INTERACTIONS IN MULTI-ROBOT SYSTEMS

A Dissertation
Presented to
The Academic Faculty

By

Yancy J. Diaz-Mercado

In Partial Fulfillment
of the Requirements for the Degree
Doctor of Philosophy in
Electrical and Computer Engineering

School of Electrical and Computer Engineering
Georgia Institute of Technology
May 2016

INTERACTIONS IN MULTI-ROBOT SYSTEMS

Approved by:

Date Approved: March 30, 2016

To my wife and parents. Thank you for your support.

ACKNOWLEDGMENT

As the African proverb states that "it takes a village to raise a child," the work in this dissertation was only possible due to the support of many individuals and the fruitful discussions and collaborations that we held.

I want to first acknowledge my research advisor, Dr. Magnus Egerstedt. During my PhD studies he was more than an advisor – he was a mentor and role model. He taught me as much about doing research and the state-of-the-art as he taught me how to carry myself professionally, teach, mentor others, and have fun doing what I do.

Many of the results presented here were achieved thanks to collaboration with brilliant individuals who were gracious enough to share their research with me, and which led us to achieve more than the sum of the parts. Jun Lu, who was a PhD student in the math department, for allowing me to implement his beautiful stochastic difference equations on our Quadcopters. Austin Jones, who was a momentarily a visiting PhD student and later a post-doc at our lab, for bringing his formal methods expertise to my motion planning framework and enabling perhaps the most interesting and extendable component of it. And my good friend Sung Gun Lee, who pioneered our first results on coverage of time-varying densities and allowed me to test my human-swarm interactions ideas using these results.

I want to thank the members of our lab, the Georgia Robotics and InTelligent Systems Laboratory (GRITS Lab), for the many fruitful discussions through the years. I especially appreciate the conversations I maintained with Thiagarajan Ramachandran, Daniel Pickem, and Matt Hale, which often helped me out of mental roadblocks. I also want to thank the more senior members who served as an inspiration and role models earlier in my studies: Smriti Chopra, J.-P. de la Croix, Greg Droge, and Philip Twu.

Last, but most definitely not least, I want to thank my parents, Jose Diaz and Nilda Mercado, for always believing in me, and Liz Bricel, my wife and world, for her constant and unyielding support through this arduous process.

TABLE OF CONTENTS

ACKNOWLEDGMENT iv

LIST OF TABLES vi

LIST OF FIGURES vii

SUMMARY ix

CHAPTER 1 INTRODUCTION 1

CHAPTER 2 LITERATURE REVIEW 4

2.1 Multi-Robot Systems and Applications 4
2.2 Human-Swarm Interactions & Coverage Control 6
2.3 Multi-Robot Motion Planning 8

CHAPTER 3 HUMAN-SWARM INTERACTIONS 12

3.1 The Coverage Problem 14
3.2 Centralized Coverage of Time-Varying Densities 18
3.3 Distributed Coverage of Time-Varying Densities 23
3.4 Designing Density Functions 28
3.5 Robotic Experiments 32
3.6 Concluding Remarks 37

CHAPTER 4 MULTI-ROBOT MOTION PLANNING 38

4.1 Inter-Robot Interactions in Multi-Robot Systems Using Braids 38
4.2 Braid Controllers 44
4.3 Formal Synthesis of Braid Strings 57
4.4 Non-Rectangular Regions 65
4.5 Implementing Braids 70
4.6 Concluding Remarks 78

CHAPTER 5 ROBOT NAVIGATION 79

5.1 Shortest Path through 3D Cluttered Environments 79
5.2 Trajectory Generation from Reference Paths 89

CHAPTER 6 CONCLUSIONS 98

REFERENCES 100

LIST OF TABLES

Table 1 Complexity comparison to other algorithms 87

Table 2 Constraint compliance during Monte Carlo runs 96

LIST OF FIGURES

Figure 1 Human-swarm interactions via coverage control 14

Figure 2 Graphical user interface used to generate density shapes 30

Figure 3 Approximation of desired density shape 31

Figure 4 User observes position of swarm and redirects it with the use of a tablet. . 33

Figure 5 Robotic implementation: coverage control using Gaussian functions . . . 34

Figure 6 GRITSBots influenced by the control of Gaussian functions HSI scheme 35

Figure 7 Robotic implementation: coverage of drawn densities 36

Figure 8 Two-robot interactions and corresponding symbolic braids 41

Figure 9 Concatenated braids generators 42

Figure 10 Geometric interpretation of braid string 44

Figure 11 Hybrid STOP-GO-STOP braid controller 48

Figure 12 Lower bound on the mixing limit using the Stop-Go-Stop braid controller 49

Figure 13 Braid string $\sigma = \sigma_1 \cdot \sigma_3 \cdot \sigma_2$ taking place in varying number of braid steps 52

Figure 14 Safety separation region for three different geometry braids 54

Figure 15 Upper bound on the mixing limit 57

Figure 16 Example product automaton 63

Figure 17 Shortest braid satisfying example specification 65

Figure 18 Agents braiding on a region that curves 66

Figure 19 Rectangular and non-rectangular regions 67

Figure 20 Agents executing a braid over curved region 68

Figure 21 Robots executing an inter-agent interaction strategy 77

Figure 22 Braid controller robotic implementation data 77

Figure 23 A robot's footprint is approximated by a ball of radius r 81

Figure 24 Shortest path with obstacles example 81

Figure 25 Illustration of geodesic tangent direction 83

Figure 26 Step function example 84

Figure 27 Three shortest and the longest paths in an environment with obstacles . . 87

Figure 28 Shortest reference and actual robot's paths 88

Figure 29 Robotic implementation of shortest paths in cluttered environments . . . 88

Figure 30 Tube representation of RNP constraints 90

Figure 31 ERLIN ONE arrival navigation chart 95

Figure 32 Trajectories obtained from Monte Carlo simulations 97

SUMMARY

The objective of this research is to develop a framework for multi-robot coordination and control with emphasis on human-swarm interactions and inter-agent interactions. We focus on two problems: in the first we address how to enable a single human operator to externally influence large teams of robots. By directly imposing density functions on the environment, the user is able to abstract away the size of the swarm and manipulate it as a whole, e.g., to achieve specified geometric configurations, or to maneuver it around. In order to pursue this approach, contributions are made to the problem of coverage of time-varying density functions.

In the second problem, we address the characterization of inter-agent interactions and enforcement of desired interaction patterns in a provably safe (i.e., collision free) manner, e.g., for achieving rich motion patterns in a shared space, or for mixing of sensor information. We use elements of the braid group, which allows us to symbolically characterize classes of interaction patterns. We further construct a new specification language that allows us to provide rich, temporally-layered specifications to the multi-robot mixing framework, and present algorithms that significantly reduce the search space of specification-satisfying symbols with exactness guarantees. We also synthesize provably safe controllers that generate and track trajectories to satisfy these symbolic inputs. These controllers allow us to find bounds on the amount of safe interactions that can be achieved in a given bounded domain.

CHAPTER 1
INTRODUCTION

The objective of the research in this dissertation is to develop a framework for multi-robot coordination and motion planning with emphasis on human-swarm and inter-agent interactions. We focus on two problems: in the first we address how to enable a single human operator to externally influence an arbitrarily-sized team of robots, e.g., so as to achieve a specified geometric configuration, or to maneuver the swarm as a whole. In the second problem we focus on the characterization of inter-agent interactions and enforcement of desired interaction patterns in a provably safe (i.e., collision free) manner, e.g., for achieving rich motion patterns in a shared space, or to exchange sensor information.

There are a number of applications and scenarios that motivate this work in control of multi-robot systems. In Section 2.1, we present a number of multi-robot applications that have been explored in the literature during the last few decades. Many of these applications and technological trends have in common is the need for some form of coordination among the agents in the multi-robot team. Further, in many of these situations there is a need for adapting to changes in the environment or the mission parameters. Although robots can be good at low-level control objectives (e.g., stabilization, formation maintaining, tracking), they lack the capacity humans possess of making high-level decisions on-the-go.

The *human-swarm interactions* (HSI) paradigm allows for a human operator to influence a swarm of robots at a high-level while the swarm continues their operations and low-level control. This allows for a combination of high-level and low-level decision making. Human operators can for example: switch swarm behavior, change overall location, or inject new objectives into the robot swarm low-level control. Section 2.2 provides a representative sample of the literature on human-swarm interactions and coverage control, the mechanism that allows us to formulate a human-swarm interactions scheme. In Section 2.3, we discuss motion planning for multi-robot systems and the general approaches found in the literature. Crucial to the development of the research in this dissertation will be the notions of symbolic approaches to motion planning and *braids in robotics*. The literature on these is also found in Section 2.3.

