of the payload is assumed to be the geometric center of p_is.

5.2.3 Direct problem: $\mathbf{n = 2}$

The configuration of the payload is determined by the positions of anchor points, p_1 and p_2, which minimize the gravity potential energy. To numerical solve the direct problem of a planar case ,$n = 2$, we formulate the optimization problem:

$$\begin{aligned} \min_{p_i} \quad & p_{1,y} + p_{2,y} \\ \text{st} \quad & ||p_1 - q_1||_2^2 - l^2 \leq 0 \\ & ||p_2 - q_2||_2^2 - l^2 \leq 0 \\ & ||p_1 - p_2||_2^2 - R^2 = 0 \end{aligned} \tag{5.2.5}$$

where we assume that y is the vertical axis and the length of the two cables are the same as l. And the length of the payload or the distance between the anchor points on the payload is R. However the last constraints in (5.2.5) is the quadratic equality constraint, which is non-convex. So, this optimization problem does not guarantee the unique solution. But we need to plan the trajectories of aerial robots, which guarantees the unique configuration, to estimate the configuration of payload for its position control. So we will find the conditions of the position of the aerial robots to achieve unique solution of the direct problem (5.2.5).

Proposition 5.2.1 (Conditions for Unique Solutions to the Direct Problem of $n = 2$)**.** *The solution to the* (5.2.5) *is unique provided that if the distance between two robots are longer than the distance between anchor point on the payload:*

$$||q_1 - q_2||_2 \geq R. \tag{5.2.6}$$

Proof. By relaxing the equality constraints in (5.2.5), we see that the program becomes convex.

$$\begin{aligned} \min_{p_i} \quad & p_{1,y} + p_{2,y} \\ \text{st} \quad & ||p_1 - q_1||_2^2 - l^2 \leq 0 \\ & ||p_2 - q_2||_2^2 - l^2 \leq 0 \\ & ||p_1 - p_2||_2^2 - R^2 \leq 0. \end{aligned} \tag{5.2.7}$$

We begin by considering the SOCP in (5.2.7). The First Order Necessary Conditions (FONC) are:

$$\begin{aligned} (\mu_1 + \mu_3)p_{1,x} - \mu_3 p_{2,x} - \mu_1 q_{1,x} &= 0 \\ \frac{1}{2} + (\mu_1 + \mu_3)p_{1,y} - \mu_3 p_{2,y} - \mu_1 q_{1,y} &= 0 \\ -\mu_3 p_{1,x} + (\mu_2 + \mu_3)p_{2,x} - \mu_2 q_{2,x} &= 0 \\ \frac{1}{2} - \mu_3 p_{1,y} + (\mu_2 + \mu_3)p_{2,y} - \mu_2 q_{2,y} &= 0 \end{aligned} \tag{5.2.8}$$

with

$$\begin{aligned} ||p_1 - q_1||_2^2 - l^2 \leq 0 \qquad & \mu_1 \geq 0 \qquad & \mu_1(||p_1 - q_1||_2^2 - l^2) = 0 \\ ||p_2 - q_2||_2^2 - l^2 \leq 0 \qquad & \mu_2 \geq 0 \qquad & \mu_2(||p_2 - q_2||_2^2 - l^2) = 0 \\ ||p_1 - p_2||_2^2 - R^2 \leq 0 \qquad & \mu_3 \geq 0 \qquad & \mu_1(||p_1 - p_2||_2^2 - R^2) = 0. \end{aligned}$$

Here, assume that $\mu_3 = 0$, which leads to $||p_1 - p_2||_2^2 - R^2 < 0$. Then the FONC (5.2.8) will be

$$\begin{aligned} \mu_1(p_{1,x} - q_{1,x}) &= 0 \\ \frac{1}{2} + \mu_1(p_{1,y} - q_{1,y}) &= 0 \\ \mu_2(p_{2,x} - q_{2,x}) &= 0 \\ \frac{1}{2} + \mu_2(p_{2,y} - q_{2,y}) &= 0. \end{aligned} \tag{5.2.9}$$

So the 2^{nd} and 4^{th} equations lead to $mu_1 \neq 0$ and $\mu_2 \neq 0$, respectively. Then, 1^{st} and 3^{rd} equations lead to $p_{1,x} = q_{1,x}$ and $p_{2,x} - q_{2,x}$, respectively. Then the optimal configuration will be

$$p_i^* = q_i + \begin{pmatrix} 0 \\ -l \end{pmatrix} \quad \text{for} \quad i = \{1, 2\} \tag{5.2.10}$$

whose optimal value is $\frac{q_{1,y}+q_{2,y}}{2} - l$. However, as $\mu_3 = 0$ this solution should satisfy the inequality of $||p_1 - p_2||_2^2 - R^2 < 0$:

$$||p_1^* - p_2^*||_2^2 - R^2 = ||q_1 - q_2||_2^2 - R^2 < 0 \tag{5.2.11}$$

which is contradiction with the assumption of (5.2.6). So the suggested condition (5.2.6) leads to the optimal solution with $\mu_3 \neq 0$, which means the optimal solution of (5.2.7), which is a convex problem, should

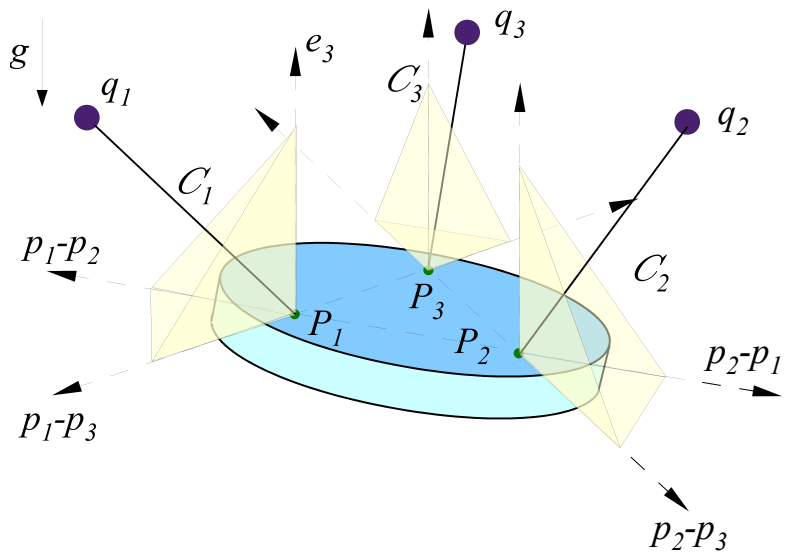


Figure 5.5: A graphical depiction of the conditions presented in Proposition 5.2.2. (This figure is taken from [40].)

satisfies $f_1(p_i) = ||p_1 - p_2||_2^2 - R^2 = 0$. Then the Direct Problem is equivalent to this relaxed problem and will have a unique solution. ■

So, under the condition of (5.2.6), we can achieve the unique solution of the direct problem in case of $n = 2$.

5.2.4 Direct problem: $\mathbf{n = 3}$

In the case of $n = 3$, we assume that the anchor points on the payload forms an equilateral triangle of edge length, R, and the center of mass is the geometric center of this equilateral triangle. Then the direct problem will be formulated as an optimization problem of

$$\begin{aligned} \min_{p_i} \quad & p_{1,z} + p_{2,z} + p_{3,z} \\ \text{st} \quad & ||p_1 - q_1||_2^2 - l^2 \leq 0 \\ & ||p_2 - q_2||_2^2 - l^2 \leq 0 \\ & ||p_3 - q_3||_2^2 - l^2 \leq 0 \\ & ||p_1 - p_2||_2^2 - R^2 = 0 \\ & ||p_2 - p_3||_2^2 - R^2 = 0 \\ & ||p_3 - p_1||_2^2 - R^2 = 0 \end{aligned} \tag{5.2.12}$$

where we assume that z is the vertical axis and the length of the three cables are the same as l. However, it is not trivial to extend the condition of (5.2.6) to three dimensional case of (5.2.12). So, we will propose a sufficient condition to achieve the unique solution of (5.2.12).

Proposition 5.2.2 (Conditions for Unique Solutions to the Direct Problem of $n = 3$)**.** *The solution to the* (5.2.12) *is unique when the i^{th} aerial robot, q_i, lies strictly inside the convex cone originated at the corresponding anchor point on the payload, p_i and formed by three vectors of $\{(p_i - p_j), (p_i - p_k), e_3\}$ for $j \neq i$, $k \neq i$, $k \neq j$ and $e = [0, 0, 1]^T$ as shown in Figure 5.5, which is called cone constraints. And we assume that the configuration of the payload satisfies that these three vectors are linearly independent.*

Proof. By relaxing the equality constraints in (5.2.12), we see that the program becomes convex.

$$\begin{aligned} \min_{p_i} \quad & p_{1,z} + p_{2,z} + p_{3,z} \\ \text{st} \quad & ||p_1 - q_1||_2^2 - l^2 \leq 0 \\ & ||p_2 - q_2||_2^2 - l^2 \leq 0 \\ & ||p_3 - q_3||_2^2 - l^2 \leq 0 \\ & ||p_1 - p_2||_2^2 - R^2 \leq 0 \\ & ||p_2 - p_3||_2^2 - R^2 \leq 0 \\ & ||p_3 - p_1||_2^2 - R^2 \leq 0. \end{aligned} \tag{5.2.13}$$

Then the FONC of (5.2.13) is

$$\begin{aligned} \mu_1(p_1 - q_1) + \mu_4(p_1 - p_2) + \mu_6(p_1 - p_3) + \frac{1}{2}e_3 = 0 \\ \mu_2(p_2 - q_2) + \mu_4(p_2 - p_1) + \mu_5(p_2 - p_3) + \frac{1}{2}e_3 = 0 \\ \mu_3(p_3 - q_3) + \mu_5(p_3 - p_2) + \mu_6(p_3 - p_1) + \frac{1}{2}e_3 = 0 \\ ||p_1 - q_1||_2^2 - l^2 \leq 0 \quad \perp \quad \mu_1 \geq 0 \\ ||p_2 - q_2||_2^2 - l^2 \leq 0 \quad \perp \quad \mu_2 \geq 0 \\ ||p_3 - q_3||_2^2 - l^2 \leq 0 \quad \perp \quad \mu_3 \geq 0 \\ ||p_1 - p_2||_2^2 - R^2 \leq 0 \quad \perp \quad \mu_4 \geq 0 \\ ||p_2 - p_3||_2^2 - R^2 \leq 0 \quad \perp \quad \mu_5 \geq 0 \\ ||p_3 - p_1||_2^2 - R^2 \leq 0 \quad \perp \quad \mu_6 \geq 0. \end{aligned} \tag{5.2.14}$$

Assume that if $\mu_1 = 0$, then the first condition in (5.2.14) will be

$$\mu_4(p_1 - p_2) + \mu_6(p_1 - p_3) + \frac{1}{2}e_3 = 0 \tag{5.2.15}$$

which violates the assumption that the three vectors forming the convex cone for $i = 1$ are linearly independent. And the same arguments can be applied for μ_2 and μ_3. So, we have $\mu_i > 0$ for all $i = \{1, 2, 3\}$. Then

the first three equations of (5.2.14) can be presented as

$$\begin{aligned} q_1 - p_1 =& \frac{\mu_4}{\mu_1}(p_1 - p_2) + \frac{\mu_6}{\mu_1}(p_1 - p_3) + \frac{1}{2\mu_1}e_3 = 0 \\ q_2 - p_2 =& \frac{\mu_4}{\mu_2}(p_2 - p_1) + \frac{\mu_5}{\mu_2}(p_2 - p_3) + \frac{1}{2\mu_2}e_3 = 0 \\ q_3 - p_3 =& \frac{\mu_5}{\mu_3}(p_3 - p_2) + \frac{\mu_6}{\mu_2}(p_3 - p_1) + \frac{1}{2\mu_3}e_3 = 0. \end{aligned} \tag{5.2.16}$$

From our assumption that the position vector $q_i - p_i$ should lie strictly inside the convex cone of $\{(p_i - p_j), (p_i - p_k), e_3\}$, the coefficients of the above equations in (5.2.16) should be all strictly positive, $\frac{\mu_j}{\mu_i} > 0$ for all $i = \{1, 2, 3\}$ and $j = \{4, 5, 6\}$. Then it is obvious that $\mu_j > 0$ for all $j = \{4, 5, 6\}$, which leads to

$$\begin{aligned} ||p_1 - p_2||_2^2 - R^2 =& 0 \\ ||p_2 - p_3||_2^2 - R^2 =& 0 \\ ||p_3 - p_1||_2^2 - R^2 =& 0, \end{aligned}$$

which means the optimal solution of (5.2.13), which is a convex problem, should satisfies $||p_j - p_k||_2^2 - R^2 = 0$ for all $j = \{1, 2, 3\}$, $k = \{1, 2, 3\}$ and $j \neq k$. Then the Direct Problem is equivalent to this relaxed problem and will have a unique solution. ■

However, the suggested condition works only when we know the desired configuration of the payload, p_is. But it is enough for path planning for aerial robots [40]. To find a condition only depending on q_is are remained as one of the future works.

5.2.5 Stability

In Sections 5.2.1, we developed the kinematic formulation of the $n = \{1, 2\}$ cases. We now consider the stability of these systems. We begin by presenting the trivial case of $n = 1$ in order to provide an intuition to the approach we use in the analysis of the $n = 2$ system.

To study the stability of the system, we are interested in considering the potential energy of the payload assuming the fixed position(s) of the robot(s). We also assume in this analysis that all cables are in tension.

Analysis: $\mathbf{n = 1}$

From Section 5.2.1, we see that the wrench balance equation is

$$\frac{\lambda}{l}\begin{bmatrix} x - r\sin\theta \\ y + r\cos\theta \\ 0 \end{bmatrix} = mg\begin{bmatrix} 0 \\ 1 \\ x \end{bmatrix}. \tag{5.2.17}$$

Therefore, the condition for static equilibrium is met when $x = 0$ and $\sin\theta = 0$, or $(x, y) =$

$(0,\ -l-r)$, $(0,-l+r)$, $(0,l-r)$, $(0,l+r)$. Computing the Hessian of the potential energy, $V = mgy$,

$$\mathcal{H}(x,\theta) = mg \begin{bmatrix} \frac{\partial^2 y}{\partial x^2} & \frac{\partial^2 y}{\partial x \partial \theta} \\ \frac{\partial^2 y}{\partial x \partial \theta} & \frac{\partial^2 y}{\partial \theta^2} \end{bmatrix} = \frac{mg}{l+2r} \begin{bmatrix} 1 & r \\ r & r(l+r) \end{bmatrix}. \tag{5.2.18}$$

A positive-definite $\mathcal{H}$ indicates the stability of a configuration, leading us to conclude that $(x,\ y) = (0,\ -l-r)$ is the only stable configuration.

While the $n = 1$ analysis is trivial, we present it to elucidate the approach to studying the stability of the next system.

Analysis: $\mathbf{n = 2}$

We begin by considering the equilibrium configurations, which correspond to the geometric point of intersection between the line of gravity and two cable forces. This condition is expressed as

$$\det \begin{bmatrix} 1 & \tan\theta & 0 \\ 1 & \tan\phi & b - a\tan\phi \\ 0 & 1 & -x_c \end{bmatrix} = 0. \tag{5.2.19}$$

The potential energy of the system is

$$V(\theta) = mgy_c = \frac{mg}{2}\left(l\sin\theta + l\sin\phi + b\right), \tag{5.2.20}$$

where ϕ is related to θ through (5.2.2). Clearly, the equilibrium points correspond to configurations in which V is stationary. Considering the first and second derivatives,

$$\frac{dV}{d\theta} = \frac{mgl}{2}\left(\cos\theta + \cos\phi\frac{d\phi}{d\theta}\right) \tag{5.2.21}$$

$$\frac{d^2V}{d\theta^2} = \frac{mgl}{2}\left(-\sin\theta - \sin\phi\left(\frac{d\phi}{d\theta}\right)^2 + \cos\phi\frac{d^2\phi}{d\theta^2}\right), \tag{5.2.22}$$

we conclude that we must find points along the coupler curve which correspond to $\frac{dV}{d\theta} = 0$ and $\frac{d^2V}{d\theta^2} > 0$.

The lines of tangency to the coupler curve may be thought of as a tangent line from infinity whose class determines the number of tangency points [36]. The class of the coupler curve (5.2.4) is twelve. A review of various coupler curves for a four-bar-linkage is provided in [51]. Figure 5.6 depicts an example of a linkage for which we can easily visualize the twelve horizontal tangents to the coupler curve and the corresponding equilibrium points. As the coupler curve is a closed curve (with one or two branches), the number of local maxima and minima are the same, and therefore six unstable and six stable equilibrium points (ignoring tension constraints, $\lambda_i > 0$). Figure 5.7 depicts these stable configurations.

5.3 Conclusion

In this chapter, we discussed the classical method of suing cables to tow payload with multiple ground or aerial robots. First, we studied the mechanics of planar, multi-robot towing a planar payload subject to fric-

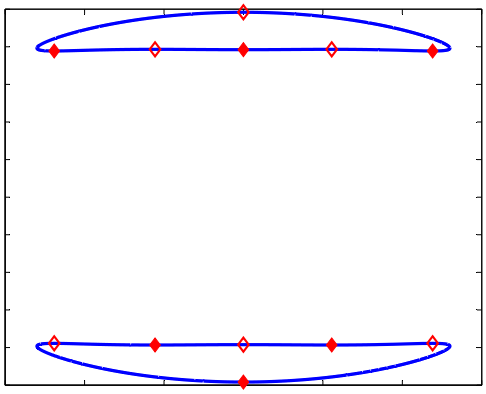


Figure 5.6: A coupler curve with twelve equilibrium configurations. The stable and unstable configurations are denoted by filled or open red diamonds. The stable configurations are shown in Figure 5.7. Note that tension constraints are ignored.

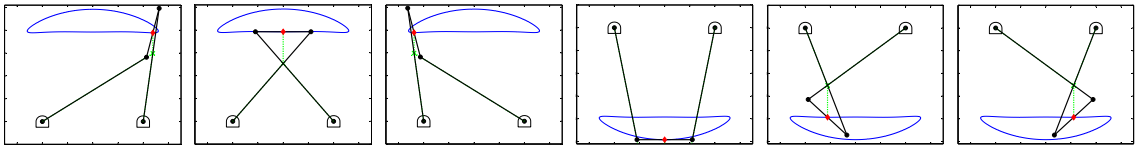


Figure 5.7: The six equilibrium configurations of Figure 5.6. Clearly Figures. 5.7(a)-5.7(c) are infeasible when considering tension constraints.

tion. The problem formulation incorporates complementarity constraints which are necessary to allow for cables becoming slack during a towing maneuver. We showed that a payload towed by one robot driving along a straight line, or two robots driving along parallel straight lines, converges to an equilibrium configuration independent of the uncertainty in the support force distribution. This result suggest a robust primitive motion of planar towing problem. However, finding another robust primitive motion, rotating about a point with finite radius, is a remaining problem.

We also address the kinematics of payloads carried by aerial robots. We address the cases with one and two aerial robots and derive the kinematics and conditions for stable static equilibrium with non-negative cable tensions. We can establish the maximum number of equilibrium positions. We derived the conditions of aerial robot configurations to achieve a unique solution of direct problem with two or three aerial robots.

Chapter 6

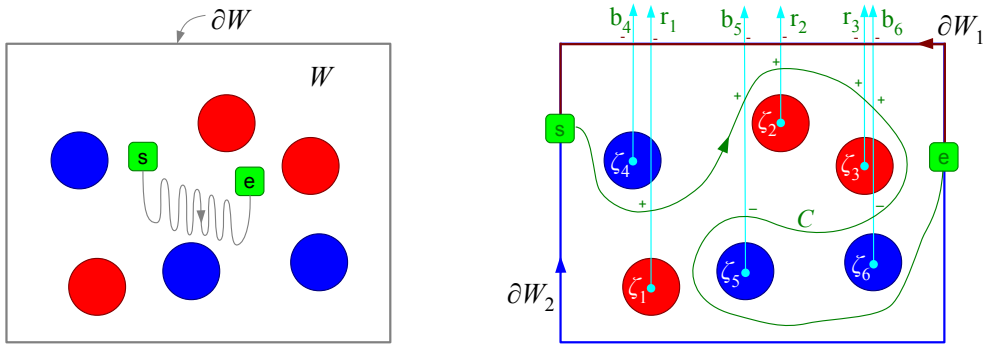
Manipulation of A Set Of Objects

In this chapter, we study the problem of manipulating and transporting multiple objects on the plane using a cable attached at each end to a mobile robot. This problem is motivated by the use of boats with booms in skimming operations for cleaning oil spills or removing debris on the surface of the water. Because the cable is flexible, the shape of the cable must be explicitly modeled in the problem. Further, the robots must cooperatively plan motions to achieve the required cable shape and gross position/orientation to separate the objects of interest and then transport them as specified. We first derive the necessary topological conditions for achieving the desired separation of objects. We then propose a distributed search-based planning technique for finding optimal robot paths for separation and transportation. We demonstrate the applicability of this method using a dynamic simulation platform with explicit models of the cable dynamics, the contact between the cable and one or more objects, and the surface drag on the cable and on the objects. The work in Section 6.1-6.5 was performed in close collaboration with Dr. Subhrajit Bhattacharya, Hordur Heidarsson and Prof. Gaurav Sukhatme, much of which was reported in [56, 57].

6.1 Introduction

Object manipulation is an important problem in robotics. Certainly conventional approaches to manipulation using robot arms with grippers has received considerable attention and is well understood [104, 30]. In contrast, we are interested in the use of mobile robots to contact and manipulate objects without special purpose effectors. This allows more versatility but leads to many challenges. One approach relies on caging an object using multiple mobile robots. This problem has been studied for planar objects [39]. However, the ratio between the number of objects manipulated at a time, and the number of robots required for doing that is small, thus making such an approach highly inefficient for manipulating a large number of objects and for separating objects in a field with obstacles. In contrast, we propose a framework for manipulating a large number of objects with only a pair of robots.

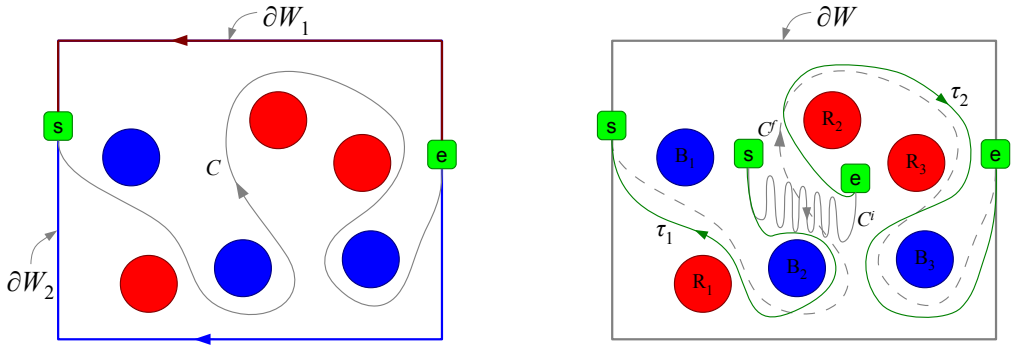
The advantages of using ropes with robots for manipulation were demonstrated by Donald *et al* [34]. An interesting problem that arises in these settings is the modeling of the shape of the cable and the motion planning for the robots to control the position and shape of the cable. Motion planning for manipulation of rope-like flexible objects is discussed in [82]. The problem of entangling and disentangling knots and the motion planning for this problem has been addressed in [60]. Our goal, however, is the motion planning that



(a) The initial configuration of the cable and the two robots in the workspace W with boundary ∂W. Red and blue circles are objects to be separated. Two green boxes are the robots. Grey curve is the cable.

(b) ζ_i are representative points inside the objects, $R_1, R_2, R_3, B_4, B_5, B_6$ (in that order), and r_i, $i = 1, 2, 3$ and b_j, $j = 4, 5, 6$ are rays emanating from the respective points. Using the bump forms, (2.2.3), corresponding to the rays in defining the H-signature, $H(C) = [1, 1, 1, 0, 0, 0]$. And, $h(C) =$ "$r_1^+ b_5^+ r_2^+ r_3^+ b_5^-$".

Figure 6.1: The problem of separating the two types of objects.



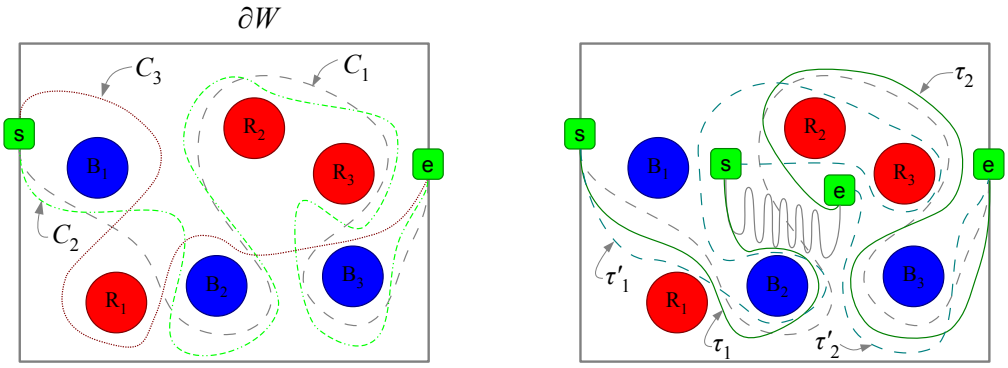
(a) A separating configuration of the cable, C, that separates the two types of objects.

(b) A possible set of paths that take the cable from the initial configuration, C^i, to a configuration homotopic to the separating configuration, C^f.

Figure 6.2: An example of separating configuration and a set of paths to the separating configuration.

is required to manipulate objects on the plane and we are less interested in the specific configuration of the cable. The use of robots to tow objects using cables is discussed in [53, 23] and in the previous Chapter 5. An extension of these ideas leads to using a cable with its ends tied to robots to cage and tow objects. Indeed this method is widely used in skimming operations on water surfaces [81, 54]. A description of the dynamics of such systems and an analysis of the problem of cooperative skimming are provided in [12, 4]. However, this work does not explicitly address the manipulation of objects.

In this chapter, we discuss the planning and control of the motions of two robots, each of which is tied to one end of a flexible cable, with the goals of (a) separating a specified set of objects from other objects; and (b) to transport the specified objects to a destination. The first step, as one might expect, is to navigate the robots around the objects so that the cable separates the objects of interest from the ones that are not of interest. The problem of finding a hypersurface separating two types of objects is studied as part of



(a) Three possible cable configurations separating the two types of objects. C_1 and C_2 are homotopic. But C_3 belongs to a different homotopy class. See the curve C' in Figure 6.1(b) for yet another cable configuration that separates the two types of objects.

(b) The robot paths (up to homotopy) that can take the cable to a desired separating configuration (up to homotopy) are not unique. In this figure, τ_1 and τ'_1 are not homotopic, neither are τ_2 and τ'_2. But either of the sets of paths, $\{\tau_1, \tau_2\}$ or $\{\tau'_1, \tau'_2\}$, take the cable to the homotopy class shown in Figure 6.2(a).

Figure 6.3: The solutions of object separating problem is not unique.

statistical classification problems [18, 93]. However such methods are susceptible to finding curves that can have disjoint components, do not have guarantees on optimality, and are statistical in nature. Moreover, the problem of finding a *separating* cable configuration (the curve) that separates the objects does not give us a necessary means of finding the paths of the robots that achieve that configuration. The first key contribution of this chapter is a topological description of the problem of separating two sets of objects and the algebraic formulation of the separation problem. The second contribution is a complete motion planning algorithm that relies on graph search [25] to drive the robots in order to achieve separation and then transport the objects to specified destinations. We also derive a decoupled algorithm that has the advantage of only requiring to plan in the individual robot's configuration space instead of the joint state-space.

6.2 Problem Description

We consider the scenario where there are two classes of objects present in a flat enclosed region, W. For convenience we will refer to the two classes as 'blue' and 'red'. Without loss of generality, one of these classes of objects will be considered to be of interest (*i.e.*, those need to be manipulated and transported), while the other consists of obstacles or objects that are not of interest. Let $\mathcal{O} = R_1 \cup R_2 \cup \cdots \cup R_r \cup B_{r+1} \cup B_{r+2} \cup \cdots \cup B_{r+b} \subseteq W$, where $R_1, R_2, \cdots, R_r$ are r counts of red objects, and $B_{r+1}, B_{r+2}, \cdots, B_{r+b}$ are b counts of blue objects. Each object, R_i or B_j, is assumed to be connected and arbitrarily shaped.

A flexible cable is attached, at its two ends, to two robots that are capable of navigating on the flat surface. Given an initial configuration of the cable and the robots (Figure 6.1(a)), we need to first make the robots follow paths to the boundary of the enclosed region, ∂W, such that the final cable configuration 'separates' the blue objects from the red, which we call the *separating configuration* (Figure 6.2(b)). Once that is achieved, the robots can move along ∂W to enclose one type of objects and "pull" them out, thus separating and transporting those objects.

Suppose $\mathbf{e}$ and $\mathbf{s}$ are the points on the boundary reached by the robots so that they split ∂W into ∂W_1

and ∂W_2 as in Figure 6.2(a). It is clear that the robot paths and cable configurations that describe the problem and achieve the desired objective are sufficiently described up to homotopy. That is, if C_1 and C_2 are two cable configurations that are in the same homotopy class [15], then, *"C_1 separates the two types of objects"* $\iff$ *"C_2 separates the two types of objects"* (Figure 6.3(a)). Likewise, if a particular set of robot paths, $\{\tau_1, \tau_2\}$, carry the cable from the initial configuration to the desired separating configuration (up to homotopy), another set of paths, $\{\tau_1', \tau_2'\}$, that are homotopic to the first set (*i.e.* $\tau_1' \sim \tau_1$ and $\tau_2' \sim \tau_2$) will achieve the same objective.

In addition to this, it should also be noted that the homotopy class of the cable configuration that achieves the separation of the two types of objects is not unique either. For example, in Figure 6.3(a), the configuration C_3 is in a different homotopy class from C_1 or C_2, but still separates the two types of objects. C' in Figure 6.1(b) is another example. Furthermore, for a given desired separating configuration of the cable (up to homotopy), the homotopy classes of the robot paths that can carry the cable from its initial configuration to the separating configuration, are not unique either (Figure 6.3(b)).

Thus, it is useful to develop a notion of optimality to more precisely define the problem objectives. It is natural to use length of the robot paths to the optimization criteria.

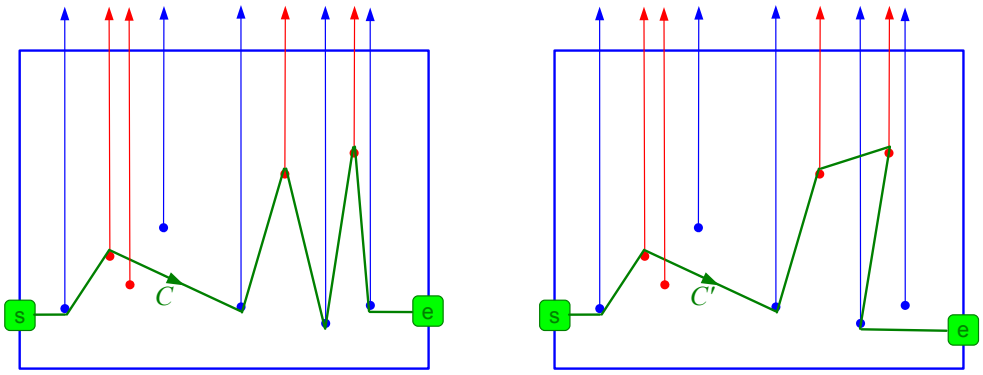
For the theoretical foundation and for setting up the optimization problem, we will make the following assumptions:

i. The objects are assumed to be stationary rigid bodies – that is, the cable cannot 'pass through' any of the objects, and that on contact of the cable with the objects the objects do not move. In the implementation (Section 6.5.2) we will however relax the conditions that the objects need to be stationary.

ii. The cable is flexible, and there is no restriction on the length of the cable (*i.e.* the cable will not fall short and tug on the robots). We assume that the cable can either be spooled out as required from a cable reel residing on the robots, or may stretch as in an elastic band.

One simple and intuitive strategy to solve this problem is to drive all the robot on one (left) boundary of the workspace and fix one robot. Then the other robot travels from this (left) boundary of the workspace to the opposite (right) boundary while passing one (red) object above it and the other (blue) object under it as shown in Figure 6.4(a). It is a simple and intuitive algorithm to find and drive to a separating configuration. However, this algorithm requires that all the obejcts should have different value of X coordinates. And this algorithm does not guarantee the shortest traveling distance. In the same environment, the cable configuration or the path of a robot shown in Figure 6.4(b) is shorter than the one in Figure 6.4(a). The path can be shorter if we allow the robot to go *backward* to minimize the overall traveling distance. Also, we should allow the robot to go backward if the objects are not size less points like this case. For example, we cannot find a path with this strategy in the environment of Figure 6.1(b). Also, this algorithm does not guarantee the shortest path of robot from the initial configuration and we need another procedure to drive the robot to a specific part of the boundary of workspace.

In this work, we divide and solve this manipulation and transportation problem into two steps. The first step is to find a separating configurations, which works as initial configuration of the simple controller for transportation, which will be discussed in Section 6.3. And the second problem is to navigate the robots to a separating configurations.



(a) A separating configuration achieved by intuitive planning. (b) A separating configuration with less travel distance.

Figure 6.4: An example of separating configurations achieve by intuition when considering point objects.

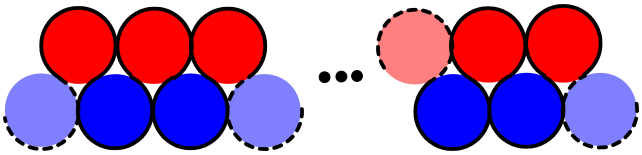


Figure 6.5: An example of separating configurations which requires smart controller for transporting.

6.3 Separating Configurations

As the separating configuration is the initial configuration of the planner or controller of transporting, we assume a simple planner/controller for transporting and there should be a feasible path or control input from this separating configuration to the final goal. Consider the example in Figure 6.5, which shows a separating configuration that satisfies the following Proposition 6.3.1. Considering that this configuration is infinitely long, the system could be stuck by pulling the cable side to side. And such configuration to cause balance between internal forces is also possible for noncircular objects, too. However, it depends on the boundary condition of this contexture configuration. If the objects at the end of the structure is free, we can decompose this contexture configuration from its ends, by driving robots or the ends of the cable up and down. If we pin nails on the proper points on the boundary of the objects on the boundary to replace the constraints forces by imaginary objects next to it, we can make this contexture configuration as a right body. (One obvious case is to pin two nails on the tangent points for the next two imaginary objects to it.) Or we can achieve such force balances by a wall tangent to the boundary object and pulling the cable with proper direction. However, we assume free workspace except the objects to manipulate, and we will not discuss about this interesting case further in this work.

As the *feasible* separating configuration should be a separating configuration that the given or designed controller for transporting can drive the cable-robot system to the goal. So, in this section, we propose a condition of separating configuration that does not represent all possible separating configurations but a set of separating configurations which can be a proper initial configuration for simple transporting controllers.

Proposition 6.3.1. *Suppose C is an embedded cable configuration such that $C(0), C(1) \in \partial W$ (*i.e. *the*

*cable ends lie on the boundary of the environment). Say the end points of C splits ∂W into two parts: ∂W_1 and ∂W_2 (which themselves are curves in $(W - \mathcal{O})$). We assign orientation to ∂W_1 and ∂W_2 such that $C \sqcup \partial W_1$ and $C \sqcup \partial W_2$ are closed loops (Figure 6.2(a)). Then, C separates the two types of objects (*i.e.*, it is a* separating configuration*) iff one of the following holds for the vector $H(C \sqcup \partial W_1)$:*

i. The first r components are all 1 or all -1, and the last b components are all 0.

ii. The last b components are all 1 or all -1, and the first r components are all 0.

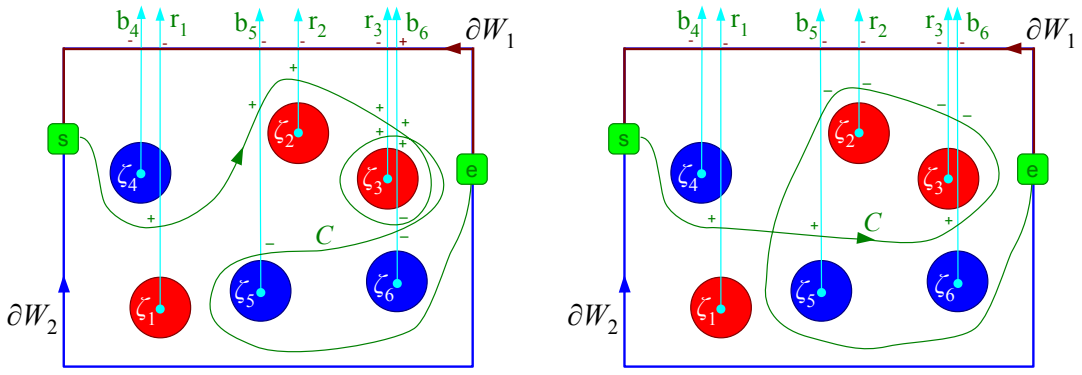
Note that from the definition of H-signature, $H(C \sqcup \partial W_1) = H(C) + H(\partial W_1)$. Also, in these conditions the choice of ∂W_1 over ∂W_2 is made without loss of generality. The conditions could have been stated in terms of ∂W_2 as well.

Sketch of Proof: The proof follows from the very definition of homology (see Figure 6.3(a)). First we note that $C \sqcup \partial W_1$ is a Jordan curve [43] inside W (since C is embedded). Hence there is a simply-connected region in W (not considering the objects) enclosed by $C \sqcup \partial W_1$. The objects (and their representative points) that this region will contain will manifest as a ± 1 in the corresponding components of the vector $H(C \sqcup \partial W_1)$. Since $C \sqcup \partial W_1$ is Jordan, it will wind around each of the enclosed points in the same direction (all clockwise or all anti-clockwise), thus making the corresponding components of the vectors either all $+1$ or all -1. All the other components will be 0. The statement of the lemma simply states that the enclosed representative points will be ones corresponding to the red objects or the blue objects, while the ones not enclosed will be ones corresponding to objects of the other color. ■

At this point it is instructive to illustrate why, in the above Proposition, we used the homology invariant instead of homotopy invariant. Consider the curve C in Figure 6.1(b), which clearly separates the red objects from blue. However we previously saw that the *reduced word* for $(C \sqcup \partial W_1)$ is, $h(C) \diamond h(\partial W_1) =$ "$r_1^+ b_5^+ r_2^+ r_3^+ b_5^-\ b_6^- r_3^- r_2^- b_5^- r_1^- b_4^-$". Likewise the *reduced word* $h(C) \diamond h(\partial W_2) =$ "$r_1^+ b_5^+ r_2^+ r_3^+ b_5^-$". Neither of these words are helpful in identifying the fact that C' separates the blue objects from the red. However, $H(C) + H(\partial W_1) = [0, 0, 0, -1, -1, -1]^T$, and $H(C) + H(\partial W_2) = [1, 1, 1, 0, 0, 0]^T$ – both satisfying the condition of Proposition 6.3.1 (note that the first 3 components of the vector correspond to R_1, R_2 & R_3, while the last 3 correspond to B_4, B_5 & B_6), thus indicating that C indeed separates the blue from the red objects.