In Chapter 3, we present a novel approach of influencing a team of robots using coverage algorithms for general time-varying density functions. This has potential implications for how human operators can interact with large teams of mobile robots, where one of the main challenges is the construction of suitable abstractions that make the entire team amenable to human control. For such abstractions to be useful, they need to scale gracefully as the number of robots increases. Density functions are promising abstractions as they are independent of the swarm's size and are instead directly affect the environment where the robot swarm operates. We construct methods for HSI to influence teams of robots tailored for obtaining desired geometric configurations and for quick manipulation of the swarm as a whole. These methods are tested through robotic implementation.

As agents in the team of robots realize rich motion patterns in a shared space, collision avoidance begins to play a bigger role. In certain applications there is also a need to satisfy desired levels of interaction between agents, e.g., for coordinated sensing, information exchange, collaborative task completion. In Chapter 4, we focus on the characterization of inter-agent interactions for the sake of multi-robot motion planning. By introducing hierarchical levels of abstractions, we are able to enforce desired interaction patterns in a provably safe (i.e., collision free) manner while conforming with the provided specifications. We first take the problem from the high-level specifications to a symbolic representation of interaction and motion patterns. In particular, we characterize interaction patterns through elements of the so-called braid group. This allows us to not focus on a particular pattern *per se*, but rather on the problem of being able to execute whole classes of patterns. These symbols become an input to motion planners, coined *braid controllers*, which generate specification conforming trajectories for the multi-robot system. The result from such a construction is a hybrid system driven by symbolic inputs that must be mapped onto actual paths that both realize the desired interaction levels and remain safe in the sense that collisions are avoided. In order to represent meaningful motion tasks in terms of interactions between agents and achieved configurations, we define a novel specification language, called Braid Temporal Logic (BTL), that allows us to specify rich, temporally-layered tasks involving agents' locations in an environment, their relative ordering, and frequency of location swaps. We use techniques from formal methods to generate symbolic inputs that conform to a given BTL specification and use the developed hybrid optimal control synthesis techniques to safely enact the synthesized pattern.

The main contributions of this work are found in Chapters 3 and 4. In addition to these, Chapter 5 presents results on robot navigation and control. In particular, contributions are made on two fronts. In Section 5.1, we present an algorithm that is implementable online to find the (globally) shortest path connecting two points in a three-dimensional cluttered environment. This is achieved by leveraging recent results on intermittent diffusion that allow us to inject sufficient noise in fast-converging local solvers to "kick" them out of locally short paths and onto the globally shortest path with a specifiable probability. A brief complexity analysis and comparison is also provided, as well as data for an online implementation. In Section 5.2, we discuss optimal trajectory generation to follow a reference path while satisfying required navigation performance criteria and required times of arrival. Using an optimal control framework, we are able to perform a reparameterization of the given reference path to accommodate the system dynamics and optimize criteria such as fuel consumption, deviations from the reference path, and arrival at spatio-temporal constraints. This work was performed in an effort to facilitate concepts for next generation air traffic control systems. Simulation results are shown for a nonlinear, six degrees-of-freedom aerodynamical model based on an arrival segment at the Hartsfield Jackson international airport. Monte Carlo simulations demonstrate the robustness of the approach under weather forecast uncertainty.

CHAPTER 2
LITERATURE REVIEW

This research in this dissertation explores the topic of human-swarm interactions where a single operator is able to externally influence a large team of robots. To characterize and enforce interactions between agents that arise from rich movement patterns in a shared space, a framework is also developed for multi-robot motion planning based on symbolic inputs. In this chapter, we provide a representative sample of the existing literature in human-swarm interactions, coverage control, and motion planning of multi-robot systems. We further identify where the research in this dissertation makes contributions to the existing body of work in the literature.

2.1 Multi-Robot Systems and Applications

Multi-agent and swarm robotics has received increased research interest in the last few decades. There are a number of benefits for considering the use of multi-robot systems: incorporating redundancy into a system for increased robustness [1, 2], task and resource allocation [3], and complementing heterogeneous capabilities [4, 5]. On the other hand, increasing the number of agents can lead to problems with the scalability of algorithms [6, 7], emergence of unaccounted artifacts in local rules at a global level (e.g., deadlocks [8], local optima and local traps [9]), and considerations on coordination and collision avoidance [9, 10].

A number of applications have been approached from a multi-robot system perspective. In the multi-robot foraging paradigm, agents wander around an environment searching for items of interest [11,12]. These items are then collected and returned to specified locations. Foraging is of interest from a biology standpoint, as many species forage for food (e.g., ants [13], birds [14]). However, the foraging task can also relate to many real-world problems, such as waste or specimen collection in hazardous environments, explosive detection and disposal, and search and rescue operations after a crisis such as floods and earthquakes.

Another application of interest in the research community and in industry is cooperative assembly [15, 16] and self-assembly [17, 18]. In the cooperative assembly scenario, multiple robots need to coordinate their motion in order to cooperatively assemble a structure using a possibly heterogeneous set of tools, end effectors, skills, or parts. Similarly, in the self-assembly scenario, multiple robots need to coordinate their motion in order to collectively form a structure or achieve a geometric configuration. In either scenario, kinematic and other constraints, such as skill or tool compatibility, play an important role in the multi-agent coordination. Real-world applications include the automotive assembly industry, space structure assembly [19], and collective construction [20, 21].

There has been a recent push for robotic farming and precision agriculture [22, 23], where a fleet of robots is sent to gather data on the status of crops, tend to and harvest them. Mobile sensor networks take advantage of sensor node mobility to relocate sensors in order to enhance sensing and maximize coverage [24, 25], provide robustness to sensor node failure [26, 27], and reduce overall energy consumption by the network [28]. In some communication architectures, mobile agents called data MULEs or message ferries [29–31] are used to transport data between sensors, access points, or base stations, in situations where it is impractical to establish physical communication. Multi-robot simultaneous localization and mapping (SLAM) takes advantage of the robot team's size to attain a more complete (or more detailed) map faster than with a single robot by combining information from multiple sources and coordinating motion [32, 33]. Other applications include transportation systems (e.g., intelligent highways, air traffic control) [34], and the convoy protection scenario (e.g., surveillance, coverage) [35].

A factor in this increased research effort is how miniaturization of technology has allowed for a rise in cost-effective but powerful microprocessors and micro-controllers, facilitating the study of multi-robot systems. In [36], a low-cost and fully scalable robot called the Kilobot was introduced that moves via the use of two vibration motors, sliding along surfaces. This led to the first 1000-robot swarm to operate without human intervention [37]. A swarm of 20 micro-quadrotors was developed and programmed in [38] to achieve tight formations and trajectory following. The experiments demonstrate agility and coordination among the robots in real time. In [39], a low-cost and scalable differential-drive micro-robot called the GRITSbot was presented as a part of a robotic ecosystem. This modularity of sensing and actuation capabilities allows for high adaptability to environmental constraints, as well as to the functional requirements of the application setting.

What all these applications and technological trends have in common is the need for some form of motion planning and coordination among agents in the multi-robot team. In some instances, there is also the need for some form of interaction between a human operator and the multi-robot system. The research in this dissertation makes contributions to these subjects by finding novel solutions to the coverage control problem and by providing a novel symbolic characterization of interactions for multi-robot systems. There is already a wealth of existing literature on these subjects. In the next sections, we provide a representative sample of the existing literature on human-swarm interactions and multi-robot motion planning, and how it compares to the research presented in this dissertation.

2.2 Human-Swarm Interactions & Coverage Control

The *human-swarm interactions* (HSI) paradigm allows for a human operator to introduce high-level decision making into a robot swarm low-level control objectives. The subject has been approached from different aspects in the literature. One approach allows human operators to guide the swarm in the environment by injecting information through interfaces. In [40], a set of trials was performed on a swarm that was autonomously performing radiation search and localization in which human input injected various levels of guidance into the swarm. These trials suggested that injecting appropriate levels of guidance information allowed the swarm to achieve its objective more efficiently. In [41], an architecture was proposed to allow the human operator to guide the swarm by introducing attraction and repulsion points for the sake of obstacle avoidance. In [42], they studied the effect of providing additional information, such as haptic feedback, to the human operator in order to help navigate the swarm around obstacles.

Other approaches focus on the converse – the swarm aids the humans achieve their tasks. For example, in [43], assistive swarm robotics was used to help navigate firefighters and support them with search and rescue operations. In [44, 45], they used the swarm of robots to obtain multiple views of human gestures, which provided multiple samples for machine learning algorithms on gesture recognition. Practical considerations for HSI were explored in [46], where the foraging paradigm was explored under bandwidth restrictions, such as one might experience in underwater missions or on other remote locations.