It is obvious that there could be separating configurations that do not satisfying Proposition 6.3.1 like examples in Figure 6.6. The configuration in Figure 6.6(a) winds one red object twice and we can separate the red objects by pulling the two robots or the cable downward or $-y$ direction. But we cannot separate the blue objects by pulling this cable upward or $+y$ direction.Also, the configuration in Figure 6.6(b) winds the red objects in different directions and we can separate only red objects by pulling cable downward. In both examples, we cannot separate the blue objects and need careful planning and control to release the red object after transporting to the destination. So, we do not consider such configurations as our *separating configuration* and it is obvious by Proposition 6.3.1.

Proposition 6.3.2. *(Refer to Figure 6.2(b)) Let C be a starting cable configuration (which has an orientation from robot '2' to robot 1', as shown in Figure 6.1(a)) and C' be a final cable configuration (which may or may not be a separating configuration). Then the paths τ_1 and τ_2 for the two robots carry the cable from initial configuration to the separating configuration (up to homotopy) if and only if the closed loop*



(a) A separating configuration which winds the same object twice. $H(C) = [1, 1, 2, 0, 0, 0]^T$.

(b) A separating configuration which winds red objects in different direction. $H(C) = [1, -1, -1, 0, 0, 0]$.

Figure 6.6: Examples of separating configurations which do not satisfy Proposition 6.3.1.

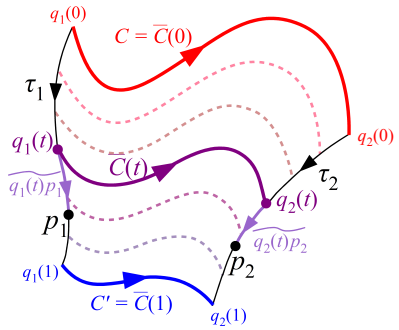
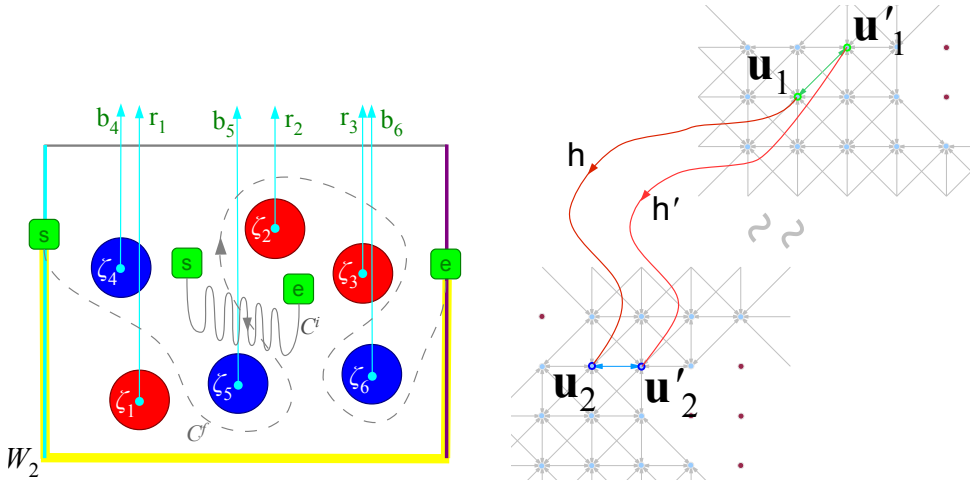


Figure 6.7: Illustration for Proof of Proposition 6.3.2.

$(C \sqcup \tau_2 \sqcup -C' \sqcup -\tau_1)$ *is* null homotopic *[49],* i.e. $h(C \sqcup \tau_2 \sqcup -C' \sqcup -\tau_1) = h(C) \diamond h(\tau_2) \diamond h(C')^{-1} \diamond h(\tau_1)^{-1} =$ *" ", is the empty word (*identity element*).*

Sketch of Proof: We note that unlike in Proposition 6.3.2 we don't have the luxury of assuming that $(C \sqcup \tau_2 \sqcup -C' \sqcup -\tau_1)$ will be Jordan (see, for example, Figure 6.2(b)). First, suppose paths τ_1 and τ_2 carries the cable from configuration C to final configuration C'. We choose two arbitrary points, p_1 and p_2, on the paths τ_1 and τ_2 respectively, as shown in Figure 6.7. Next consider the sequence of cable configurations from C to C' as the robots carry it. We can thus construct a continuous function (a homotopy), $\overline{C} : [0, 1] \times [0, 1] \to (W - \mathcal{O})$, such that $\overline{C}(0, \cdot) \equiv C(\cdot)$ and $\overline{C}(1, \cdot) \equiv C'(\cdot)$, and $\overline{C}(t)$ is a general intermediate cable configuration. Such a curve, $\overline{C}(t)$, has its end points $q_1(t) \in \tau_1$ and $q_2(t) \in \tau_2$ (Figure 6.7). We consider the curve connecting $q_1(t)$ to p_1 and lying on τ_1 (call it $\widetilde{q_1(t)p_1}$), and the one connecting $q_2(t)$ to p_2 and lying on τ_2 (call it $\widetilde{q_2(t)p_2}$). Thus, the sequence of curves, $D(t) := \left(-(\widetilde{q_1(t)p_1}) \sqcup \overline{C}(t) \sqcup (\widetilde{q_2(t)p_2})\right)$, defines a homotopy between curves connecting p_1 and p_2. Thus, $D(0) \sqcup -D(1)$ is null-homotopic. That is, $\left(-(\widetilde{q_1(0)p_1}) \sqcup \overline{C}(0) \sqcup (\widetilde{q_2(0)p_2})\right) \bigsqcup - \left(-(\widetilde{q_1(1)p_1}) \sqcup \overline{C}(1) \sqcup (\widetilde{q_2(1)p_2})\right) \equiv (C \sqcup \tau_2 \sqcup -C' \sqcup -\tau_1)$, is null-homotopic.

Conversely, if $(C \sqcup \tau_2 \sqcup -C' \sqcup -\tau_1)$ is null-homotopic, one can construct a homotopy, D, as before, and hence construct a sequence of curves $\overline{C}$, that takes the cable from C to C'. ■



(a) The rectangular environment with the goal of the robots being the left (cyan line) and right (magenta line) boundaries. ∂W_2 does not intersect any of the rays (all of which point in the positive Y direction).

(b) Robots 1 and 2 navigating on copies of graph, $\mathcal{G}$, formed by uniform discretization of configuration space. Change in the h-signature of the cable due to transitions within the graph is also illustrated.

Figure 6.8: The environment and its discretization.

6.4 Implementation

For simplicity, we assume that the environment, W, is a rectangular region ($x \in [x_l, x_r], y \in [y_b, b_t]$), and all the rays, $r_j,\ j = 1, 2, \cdots, r$ and $b_j,\ j = r+1, r+2, \cdots, r+b$, are parallel, pointing along the positive Y axis. Furthermore, we restrict the final goals of the robots to the left and right boundaries of the environment (∂W_l at $x = x_l$ and ∂W_r at $x = x_r$ respectively), but they need to reach the opposite edges. Thus a part of the boundary, ∂W_2, will never intersect any of the rays (Figure 6.8(a)), and hence $H(\partial W_2) = [0, 0, \cdots, 0]^T$ and $h(\partial W_2) =$"". This simplifies the computation of $H(C^f \sqcup \partial W_2)$ for Proposition 6.3.1 to the computation of $H(C^f)$.

We use a discrete representation of the environment, and construct a graph, $\mathcal{G}$, by placing a vertex in every discrete cell and by establishing an edge between the vertices of adjacent cells. From such a graph we can construct an H-augmented graph, $\mathcal{G}_H$ (for keeping track of the homology invariants), or an h-augmented graph, $\mathcal{G}_h$ (for keeping track of the homotopy invariants), as described in [15].

While the graph, $\mathcal{G}$, itself can be quite arbitrary, for simplicity we used a uniform 8-connected discrete representation (see Figure 6.8(b)) of the environment for all our simulations and experiments.

6.4.1 Planning in Joint State-space

The problem under consideration is to plan optimal paths that would take a given initial cable configuration, C^i, to a separating cable configuration, and the robot 1 reaches the left (or right) edge of W, while robot 2 reaches the right (or left) edge. In the first approach we plan paths in the joint state-space of the two robots. A graph, $\mathcal{J} = \mathcal{G} \times \mathcal{G}$, is defined as the *graph Cartesian product* of two copies of $\mathcal{G}$. Thus, for every pair of vertices, $\mathbf{u}_1, \mathbf{u}_2 \in V(\mathcal{G})$, a vertex in $V(\mathcal{J})$ is of the form $(\mathbf{u}_1, \mathbf{u}_2)$. We are given an initial vertex in the joint state-space, $(\mathbf{u}_1^i, \mathbf{u}_2^i)$, and an initial configuration of the cable (up to homotopy) in form of the h-signature

of the cable, $\mathfrak{h}_i$ (which, as defined earlier, is a *reduced word*).

We define an augmented graph, $\mathcal{J}_h$, such that a vertex in this graph contains the additional information of the h-signature of the cable that is being carried by the robots. This, in essence, is similar to the H-augmented graph construction detailed in [15]. Thus the initial vertex in the graph is $\mathbf{v}_i = (\mathbf{u}_1^i, \mathbf{u}_2^i, \mathfrak{h}^i)$, which contain the information about the initial positions of the robots and the h-signature of the initial cable configuration, $\mathfrak{h}^i = h(C^i)$. A transition of the robots from $(\mathbf{u}_1, \mathbf{u}_2, \mathfrak{h})$ to $(\mathbf{u}_1', \mathbf{u}_2', \mathfrak{h}')$ will mean (due to Proposition 6.3.2) that the h-signature of the resultant cable configuration is equal to $\mathfrak{h}' = h(-\tau_2) \diamond \mathfrak{h} \diamond h(\tau_1)$ (recall, '$\diamond$' is concatenation, followed by reduction), where τ_1 and τ_2 are paths taken by the robots for the transition (see Figure 6.8(b)). Thus, for each edge $[(\mathbf{u}_1, \mathbf{u}_2) \rightsquigarrow (\mathbf{u}_1', \mathbf{u}_2')] \in E(\mathcal{J})$, the vertex $(\mathbf{u}_1, \mathbf{u}_2, \mathfrak{h})$ is connected to neighbors $(\mathbf{u}_1', \mathbf{u}_2', h(\overrightarrow{\mathbf{u}_2'\mathbf{u}_2}) \diamond \mathfrak{h} \diamond h(\overrightarrow{\mathbf{u}_1\mathbf{u}_1'}))$ (where, $[\mathbf{a} \rightsquigarrow \mathbf{b}]$ is used to indicate an edge in edge set, $E(\mathcal{G})$, from vertex $\mathbf{a}$ to $\mathbf{b}$, and $\overrightarrow{\mathbf{ab}}$ is the curve/line segment that constitutes the edge. $\overrightarrow{\mathbf{ba}}$ is the same curve but with opposite orientation).

We choose the optimization objective to be the sum of the length of the robot paths. Thus, the cost of the edge $[(\mathbf{u}_1, \mathbf{u}_2, \mathfrak{h}) \rightsquigarrow (\mathbf{u}_1', \mathbf{u}_2', \mathfrak{h} \diamond h(\overrightarrow{\mathbf{u}_1\mathbf{u}_1'}) \diamond h(\overrightarrow{\mathbf{u}_2'\mathbf{u}_2}))] \in E(\mathcal{J}_h)$ is chosen to be the sum of the lengths of the edges $[\mathbf{u}_1 \rightsquigarrow \mathbf{u}_1']$ and $[\mathbf{u}_2 \rightsquigarrow \mathbf{u}_2']$ in $E(\mathcal{G})$. For this cost and with the left and right boundaries as goal, an *admissible heuristic function* is $f(\mathbf{u}_1, \mathbf{u}_2, \mathfrak{h}) = \min\left((u_{1,x} - x_l) + (x_r - u_{2,x}), (u_{2,x} - x_l) + (x_r - u_{1,x})\right)$, which is a lower bound on the cost to reach a goal from $(\mathbf{u}_1, \mathbf{u}_2, \mathfrak{h})$ (where, $u_{j,x}$ is the X coordinate at a vertex $\mathbf{u}_j$).

Starting at $(\mathbf{u}_1^i, \mathbf{u}_2^i, \mathfrak{h}^i)$ we thus keep expanding the vertices in the graph, $\mathcal{J}_h$, using a search algorithm (we use Dijkstra's [29] or A* [48] since they are complete, optimal and deterministic). A vertex $(\mathbf{u}_1, \mathbf{u}_2, \mathfrak{h})$ is deemed as goal if $\mathbf{u}_1 \in \partial W_l$ and $\mathbf{u}_2 \in \partial W_r$ (or vice-versa), and if $\mathfrak{h}_*(\mathfrak{h}) + H(\partial W_2)$ $(= \mathfrak{h}_*(\mathfrak{h}))$ satisfies the condition of Proposition 6.3.1 (*i.e.*, it is a separating cable configuration).

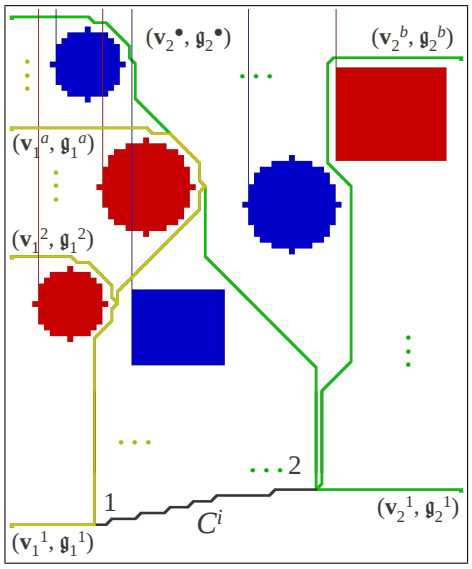
Planning in the joint state-space gives the flexibility of easily incorporating additional constraints like inter-robot collision avoidance, communication constraints, etc.

6.4.2 Decoupled Planning: A Distributed Approach

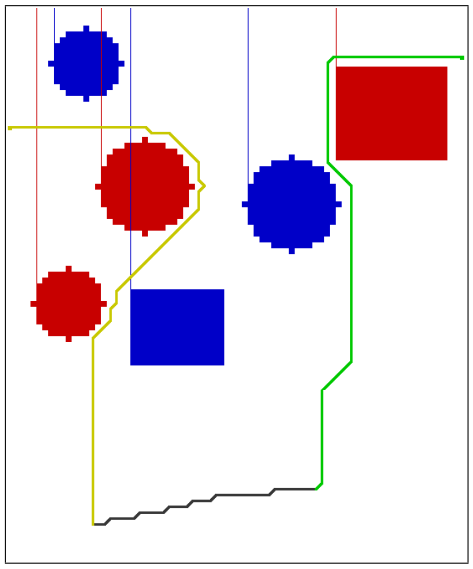
While the approach of planning in joint state-space is complete and optimal, it suffers from the obvious drawback of being slow and inefficient since the graph, $\mathcal{J}$, is very large and is of high degree, being a discrete representation of a 4-dimensional space. However, it is possible to decouple the searches for the two robots in two copies of $\mathcal{G}_h$ (the h-augmented graph of $\mathcal{G}$, described next), and run those searches in parallel (parallel threads in our C++ implementation), comparing the solutions obtained from each parallel process as they progress, and being able to conclude when the optimal solution is found, and thus halting the threads.

The h-augmented graph, $\mathcal{G}_h$, is very similar to the concept of the H-signature augmented graph, $\mathcal{G}_H$ described in [15], only with the homology invariants being replaced by the homotopy invariants. Corresponding to a given $\mathbf{u} \in V(\mathcal{G})$, there exists discrete number of the augmented states, $(\mathbf{u}, \mathfrak{h}) \in V(\mathcal{G}_h)$, for each homotopy class of paths (with h-signature $\mathfrak{h}$) from an initial vertex, $\mathbf{u}^i$, to the vertex $\mathbf{u}$. Edges emanating from $(\mathbf{u}, \mathfrak{h})$ are thus of the form $[(\mathbf{u}, \mathfrak{h}) \rightsquigarrow (\mathbf{u}', \mathfrak{h} + h(\overrightarrow{\mathbf{u}\mathbf{u}'}))] \in E(\mathcal{G}_h)$, corresponding to every $[\mathbf{u} \rightsquigarrow \mathbf{u}'] \in E(\mathcal{G})$. The cost of such an edge is chosen to be the Euclidean length of $\overrightarrow{\mathbf{u}\mathbf{u}'}$. An *admissible heuristic function* for this choice of cost, and with goal as $\partial W_l \cup \partial W_r$, is $f(\mathbf{u}, \mathfrak{h}) = \min(u_x - x_l, x_r - u_x)$.

Thus, we start with two copies of the augmented graph, $\mathcal{G}_{h,1}$ and $\mathcal{G}_{h,2}$, in two parallel threads (that branch off from a main thread), for robots 1 and 2. In robot j's copy of the graph, we start expanding the vertices



(a) Optimal paths in different homotopy classes corresponding to goal vertices $(\mathbf{v}_j^1, \mathfrak{g}_j^1), (\mathbf{v}_j^2, \mathfrak{g}_j^2), \cdots \in \mathcal{G}_{h,j}$ are found in parallel threads for robot $j = 1$ (yellow) and robot $j = 2$ (green).



(b) A compatible combination of paths (with end vertices $(\mathbf{v}_1^a, \mathfrak{g}_1^a)$ and $(\mathbf{v}_2^b, \mathfrak{g}_2^b)$) is such that the end points, $\mathbf{v}_1^a$ & $\mathbf{v}_2^a$, lie on the opposite edges of W, and $\mathsf{h}_*((\mathfrak{g}_2^b)^{-1} \diamond h(C^i) \diamond \mathfrak{g}_1^a)$ satisfies the condition of Proposition 6.3.1.

Figure 6.9: Decoupled and distributed planning: Optimal paths with different h-signatures found for the two robots in parallel threads, and costs of compatible pairs are compared to find the optimal compatible pair.

from (*i.e.*, initiate the *open set* with) the vertex $(\mathbf{u}_j^i, \text{“ ”}) \in \mathcal{G}_{h,j},\ j = 1, 2$. We keep expanding the vertices in the respective graphs, and keep storing a path every time ∂W_l or ∂W_r is reached via a new homotopy class for the robot (*i.e.* if $(\mathbf{v}, \mathfrak{g})$ is expanded, with $\mathbf{v} \in \partial W_l \cup \partial W_r$, then the vertex is bookmarked if the homotopy class $\mathfrak{g}$ is not same for any of the previously bookmarked vertices for the robot). It is important to note that for each of the robots such optimal paths with different h-signatures are found in the order of their costs since we use an optimal search algorithm (Dikjstra's/A* [48]). Suppose for robot 'j' such goal vertices are $\{(\mathbf{v}_j^1, \mathfrak{g}_j^1), (\mathbf{v}_j^2, \mathfrak{g}_j^2), (\mathbf{v}_j^3, \mathfrak{g}_j^3), \cdots\}$ with costs of the respective optimal paths $c_j^1 \leq c_j^2 \leq c_j^3 \leq \cdots$, for $j = 1, 2$.

We define a *partial order* [90], $\preccurlyeq$, on $\mathbb{R}^2$, to compare the cost of pairs of paths of robots 1 and 2. One obvious choice is to compare the sum of the path costs: $(\alpha_1, \alpha_2) \preccurlyeq (\beta_1, \beta_2) \Leftrightarrow \alpha_1 + \alpha_2 \leq \beta_1 + \beta_2$. However, one would desire that the task of carrying the cable is evenly distributed among the two robots, and not one of the robots end up traveling the most of the distance while the other travels very little. For this, we choose to minimize the maximum of the costs of the two paths (rather than their sum). Thus, we define the partial order to be

$$(\alpha_1, \alpha_2) \preccurlyeq (\beta_1, \beta_2) \quad \Longleftrightarrow \quad \begin{array}{l} \max(\alpha_1, \alpha_2) < \max(\beta_1, \beta_2) \ \text{ or} \\ \max(\alpha_1, \alpha_2) = \max(\beta_1, \beta_2) \ \text{ and } \ \min(\alpha_1, \alpha_2) \leq \min(\beta_1, \beta_2) \end{array} \tag{6.4.1}$$

which we call the *sorted lexicographic order*.

Thus, as the main thread of the program receives the two sequences of optimal paths to the left/right

boundaries with different h-signatures from the two different threads, it keeps checking them in pairs. A pair, $(\mathbf{v}_1^a, \mathfrak{g}_1^a)$ and $(\mathbf{v}_2^b, \mathfrak{g}_2^b)$, is deemed '*compatible*' (Figure 6.9(b)) if the corresponding final cable configuration (whose h-signature, by Proposition 6.3.2, is equal to $(\mathfrak{g}_2^b)^{-1} \diamond h(C^i) \diamond \mathfrak{g}_1^a$) is a separating configuration. That is, due to Proposition 6.3.1, a pair is compatible if $\mathsf{h}_*((\mathfrak{g}_2^b)^{-1} \diamond h(C^i) \diamond \mathfrak{g}_1^a) + H(\partial W_2)$ is a vector with first r components ± 1 and rest zeros, or last b components ± 1 and rest zeros. We keep record of the most optimal compatible pair (*i.e.*, one with *lowest* (c_1^a, c_2^b), where comparisons are made using '$\preccurlyeq$').

Say at an instant the most optimal pair has cost (c_1^*, c_2^*). Since the optimal paths with different h-signatures are found in order of there costs, if robot j finds a path such that its cost is greater than current value of $\max(c_1^*, c_2^*)$ (or, if we were using the sum of the pairs in defining the partial order, then $c_1^* + c_2^*$), we can say for sure that none of the paths to be discovered for robot j after that point can be part of a more optimal pair. Hence we stop the search for robot j. When the searches for both the robots end, the current optimal pair is the global optimal one.

6.4.3 Sequential Planning

While the approach of decoupled planning in the previous Section is efficient method for large map, it requires large memory and heavy computation with large number of objects. For example, in Figure 6.9(a), the planner of robot $j = 1$ finds the path to $(\mathbf{v}_1^1, \mathfrak{g}_1^1)$ first, because it is the minimum cost path to the boundary of the workspace. Then keep finding paths to the left vertices on the left boundary in the homotopy class of $\mathfrak{g}_1^1$ then it will find the optimal path in different homotopy class of $(\mathbf{v}_1^2, \mathfrak{g}_1^2)$. But, we need only one optimal path in each homotopy class. So, this decoupled planning can be and should be improved.

We divide this problem into two steps. The first step is find proper combination of homotopy class. Here we do not allow to wind the same object twice. In other word, we do not allow that r_k^+ or r_k^- $(k = 1, 2, \ldots, r)$ appears in $\mathfrak{g}_j^k$ more than once. If we have $n = r + b$ objects the upper bound of possible number of homotopy class is

$$N \leq 1 + 2n + 2n(2n-2) + 2n(2n-2)(2n-3) + 2n(2n-2)(2n-3)(2n-4) + \cdots \tag{6.4.2}$$

$$= 1 + 2n + \sum_{k=2}^{2n-1} 2n \prod_{m=1}^{k} (2n-k)) + 2n \prod_{m=1}^{2n-1} (2n-k)) \tag{6.4.3}$$

where the first 1 is for empty word or the word with length 0. For word of length 1 we can choose arbitrary reference ray in any direction. So the number of possible words is $2n$. For the k^{th} letter, it cannot be the $k-1$ letters which are already appeared and the letter of different sign with $k-1^{th}$ letter. As, the $2n^{th}$ letter has no choice, number of possible word of length $2n$ is the same with $2n-1$. However, in this computation, we did not consider the case that the letter of different sign with the previous, $(k-1)^{th}$, letter is already appeared. So, the Equation (6.4.2) can give us upper bound but not exact number of possible words. However, it is obvious that the possible homotopy class is finite and we can find the corresponding word to each homotopy class.

The decoupled planning works with heuristic function which gives the minimum cost to the boundary while not considering the homotopy class of the path. This function is a proper heuristic function for decoupled planning because there is no *desired homotopy class* in decoupled planning. However, we have desired homotopy class of the path in sequential planning, we need a heuristic function which consider the goal homotopy class. The basic concept of this heuristic function is that the robot should visit the reference ray

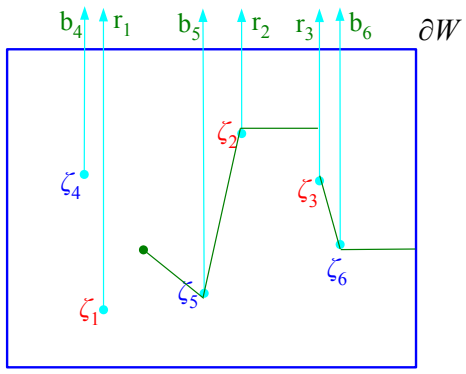


Figure 6.10: An example of heuristic cost(the sum of the length of green lines) of the path start from the green circle to the boundary while the desired homotopy class is $h_d =$ "$r_2^+ r_3^+$". In this example, we ignore the feasibility of the path with respect to objects.

in order described in the given *word* or h-signature to find the path in the desired homotopy class. While traveling between reference rays, other reference ray will work like obstacle to be avoided not to change the h-signature of the path. In Figure 6.10, the heuristic cost from given point, the green circle, to the boundary while the desired homotopy class is given by $h_d =$ "$r_2^+ r_3^+$" will be the sum of the length of green lines. As we ignore the feasibility with respect to the objects, the cost will be the sum of each length of path between start configuration to the first reference ray, paths between reference rays; and the path between the last reference ray and the boundary of the workspace while considering other reference rays as obstacles. Also, we do not consider the cost between segments of paths, it is underestimation and an admissible heuristic function. We add some detailed example in Appendix A.

The sequential planning we define a vertex as

$$c^{a,b} = (\mathfrak{g}_1^a, \mathfrak{g}_2^b, c_1^a, c_2^b, realcost_1^a, realcost_2^b) \tag{6.4.4}$$

where is $\mathfrak{g}_j^a$ is the homotopy class of thej^{th} robot, c_j^a is the cost to the corresponding path, $realcost_j^a$ is boolean variable if corresponding cost c_j^a is the real cost or heuristic cost. Then the following algorithm describe the sequential planning

Algorithm 1**:** Pseudocode for *Sequential Planning*:

1. |Find all possible homotopy class for each robot.
2. |Build the set of all possible combinations $c^{a,b} \in C$ satisfies the condition of Proposition 6.3.1.
3. |**for** all $c^{a,b} \in C$
4. |Calculate heuristic cost and update c_1^a and c_2^b. Set $realcost_1^a = false$ and $realcost_2^b = false$.
5. |**end for**
6. |**while** TRUE
7. |Sort C and find the combination of minimum cost, $c^{a,b} \in C$
8. |**if** $realcost_1^a \& realcost_2^b$
9. |**break**
10. |**else**
11. |**if** $notrealcost_1^a$

12. |Find optimal path of the first robot in homotopy class of $\mathfrak{g}_1^a$
13. |Update c_1^a. Set $realcost_1^a = true$.
14. |**end** if
15. |**if** $notrealcost_2^b$
16. |Find optimal path of the second robot in homotopy class of $\mathfrak{g}_2^b$
17. |Update c_2^b. Set $realcost_2^b = true$.
18. |**end** if
19. |**end** while
20. |**return** $c^{a,b}$

Then the result combination is the optimal one. The unexpanded vertices cannot be better because the estimated cost is underestimation. We need to find optimal paths in each homotopy class to update the cost with real value. However, we do not need to non-optimal paths in each homotopy class to reduce the memory requirements and computation power.

6.5 Result

In this section, we demonstrate the performance of the algorithm and implementation in the previous section by various ways.

6.5.1 Simulation Results

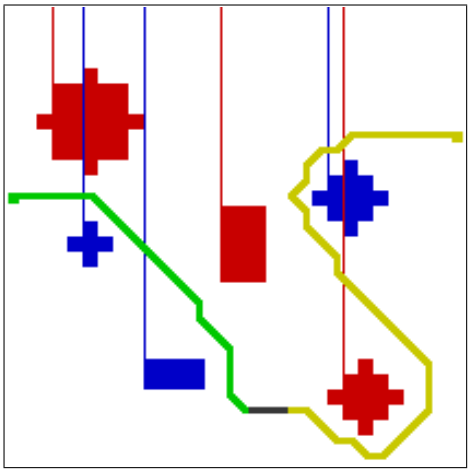
We implemented the search in the joint state-space as well as the decoupled search in C++ programming language with ROS integration, and used A* search algorithm. All computations were performed on a system with dual-core processor with clock speed 2.6 MHz and 4 Gb memory. Throughout this thesis we consider an uniform discretization of the environment for simplicity. However, the techniques developed in this thesis is not restricted to any specific discretization scheme or even a specific search algorithm. A more detailed discussion on the generality of the technique can be found in [15].

Joint State-space Plan

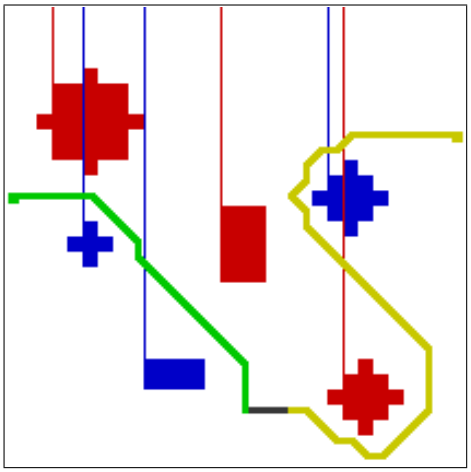
The search in this 4-dimensional environment is prohibitively expensive for large environments. Figure 6.11(a) shows the result in a simple environment, 30×30 discretized, and with 3 objects of each type. The search took about 4250 s and expanded 1484999 vertices in $\mathcal{J}_h$. Figure 6.11(b) shows the result obtained for same problem, but using the decoupled planning (and using sum of the cost of the paths for defining the partial order, $\preccurlyeq$, for being consistent). The result has the same optimal cost as the joint state-space planning, but took less than 1 s with 19144 and 19593 vertices being expanded in $\mathcal{G}_{h,1}$ and $\mathcal{G}_{h,2}$. All objects were *inflated* to avoid collision.

Decoupled Planning

In this section we present results obtained using the decoupled, distributed implementation. The sorted lexicographic order was used for '$\preccurlyeq$'. Figure 6.12(a) show the plans obtained for two robots in a 100×100 discretized environment. The planning took about 1.3 s, and expanded 39764 and 40066 vertices in

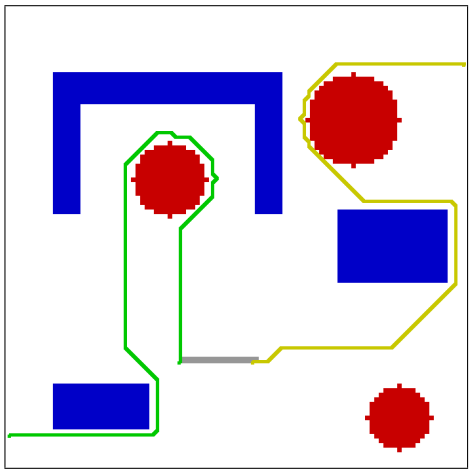


(a) Planning in the joint state-space took 4250s. The sum of the costs of two paths is 65.598 discretization units.

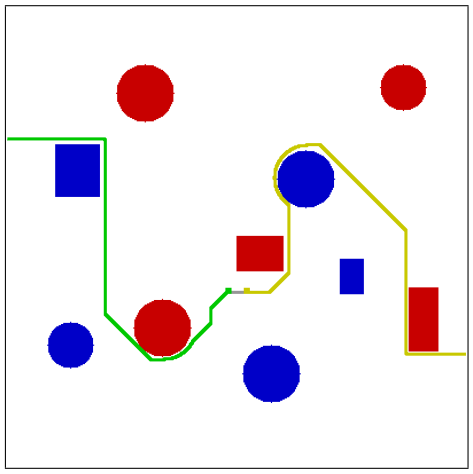


(b) The distributed decoupled planning gives result with the same optimal cost, but takes about $2s$ to run.

Figure 6.11: A simple 30×30 environment with $r = b = 3$. The green & yellow are the paths of the robots. The rays emanating from ζ_j are also shown. The dark gray segment indicates the initial cable configuration.



(a) The planned paths in a 100×100 discretized environment.



(b) The planned paths in a 400×400 discretized environment.

Figure 6.12: Decoupled, distributed plans. Initial cable is shown in gray/black. Paths are in green and yellow.

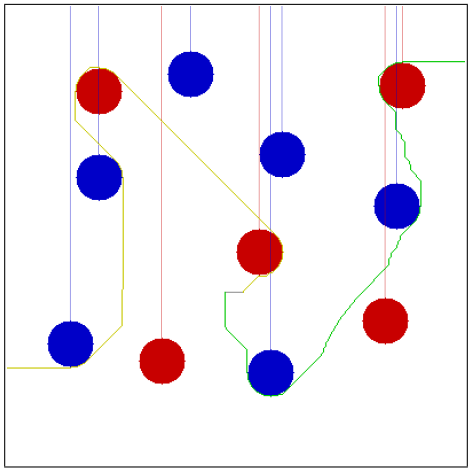


Figure 6.13: Sequential plan. Initial cable is shown in gray/black. Paths are in green and yellow.

the graphs of the two robots. Figure 6.12(b) shows the result in a much larger (400×400 discretized) environment. The planning time for this case was 490 s, with 1086182 and 1079670 vertices being expanded.

Sequential Planning

In this section we present results obtained using the sequential planning implementation. The decoupled algorithm shows fast computation with several examples. However. the decoupled algorithm failed to find optimal paths in the case of eleven objects in Figure 6.13 because of lack of memory. The sequential planner find this optimal path in 350 s.

6.5.2 Dynamic Simulation and Fast Re-planning

So far we have planned the paths with the assumptions that the object remain stationary as the robots follow the planned paths. However, in a practical implementation, where the objects will be free to move on the surface, the interaction between the cable and the objects will change the configuration of the environment. Consequently there comes the need for re-planning.

Dynamic Simulation

For the purpose of testing this scenario we build an accurate real-time dynamic simulation platform for the cable (modeled as a serial chain) and freely floating disk-shaped objects on a fluid. Using Lagrangian mechanics we developed the equations of motion with realistic modeling of drag forces [12], and modeled the contacts using linear complementarity conditions [1]. We use a simple feedback (PD) controller to make each robot follow the paths generated by the planner.

In this section, we extend the dynamics simulation platform proposed in [12]. We adopt the discrete dynamic model of the fixed-length cable by modeling the cable as a series of n rigid cylindrical segments connected by revolute joints. The i^{th} segment is modeled as an uniform cylinder of length L_i, and its

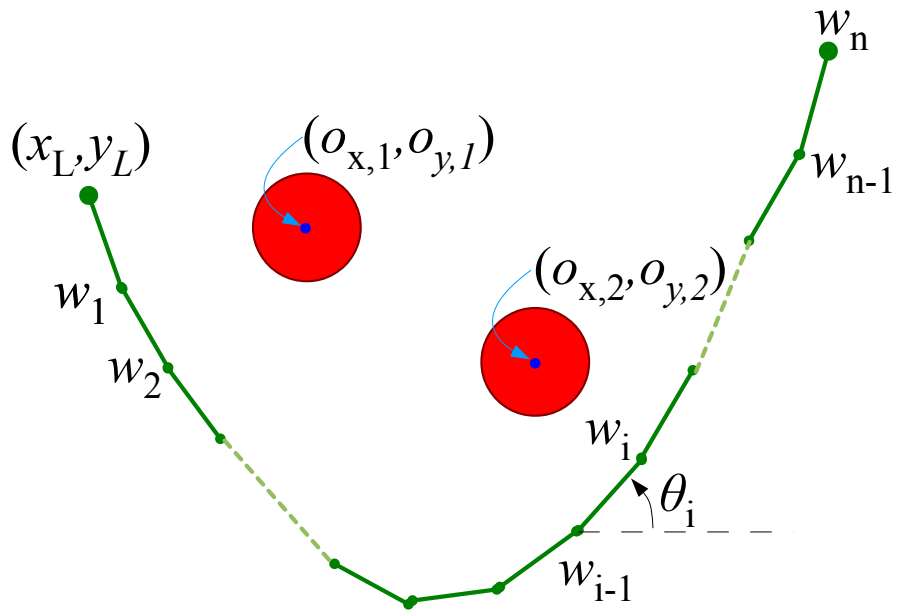


Figure 6.14: The dynamic model showing a discrete model of the cable consisting of n rigid segments and two rigid circular objects.

diameter, d_i, is assumed to be much smaller than its length. This model of the cable has $n + 2$ degrees of freedom.

We also consider the dynamics of n_o count of objects freely floating on the water. Although it is straightforward to extend them to non circular objects, for simplicity, the objects are modeled as uniform disks of radius $R_j,\ j = 1, 2, \cdots, n_o$. Each disk-shaped object is assumed to have only two degrees of freedom – the x and y coordinates of their centers. The objects' rotational degree of freedom are ignored due to symmetry of the disk shapes, and due to the assumption that the coefficient of friction on the surface of the objects are zero, thus allowing only normal forces to be imparted on the objects.

Thus the system has $(2 + n) + 2n_o$ degrees of freedom. We choose the generalized coordinates to be $\mathbf{q} = [x_L, y_L, \theta_1, \ldots, \theta_n, o_{1,x}, \ldots, o_{n_o,x}, o_{1,y}, \ldots, o_{n_o,y}]^T$, where $\mathbf{w}_0 := [x_L, y_L]^T$ is the coordinates of the point at which the cable is attached to the left boat, θ_i is the angles made by the i^{th} cylindrical segment with respect to positive X axis of the global inertial frame of reference, and $\mathbf{o}_j = [o_{j,x}, o_{j,y}]^T$ are the coordinates of the center of the j^{th} object (see Figure 6.14).

Using Lagrangian mechanics, the equations of motion for the system can be written as,

$$\frac{d}{dt}\left(\frac{\partial K}{\partial \dot{q}_l}\right) - \frac{\partial K}{\partial q_l} - Q_{q_l} = 0 \tag{6.5.1}$$
$$\forall q_l \in \{x_L, y_L, \theta_1, \ldots, \theta_n, o_{1,x}, \ldots, o_{n_o,x}, o_{1,y}, \ldots, o_{n_o,y}\}$$

where, K, the kinetic energy of the system, is given by,

$$K = \sum_{i=1}^{n} \left(\frac{1}{2} m_i \left(\dot{\mathbf{p}}_i \cdot \dot{\mathbf{p}}_i \right) + \frac{1}{2} \frac{m_i L_i^2}{12} \dot{\theta}_i^2 \right) + \sum_{j=1}^{n_o} \frac{1}{2} M_j \left(\dot{\mathbf{o}}_j \cdot \dot{\mathbf{o}}_j \right) \tag{6.5.2}$$

where $\mathbf{p}_i$ is the center of mass of the i^{th} segment of the cable, m_i is the mass of the i^{th} segment of the cable, and M_j is the mass of the j^{th} object. Since we assume frictionless contact with the disk-shaped objects, the kinetic energy due to rotation of the objects will remain unchanged (*i.e.* derivatives w.r.t. the coordinates and time will be zero), and hence not considered as part of the expression for kinetic energy in the Lagrange equations.