There have also been approaches which focus on finding a medium that affords the control and manipulation of a swarm of robots. In [47], they used clay as a deformable medium to specify the desired geometric configuration of the swarm. The desired configuration was then processed using image recognition algorithms and distributed formation controllers were used to achieve desired configurations. In our work, we take this latter approach – enabling human-swarm interactions by providing the human operator a mechanism that affords the manipulation of the swarm and that scales with the number of agents in the robot swarm. In particular, we will impose the desired concentration, *or density*, of agents directly on the environment. This way, the robots' task is to spread or concentrate in the different parts of the environment in order to satisfy the desired density of agents imposed by the user. We will achieve this by performing *coverage control* over a human-provided density function.

We extend the work of [48] on optimal coverage for mobile robots as a mechanism to externally influence a swarm of robots. The coverage problem over static densities has been extensively studied before. In [49], it was proposed that for a Voronoi partition of the space, the center of mass locations of each region provides the best coverage. There it was proposed to set the location of the Voronoi seeds to the center of mass of each cell at every iteration, what later came to be known as Lloyd's algorithm. For the static case, it is shown in [50] that the center of mass locations of each cell in a Voronoi tessellation are critical points to the so-called locational cost. Further, if the locational cost is locally strictly convex around the center of mass locations, then the center of mass locations satisfy the necessary conditions to be contraction points. In [48] it was shown that for static densities, a continuous version of Lloyd's algorithm asymptotically achieves a centroidal Voronoi tessellation, a necessary condition for optimality with respect to the locational cost. However, relatively little work has been done on coverage of time-varying densities. Cortes et al. also proposed in [48, 51] the use of a distributed control law to achieve centroidal Voronoi tessellations for the time-varying density case. However, these control laws require that the density be designed in a way not amenable to human-swarm interactions. We construct distributed approximations to a centralized control law that provably achieves centroidal Voronoi tessellations, translate human-provided time-varying densities into robotic movement, scale with the number of robots, and do not make stringent assumptions on the provided densities.

2.3 Multi-Robot Motion Planning

In previous sections we presented a number of applications proposing the use of multi-robot systems. At a certain level of abstraction, most of the tasks therein can be described as a coordinated effort between agents to collaboratively achieve some motion or geometric configuration, while satisfying a set of constraints such as kinematic constraints, spatio-temporal constraints, and collision avoidance. Motion planning of multi-agent systems has been approached from a number of different ways in the literature. We now present a representative sample of these approaches. In [52], motion planning methods are grouped under three general approaches:

Roadmap Approaches: These methods consist of capturing the connectivity of a network of one-dimensional curves in the robot's free space, i.e., a network of connecting paths that do not intersect with the interior of any obstacle. Schemes like visibility graphs, Voronoi diagrams, and silhouette fall under this category. As an example, in [53], paths were represented as the set of junction points between the robot's free space and the boundary of obstacles, reducing the problem of finding the shortest path from an infinite-dimensional problem to a finite-dimensional one. By evolving these junctions with computationally-efficient algorithms, they are able to obtain the (global) shortest path between two points within a prescribed probability, and without the standard assumption of polyhedral obstacles, or having to discretize the environment into a grid.

Cell Decomposition: As the name suggests, these approaches consist of decomposing the robot's free space into simple regions such that paths can be generated between regions. A connectivity graph is then constructed describing the adjacency relation between regions, and this graph is searched for a path that satisfies the constraints. Cell decomposition is one of the most popular methods in robot motion planning. For example, in [54] they use cell decomposition and model the robot's motion between cells as a Markov decision process. They then inform the model's transition probabilities using probabilistic computation tree logic and linear temporal logic such that the robot mission space accommodates for increased expressivity in the specification language, e.g., in addition to "go to *A* without colliding with obstacles," the robot should also "visit location *B* before arriving at *A*."

Potential Fields: By introducing an artificial potential over the environment, these methods allow a robot to be influenced by the goal and obstacles locations with attractive and repulsive potential, respectively. A typical approach is to flow along the negated gradient direction of the potential field. Since no search is required, these methods tend to be very computationally efficient. However, they also tend to suffer from problems such as getting stuck in local minima. To get around these issues, the use of powerful mechanisms have been suggested to escape from local minima [55], or alternatively, it has been suggested to design the potential field such that there is a unique minimum [56].

The research on multi-robot motion planning in this dissertation focuses on two aspects. The first problem is that of characterizing inter-robot interactions for the sake of coordination and collision avoidance. This will be done within the setting of symbolic approaches and trajectory tracking. Symbolic inputs allow us to abstract away the geometry and physical constraints involved in the multi-robot motion planning. As described in [57], this provides the advantage of hierarchical abstractions that are typically broken into three stages: at the top layer is the *specification level*, which describes the motion tasks (such as robots A and B should interact, and arrive at the goal simultaneously). The second layer is the *execution level*, which describes *how* to obtain the motion, e.g., by using an optimality condition that penalizes deviations from a path. The bottom layer is the *implementation level*, which concerns itself with constructing the robot controller, e.g., to track a reference trajectory.

The second aspect extends the notions presented in [58] for trajectory tracking of virtual vehicles. This idea requires the physical vehicles to track a virtual vehicle, as opposed to a path (or trajectory) itself. The virtual vehicle, which is being controlled directly with simplified dynamics, can be controlled in order to track a reference path while satisfying the constraints, compensating for the physical vehicle's dynamics and other disturbances. Assuming the virtual vehicle always remains on the reference path, this problem may be viewed as finding the optimal parameterization of the given reference path (thus turning the path into the desired trajectory) while the physical vehicle performs trajectory tracking.

We will use elements of algebraic topology, namely the braid group [59,60], as the symbols describing interactions and motion plans. The use of braids for robot motion planning has been considered before. In [61], they used the *graph braid group* as the fundamental group of the configuration space of graphs that describe robot motion (e.g., roadmaps, as described above). There, each graph in the configuration space represents discretized collision-free robotic motion, where at each discrete time the graph vertices represent the positions of robots and (possibly moving) obstacles, and the edges represent fixed tracks connecting these vertices. They presented an algorithm to construct the presentation of the graph braid group of *n* agents, where the group generators (i.e., the braids) represent actual paths between configurations of robots on the graph, i.e., the motion plan to transition from one configuration to another. However, they only considered zero-size robots where they relied on a "one edge separation" between points at all times to avoid collisions, and as such, their approach is purely theoretical and mainly focuses on the combinatorics of ideal robots moving on a graph. We use the braid group's generators as symbolic inputs to multi-robot hybrid controllers which characterize and enforce collision-free interactions, take into account kinematic and geometric constraints, and are executable on actual robotic platforms. To the best of our knowledge, this is the first approach that uses braids in multi-robot motion planning to symbolically characterize and enforce rich interaction patterns, implementable on actual robotic platforms.

CHAPTER 3

HUMAN-SWARM INTERACTIONS VIA COVERAGE OF TIME-VARYING DENSITY FUNCTIONS

Coverage control is one area of multi-agent control that has received significant attention lately, e.g., [62], [63], and it is concerned with how to position agents in such a way that "surveillance" of a domain of interest is maximized. In this context, an idea that has been widely adopted to describe how interesting a "domain of interest" is, is to associate a density function to the domain, as was done in [48, 51, 62, 64–66]. However, the focus of previous coverage algorithms has largely been on static density functions, which does not provide enough flexibility when human operators are to adaptively interact with a team through a dynamic re-shaping of the density functions.

To enable this line of inquiry, we require an algorithm that can guarantee multi-robot optimal coverage given general time-varying density functions. Applications to this beyond the means for multi-robot influence can be found in a number of domains. For example, in search and rescue scenarios, the density function could represent the probability of a lost person being at a certain point in an area, e.g., [67]. Additionally, optimal coverage of density functions for multi-robot surveillance and exploration was used in [66], where the density function was modeled to be a function of the explored "frontier." (For other examples, see [68] and references therein.)

To date, relatively little work has been done on coverage with time-varying density functions. In [51], the time-varying case was investigated under a set of simplifying assumptions on the density functions. Another stab at the problem was pursued in [65], where time-varying density functions where used as a means to tracking moving targets. While simulations and experiments verified that coverage was indeed achieved, formal guarantees were absent.

In contrast to [51] and [65], in this paper we derive an algorithm that guarantees optimal coverage with quite general, time-varying functions [69]. As with the algorithms for time-invariant density functions, centroidal Voronoi tessellations (CVT) will play a key role. A CVT is a configuration where the positions of each robot coincide with the centroids of their Voronoi cells, given a so-called Voronoi tessellation of the domain of interest. The CVT may result in the optimality of a commonly used metric on the coverage of a domain by the robot swarm. To that end, the control laws presented here seek to drive the agents into a CVT given a user-provided density describing the areas of importance.