The generalized forces are

$$Q_{x_L} = f_{Lx} + f_{Rx} + \sum_{j=1}^{n} F_{j,x} \tag{6.5.3}$$

$$Q_{y_L} = f_{Ly} + f_{Ry} + \sum_{j=1}^{n} F_{j,y} \tag{6.5.4}$$

$$Q_{\theta_i} = f_{Rx} L_i \sin\theta_i + f_{Ry} L_i \cos\theta_i + \tau_i + \sum_{k=1}^{n} \mathbf{F}_k \cdot \frac{\partial \mathbf{p}_k}{\partial \theta_i} \qquad \forall i \in \{1, 2, \ldots, n\} \tag{6.5.5}$$

$$Q_{o_{j,x}} = - c_j \dot{o}_{j,x} \tag{6.5.6}$$

$$Q_{o_{j,y}} = - c_j \dot{o}_{j,y} \tag{6.5.7}$$

where $[f_{Lx}, f_{Ly}]^T$ and $[f_{Rx}, f_{Ry}]^T$ are the input forces on the left and right ends of the cable respectively;

$$\mathbf{F}_i := \begin{bmatrix} F_{i,x} \\ F_{i,y} \end{bmatrix} = - \int_{-L_i/2}^{L_i/2} \left(c_V \mathbf{v}_i^{\parallel}(s) + c_S \mathbf{v}_i^{\perp}(s) \right) ds \tag{6.5.8}$$

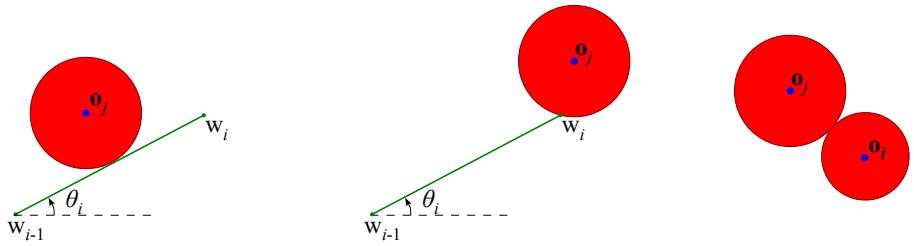
$$\tau_i := - \int_{-L_i/2}^{L_i/2} \mathbf{r}_i(s) \times \left(c_V \mathbf{v}_i^{\parallel}(s) + c_S \mathbf{v}_i^{\perp}(s) \right) ds \tag{6.5.9}$$

are the forces and torques due to drags on the cylindrical segments of the cable, where c_V and c_S are constants that are functions of fluid properties and Reynolds number [42] and $\mathbf{r}_i$ is the moment arm with respect to the left end of cylindrical segment. For these constants we use the same values suggested in [12]. In writing the above equations for the forces and torques, the relative flow velocity along each segment has been decomposed into two components – one parallel to the axis ($\mathbf{v}_i^{\parallel}$), other perpendicular to the axis ($\mathbf{v}_i^{\perp}$), which, using a parameter, s, along the length of the segments, are given by,

$$\mathbf{v}_i^{\parallel}(s) = \left(\begin{bmatrix} \cos\theta_i \\ \sin\theta_i \end{bmatrix} \cdot \dot{\mathbf{r}}_i(s) \right) \begin{bmatrix} \cos\theta_i \\ \sin\theta_i \end{bmatrix} \tag{6.5.10}$$

$$\mathbf{v}_i^{\perp}(s) = \left(\begin{bmatrix} -\sin\theta_i \\ \cos\theta_i \end{bmatrix} \cdot \dot{\mathbf{r}}_i(s) \right) \begin{bmatrix} -\sin\theta_i \\ \cos\theta_i \end{bmatrix} \tag{6.5.11}$$

under the assumption that the fluid is stationary. Also, the generalized forces on the j^{th} object due to the drag (Equation (6.5.6) and (6.5.7)) are assumed to be proportional to the speed of the flow at low Reynold's



(a) The first type of contact. (b) The second type of contact. (c) The third type of contact.

Figure 6.15: The three types of contacts considered in the model.

number. For a 3-dimensional sphere moving through a fluid, Stokes' law give a drag equal to $F_{D,sphere} = 3\pi\mu U d$ where U and d are speed and diameter of the sphere respectively [101], the direction of which is opposite to the direction of motion of the sphere. We assume that each object is a sphere of radius R_j, the bottom half of which is under the water surface. Thus, the drag on each object being half of that on the sphere,

$$\mathbf{F}_{D,j} = \frac{-3\pi\mu \mathbf{U} d}{2} = -3\pi\mu R_j \dot{\mathbf{o}}_j. \tag{6.5.12}$$

Thus, $c_j = 3\pi\mu R_j$ in Equations (6.5.6) and (6.5.7), where the kinematic viscosity of seawater is $\mu = 1.07 \times 10^{-3} kg/(m \cdot s)$ at $20\,^{\circ}\mathrm{C}$ [101].

Using first order approximation of the acceleration in the equations of motion, Equation (6.5.1), we can write at the k^{th} time instant,

$$M(\mathbf{q}^k)\frac{\dot{\mathbf{q}}^{k+1} - \dot{\mathbf{q}}^k}{\Delta t^k} \; - \; \mathbf{V}(\dot{\mathbf{q}}^k, \mathbf{q}^k) \; - \; \mathbf{Q}^k \;\; = \;\; 0 \tag{6.5.13}$$

where $\mathbf{V}(\dot{\mathbf{q}}^k, \mathbf{q}^k) \in \mathbb{R}^{n+2n_o+2}$ and $\mathbf{Q}^k = \left[Q_{q_1}, \ldots, Q_{q_{n+2n_o+2}}\right]^T \in \mathbb{R}^{n+2n_o+2}$ is the vector of generalized forces. Thus, using Euler integration, we can find the generalized states and velocities at the $(k+1)^{th}$ time step.

We next consider the contact between cable and objects. Since we assume frictionless contacts between the rigid bodies, we adopt a time-stepping algorithms to solve Linear Complementarity Problem at every step [1, 3, 91]. In order to account for *noninterpenetration* between the rigid bodies with smooth distance function, we assume that all the objects are disk-shaped, and that the thickness of cable segments are negligible, which is consistent with our earlier assumptions. In our model we consider three different cases of contacts or noninterpenetration conditions.

The first case (Figure 6.15(a)) is the *tangential contact* between the segments of the cable and the disk-shaped object. For this, we need to consider the distance between the i^{th} segment and the j^{th} disk-shaped object only when the center of the object lies in the strip perpendicular to the segment and containing the segment (the yellow region in Figure 6.16). If the i^{th} segment and the j^{th} disk-shaped object do not satisfy the both conditions, we do not need to consider contact between them.

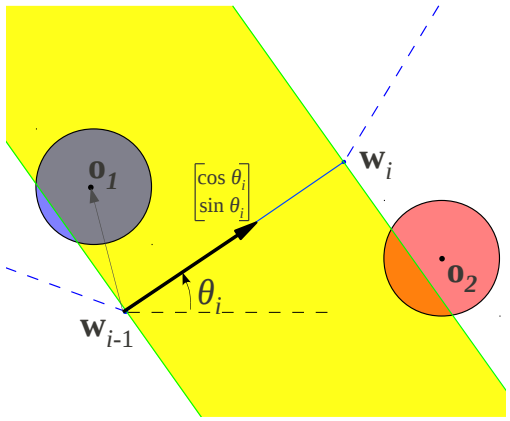


Figure 6.16: When the center of an object lies in the yellow region, we need to check for contact between the i^{th} segment of the cable and the object. The boundary of yellow region (*i.e.* the green lines) are perpendicular to $(\mathbf{w}_i - \mathbf{w}_{i-1})$. In this example, we need to check for contact between i^{th} segment and o_1, but not o_2.

The inequalities that need to be satisfied for that to happen are,

$$\begin{bmatrix} \cos\theta_i \\ \sin\theta_i \end{bmatrix} \cdot (\mathbf{o}_j - \mathbf{w}_{i-1}) \geq 0 \tag{6.5.14}$$

$$-\begin{bmatrix} \cos\theta_i \\ \sin\theta_i \end{bmatrix} \cdot (\mathbf{o}_j - \mathbf{w}_i) \geq 0 \tag{6.5.15}$$

where $\mathbf{w}_i$ is the coordinate of the right end of the i^{th} segment:

$$\mathbf{w}_i = \mathbf{w}_{i-1} + L_i \begin{bmatrix} \cos\theta_i \\ \sin\theta_i \end{bmatrix} \tag{6.5.16}$$

with $w_0 = [x_L, y_L]^T$ being the left end of the cable. Then the distance between the i^{th} segment and j^{th} object is

$$D_{LO,ij} = \left| \begin{bmatrix} \cos\theta_i \\ \sin\theta_i \end{bmatrix} \times (\mathbf{o}_j - \mathbf{w}_{i-1}) \right| - R_j. \tag{6.5.17}$$

The second case (Figure 6.15(b)) is the *non-tangent contact* between an end of a segment and an object. Such a case can arise when at least one of the Equation (6.5.14) and (6.5.15) is not satisfied. Then the distance function is

$$D_{PO,ij} = \|\mathbf{w}_i - \mathbf{o}_j\| - R_j \tag{6.5.18}$$

for $\forall i \in \{0, 1, \ldots, n\}$ and $\forall j \in \{1, \ldots, n_o\}$.

The last case (Figure 6.15(c)) is the contact between two objects. The distance function in this case is

$$D_{OO,ij} = \|\mathbf{o}_i - \mathbf{o}_j\| - R_i - R_j \tag{6.5.19}$$

for $\forall i, j \in \{1, \ldots, n_o\}$.

At a given state, $\mathbf{q}_k$, at time t^k, we assume all the distances are nonnegative. We select and stack up all the distance functions whose values are less than some threshold, δ, (say m of them) into a vector, $\mathbf{f}(\mathbf{q}^k) = \left[f_1(\mathbf{q}^k), \ldots, f_m(\mathbf{q}^k)\right]^T$, *i.e.* $f_p(\mathbf{q}^k) < \delta$ for $p = 1, 2, \ldots, m$. Then our goal is to find the state in the next step which satisfies,

$$f_p(\mathbf{q}^{k+1}) \geq 0 \tag{6.5.20}$$

which results in a nonlinear problem. We thus linearize the distance functions around $\mathbf{q}^k$ using a first order approximation as follows [1, 3, 91],

$$f_p(\mathbf{q}^{k+1}) \;\simeq\; f_p(\mathbf{q}^k) \;+\; \Delta t^k \nabla_{\mathbf{q}} f_p(\mathbf{q}^k) \dot{\mathbf{q}}^{k+1} \tag{6.5.21}$$

which is linear with respect to the generalized velocity of the next time step.

Now we add the impulse due to contact into the generalized force term of Equation (6.5.13), and thus construct the equations of motion which satisfy the noninterpenetration constraints

$$\begin{aligned} M^k \dot{\mathbf{q}}^{k+1} =& M^k \dot{\mathbf{q}}^k + \Delta t^k \left(\mathbf{V}^k + \mathbf{Q}^k\right) + \nabla_{\mathbf{q}} \mathbf{f}(\mathbf{q}^k)^T \lambda^k \\ =& M^k \dot{\mathbf{q}}_u^k + \nabla_{\mathbf{q}} \mathbf{f}(\mathbf{q}^k)^T \lambda^k \end{aligned} \tag{6.5.22}$$

where $M^k = M(\mathbf{q}^k)$, $\mathbf{V}^k = \mathbf{V}(\dot{\mathbf{q}}^k, \mathbf{q}^k)$, $\mathbf{Q}^k = \mathbf{Q}(\mathbf{q}^k)$ and $\lambda^k = \left[\lambda_1^k, \ldots, \lambda_m^k\right]^T$ is a vector of nonnegative components (since the impulse due contact of two rigid bodies should be in a direction such that the distance functions increase). To simplify the notation, we define $\dot{\mathbf{q}}_u^k$ to be the generalized velocity of the next time step when there is no impulse due to noninterpenetration constraints, which satisfies the Equation (6.5.13). Then we substitute this equation into the noninterpenetration constraints to achieve a linear inequality,

$$\begin{aligned} & \mathbf{f}(\mathbf{q}^k) + \Delta t^k \nabla_{\mathbf{q}} \mathbf{f}(\mathbf{q}^k) \left(M^k\right)^{-1} \left(M^k \dot{\mathbf{q}}_u^k + \nabla_{\mathbf{q}} \mathbf{f}(\mathbf{q}^k)^T \lambda^k\right) \\ =& \mathbf{f}(\mathbf{q}^k) + \Delta t^k \nabla_{\mathbf{q}} \mathbf{f}(\mathbf{q}^k) \dot{\mathbf{q}}_u^k + \Delta t^k \nabla_{\mathbf{q}} \mathbf{f}(\mathbf{q}^k) \left(M^k\right)^{-1} \nabla_{\mathbf{q}} \mathbf{f}(\mathbf{q}^k)^T \lambda^k \\ =& \mathbf{b}^k + A^k \lambda^k \geq 0 \end{aligned} \tag{6.5.23}$$

where $\mathbf{b}^k = \mathbf{f}(\mathbf{q}^k) + \Delta t^k \nabla_{\mathbf{q}} \mathbf{f}(\mathbf{q}^k) \dot{\mathbf{q}}_u^k \in \mathbb{R}^m$ and $A^k = \Delta t^k \nabla_{\mathbf{q}} \mathbf{f}(\mathbf{q}^k) \left(M^k\right)^{-1} \nabla_{\mathbf{q}} \mathbf{f}(\mathbf{q}^k)^T \in \mathbb{R}^{m \times m}$ are known, given by the current generalized state and velocity, $\mathbf{q}^k$ and $\dot{\mathbf{q}}^k$. Hence we need to solve a linear complementarity problem (LCP),

$$\begin{aligned} \mathbf{b}^k + A^k \lambda^k &\geq 0 \\ \lambda^k &\geq 0 \\ \lambda^k \cdot \left(\mathbf{b}^k + A^k \lambda^k\right) &= 0 \end{aligned} \tag{6.5.24}$$

at each time step. We can solve this LCP efficiently with Lemke's method [26, 38]. We then substitute the solution of λ^k into the following equation (derived from Equation (6.5.22)) to find the generalized state and

velocity for the next time step.

$$\dot{\mathbf{q}}^{k+1} = \dot{\mathbf{q}}_u^k + \left(M^k\right)^{-1} \nabla_{\mathbf{q}} \mathbf{f}(\mathbf{q}^k)^T \lambda^k \tag{6.5.25}$$

$$\mathbf{q}^{k+1} = \mathbf{q}^k + \Delta t^k \dot{\mathbf{q}}^{k+1} \tag{6.5.26}$$

Re-planning

Throughout the Section 6.4, we presented algorithms to solve this path planning problem more fast in large envirnoment with objects. However, the simulation results in the previous Section 6.5.1 showed that it is hard to implement real-time planner in large envirnoment. However, in the dynamics environment, when the objects are fre to move, we need real-time re-planning for successful manipulation. So in this section, we will briefly present a real-time replanning algorithm for real-time simulation or experiments in dynamic environments. Instead of solving the path planning problem to adapt the change of the environment, we find the optimal paths of the robots which lead to the same homotopy class of the initial path.

Instead of solving the entire problem every time whenever the environment changes, we invoke a re-planning algorithm whenever we need replanning:

- Any two objects exchange the order of the X coordinates of their representative points (*i.e.*, the *rays* emanating from ζ_j cross each other)
- One of the planned paths becomes invalid (due to an object moving on top of it).

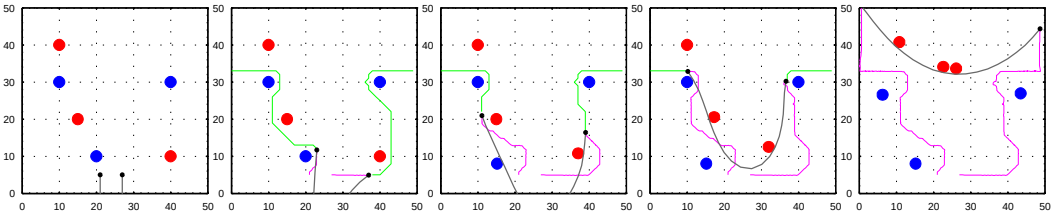
Let $\mathfrak{g}_{sc} = (\mathfrak{g}_2^b)^{-1} \diamond h(C^i) \diamond \mathfrak{g}_1^a$ be the h-signature of the separating configuration of the iniatial path planning, which can be generated by any algorithm described in the previous Section 6.4. Assume we keep tracking the cable configuration and its homotopy class, $\mathfrak{g}_c = h(C)$. If the trigger was caused due to switching of the X coordinates of two representative points, we interchange the positions of the corresponding letters in the words ($\mathfrak{g}_{sc}$ and $\mathfrak{g}_c$) wherever they appear side-by-side. Then the goal h-signature of the paths of two robots should be determined so that $\mathfrak{g}_{sc} = (\mathfrak{g}_2^{b'})^{-1} \diamond \mathfrak{g}_c \diamond \mathfrak{g}_1^{a'}$. We can simply split the h-signature of the separating configuration as $\mathfrak{g}_{sc} = \mathfrak{g}_{sc}^f \diamond \mathfrak{g}_{sc}^b$ and the number of possible configurations of $(\mathfrak{g}_{sc}^f, \mathfrak{g}_{sc}^b)$ is $length(\mathfrak{g}_{sc})+1$. And we can split the h-signature of the current cable configuration as $\mathfrak{g}_c = \mathfrak{g}_c^f \diamond \mathfrak{g}_c^b$ in the same manner to find $length(\mathfrak{g}_c) + 1$ combinations. Then we will achieve $(length(\mathfrak{g}_{sc}) + 1) \times (length(\mathfrak{g}_{sc}) + 1)$ combiniations of

$$\begin{aligned} \mathfrak{g}_1^{a'} &= (\mathfrak{g}_c^b)^{-1} \diamond \mathfrak{g}_{sc}^b \\ \mathfrak{g}_2^{b'} &= (\mathfrak{g}_{sc}^f \diamond (\mathfrak{g}_c^f)^{-1})^{-1} = \mathfrak{g}_c^f \diamond (\mathfrak{g}_{sc}^f)^{-1} \end{aligned} \tag{6.5.27}$$

which will drive the cable-robot system to the same separating configuration of the initial path:

$$\begin{aligned} (\mathfrak{g}_2^{b'})^{-1} \diamond \mathfrak{g}_c \diamond \mathfrak{g}_1^{a'} &= (\mathfrak{g}_c^f \diamond (\mathfrak{g}_{sc}^f)^{-1})^{-1} \diamond \left(\mathfrak{g}_c^f \diamond \mathfrak{g}_c^b\right) \diamond \left((\mathfrak{g}_c^b)^{-1} \diamond \mathfrak{g}_{sc}^b\right) \\ &= \mathfrak{g}_{sc}^f \diamond (\mathfrak{g}_c^f)^{-1} \diamond \mathfrak{g}_c^f \diamond \mathfrak{g}_c^b \diamond (\mathfrak{g}_c^b)^{-1} \diamond \mathfrak{g}_{sc}^b = \mathfrak{g}_{sc}^f \diamond \mathfrak{g}_{sc}^b = \mathfrak{g}_{sc}. \end{aligned} \tag{6.5.28}$$

As we have finite number of possible combinations of h-signatures of pthas of robots, we find the optimal paths of the robots from current positions by using the same algorithm used in the sequential planning



(a) $t = 0.00$ s (b) $t = 58.06$ s (c) $t = 178.35$ s (d) $t = 298.65$ s (e) $t = 479.10$ s

Figure 6.17: Dynamic simulation for separation of objects. The gray curve is the cable, with black dots marking robots at its ends. Green curves are the planned paths. Magenta curves are the robot footprints. Red & blue disks are the rigid freely-floating objects. See http://youtu.be/GyCn-8yDzO0 for video.

Figure 6.18: Experiments with Autonomous Boats conducted by H. K. Heidarsson, University of Southern California [56]. Red and blue circles are Buoys (objects). Thin gray curve is the planned paths of two ASVs. The Black curve is the current cable configuration. See http://youtu.be/vGgca2w2UdA for video.

described in 6.4.3, which finds the optimal combination of $h-$-signatures and paths of the robots. The only difference is how we build the set of combinations, C.

Figure 6.17 shows the simulation result. Figure 6.17(a) shows the initial configuration of the system. As the objects move and the map change, the planned paths of robot are re-computed (shown by green curves in Figures 6.17(b)-(e)). We are able to successfully separate the red objects from the blue ones.