We further find decentralized approximations to the control laws presented here that allow the individual agents in the swarm to compute their update law with only local information. The fact that only adjacency information is required in their update law means that a single operator could potentially influence arbitrarily large number of agents. Due to the scalability of the algorithm, and the level of abstraction that the generation of density functions offers (in that it does not care for the number of agents performing the coverage), we consider these decentralized update laws as tools for HSI [70]. The idea presented here is to allow a human operator to influence teams of robots by generating density functions, letting robot swarm perform coverage over this density.

In subsequent sections, we offer two different means of generating density functions for HSI. The first method allows the user to easily specify the geometric configuration of the swarm and "move" them around, but at an added computational cost. The second method reduces computational cost by using predefined Gaussian functions to allow the human operator to quickly assign importance to the area of interest. It is fast and allows the user to "move" the team of robot easily, but is not as amenable to "shaping" the swarm as the first method. Figure 1 illustrates how these approaches allow a user to manipulate a team of robots to drive to certain areas of the environment. Towards the end of the chapter, we present more discussion on the robotic implementation of these HSI methods.

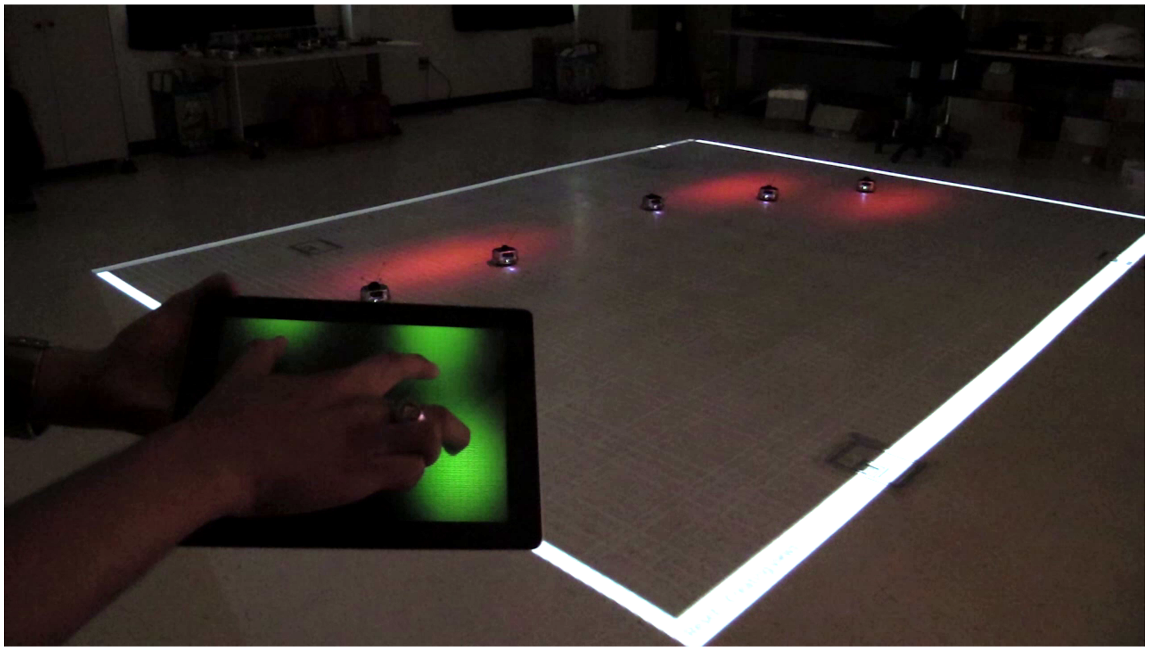


Figure 1. A team of five Khepera robots are influenced by a human operator to drive to certain parts of the environment. The tablet is representative of the robot environment. An overhead projector allows us to overlay the broadcasted density on the environment in real-time.

3.1 The Coverage Problem

We are on the hunt for decentralized update laws that allow a human operator to influence the swarm of agents by providing a density function characterizing the areas of importance within the domain. The agents in the robot swarm will seek to optimize their coverage over the domain under the influence of the user-provided density. In order to discuss optimal coverage, one first has to associate a cost to a robot configuration that describes how well a given area is being covered. For this, we will follow the construction of this so-called locational cost, as was done, for example, in [51].

Let $D \subset \mathbb{R}^2$ be the two-dimensional[1] convex domain representing the robot workspace. Moreover, let $\phi : D \times [0, \infty) \to (0, \infty)$ be the associated density function, which we will assume is bounded and continuously differentiable in both arguments, and where $\phi(q, t)$ captures the relative importance of a point $q \in D$ at time t. For the sake of HSI, this density

[1]Here we focus on the two-dimensional case, but note that the math also extends to higher dimensional spaces as well.

function will be an user-provided input to the robot swarm.

The coverage problem involves placing n robots in D, and we let $p_i \in D$, $i = 1, \dots, n$ be the position of the i^{th} robot. Moreover, the domain itself will be divided into regions of dominance, e.g., [48], $P_1, \dots, P_n$ (forming a proper partition of D), where the idea is to let robot i be in charge of covering region P_i. One can then ask how good the choice of p and P is, where $p = [p_1^T, \dots, p_n^T]^T$, and $P = \{P_1, \cdots, P_n\}$. The final piece needed to answer this question is a measure of how well a given point $q \in D$ is covered by robot i at position $p_i \in D$ (see [50] and references therein). In the most general setting, one can consider the locational cost given by

$$\mathcal{H}(p, P, t) = \sum_{i=1}^{n} \int_{P_i} f(\|q - p_i\|)\phi(q, t)dq.$$

for $f : \mathbb{R} \to [0, \infty)$. As the performance of a large class of sensors deteriorate with a rate proportional to the square of the distance, [71, 72], here we will instead consider the locational cost given by

$$\mathcal{H}(p, P, t) = \sum_{i=1}^{n} \int_{P_i} \|q - p_i\|^2 \phi(q, t)dq. \tag{1}$$

At a given time t, when a configuration of robots p together with the partition P minimize (1), the domain is said to be optimally covered with respect to ϕ. However, it is possible to view the minimization problem as a function of p alone, [51], by observing that given p, the choices of P_i that minimize (1) is

$$V_i(p) = \{q \in D \mid \|q - p_i\| \leq \left\|q - p_j\right\|, i \neq j\}.$$

This partition of D is a Voronoi tessellation — hence the use of V_i to denote the region. With this choice of region, we can remove the partition as a decision variable and instead focus on the locational cost

$$\mathcal{H}(p, t) = \sum_{i=1}^{n} \int_{V_i(p)} \|q - p_i\|^2 \phi(q, t)dq \tag{2}$$

where $\phi(q, t)$ is the user-provided input to the system.

In [50, 73] it was shown that

$$\frac{\partial \mathcal{H}}{\partial p_i} = \int_{V_i} -2(q - p_i)^T \phi(q,t) dq, \tag{3}$$

and since $\phi > 0$, one can define the mass m_i and center of mass c_i of the i^{th} Voronoi cell, V_i, as

$$m_i(p,t) = \int_{V_i(p)} \phi(q,t) dq, \tag{4}$$

$$c_i(p,t) = \frac{\int_{V_i(p)} q\phi(q,t) dq}{m_i}. \tag{5}$$

For convenience, we will stack the masses into a single diagonal (positive-definite) matrix denoted by

$$M = \begin{bmatrix} m_1 & & & \\ & m_2 & & \\ & & \ddots & \\ & & & m_n \end{bmatrix} \otimes I_d, \tag{6}$$

where "⊗" is the Kronecker product and I_d is the $d \times d$ identity matrix where d corresponds to the dimension of the domain ($d = 2$ is considered here). We also group the centers of mass as a single vector $c = \left[c_1^T, \ldots, c_n^T\right]^T$. As was shown in [50, 73], using these quantities allows us to rewrite the partial derivative in (3) as

$$\frac{\partial \mathcal{H}}{\partial p_i} = 2m_i(p_i - c_i)^T. \tag{7}$$

From this expression, we can see that a critical point of (2) is

$$p_i(t) = c_i(p,t), \quad i = 1, \cdots, n, \tag{8}$$

and a minimizer to (2) is necessarily of this form, [74]. Moreover, when (8) is satisfied, p is a so-called centroidal Voronoi tessellation (CVT).

The robots being in a CVT configuration does not, however, imply that the global minimum of (2) is attained, e.g., [50]. In fact, the CVT is in general not unique given a density function ϕ. Finding the globally minimizing configuration is a difficult problem due to the nonlinearity and nonconvexity of (2), as discussed in [75]. As such, in this paper, we are interested in designing algorithms that guarantee convergence to local minima with respect to time-varying density functions (as our goal is to merely be able to influence the swarm), and we make no claims about finding the global minimum.

In light of (7), the gradient direction (with respect to p_i) is given by $\frac{(p_i - c_i)}{\|p_i - c_i\|}$. As such, a (scaled) gradient descent motion for the individual robots to execute would be

Lloyd:

$$\dot{p}_i(t) = -\kappa(p_i(t) - c_i(p,t)) \tag{9}$$

where κ is a positive gain. This is a continuous-time version of Lloyd's algorithm [49] for obtaining CVTs as long as ϕ does not depend on t. The way to see this, as was done in [48], is to take $\mathcal{H}(p)$ in (2) (note that we assume that $\mathcal{H}$ only depends on p and not on t for the purpose of this argument) as the Lyapunov function,

$$\frac{d}{dt}\mathcal{H}(p) = \sum_{i=1}^{n} \frac{\partial}{\partial p_i}\mathcal{H}(p)\dot{p}_i = -2\kappa \sum_{i=1}^{n} m_i \|p_i - c_i(p)\|^2 .$$

By LaSalle's invariance principle, the multi-robot system asymptotically converges to a configuration $\{\|p_i - c_i(p)\|^2 = 0,\ i = 1, \cdots, n\}$, i.e., to a CVT ([48]).

However, if ϕ is time-varying, the same control law does not stabilize the multi-robot system to a CVT. This point can be hinted at by investigating the evolution of a time-dependent $\mathcal{H}(p,t)$,

$$\begin{aligned}\frac{d}{dt}\mathcal{H}(p(t),t) &= \sum_{i=1}^{n} \frac{\partial}{\partial p_i}\mathcal{H}(p(t),t)\dot{p}_i + \frac{\partial}{\partial t}\mathcal{H}(p(t),t)\\ &= \sum_{i=1}^{n} \int_{V_i} \|q - p_i(t)\|^2 \frac{\partial \phi}{\partial t}(q,t)dq - 2\kappa \sum_{i=1}^{n} m_i \|p_i(t) - c_i(p(t),t)\|^2 .\end{aligned}$$

There is no reason, in general, to assume that this expression is negative since we do not want to impose assumptions on slowly varying, or even quasi-static, density functions. Instead, what is needed is a new set of algorithms for handling the time-varying case.

To get around the problem associated with non-slowly varying density functions, timing information must be included in the motion of the robots. In [51], this was done through the assumption that $\phi(q,t)$ is such that

$$\frac{d}{dt}\left(\sum_{i=1}^{n}\int_{V_i}\|q-c_i\|^2\,\phi(q,t)dq\right)=0.$$

Letting

$$\dot{m}_i=\int_{V_i}\dot{\phi}(q,t)dq,\quad \dot{c}_i=\frac{1}{m_i}\left(\int_{V_i}q\dot{\phi}(q,t)dq-\dot{m}_i c_i\right),$$

the algorithm in [51] for time-varying density functions is given by

Cortes:

$$\dot{p}_i=\dot{c}_i-(\kappa+\frac{\dot{m}_i}{m_i})(p_i-c_i). \tag{10}$$

Under the previously mentioned assumption on ϕ, $\mathcal{H}(p,t)$ again becomes a Lyapunov function when the agents move according to (10), and convergence to a time-varying CVT is established.

Unfortunately, the assumption required to make (10) work is rather restrictive for the sake of HSI as it basically states that the density function must be designed in such a way that the locational cost achieves a minimum that remains constant as the agents move. Thus, for the remainder of the paper, we will develop new methods for handling time-varying density functions that do not impose major assumptions on $\phi(q,t)$. In fact, if the density function is to be thought of as an external, human-generated input to the system, there are no a priori reasons why the human operator would restrict the interactions to satisfy particular regularity assumptions on ϕ.

3.2 Centralized Coverage of Time-Varying Densities

We are now in pursuit of a control laws that allow us achieve a CVT, even if the density function is time-varying. In order to move forward with our analysis, we need to understand some of the properties associated with the center of mass of a given Voronoi tessellation. We first note that the partial derivatives of the center of mass with respect to the agents positions has eigenvalues bounded by one, as long as the agents are close enough to a CVT, and under the assumption that at the CVT the locational cost is locally strongly convex.

Lemma 1. *If the CVT is locally strongly convex, then there exists* $\delta > 0$ *such that if* $\|p(t) - c(p(t), t)\| < \delta$ *then* $\mathrm{Re}(\lambda) < 1$ *for all* $\lambda \in \mathrm{eig}(\frac{\partial c}{\partial p})$.

Proof. As it was shown in [76], under the assumption that the locational cost is locally strongly convex around the CVT, we have that

$$\begin{aligned} \frac{\partial^2 \mathcal{H}}{\partial p^2}\Big|_{p=c} &= 2M(I - \frac{\partial c}{\partial p}) \succ 0 \\ &\Rightarrow M^{1/2}(I - \frac{\partial c}{\partial p})M^{-1/2} \succ 0 \\ &\Rightarrow (I - M^{1/2}\frac{\partial c}{\partial p}M^{-1/2}) \succ 0 \end{aligned}$$

where M is the positive-definite diagonal matrix of Voronoi cell masses defined in (6), which implies that for all $\lambda \in \mathrm{eig}(\frac{\partial c}{\partial p})$ we have that $\mathrm{Re}(\lambda) < 1$.

Let $f(\lambda) = \det\left(\lambda I - \frac{\partial c}{\partial p}\right) = \sum_{i=1}^{2N} a_i \lambda^i$ be the characteristic polynomial for $\frac{\partial c}{\partial p}$, whose roots ($\lambda$ such that $f(\lambda) = 0$) are the eigenvalues of $\frac{\partial c}{\partial p}$. The coefficients a_i, $i = 1, \ldots, 2N-1$, are themselves polynomials of the entries of $\partial c/\partial p$, with $a_{2N} = 1$.

Since $m_i > 0$ for all i, the different entries of $\partial c/\partial p$ are continuous (as we will see below). Further, since the roots of any polynomial vary continuously with all but its highest order coefficient [77], we have that the eigenvalues of $\partial c/\partial p$ are continuous.

One can apply a chain of continuity arguments to show that for any $\epsilon > 0$, there exists $\delta > 0$ so $\mathrm{Re}(\lambda) < 1$ whenever $\|p(t) - c(p(t), t)\| < \delta$. □

Corollary 1. *As long as the agents are close enough to a CVT configuration for which the locational cost is locally strongly convex, the matrix inverse* $(I - \frac{\partial c}{\partial p})^{-1}$ *exists.*

We use this corollary to obtain our first result on maintaining a CVT for all time, as originally stated in [69].

Theorem 1 (Maintaining CVT [69]). *Let $p(t_0) = c(p(t_0), t_0)$. If*

$$\dot{p} = \left(I - \frac{\partial c}{\partial p}\right)^{-1} \frac{\partial c}{\partial t}, \qquad t \geq t_0$$

then

$$\| p(t) - c(p(t), t) \| = 0, \qquad t \geq t_0$$

as long as the inverse $(I - \partial c/\partial p)^{-1}$ is well-defined.

Proof. Assume the agents begin from a CVT configuration, i.e., $p(t_0) = c(p(t_0), t_0)$. We need to ensure that

$$\frac{\mathrm{d}}{\mathrm{d}t}(p(t) - c(p(t), t)) = 0, \qquad \forall t \geq t_0.$$

But this implies that $\dot{p} = \dot{c} = \frac{\partial c}{\partial p}\dot{p} + \frac{\partial c}{\partial t}$, which can be rearranged into the form

$$\dot{p} = \left(I - \tfrac{\partial c}{\partial p}\right)^{-1} \tfrac{\partial c}{\partial t}$$

as long as the inverse is well-defined. From Lemma 1, a sufficient condition for this is that the initial CVT makes the locational cost locally strongly convex. □

In order for this theorem to be meaningful, we require that the agents first achieve a CVT configuration which is then maintained. This control law also assumes the agents will be able to instantaneously accelerate to the required velocities, which may not be possible in practice. In order to compensate for saturation, modeling errors and deviations from the CVT, in [69] a proportional term is introduced that forces the agents into a CVT. This update law was called the *TVD-C* which stands for time-varying densities, centralized case.

$$\dot{p}(t) = \left(I - \frac{\partial c}{\partial p}\right)^{-1} \left(-\kappa (p(t) - c(p(t), t)) + \frac{\partial c}{\partial t}\right) \tag{11}$$

where $\kappa > 0$ is a proportional gain. It is noteworthy that this proportional term influences the team of robots to move towards a (scaled) gradient descent direction to achieve a CVT configuration, and that once a CVT is achieved the proportional term does not contribute to the update law and Theorem 1 holds.

As long as the inverse is well-defined, this update law will drive the agents into a CVT. This is encapsulated in the following theorem.