6.5.3 Experiment Results

To demonstrate the performance of the suggested algorithm, the field experiments were conducted in Puddingstone Lake, San Dimas, CA. The experiment was performed by Hordur Kristinn Heidarsson under the supervision of Prof. Gaurav Sukhatme of University of Southern California [56].

The two Autonomous Surface Vessels (ASV) are identical, each around 2 m long and 0.8 m wide, capable of speeds up to 1.6 m/s, using two electric thrusters and a rudder for control. Both are equipped with a GPS, an IMU with integrated compass and an onboard computer for control.

The two ASVs have a 40 m long floating rope attached between them with periodically spaced markers

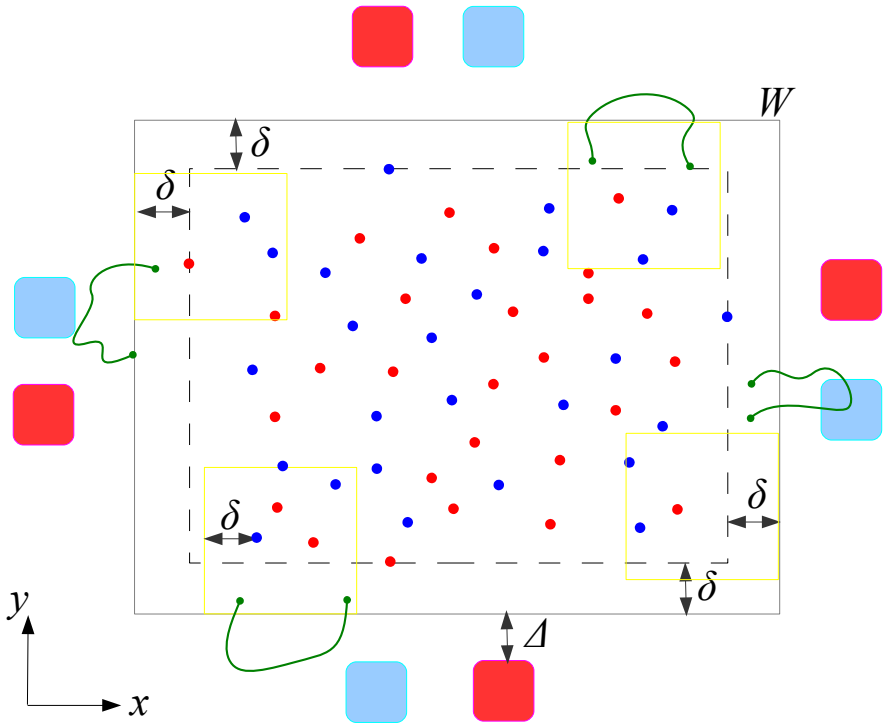


Figure 6.19: A large problem. Red and blue dots are object. Green curves are cable-robot teams. Light blue and red boxes are the baskets to bring objects. The dashed line is a smallest box to enclose all objects to be manipulated. Gray box is the workspace of the problem. The yellow boxes are the workspace of each cable-robot team.

on it for increased visibility and to use as fixed sampling points. Buoys (the objects to be separated) are placed in the water and anchored in place in an area 50 m $\times$ 50 m, and their approximate locations recorded using GPS.

To record the experiment, particularly the position and shape of the rope, a camera, in an adjustable tilt mount, was mounted on top of a 30 ft mast which stood on shore close to the experiment area, overlooking it. Figure 6.18 shows a snapshot of the experimental. The video of experiment is available at http://youtu.be/vGgca2w2UdA.

The experiment result showed that the planned paths were feasible and the ASVs followed the planned paths successfully.

6.6 Sequential Manipulation of Large Number of Objects

We presented various implementation to decrease the computation time and solve this planning problem with more number of objects in larger environments in Section 6.4. However, it is practically impossible to find a solution with a large number of objects due to heavy computation. Also, as the size of the workspace and number of objects increases, we need longer cable and more powerful actuator on robots to manipulate and transport obejcts. So, it is not practical to manipulate and transprt a large number of objects with the

same strategy described in the previous sections. In this section, we will consider the problem to manipulate and transport a large number of objects with multiple cable-robot teams (see Figure 6.19). There are a large number of red and blue objects. Each cable-robot team will sequentially manipulate and transport the objects to the corresponding baskets. To simplify the algorithm, we assume that the number of cable-robot team is equal to the number of pairs of red/blue baskets. And each cable-robot team is assigned to its own pair of baskets.

6.6.1 Algorithm

In this section, we will describe how we can sequentially manipulate and transport objects to basket efficiently and successfully. As the size of the planning problem become enormous comparing to the problem of single cable-robot team, it is practically impossible to find paths of all the cable-robot team in a single planner; the dimension of the search space will be too large for real-time planning. So, we will reduce the size of the problem by splitting the large planning problem into a planning problem of single cable-robot team like the previous problem. We need to decentralize the planning so that each cable-robot team finds its own path.

Workspace of each cable-robot team

As there are other cable-robot teams, each cable-robot team should consider the configurations and paths of other cable-robot teams to avoid collisions and cable entanglements. As the configurations and paths of other cable-robot teams keep changing and the manipulation will happen only in a part of whole workspace, it is not an efficient way to find path of cable-robot team in the whole workspace. So, we assign the workspace, $W_j \subset W$, as a subset of the whole workspace for the j^{th} cable-robot team. To determine the workspace of each robot, we will explain with the robot on the bottom of Figure 6.19. For other robot will work with the same manner by proper transformation(rotation) of the coordinates. In Figure 6.19, the dashed lines are the boundary of the smallest rectangle including all the red and blue object to be manipulate while its edge are parallel to x or y axis. Then the workspace $W = \{(x, y)|x_l \leq x \leq x_r, y_b \leq y \leq y_t\}$ will have δ margins from this box. Then the workspace of the j^{th} cable robot team whose cable length is L_j is a square box of

$$W_j = \{(x, y)|x_j \leq x \leq x_j + l_{j,x}, y_b \leq y \leq y_b + l_{j,y}\} \tag{6.6.1}$$

where l_j is the length of the edge while $\sqrt{2}l_{j,x} \leq L_j$ and $\sqrt{2}l_{j,y} \leq L_j$, which will guarantee that the maximum distance between two robots will be less than the cable length while navigating inside the workspace, W_j. Also, we reduce $l_{j,x}$ and $l_{j,y}$ properly to avoid overlapping with workspace of other cable-robot teams. And x_j is the left boundary of the workspace W_j which is determined by the minimum value of x coordinate of the objects in W whose y coordinate lies between y_b and $y_b + l_j$. If we assume there is alway certain distances between objects so that the robots can travel, we can always find paths that can separate this objects. And we can manipulate at least one object for each manipulation and can separate all the objects.

Coarse grid and search

To reduce the computation time, we will adapt a coarse grid map for our search algorithm (see Figure 6.20). The workspace is divided into set of cells by the rays from the reference points inside each objects, ζ_j in four

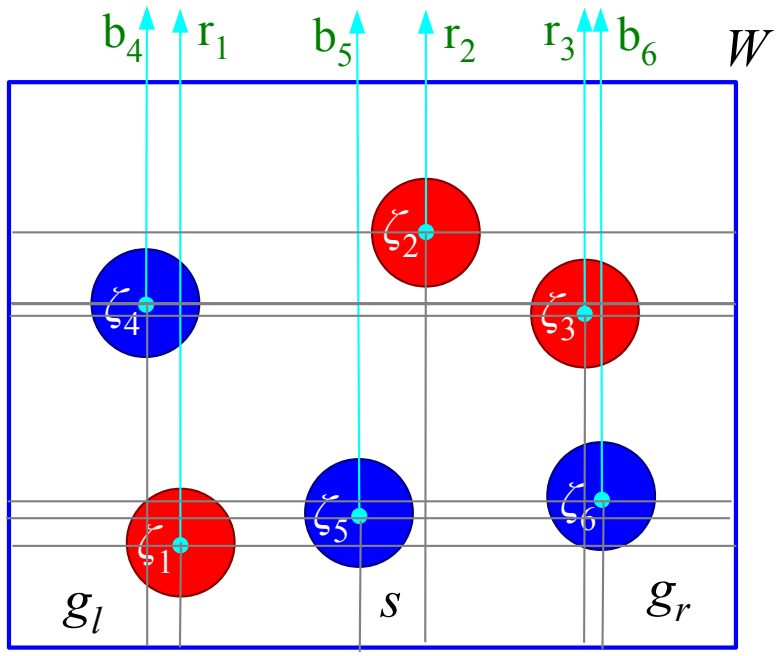


Figure 6.20: An example of coarse grid. the given workspace is split into set of cells whose boundaries are the reference rays, the cyan lines, and the grey lines. the topology class of path does not change when crossing the grey lines.

directions, $(+x, -x, +y, -y)$. And we consider 4-way connected map; the cell is connected to the neighbor cells who share edges, not a vertex. The topology class of path does not change when move to the neighbor cell through grey lines. And the topology class of the cell changes only when passing to the neighbor cell through reference rays, the cyan lines in Figure 6.20. So, we can track the change of topology of the path.

Then we drive the two robots to the cell in the middle of bottom, which is marked as s in Figure 6.20. This cell is the initial node of search. The goal of each robot will be the cells on the left and right bottom corners, g_l and g_r in Figure 6.20.

Efficient sequential manipulation

We will sequentially manipulate and transport the objects. We will find paths of two robot for each cable-robot team inside its own workspace, W_j. However, we cannot guarantee that there is always a solution that satisfies the Proposition 6.3.1. Eventhough such a path exists, it could no be optimal in some other point of view. For example, if only one object is on the upper left corner and all others are on the right bottom, it will be efficient to remain the one on the left corner for other cable-robot team or next manipulation. So, we will follow the decoupled planning algorithm described in the previous section to find the set of paths for each robot $\{\tau_j^1, \tau_j^2, \tau_j^3, \dots\}$ for $j \in \{l, r\}$ where tau_j^k start from the cells and reach the goal cell g_j. Then $h_j^k = h(\tau_j^k)$ is the homotopy class of corresponding path of the robots. Then we will find the optimal combination of the paths that maximize the efficiency of each manipulation:

$$\max_{a,b} \quad \frac{sum(H(-\tau_l^a \sqcup \tau_l^b))}{\max(c_l^a, c_r^b) + 2\Delta} \tag{6.6.2}$$

where Δ is the underestimated cost between the basket and the workspace as shown in Figure 6.19. Also, this pair of paths should be embed. And the components of $H(-\tau_l^a \sqcup \tau_l^b)$ should be 1 or 0 or one kind of object (for example *red*) and should be 0 for the other kind (for example *blue*). Then $sum(H(-\tau_l^a \sqcup \tau_l^b)$ is the number of objects (for example *red*) we can manipulate and transport by pulling the cable in $-y$ direction to the basket. And these objects have the same color or are the same kind. By maximizing the Equation (6.6.2), we can maximize the number of objects, numerator, to separate with respect to the cost of the manipulation, denominator.

6.6.2 Simulation Result

For the simulation for this large problem, we extend the dynamics simulator described in Section 6.5.2 to consider a large number of objects with multiple cable-robot teams. To reduce the computation time, we do not consider the cable-cable contacts. As each cable team will work in there own workspace, there is little possibility that cable-cable contact occurs. So we demonstrate the suggested algorithm with simulation of 100 objects (50 red objects and 50 blue objects).

The Figure 6.21 shows the simulation result with 4 cable-robot teams. Each cable robot team separates and transports the objects to its own baskets. The size of the workspace for each cable-robot team is properly reduced to avoid overlapping like Figure 6.21(i).

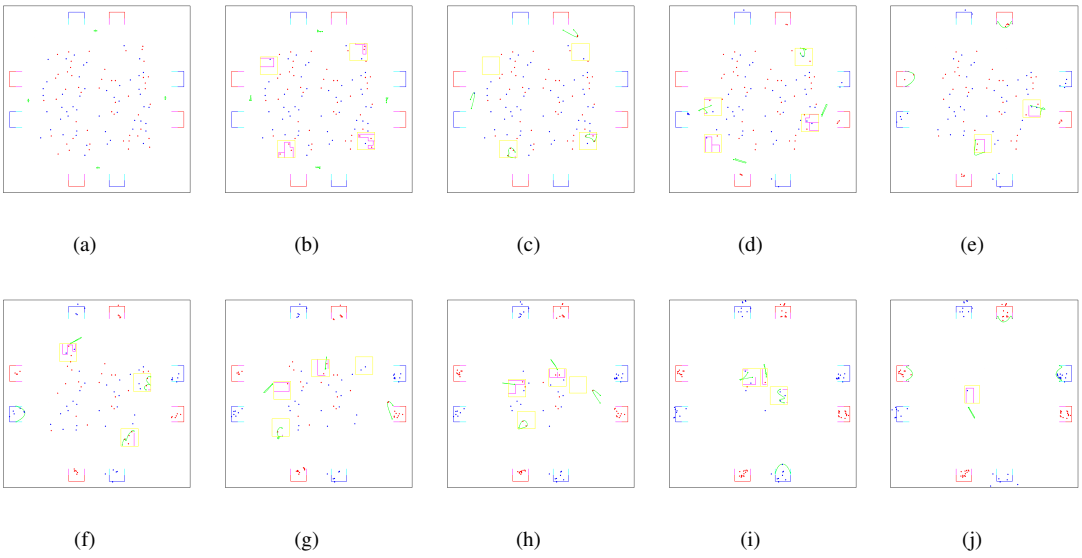


Figure 6.21: Dynamic simulation for separation of a large number of objects with multiple cable-robot teams via sequential manipulation. The red and blue dots are the objects. The green curves are the cables. The red and blue ⊔'s are the baskets. Yellow boxes are the workspace of each cable-robot team. Magenta curves are the paths of the robots. See http://youtu.be/ZHrEIo8dGDA for video.

6.7 Conclusion

In this chapter, we present a formal mathematical description of the problem of planning and control for a flexible cable towed by two robots so as to separate two types of objects in a planar environment. We develop a graph search-based implementation, and distribute the computation for efficiency. We demonstrate the working of the algorithms through simulations, and the practical applicability of the method using a complete dynamic simulation. We also extend this problem to separate a large number of objects with multiple cable-robot team via sequential manipulation and transportation. We presented various algorithms to reduce the computation time but still the algorithm is not real-time. We build the graph from fine grid of the environment. However, we can adapt other graphs like visibility graph or Voronoi tessellation to reduce the size of search space. We need to choose one which is easy and fast to build while the topology of its edges can be easily calculated. Also, to manipulate large number of objects more efficiently, we need to find more systematical method to assign workspace of each cable-robot team. Also, in real debris clean-up or oil skimming with fixed obstacles like rocks, we need to develop proper planner and controller for transporting procedure to avoid obstacles while not losing objects or oil.

Chapter 7

Conclusion

7.1 Summary

In this thesis, we have presented different trajectory generation or path planning problems considering topology classes. In Chapter 3, we presented a method to find a smooth optimal trajectory subject to geometric and kinematic constraints, and restricted to a specific topology class. In Chapter 4, we proposed an algorithm to explore an unknown or partially known environment by gradually building a topological description of the environment. Using the notion of quotient spaces, optimal paths in different topological classes leading up to the unknown region were found by searching in the H-augmented graph. We also addressed the forward and inverse kinematics of payloads carried by aerial robots. In Chapter 6, we presented a formal mathematical description of the problem of planning and control for a flexible cable towed by two robots so as to separate two types of objects in a planar environment. We also extended this problem to separate a large number of objects with multiple cable-robot team via sequential manipulation and transportation.

7.2 Main Contributions

There are three key contributions in this thesis.

The first contribution is to present optimal control problem to generate optimal trajectories under topology class constraints. As the H-signature is a homology class invariant function, the gradient would be zero almost everywhere. So, it is not a proper constraints for optimal trajectory generation problem formulated as optimal control problem. Because it is practically impossible to find analytical solution of the optimal control problems and we depends on numerical solvers to find solution. Most of the numerical solvers depends on gradients of cost function and constraints function to improve the solution. So, we cannot expect to find a solution if the gradients are zero in infeasible region of the search space. In this work, we introduced a H-signature calculated from integer variables corresponding to the intermediate point on the trajectory. To our knowledge, it is the first work to define homology class invariant function, from coarse information of the trajectory, whose gradient is not zero. By adapting this new H-signature, we formulate the optimal trajectory generation under homology class constraints as MIQP to guarantee the global optimal solution.

We also presented optimal control problem to generate optimal trajectories under homotopy class constraints. As the integer variables in MIQP provide the coarse representation of the trajectory, we can build

a set of integer variables which correspond to the trajectories in the same homotopy class. By finding the integer variables in the given homotopy class, we can find optimal trajectory under the given homotopy class constraints by solving QP while adjusting time distribution.

The second contribution of this thesis deals with how the topological constraints can be used to solve practical problems. We suggest the topological exploration algorithm based on H-signature as the alternative method of frontier-based planning.

Most of the exploration algorithms with multiple robots depend on frontier-based planning to maximize the information gain. However, we can deploy the group of robots into different paths by comparing the H-signature of paths instead of the goal frontiers. As the calculation time of H-signature is negligible and we do not need preprocessing of finding frontiers, this algorithm can work fast and efficient.

Finally, the last contribution is the manipulation using cable. The cables are widely used in robotic or mechanical systems to transport powers. However, we can also use cables to tow payload with single or multiple robots. In this thesis, we presented a stable motion primitive of straight line motion to two a payload with single or two mobile robot. Also, we discussed the kinematics of payload and aerial robots connected by cables. The unique configuration of payload can be achieved if the aerial robots satisfy the suggested conditions.

As the cable is a curve in plane or three dimensional space, it has infinite dimensions and it is practically impossible to control the cable configuration with finite number of robots. In this work, instead of control the configuration of the cable, we plan and control the homotopy class of the cable with only two robots, which are attach both ends of the cable, to manipulate and transport a set of objects.

First, we define the separating configuration as the topological constraints of the cable to manipulate and transport only one kind of objects. Then we implemented different algorithms to navigate the robots to reach the separating configuration from the initial configuration while minimizing the travel distance.

We also extend this manipulation problem to consider a large number of objects with multiple cable-robot team via sequential manipulation and transportation. We defined the efficiency of each manipulation to increase the performance of whole manipulation problem.

7.3 Future Work

There are many directions for future work. One direction of the future work is to increase the performance of the suggested algorithm. And the other direction is to extend the algorithm to solve more complicated problems.

The suggested algorithms to find optimal trajectories under topological constraints can generate anytime solution or guarantee the global optimal solution. But in practical implementation, the size of the problem is too large to find solution in real-time. So, it is important to reduce the size of the problem to reduce the computation load to make this approach practical. This improvement is one of the key direction of the future works.

In addition, we plan to extend the this optimal trajectory generation problem to three-dimensional space.

We introduced the topological exploration algorithm as an alternative criteria of Frontier-based planning. However, the current implementation is centralized and necessitates the integration and sharing of all the sensor information of all the robots. If we implement this exploration algorithm in decentralized manner, it would be robust to the failure or loss of a groups or robot, or temporary disconnection of communications.

For decentralized algorithm, we need properly sharing the map between groups of robots when they are inside the communication range. Developing proper communication protocol to share topological information is remained for future research. Also, we need to verify the decentralized algorithm with by conducting experiments with more than one robot.

One of the most intriguing direction of future work is to solve the problem of clean up surface oil or debris in the presence of fixed obstacles. In this work, we assume a simple feedback control for transportation process. However, in the presence of fixed obstacle, which cannot be removed by the skimming operation, the fixed obstacles should be avoided in the transportation process and both robots and cable should avoid these obstacles by considering the topology of the paths of the robots. Also, the robots should maintain proper distance inside the cable length not to leak any objects to transport. Finally, we need to reduce the dimension of the search space for fast planning.