Theorem 2 (Convergence of TVD-C)**.** *If* $1 \notin \operatorname{eig}\left(\frac{\partial c}{\partial p}\right)$ *and we let*

$$\dot{p} = \left(I - \frac{\partial c}{\partial p}\right)^{-1} \left(-\kappa \left(p - c\right) + \frac{\partial c}{\partial t}\right), \qquad t \geq t_0$$

then $\|p(t) - c(p(t), t)\| \to 0$ *as* $t \to \infty$ *with rate of decrease* $\exp\left(-\kappa(t - t_0)\right)$.

Proof. Consider the total derivative for $\|p(t) - c(p(t), t)\|^2$ below

$$\begin{aligned}
\frac{\mathrm{d}}{\mathrm{d}t}\left(\|p - c\|^2\right) &= 2\left(p - c\right)^T \left(\dot{p} - \dot{c}\right) = 2\left(p - c\right)^T \left(\dot{p} - \frac{\partial c}{\partial p}\dot{p} - \frac{\partial c}{\partial t}\right) \\
&= 2\left(p - c\right)^T \left(\left(I - \frac{\partial c}{\partial p}\right)\dot{p} - \frac{\partial c}{\partial t}\right) \\
&= 2\left(p - c\right)^T \left(\left(I - \frac{\partial c}{\partial p}\right)\left(I - \frac{\partial c}{\partial p}\right)^{-1} \left(-\kappa \left(p - c\right) + \frac{\partial c}{\partial t}\right) - \frac{\partial c}{\partial t}\right) \\
&= -2\kappa \|p - c\|^2
\end{aligned}$$

which tells us that

$$\frac{\mathrm{d}}{\mathrm{d}t}\left(\|p - c\|^2\right) = -2\kappa \|p - c\|^2$$

This is a homogeneous linear differential equation. For initial condition $\|p(t_0) - c(p(t_0), t_0)\|$ it is known to have a unique solution given by

$$\begin{aligned}
&\|p(t) - c(p(t), t)\|^2 = \exp(-2\kappa(t - t_0))\,\|p(t_0) - c(p(t_0), t_0)\|^2 \\
\Rightarrow \quad &\|p(t) - c(p(t), t)\| \ = \exp(-\kappa(t - t_0))\,\|p(t_0) - c(p(t_0), t_0)\|\,.
\end{aligned}$$

Hence $\|p(t) - c(p(t), t)\| \to 0$ as $t \to \infty$. Further, we see that

$$\begin{aligned} \|p(t) - c(p(t), t)\| &= \exp(-\kappa(t - t_0)) \, \|p(t_0) - c(p(t_0), t_0)\| \\ \Rightarrow \quad \frac{\|p(t) - c(p(t), t)\|}{\|p(t_0) - c(p(t_0), t_0)\|} &= \exp(-\kappa(t - t_0)) \end{aligned}$$

such that the rate of decrease is given by $\exp\left(-\kappa(t - t_0)\right)$. $\square$

Recalling Lemma 1, as long as the agents are close enough to a CVT and the locational cost is locally strongly convex around the CVT, the inverse in control law TVD-C will drive the agents to the CVT. However, note that it is not necessary for the locational cost to be locally strongly convex at a CVT. Indeed, in [76] they present several CVT configurations that result in saddle points instead. What has been observed in robotic implementations of these algorithms (Section 3.5) is that the addition of the gradient descent term seeks to reduce the overall locational cost. It is also observed that all the disturbances seem to drive the agents into stable CVT configurations. This in turn suggests the achieved CVT are in fact local minimizers to the locational cost, as opposed to saddle points.

It is also noteworthy that Theorem 2 only requires that $1 \notin \text{eig}(\partial c/\partial p)$, and as such the agents are driven into a CVT even if $\text{Re}(\lambda) > 1$ for any $\lambda \in \text{eig}(\partial c/\partial p)$. If this is the case, then the problem lies in that due to the continuity of these eigenvalues, we will inevitably obtain $1 \in \text{eig}(\partial c/\partial p)$. In practice, however, these ill-conditioned matrices are hardly a problem as the set $\{t | 1 \in \text{eig}(\frac{\partial c}{\partial p}(p(t), t))\}$ seems to often be a zero-measure set.

We would still like to avoid the case of ill-conditioned matrices all together, and perhaps more importantly, this matrix inverse has a dense structure, which makes this a centralized approach — which will not scale well, and will hinder our effort in manipulating large teams of robots. We will address these issues in the following section where we proceed to obtain distributed approximations to our centralized update laws.

3.3 Distributed Coverage of Time-Varying Densities

In order to implement update law (11), we need to deal with two subtle difficulties associated with it. The first is the computation of the deceivingly innocent looking terms $\partial c/\partial p$ and $\partial c/\partial t$, the second is addressing the computation of the matrix inverse $(I - \partial c/\partial p)^{-1}$. We will address the matrix inverse difficulty in Section 3.3.2. In the next section we address the issue of the computation of the partial derivatives.

3.3.1 Computing the Partial Derivatives

Recall that by combining equations (4) and (5), we have that

$$c_i\,(p,t) = \frac{\int_{V_i(p)} q\,\phi(q,t)\;\mathrm{d}q}{\int_{V_i(p)} \phi(q,t)\;\mathrm{d}q},$$

which depends on p in the boundary of the area over which the two integrals are taken.

In order to compute these partials, we first need to make use of Leibniz rule, e.g., [76].

Lemma 2. *Let $\Omega(p)$ be a region that is a smooth function of p such that the unit outward normal vector n is uniquely defined almost everywhere on $\partial\Omega$, which is the boundary of Ω. Let*

$$F = \int_{\Omega(p)} f(q)dq.$$

Then

$$\frac{\partial F}{\partial p} = \int_{\partial\Omega(p)} f(q)\frac{\partial q}{\partial p}\cdot n(q)dq$$

where $\frac{\partial q}{\partial p}$ is the derivative of the points on $\partial\Omega$ with respect to p.

In [76], it was investigated how Voronoi cells changed as functions of p_i. In fact, it was shown in [76] that for any point $q \in \partial V_{ij}$ (the boundary between adjacent cells V_i and V_j),

$$\frac{\partial q}{\partial p_j^{(b)}}\cdot(p_j - p_i) = \frac{1}{2}e_b\cdot(p_j - p_i) - e_b\cdot\left(q - \frac{p_i + p_j}{2}\right),$$
$$\frac{\partial q}{\partial p_i^{(b)}}\cdot(p_j - p_i) = \frac{1}{2}e_b\cdot(p_j - p_i) + e_b\cdot\left(q - \frac{p_i + p_j}{2}\right),$$

where $p_j^{(b)}$ denotes the bth component of the vector p_j and e_b is the bth elementary unit vector. Note that in this chapter, $b = 1, 2$ since we are considering the case $D \subset \mathbb{R}^2$ only.

Substituting this into Leibniz rule, we obtain

$$\frac{\partial c_i^{(a)}}{\partial p_j^{(b)}} = -\left(\int_{\partial V_{ij}} \phi q^{(a)} \frac{q^{(b)} - p_j^{(b)}}{\left\|p_j - p_i\right\|} dq\right) \Big/ m_i + \left(\int_{V_i(P)} \phi q^{(a)} dq\right)\left(\int_{\partial V_{ij}} \phi \frac{q^{(b)} - p_j^{(b)}}{\left\|p_j - p_i\right\|} dq\right) \Big/ m_i^2 \quad (12)$$

where $a = 1, 2$, $b = 1, 2$ and where $i \neq j$. When $i = j$ we must consider the contribution from all neighbors

$$\frac{\partial c_i^{(a)}}{\partial p_i^{(b)}} = \sum_{k \in N_{V_i}} \left[\left(\int_{\partial V_{ik}} \phi q^{(a)} \frac{q^{(b)} - p_i^{(b)}}{\|p_k - p_i\|} dq\right) \Big/ m_i - \left(\int_{V_i(P)} \phi q^{(a)} dq\right)\left(\int_{\partial V_{ik}} \phi \frac{q^{(b)} - p_i^{(b)}}{\|p_k - p_i\|} dq\right) \Big/ m_i^2\right]. \quad (13)$$

We can rewrite these equations more compactly using block matrix notation

$$\left[\frac{\partial c}{\partial p}\right]_{ij} = \frac{\partial c_i}{\partial p_j} = -\frac{\int_{\partial V_{ij}} (q - c_i)(q - p_j)^T \phi \, \mathrm{d}q}{m_i \left\|p_j - p_i\right\|}, \quad (14)$$

$$\left[\frac{\partial c}{\partial p}\right]_{ii} = \frac{\partial c_i}{\partial p_i} = \sum_{k \in N_{V_i}} \frac{\int_{\partial V_{ik}} (q - c_i)(q - p_i)^T \phi \, \mathrm{d}q}{m_i \left\|p_j - p_k\right\|}. \quad (15)$$

where $[\cdot]_{ij}$ corresponds to the ith, jth $d \times d$ block matrix.

It is noteworthy that given a continuously differentiable density function ϕ, computing $\partial c / \partial p$ at any given time t becomes an exercise in finding line and area integrals. In implementation, it suffices to use numerical approximations to compute these integrals (e.g., Riemann sums, Gaussian quadrature rule).

One more partial derivative is required for update law (11), namely $\partial c / \partial t$. Another application of Leibniz rule results in

$$\frac{\partial c_i}{\partial t} = \frac{m_i \int_{V_i} q \frac{\partial \phi}{\partial t}(q, t) \, \mathrm{d}q - \int_{V_i} q \phi(q, t) \, \mathrm{d}q \int_{V_i(p)} \frac{\partial \phi}{\partial t}(q, t) \, \mathrm{d}q}{m_i^2} \quad (16)$$

or more compactly

$$\frac{\partial c_i}{\partial t} = \frac{\int_{V_i} (q - c_i) \frac{\partial \phi}{\partial t} \, \mathrm{d}q}{m_i}. \quad (17)$$

with $\frac{\partial c}{\partial t}^T = [\frac{\partial c_1}{\partial t}^T, \ldots, \frac{\partial c_n}{\partial t}^T]$.

When implementing the update law, it suffices to numerically compute the integrals in $\partial c/\partial t$. However, unlike with the computation of the integrals in $\partial c/\partial p$, we require knowledge of $\partial\phi/\partial t$. If ϕ is not provided analytically in t, then one could:

1. Utilize a finite difference scheme to approximate $\partial\phi/\partial t$. This could however give rise to difficulties with noisy measurements.

2. Alternatively, the user could provide the time evolution of the density function directly via a continuous function $\partial\phi/\partial t$ such that $\phi(q,t) = \int_{t_0}^{t} \frac{\partial\phi}{\partial t}(q,\tau)\,\mathrm{d}\tau$. In the implementations found in Section 3.4, the "shape" of the density is defined as a continuously differentiable function $\phi(q,t_0)$ defined over D, and the as the user provided input corresponds to $\partial\phi/\partial t$.

As a final implementation note, finding the integrals over V_i may be computationally intensive. However, the expanded versions of these partial derivatives reveal that only a few integrals actually need to be computed per agent. To compute all expressions it suffices to compute the integrals of ϕ, $d\phi/dt$, $q\phi$, and $qd\phi/dt$ over V_i, which may be computed once for each agent (and may be computed in a distributed fashion).

3.3.2 Circumventing the Matrix Inverse Computation

The second subtle difficulty with update law (11) is ensuring that the inverse $(I - \partial c/\partial p)^{-1}$ is well defined, which is generally difficult. Lemma 1 provides us with a sufficient condition for the existence of this matrix. In [76] it was also shown that in the time-invariant case, the inverse is well-defined as long as $\phi(p)$ is a log-concave function of p. We would also need ϕ to be continuously differentiable in both arguments, so these two conditions are enough to ensure that the inverse exists. Since the motivation for this work is to have a human operator influence the team of robots by generating these density functions, the former constraint is quite an unsatisfying one, for it would greatly reduce the types of density functions allowable. Moreover, the computation of this $2n \times 2n$ matrix inversion does not scale well with increase in number of agents, and requires information on every agent which makes it a centralized scheme. Fortunately, it is possible to alleviate these concerns by performing a distributed approximation of update law (11). The desired approximation can be found by using the Neumann series, e.g., [78].

Lemma 3 (Neumann series)**.** *Let A be a square matrix. If $\lim_{k\to\infty} A^k = 0$, then $I - A$ is invertible and*

$$(I - A)^{-1} = I + A + A^2 + A^3 + \dots .$$

Moreover, for a $m \times m$ square matrix A, $\lim_{k\to\infty} A^k = 0$ if and only if $|\lambda_i| < 1$ for all $i = 1, 2, \cdots, m$, where λ_i are the eigenvalues of A. As such, let λ_{max} denote the eigenvalue with the largest magnitude of the matrix $\partial c/\partial p$. Using the Neumann series, we can express $(I - \partial c/\partial p)^{-1}$ as

$$\left(I - \frac{\partial c}{\partial p}\right)^{-1} = I + \frac{\partial c}{\partial p} + \left(\frac{\partial c}{\partial p}\right)^2 + \dots \tag{18}$$

as long as $|\lambda_{max}| < 1$.

Our goal will be to truncate this series to obtain the well-defined approximation to the matrix inverse, but then the question arises: how many terms should be kept? The answer lies in the sparsity structure of $\partial c/\partial p$.

Given a Voronoi partition of the area of interest, we denote the boundary between the two cells V_i and V_j by ∂V_{ij}. Since we are only considering the planar case, there are three possibilities for ∂V_{ij}:

1. ∂V_{ij} is empty, meaning that cells V_i and V_j do not intersect.
2. ∂V_{ij} consist of a single point, meaning that cells V_i and V_j share a single vertex.
3. ∂V_{ij} is a line, meaning that cells V_i and V_j share a face.

We will denote N_{V_i} to be set of indexes pertaining to the agents whose Voronoi cells V_j share a face with agent i's Voronoi cell V_i.

Lemma 4. $j \notin N_{V_i} \implies \dfrac{\partial c_i}{\partial p_j} = 0.$

Proof. For the first two cases, i.e., ∂V_{ij} is either empty or consists of a singleton, from (12) and (13) we see that the integrals over ∂V_{ij} would be zero. Note that for these two cases, this will be true for all four elements in $\partial c_i / \partial p_j$. Since these two cases correspond to agents i and j not sharing a face, we conclude that $j \notin N_{V_i}$ implies that $\partial c_i / \partial p_j = 0$. □

This lemma tells us that $\partial c / \partial p$ actually encodes adjacency information of the graph induced by the Voronoi tessellation. This induced graph is known as the *Delaunay graph*.

To obtain a distributed update law, we must insist that the update for $\dot{p}_i$ depends only on information from itself (p_i and $\phi(q,t)$ for $q \in V_i$) and information on neighboring agents (p_j and $\phi(q,t)$ for $q \in V_j$, for all $j \in N_{V_i}$). To this end, we truncate the Neumann series in (18) after just two entries, i.e., $(I - \partial c/\partial p)^{-1} \approx I + \partial c/\partial p$. By modifying update law (11) with this approximation, we obtain the update law

$$\dot{p} = \left(I + \frac{\partial c}{\partial p}\right)\left(-\kappa\,(p - c) + \frac{\partial c}{\partial t}\right),$$

which at the individual robot level results in the update law called *TVD-D*$_1$ for Time-Varying Densities, Distributed case with 1-hop adjacency information:

$$\dot{p}_i = \frac{\partial c_i}{\partial t} - \kappa\,(p_i - c_i) + \sum_{j \in \overline{N}_{V_i}} \frac{\partial c_i}{\partial p_j}\left(\frac{\partial c_j}{\partial t} - \kappa\left(p_j - c_j\right)\right) \tag{19}$$

where $\overline{N}_{V_i}$ is the closed neighborhood set to Voronoi cell i in the Delaunay graph.

It should be noted that (19) is always well defined (as long as ϕ is continuously differentiable). In other words, even if the Neumann series is not convergent or if the inverse does not exist, the entries in (19) are well defined. In fact, it turns out that during the robotic experiment, even in cases where $|\lambda_{max}| > 1$, the robots consistently evolve in a manner that achieves coverage. In [69], a comparison is made between both the centralized and decentralized approaches shown here to other available techniques for coverage of time-varying densities — we refer the readers to it for a discussion on how they compare.

One can now investigate what happens when higher order terms are kept in the Neumann series. For this, we let $dist(i, j)$ denote the edge distance, or number of edges in the shortest path, between i and j in the Delaunay graph induced by the Voronoi tessellation. And, as $\partial c/\partial p$ is a (block) adjacency matrix, we have that

$$\left[(\partial c/\partial p)^k\right]_{ij} \neq 0 \;\Rightarrow\; dist(i, j) = k,\; k = 0, 1, 2, \dots$$

where $[\;\cdot\;]_{ij}$ denotes the block corresponding to cell c_i and robot position p_j.

The k-hop version of $TVD\text{-}D_1$ thus becomes

$$\dot{p} = \sum_{\ell=0}^{k} \left(\frac{\partial c}{\partial p}\right)^{\ell} \left(-\kappa(p - c) + \frac{\partial c}{\partial t}\right). \tag{20}$$

3.4 Designing Density Functions

At the heart of our HSI implementation lies update law (19) which allows a human operator to influence the position of the swarm by broadcasting the time-varying density ϕ defined over the domain D. In this section, we describe two different design methods that allow a human operator to generate the density function based on simple inputs such as the general shape of the density or its position in the domain.