The first direction of future work in this section discuss about finding optimal trajectories faster for practical problem. The heavy computation of the optimal control problem prevents us to apply this algorithm to more complicated problems. For example, we used graph-search-based planning for the topological exploration and manipulation problems in Chapter 4 and 6 to simplify the problem and reduce the computation time. So, one of the direction of future work is to implement the algorithm for these applications as optimal control problems. For real-time implementation, we need to reduce the size the optimization problem. One way to reduce the problem size it to adapt model predictive control. For example, we find the coarse path, or the integer variables, from graph-search-based planner and find smooth optimal trajectories for finite horizon.

Another direction of future work is to extend the suggested algorithm to dynamic environments. In the manipulation of a set objects, we consider all the objects are stationary. However, some objects could move actively to avoid our robots or randomly by external forces. In this work, we proposed a fast-replanning algorithm but there could be better strategy to consider dynamic environments. One good example of planning and control dynamic environments is elastic band [78], strips [22] and roadmaps [103]. Merging the suggested topological constraints into these algorithm is one of obvious and trivial ways to extend this problem with moving objects. However, when the objects are moving, we need to sense the cable configuration or estimate the homotopy class of the cable. As the objects can pass the previously executed paths of robots, we cannot just estimate the homotopy class of the cable from current object configurations and the previously executed paths of robots. Even though, the robots are stationary, the homotopy class of the cable can be changed by the movements of objects. As we estimate the homotopy class of cable only from paths of robots, it is an interesting problem to consider moving objects or change of reference rays.

One obvious and intriguing direction of future work is to manipulate and transport a set of objects with a net, a surface in three dimensional space. To extend the suggested algorithm to this three-dimentional problem, we need some mathematical framework. First, we need a topological, or any coarse, representation of a surface like homotopy or homology of cable. Then we need to define the separating configuration based on this topological, or coarse, representation as we described in this work. And the last part of this problem is to find the relation between the topology class, or coarse presentation, of surface and paths/trajectories of robots like Proposition 6.3.2. Also, we need to consider what is the minimum number of robots to control the topology class, or coarse representation, of the surface.

The last direction of future work is find and solve novel and practical problems by considering topology of robot systems. It is obvious that topology class is the coarse representation of the path/trajectory or

cable. So, we can add some abstract constraints into our path planning problems in the form of topological constraints. For example, wireless communication allows us to share information between robots without physical contact but requres the robots to stay inside certain range. However, the wireless communication will not work in extreme conditions like mission under the nuclear pollution or communication jam by other agents. In such circumstances, we need wired communication between robot-robot or robot-user for stable and robust communication. another issue is the power supply of robots. The battery is the most commner source of power for mobile or aerial robots. And the capacity of the battery keeps increasing and the size or weight of battery keeps decreasing. However, the limitation of actuation and the size of the robot prevent to install arbitrary size or number of batteries on the robot. And the wired power supply is the most reliable source of energy. So, the mission in extreme environments, it is necessary to consider the configuration of cable connecting robot-robot or robot-user in planning and control. In this case, the reachable region does not simply depend on the length of the cable but the history of the paths of robots. So, we need to plan the paths of robots considering the topology of the previously executed paths and build the map of reachable region.

Appendix A

Heuristic distance function considering the homotopy class constraints

A.1 Algorithm

In this section, we will discuss heuristic distance function to the boundary of the workspace while considering the desired homotopy class of the path. The workspace is a bounded box of $W = \{(x, y)|x_l \leq x \leq x_r,\ y_b \leq y \leq y_t\}$. In this section, we assume that all the N objects, O_j, are disk of radius R_j for $j = 1, 2, \ldots, N$. And the reference rays, r_j emits from the center of each objects $(\zeta_{j,x}, \zeta_{j,y})$ in the $+y$ direction.

The initial configuration or position is given as $q_i = (x_i, y_i) \in W$, which does not lie on any reference rays, and the desired homotopy class of the path is given as $h_d =$ "$r_{k_1}^{s_1} r_{k_2}^{s_2} \cdots r_{k_n}^{s_n}$" of length n, where $k_j \in \{1, 2, \ldots n\}$ is the index of reference ray to visit and $s_j \in \{+, -\}$ is the sign or direction of visit. Then the heuristic cost to the boundary is the sum of the cost of the segments of path sequentially visit the reference ray in order:

$$c^h(q_i) = c^h_{pr}(q_i, r_{k_1}^{s_1}) + c^h_{rr}(r_{k_1}^{s_1} r_{k_2}^{s_2}) + c^h_{rr}(r_{k_2}^{s_2} r_{k_3}^{s_3}) + \cdots + c^h_{rr}(r_{k_{n-1}}^{s_{n-1}} r_{k_n}^{s_n}) + min(c^h_{rr}(r_{k_n}^{s_n} r_l^-), c^h_{rr}(r_{k_n}^{s_n} r_r^+)). \tag{A.1.1}$$

where $c^h_{pr}(q_i, r_k^s)$ is the admissible heuristic cost from a point q to the k^{th} reference ray with sign s and $c^h_{rr}(\cdot, \cdot)$ is the admissible heuristic cost between two reference rays with proper signs. We consider the left and right boundary as reference rays emitting from (x_l, y_b) and (x_r, y_b) respectively and choose the minimum value to the boundaries. So, we need to find proper function to calculate these distances.

To calculate $c^h_{pr}(q_i, r_k^s)$, we will consider the case that the initial point lies on the left side of reference ray, $x_i \leq \zeta_{k,x}$. If $x_i > \zeta_{k,x}$, we can use the symmetry of the problem as $c^h_{pr}(q_i, r_k^s) = c^h_{pr}(q_i + 2[\zeta_{k,x} - x_i, 0]^T, r_k^{-s})$ where $-s$ means to change sign of s. While considering other reference rays as obstacles., we will consider only one reference line with minimum y_j value which lies between q_i and goal reference line:

$$\begin{aligned} \min_{o \in \{1,2,\ldots,N\}} \quad & \zeta_{o,y} \\ \text{s.t.} \quad & x_i < \zeta_{o,x} < \zeta_{k,x}. \end{aligned} \tag{A.1.2}$$

If there is no solution for the optimization problem (A.1.2), in other words, there is no reference line between q_i and the k^{th} reference ray, we consider $\zeta_{o,y} = y_t$.

Then the admissible heuristic distance function from q_i to k^{th} reference ray while avoiding the o^{th} reference ray is illustrated in Algorithm A.1.1 and Figure A.1.

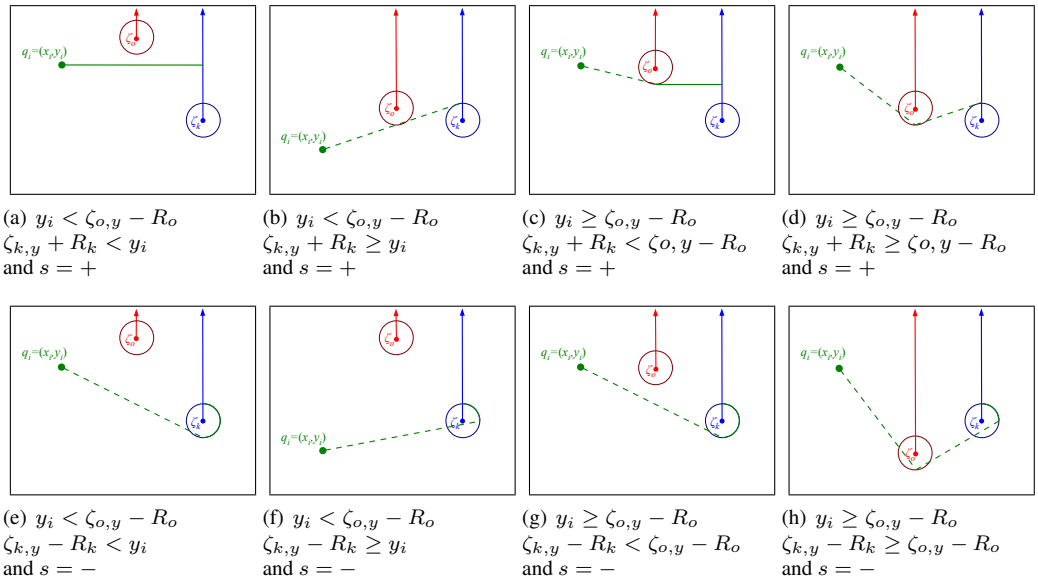
Algorithm 1.1.1 HeuristicDistanceFromPointToReferenceRay(q, ζ_o, R_o, ζ_k, R_k, s): Find heuristic distance from point q_i to the k^{th} reference ray with direction of s, while avoiding the o^{th} reference ray.

if $s == +$ **then**
  if $y_i < \zeta_{o,y} - R_o$ **then**
    if $\zeta_{k,y} + R_k < y_i$ **then**
      return $(\zeta_{k,x} - x_i)$ {see Figure A.1(a)}
    else
      return $D_h(q_i, \zeta_k + [0, R_k]^T)$ {see Figure A.1(b)}
    end if
  else
    if $\zeta_{k,y} + R_k < \zeta_{o,y} - R_o$ **then**
      return $D_h(q_i, \zeta_o - [0, R_o]^T) + (\zeta_{k,x} - \zeta_{o,x})$ {see Figure A.1(c)}
    else
      return $D_h(q_i, \zeta_o - [0, R_o]^T) + D_h(\zeta_o - [0, R_o]^T, \zeta_k + [0, R_k]^T)$ {see Figure A.1(d)}
    end if
  end if
else
  if $y_i < \zeta_{o,y} - R_o$ **then**
    if $\zeta_{k,y} - R_k < y_i$ **then**
      return $D_h(q_i, \zeta_k - [0, R_k]^T) + \pi R_k$ {see Figure A.1(e)}
    else
      return $D_h(q_i, \zeta_k + [R_k, 0]^T) + \frac{\pi}{2} R_k$ {see Figure A.1(f)}
    end if
  else
    if $\zeta_{k,y} - R_k < \zeta_{o,y} - R_o$ **then**
      return $D_h(q_i, \zeta_k - [0, R_k]^T) + \pi R_k$ {see Figure A.1(g)}
    else
      return $D_h(q_i, \zeta_o - [0, R_o]^T) + D_h(\zeta_o - [0, R_o]^T, \zeta_k + [R_k, 0]^T + \frac{\pi}{2} R_k)$ {see Figure A.1(h)}
    end if
  end if
end if

where $D_h(\cdot, \cdot)$ is the admissible heuristic distance function between two points based on how we build the connectivity of the grid. In continuous space, it will be Euclidean distance. In 8-connected graph, it will be $D_h(\mathbf{u}, \mathbf{v}) = \sqrt{2}\min(|u_x - v_x|, |u_y - v_y|) + \big||u_x - v_x| - |u_y - v_y|\big|$. In 4-connected graph, it will be $D_h(\mathbf{u}, \mathbf{v}) = |u_x - v_x| + |u_y - v_y|$.

Now we will describe the admissible heuristic distance function from the j^{th} to k^{th} reference rays while avoiding the o^{th} reference ray in Algorithm A.1.2.This Algorithm A.1.2 finds proper initial point on the j^{th} reference ray or on the j^{th} object which has the shortest distance to the k^{th} reference ray. In Algorithm A.1.2, we will only consider the case that the j^{th} reference ray lies on the left side of the k^{th} reference ray, $\zeta_{j,x} < \zeta_{k,x}$. If $\zeta_{j,x} > \zeta_{k,x}$, we can use the symmetry of the problem as $c^h_{rr}(r_j^{s_j}, r_k^{s_k}) = c^h_{rr}(r_k^{-s_k}, r_j^{-s_j})$ where $-s_i$ means to change sign of s_i.

Algorithm 1.1.2 HeuristicDistanceBetweenTwoReferenceRay(ζ_j, R_j, s_j, ζ_o, R_o, ζ_k, R_k, s_k): Find



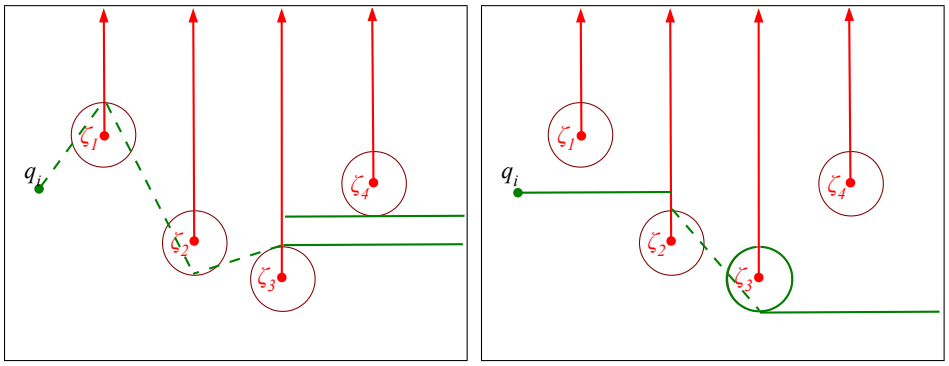
(a) $y_i < \zeta_{o,y} - R_o$ $\zeta_{k,y} + R_k < y_i$ and $s = +$
(b) $y_i < \zeta_{o,y} - R_o$ $\zeta_{k,y} + R_k \geq y_i$ and $s = +$
(c) $y_i \geq \zeta_{o,y} - R_o$ $\zeta_{k,y} + R_k < \zeta o, y - R_o$ and $s = +$
(d) $y_i \geq \zeta_{o,y} - R_o$ $\zeta_{k,y} + R_k \geq \zeta o, y - R_o$ and $s = +$
(e) $y_i < \zeta_{o,y} - R_o$ $\zeta_{k,y} - R_k < y_i$ and $s = -$
(f) $y_i < \zeta_{o,y} - R_o$ $\zeta_{k,y} - R_k \geq y_i$ and $s = -$
(g) $y_i \geq \zeta_{o,y} - R_o$ $\zeta_{k,y} - R_k < \zeta_{o,y} - R_o$ and $s = -$
(h) $y_i \geq \zeta_{o,y} - R_o$ $\zeta_{k,y} - R_k \geq \zeta_{o,y} - R_o$ and $s = -$

Figure A.1: The cost of $c^h_{pr}(q_i, r^s_k)$ is sum of the length of green lines. This Figure illustrate the case when there is no reference line between the initial configuration and goal reference line. The length of dashed green line can be replaced by proper admissible heuristic function based on graph structure.

heuristic distance from the j^{th} reference ray with direction of s_j to the k^{th} reference ray with direction of s_k, while avoiding the o^{th} reference ray.

if $j == k$ **then**
  if $s_j == s_k$ **then**
    return $2\pi R_j$
  else
    return 0
  end if
end if
if $s_j == +$ **then**
  if $\zeta_{j,y} + R_j < \zeta_{o,y} + R_o$ **then**
    if $s_k == +$ & $\zeta_{j,y} + R_j < \zeta_{k,y} + R_k$ **then**
      return $\min(c^h_{pr}([\zeta_{j,x}, \zeta_{o,y} - R_o]^T, r^{s_k}_k), c^h_{pr}(\zeta_j + [0, R_j]^T, r^{s_k}_k), c^h_{pr}([\zeta_{j,x}, \zeta_{k,y} + R_k]^T, r^{s_k}_k))$
    else
      if $s_k == -$ & $\zeta_{j,y} + R_j < \zeta_{k,y} - R_k$ **then**
        return $\min(c^h_{pr}([\zeta_{j,x}, \zeta_{o,y} - R_o]^T, r^{s_k}_k), c^h_{pr}(\zeta_j + [0, R_j]^T, r^{s_k}_k), c^h_{pr}([\zeta_{j,x}, \zeta_{k,y} - R_k]^T, r^{s_k}_k))$
      else
        return $\min(c^h_{pr}([\zeta_{j,x}, \zeta_{o,y} - R_o]^T, r^{s_k}_k), c^h_{pr}(\zeta_j + [0, R_j]^T, r^{s_k}_k))$
      end if
    end if
  else
    return $c^h_{pr}(\zeta_j + [0, R_j]^T, r^{s_k}_k))$
  end if
else
  return $\pi R_j + c^h_{pr}(\zeta_j - [0, R_j]^T, r^{s_k}_k))$
end if

We suggested the Algorithm A.1.1 and A.1.2 to calculate the admissible heuristic functions considering



(a) h_d“ $= r_1^+ r_3^+$” (b) h_d“ $= r_2^+ r_3^-$”

Figure A.2: Examples of calculation of heuristic cost function.

homotopy class of the path. However, the suggested algorithms was designed to consider the graph-search on 8-connected graph. These algorithms can be improved by considering the structure of the search space.

A.2 Examples

In this section, we will present several examples to help understanding the algorithm of the previous section. Figure A.2 shows the examples. Each example has the same initial configuration and environments. The goal is the right boundary of the workspace.

Figure A.2(a) shows the case of h_d“ $= r_1^+ r_3^+$”. Then

$$c^h(q_i) = c^h_{pr}(q_i, r_1^+) + c^h_{rr}(r_1^+ r_3^+) + c^h_{rr}(r_3^+ r_r^+).$$

Then let's find the proper case for each distance function. For $c^h_{pr}(q_i, r_1^+)$, as there is no object between q_i and the reference ray r_1, $\zeta_{o,y} = y_t$ and $R_o = 0$. So, it will be the case in Figure A.1(b). For $c^h_{rr}(r_1^+ r_3^+)$, the reference ray 2 will work as obstacle, $o = 2$. As $\zeta_{1,y} + R_1 \geq \zeta_{2,y} + R_o$, the cost will be $c^h_{pr}(\zeta_1 + [0, R_1]^T, r_3^+)$, which will be the case in Figure A.1(d). For the cost to the right edge, $c^h_{rr}(r_3^+ r_r^+)$, the reference ray r_4 works as obstacle ray but ζ_4. then this cost will be the case $c^h_{rr}(r_3^+ r_r^+) = \min(c^h_{pr}([\zeta_{3,x}, \zeta_{4,y} - R_4]^T, r_r^+), c^h_{pr}(\zeta_3 + [0, R_3]^T, r_r^+))$ but both cost function will have the same cost in this case as the solid green lines in Figure A.2(a). Then the heuristic cost of this example will be the sum of the three dashed green lines, which will be the proper heuristic cost between end points depending on the graph structure, and one of the length of the solid green line from r_3 to the right boundary.

The second example is the case of h_d“ $= r_2^+ r_3^-$” in Figure A.2(b). We have the cost function of

$$c^h(q_i) = c^h_{pr}(q_i, r_2^+) + c^h_{rr}(r_2^+ r_3^-) + c^h_{rr}(r_3^- r_r^+).$$

For $c^h_{pr}(q_i, r_2^+)$, r_1 works as obstacle ray and this cost is the case in Figure A.1(a). For $c^h_{rr}(r_2^+ r_3^-)$ there is no obstacle ray. So we have $c^h_{rr}(r_2^+ r_3^-) = c^h_{pr}(\zeta_2 + [0, R_2]^T, r_3^-)$, which is the case in Figure A.1(e). For $c^h_{rr}(r_3^- r_r^+)$, as we start from r_3 with negative sign, we have $c^h_{rr}(r_3^- r_r^+) = \pi R_3 + c^h_{pr}(\zeta_3 - [0, R_3]^T, r_r^+)$, which is the case in Figure A.1(a). So the heuristic cost of this example is sum of all the green solid

lines(including the green curves on the obstacle 3) and proper heuristic cost of the green dashed line in Figure A.2(b). Here, note that the right half of the green circle is the heuristic cost for $c^h_{rr}(r_2^+ r_3^-)$ and the left half is for $c^h_{rr}(r_3^- r_r^+)$.

Bibliography

[1] M. Anitescu. A fixed time-step approach for multibody dynamics with contact and friction. In *Proc. of the IEEE/RSJ Int. Conf. on Intelligent Robots and Systems*, pages 3725–3731, Coimbra, Portugal, October 2003.

[2] M. Anitescu and F. A. Potra. A time-stepping method for stiff multibody dynamics with contact and friction. *International Journal of Numerical Methods Engineering*, pages 753–784, July 2002.

[3] Mihai Anitescu and Gary D. Hart. A fixed-point iteration approach for multibody dynamics with contact and small friction. *Mathematical Programming*, 101(1):3–32, 2004.

[4] Joaqun Aranda, Pablo Gonzlez de Santos, and Jess Manuel de la Cruz. *Robotics and Automation in the maritime industries*. Produccin Grfica Multimedia (PGM), 2006.

[5] Devin Balkcom and Matthew T. Mason. Geometric construction of time optimal trajectories for differential drive robots. In *Proceedings of the Workshop on the Algorithmic Foundations of Robotics (WAFR '00)*, 2000.

[6] Devin J. Balkcom. Geometric construction of time optimal trajectories for differential drive robots. In *Fourth Workshop on Algorithmic Foundations of Robotics*, pages 1–13, 2000.

[7] M.A. Benayad, G. Campion, V. Wertz, and M.E. Achhab. Steering a mobile robot: Selection of a velocity profile satisfying dynamical constraints. *Asian Journal of Control*, 2(4):219–229, 2000.

[8] Spring Berman. *Abstractions, Analysis Techniques, and Synthesis of Scalable Control Strategies for Robot Swarms*. PhD thesis, University of Pennsylvania, 2010.

[9] Subhrajit Bhattacharya. *Topological and Geometric Techniques in Graph-Search Based Robot Planning*. PhD thesis, University of Pennsylvania, January 2012.

[10] Subhrajit Bhattacharya. *Topological and Geometric Techniques in Graph-Search Based Robot Planning*. PhD thesis, University of Pennsylvania, January 2012.

[11] Subhrajit Bhattacharya, Robert Ghrist, and Vijay Kumar. Multi-robot coverage and exploration in non-euclidean metric spaces. In *Proceedings of The Tenth International Workshop on the Algorithmic Foundations of Robotics*, 13-15 June 2012.

[12] Subhrajit Bhattacharya, Hordur Heidarsson, Gaurav S. Sukhatme, and Vijay Kumar. Cooperative control of autonomous surface vehicles for oil skimming and cleanup. In *Proceedings of IEEE International Conference on Robotics and Automation (ICRA)*, 9-13 May 2011.

[13] Subhrajit Bhattacharya, Vijay Kumar, and Maxim Likhachev. Search-based path planning with homotopy class constraints. In *AAAI Conf. on Artificial Intelligence*, July 2010.

[14] Subhrajit Bhattacharya, Maxim Likhachev, and Vijay Kumar. Identification and representation of homotopy classes of trajectories for search-based path planning in 3D. In *Robotics: Science and Systems*, June 2011.

[15] Subhrajit Bhattacharya, Maxim Likhachev, and Vijay Kumar. Topological constraints in search-based robot path planning. *Autonomous Robots*, 33(3):273–290, October 2012. doi: 10.1007/s10514-012-9304-1.

[16] Subhrajit Bhattacharya, David Lipsky, Robert Ghrist, and Vijay Kumar. Invariants for homology classes with application to optimal search and planning problem in robotics. *Electronic pre-print*, Aug 2012. http://arxiv.org/abs/1208.0573 [math.AT].

[17] Subhrajit Bhattacharya, Nathan Michael, and Vijay Kumar. Distributed coverage and exploration in unknown non-convex environments. In *Proceedings of 10th International Symposium on Distributed Autonomous Robotics Systems*. Springer, 1-3 Nov 2010.

[18] D.A. Binder. Approximations to bayesian clustering rules. *Biometrika*, 68:275–285, 1981.

[19] J. E. Bobrow, B. Martin, G. Sohl, E. C. Wang, F. C. Park, and Junggon Kim. Optimal robot motions for physical criteria. *J. of Robotic Systems*, 18:2001, 2001.

[20] P. Bosscher and I. Ebert-Uphoff. Wrench based analysis of cable-driven robots. In *Proceedings of the IEEE International Conference on Robotics and Automation*, pages 4950–4955, April 2004.

[21] R. Bott and L.W. Tu. *Differential Forms in Algebraic Topology*. Graduate texts in mathematics. Springer-Verlag, 1982.

[22] Oliver Brock and Oussama Khatib. Elastic strips: A framework for motion generation in human environments. *The International Journal of Robotics Research*, 21(12):1031–1052, 2002.

[23] P. Cheng, J. Fink, and V. Kumar. Cooperative towing with multiple robots. *ASME Transactions: Journal of Mechanisms and Robotics*, 1, February 2009.

[24] Peng Cheng, Jonathan Fink, Soonkyum Kim, and Vijay Kumar. Cooperative towing with multiple robots. In Gregory Chirikjian, Howie Choset, Marco Morales, and Todd Murphey, editors, *Algorithmic Foundation of Robotics VIII*, volume 57 of *Springer Tracts in Advanced Robotics*, pages 101–116. Springer Berlin / Heidelberg, 2009.

[25] T. H. Cormen, C. E. Leiserson, R. L. Rivest, and C. Stein. *Introduction to algorithms*. MIT Press, 2nd edition, 2001.

[26] R. Cottle, J.S. Pang, and R.E. Stone. *The linear complementarity problem*. Classics in applied mathematics. Society for Industrial and Applied Mathematics (SIAM, 3600 Market Street, Floor 6, Philadelphia, PA 19104), 1992.

[27] Douglas Demyen and Michael Buro. Efficient triangulation-based pathfinding. In *National Conf. on Artificial Intelligence*, pages 942–947, 2006.

[28] M Bernardine Dias, Robert Michael Zlot, Nidhi Kalra, and Anthony (Tony) Stentz. Market-based multirobot coordination: A survey and analysis. Technical Report CMU-RI-TR-05-13, Robotics Institute, Pittsburgh, PA, April 2005.

[29] Edsger W. Dijkstra. A note on two problems in connexion with graphs. *Numerische Mathematik*, 1:269–271, 1959.

[30] Mehmet Dogar, Kaijen Hsiao, Matei Ciocarlie, and Siddhartha Srinivasa. Physics-based grasp planning through clutter. In *Robotics: Science and Systems VIII*, July 2012.

[31] B. R. Donald. On information invariants in robotics. *Artificial Intelligence Journal*, 72:217–304, 1995.

[32] B. R. Donald. Information invariants for distributed manipulation. *International Journal of Robotics Research*, 16, 1997.

[33] B. R. Donald, L. Gariepy, and D. Rus. Distributed manipulation of multiple objects using ropes. In *Proceedings IEEE International Conference on Robotics & Automation*, 2000.

[34] Bruce Donald, Larry Gariepy, and Daniela Rus. Distributed manipulation of multiple objects using ropes. In *In IEEE International Conference on Robotics and Automation*, pages 450–457, 2000.

[35] Bruce Donald, Patrick Xavier, John Canny, and John Reif. Kinodynamic motion planning. *J. of ACM*, 40(5):1048–1066, November 1993.

[36] P. S. Donelan and C. P. Scott. Real inflections of hinged planar four-bar coupler curves. *Mechanism and Machine Theory*, 30(8):1179–1191, November 1995.

[37] L. E. Dubins. On curves of minimal length with a constraint on average curvature, and with prescribed initial and terminal positions and tangents. *American Journal of Mathematics*, 79(3):pp. 497–516, 1957.

[38] Paul L. Fackler and Mario J. Miranda. Lemke: Solver for standard linear complementarity problems (lcps), 2002.

[39] Jonathan Fink, M. Ani Hsieh, and Vijay Kumar. Multi-robot manipulation via caging in environments with obstacles. In *IEEE International Conference on Robotics and Automation (ICRA)*, Pasedena, CA, May 2008.

[40] Jonathan Fink, Nathan Michael, Soonkyum Kim, and Vijay Kumar. Planning and control for cooperative manipulation and transportation with aerial robots. *I. J. Robotic Res.*, 30(3):324–334, 2011.

[41] D. Fox, J. Ko, K. Konolige, B. Limketkai, D. Schulz, and B. Stewart. Distributed multirobot exploration and mapping. *Proceedings of the IEEE*, 94(7):1325 –1339, july 2006.

[42] Robert W. Fox, Alan T. McDonald, and Philip J. Pritchard. *Introduction to Fluid Mechanics*. Wiley, 2005.

[43] Theodore W. Gamelin. *Complex analysis*. Springer Science, 2001.

[44] Brian P. Gerkey. amcl ros package. http://www.ros.org/wiki/amcl.

[45] P.G. Goerss and J.F. Jardine. *Simplicial Homotopy Theory*. Progress In Mathematics. Birkhäuser, 1999.

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

[47] Yi Guo and Lynne E. Parker. A distributed and optimal motion planning approach for multiple mobile robots. In *In Proceedings of IEEE International Conference on Robotics and Automation*, pages 2612–2619, 2002.

[48] P. E. Hart, N. J. Nilsson, and B. Raphael. A formal basis for the heuristic determination of minimum cost paths. *IEEE Transactions on Systems, Science, and Cybernetics*, SSC-4(2):100–107, 1968.

[49] Allen Hatcher. *Algebraic Topology*. Cambridge Univ. Press, 2001.

[50] J. Hershberger and J. Snoeyink. Computing minimum length paths of a given homotopy class. *Comp. Geom. Theory and Applications*, 4:331–342, 1991.

[51] K.H. Hunt. *Kinematic Geometry of Mechanisms*. Oxford University Press, 1978.

[52] International Business Machines Corporation. *IBM ILOG CPLEX V12.1: User's Manual for CPLEX*, 2009.

[53] Qimi Jiang and Vijay Kumar. The inverse kinematics of 3-d towing. *Advances in Robot Kinematics: Motion in Man and Machine*, pages 321–328, 2010.

[54] Richard A. Kerr. A lot of oil on the loose, not so much to be found. *Science*, 329(734), 2010.

[55] Soonkyum Kim, Subhrajit Bhattacharya, Robert Ghrist, and Vijay Kumar. Topological exploration of unknown and partially known environments. In *Proceedings of IEEE International Conference on Intelligent Robots and Systems*, Tokyo, Japan, Aov 3-8 2013.

[56] Soonkyum Kim, Subhrajit Bhattacharya, Hordur Heidarsson, Gaurav Sukhatme, and Vijay Kumar. A topological approach to using cables to separate and manipulate sets of objects. In *Proceedings of the Robotics: Science and System (RSS)*, June 24-28 2013.

[57] Soonkyum Kim, Subhrajit Bhattacharya, and Vijay Kumar. Dynamic simulation of autonomous boats for cooperative skimming and cleanup. In *Proceedings of the ASME International Design Technical Conferences and Computer and Information in Engineering Conference*, Portland, OREGON, Aug 4-7 2013.

[58] Soonkyum Kim, Koushil Sreenath, Subhrajit Bhattacharya, and Vijay Kumar. Optimal trajectory generation under homology class constraints. In *51st IEEE Conference on Decision and Control*, 10-13 Dec 2012.

[59] Soonkyum Kim, Koushil Sreenath, Subhrajit Bhattacharya, and Vijay Kumar. Trajectory planning for systems with homotopy class constraints. In *13th International Symposium on Advances in Robot Kinematics (ARK)*, pages 83–90, Innsbruck, Austria, jun 2012. Springer, Netherlands.

[60] F. Lamiraux and L. E. Kavraki. Planning paths for elastic objects under manipulation constraints. *International Journal of Robotics Research*, 20(3):188–208, 2001.

[61] Steven M. Lavalle and James J. Kuffner. Rapidly-exploring random trees: Progress and prospects. In B. R. Donald, K. M. Lynch, and D. Rus, editors, *Algorithmic and Computational Robotics: New Directions*, pages 293–308. A K Peters, 2001.

[62] H. Lipkin and J. Duffy. The elliptic polarity of screws. *Trans. ASME, Journal of Mechanisms, Transmissions and Automation in Design*, 107(3):377–386, September 1985.

[63] Matthew T. Mason. *Mechanics of robotic manipulation*. MIT Press, 2001.

[64] D. Mellinger and V. Kumar. Minimum snap trajectory generation and control for quadrotors. In *Proceedings of the IEEE International Conference on Robotics and Automation (ICRA)*, May 2011.

[65] N. Michael, J. Fink, and V. Kumar. Experimental testbed for large multi-robot teams: Verification and validation. *ram*, 15(1):53–61, March 2008.

[66] N. Michael, J. Fink, and V. Kumar. Cooperative manipulation and transportation with aerial robots. In *Robotics: Science and Systems*, Seattle, WA, June 2009. Submitted.

[67] N. Michael, S. Kim, J. Fink, and V. Kumar. Kinematics and statics of cooperative multi-robot aerial manipulation with cables. In *ASME Int. Design Engineering Technical Conf. Computers and Information in Engineering Conf.*, San Diego, CA, August 2009. To Appear.

[68] Mark B. Milam, Kudah Mushambi, and Richard M. Murray. A new computational approach to real-time trajectory generation for constrained mechanical systems. In *IN IEEE CONFERENCE ON DECISION AND CONTROL*, pages 845–851, 2000.

[69] James Munkres. *Topology*. Prentice Hall, 1999.

[70] R. M. Murray. Trajectory generation for a towed cable system using differential flatness. In *IFAC World Congress*, San Francisco, CA, July 1996.

[71] Richard Murray and S. Shankar Sastry. Nonholonomic motion planning: Steering using sinusoids. *IEEE Transactions on Automatic Control*, 38:700–716, 1993.

[72] Richard M. Murray, Muruhan Rathinam, and Willem Sluis. Differential flatness of mechanical control systems: A catalog of prototype systems. In *Proceedings of the 1995 ASME International Congress and Exposition*, 1995.

[73] S. R. Oh and S. K. Agrawal. Cable suspended planar robots with redundant cables: controllers with positive tensions. *IEEE Transactions on Robotics*, 21(3):457–465, June 2005.

[74] S. R. Oh and S. K. Agrawal. A control lyapunov approach for feedback control of cable-suspended robots. In *Proc. of the IEEE Int. Conf. on Robotics and Automation*, pages 4544–4549, Rome, Italy, April 2007.

[75] S.R. Oh and S.K. Agrawal. Cable-suspended planar parallel robots with redundant cables: Controllers with positive cable tensions. In *Proc. of IEEE International Conference on Robotics and Automation*, pages 3023–3028, 2003.

[76] J. Phillips. *Freedom in Machinery: Volume 1*. Cambridge University Press, 1990.

[77] Richard Pito. A solution to the next best view problem for automated surface acquisition. *IEEE Trans. Pattern Anal. Mach. Intell.*, 21(10):1016–1030, October 1999.

[78] S. Quinlan and O. Khatib. Elastic bands: connecting path planning and control. In *Robotics and Automation, 1993. Proceedings., 1993 IEEE International Conference on*, pages 802–807 vol.2, 1993.

[79] J. A. Reeds and L. A. Shepp. Optimal paths for a car that goes both forwards and backwards. *Pacific Journal of Mathematics*, 145(2):367–393, 1990.

[80] Arthur Richards and Jonathan P. How. Aircraft trajectory planning with collision avoidance using mixed integer linear programming. In *American Control Conf.*, volume 3, pages 1936–1941, May 2002.

[81] Campbell Robertson and Clifford Krauss. Gulf spill is the largest of its kind, scientists say. *The New York Times*, August 2010.

[82] Mitul Saha, Pekka Isto, and Jean claude Latombe. Motion planning for robotic manipulation of deformable linear objects. In *in Proc. IEEE Int. Conf. Robot. Autom*, pages 2478–2484, 2006.

[83] E. Schmitzberger, J. L. Bouchet, M. Dufaut, D. Wolf, and R. Husson. Capture of homotopy classes with probabilistic road map. In *IEEE/RSJ Int. Conf. on Intelligent Robots and Systems*, pages 2317–2322, October 2002.

[84] T. Schouwenaars, B. De Moor, E. Feron, and J. How. Mixed integer programming for multi-vehicle path planning. In *European Control Conference*, pages 2603–2608. Citeseer, 2001.

[85] M. Schwager, P. Dames, D. Rus, and V. Kumar. A multi-robot control policy for information gathering in the presence of unknown hazards. In *International Symposium on Robotics Research*, Aug. 2011.

[86] W.R. Scott and W.R. Scott. *Group Theory*. Dover Books on Mathematics Series. Dover Publ., 1964.

[87] C. Stachniss. *Exploration and Mapping with Mobile Robots*. PhD thesis, University of Freiburg, Freiburg, Germany, April 2006.

[88] C. Stachniss. *Robotic Mapping and Exploration*. Springer Tracts in Advanced Robotics. Springer, 2009.

[89] C. Stachniss, G. Grisetti, and W. Burgard. Information gain-based exploration using rao-blackwellized particle filters. In *Robotics: Science and Systems*, pages 65–72, Cambridge, MA, June 2005.

[90] R.P. Stanley and G.C. Rota. *Enumerative Combinatorics:*. Cambridge studies in advanced mathematics. Cambridge University Press, 2000.

[91] David Stewart and J. C. Trinkle. An implicit time-stepping scheme for rigid body dynamics with coulomb friction. *International Journal Of Numerical Methods In Engineering*, 39:2673–2691, 1996.

[92] E. Stump and V. Kumar. Workspaces of cable-actuated parallel manipulators. *ASME Journal of Mechanical Design*, 128(1):159–167, January 2006.

[93] J. A. K. Suykens and J. P. L. Vandewalle. Least squares support vector machine classifiers. *Neural Processing Letters*, 9(3):293–300, Jun 1999.

[94] Russ Tedrake, Ian R. Manchester, Mark M. Tobenkin, and John W. Roberts. LQR-trees: Feedback motion planning via sums of square verification. *The Int. J. of Robotics Research*, 29(8):1038–1052, July 2010.

[95] S. Thrun, W. Burgard, and D. Fox. *Probabilistic Robotics (Intelligent Robotics and Autonomous Agents)*. The MIT Press, 2005.

[96] B. Tovar, F. Cohen, L. Bobadilla, J. Czarnowski, and S. M. LaValle. Combinatorial filters: Sensor beams, obstacles, and possible paths. *ACM Transactions on Sensor Networks*, 2012. Under review.

[97] Benjamin Tovar, Fred Cohen, and Steven LaValle. Sensor beams, obstacles, and possible paths. In Gregory Chirikjian, Howie Choset, Marco Morales, and Todd Murphey, editors, *Algorithmic Foundation of Robotics VIII*, volume 57 of *Springer Tracts in Advanced Robotics*, pages 317–332. Springer Berlin / Heidelberg, 2009.

[98] Benjamn Tovar, Fred Cohen, and Steven M. LaValle. Sensor beams, obstacles, and possible paths. In *Workshop on the Algorithmic Foundations of Robotics*, pages 317–332, 2008.

[99] R. Verhoeven. *Analysis of the Workspace of Tendon-based Stewart Platforms*. PhD thesis, University Duisburg-Essen, Essen, Germany, July 2004.

[100] Arnoud Visser and Bayu Slamet. Balancing the information gain against the movement cost for multi-robot frontier exploration. In Herman Bruyninckx, Libor Preucil, and Miroslav Kulich, editors, *Second European Robotics Symposium 2008, EUROS 2008, Prague, Czech Republic*, volume 44 of *Springer Tracts in Advanced Robotics*, pages 43–52. Springer, 2008.

[101] Frank M. White. *Fluid mechanics*. McGraw-Hill, 2008.

[102] B. Yamauchi. A frontier-based approach for autonomous exploration. In *Computational Intelligence in Robotics and Automation, 1997. CIRA'97., Proceedings., 1997 IEEE International Symposium on*, pages 146 –151, jul 1997.

[103] Yuandong Yang and Oliver Brock. Elastic roadmapsmotion generation for autonomous mobile manipulation. *Autonomous Robots*, 28(1):113–130, 2010.

[104] Dmitry Zarubin, Vladimir Ivan, Marc Toussaint, Taku Komura, and Sethu Vijayakumar. Heirachical motion planning in topological representations. In *Proc. Robotics: Science and Systems (RSS)*, Sydney, Australia, 2012.

[105] H. Zhang, V. Kumar, and J. Ostrowski. Motion planning under uncertainty. In *IEEE International Conference on Robotics and Automation*, Leuven, Belgium, May 16-21 1998. This paper was nominated for the best paper award at the 1998 IEEE International Conference on Robotics and Automation.