3.4.1 Diffusion of Drawn Geometric Configurations

We now present an approach that allows the user to specify the geometric configuration of the swarm. The process consists of allowing the human operator to draw the desired shape in the tablet-like interface. This drawing, taken as a binary image over the area of interest $\Psi : D \to \{0, 1\}$, can be made smooth and turned into a continuously differentiable function in the spatial argument by evolving it as a diffusion process [79]. The result is a smooth, non parametric density function with the pixels determining the density intensities.

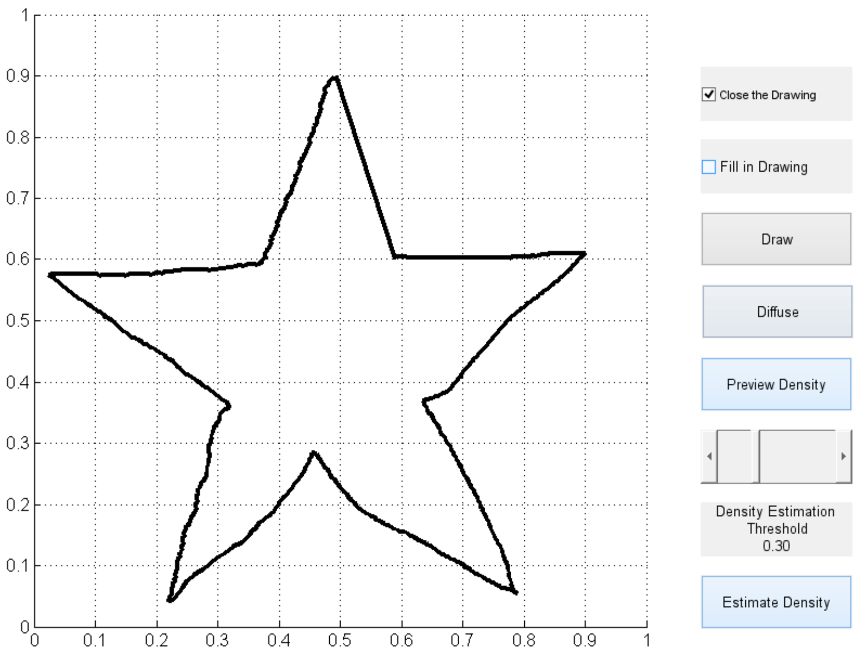
One could then simply pass the image to the numerical integrating tools to obtain update law for the agents, interpolating between pixel values as needed. Figure 2a illustrates a potential graphical user interface to provide the input density to the swarm by allowing a user to draw the desired configuration. Figure 2c illustrates an example of the smoothed drawing that represents the desired density.

However, in order to reduce the amount of information needed to be communicated to the agents, it is possible to use the level sets of the resulting image to come up with a Gaussian Mixture Model (GMM), which would collapse the dimensionality from the pixel count (which could be significant) to k centroids and covariances, k being the amount of Gaussian functions desired to approximate the non parametric density, thus reducing significantly the amount of information required to represent the density. A GMM is given in the form

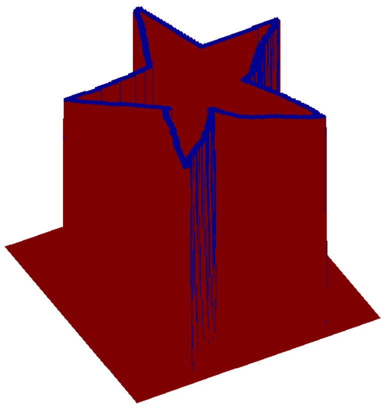
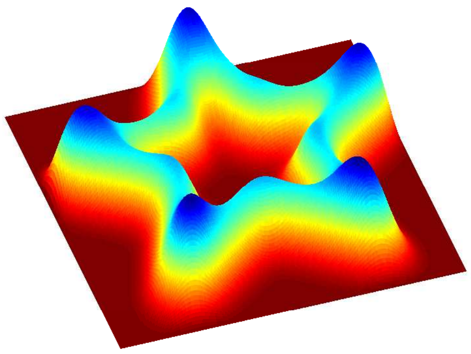
$$\phi_{shape}(q) = \sum_{i=1}^{k} f_i(q) = \sum_{i=1}^{k} \frac{\alpha_i}{2\pi\sqrt{|\Sigma_i|}} \exp\left(-\frac{1}{2}(q-\mu_i)^T \Sigma_i^{-1} (q-\mu_i)\right)$$

where μ_i, Σ_i and α_i are the mean, covariance and weight of the i^{th} mixture, with $\sum \alpha_i = 1$ and $\sum \mu_i = 0$.

In [80], a process to obtain a GMM from a data set with redundancies over the data is presented. This process can be utilized to obtain a parametric approximation of the desired density. In order to generate the required redundant data sets, sample points can be selected from the contour level sets of the desired density (Figure 3a). The points in this data set are then grouped into k clusters, where the design parameter k is the number of Gaussian models used in the GMM. A larger k can be used for a finer approximation of the desired density whereas a smaller k can be used for coarser approximation. The data from each cluster is used to determine the parameters for each of the Gaussian models. Figure 3b illustrates the data set obtained from the original drawing with the redundant data points obtained from the level set contours of the desired density function. Figure 3b also illustrates the GMM found to approximate the density with the use of ten Gaussian functions.



(a) Graphical user interface allows the human operator to draw the desired density shape.

(b) The drawn shape is parsed as a binary image over the domain.

(c) Binary image is turned smooth by a diffusion process.

Figure 2. Graphical user interface allows the user to generate density shapes.

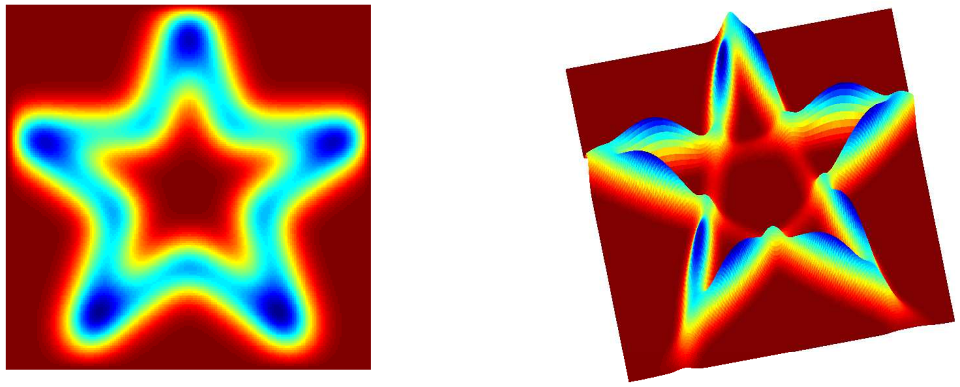

(a) Level sets of desired function. (b) GMM approximation of the desired density shape.

Figure 3. GMM approximation of desired density shape based on level sets obtained from the diffused image.

In order to move the density around, one could evolve the mixture means to track the desired user-provided position, e.g.,

$$\phi(q,t) = \sum_{i=1}^{k} f_i\left(q - v(t)\right), \qquad \frac{\partial \phi}{\partial t} = \sum_{i=1}^{k} f_i\left(q - v(t)\right)\left(q - v(t) - \mu_i\right)^T \Sigma_i^{-1} \dot{v}(t)$$

where $v(t)$ is the solution to $\dot{v}(t) = -\kappa\left(v(t) - r(t)\right)$ given $v(t_0) \in D$ for some $\kappa > 0$, and $r(t) \in D$ is the human-operator provided desired location of the density at t.

3.4.2 Control of Gaussian Functions

The previous approach allows an user to generate a density function that captures the important areas to be covered by simply drawing over the domain. In order to reduce the amount of information required to describe the generated non parametric density, this was approximated by a Gaussian Mixture Model (GMM) with k Gaussian functions. If the geometric shape the swarm takes is not as important as actively manipulating the swarm, then the GMM concept from the previous method can be modified by discarding the a priori weights and fixing the general “shape” of the Gaussian functions.

In this method the human operator “taps” on the desired location of the swarm on a tablet representative of the domain D. Influence on the team of robots is achieved by performing coverage of the density function made of Gaussian functions in the form

$$\phi(q,t) = \epsilon_0 + \sum_{i=1}^{R} \ell_i(t) e^{-\frac{1}{2}(q-\mu_i(t))^T \Sigma^{-1}(q-\mu_i(t))}$$

$$\frac{\partial \phi}{\partial t}(q,t) = \sum_{i=1}^{R} \left[\ell_i(t)\,(q-\mu_i(t))^T\,\Sigma^{-1}\dot{\mu}_i(t) + \dot{\ell}_i(t)\right] e^{-\frac{1}{2}(q-\mu_i(t))^T \Sigma^{-1}(q-\mu_i(t))}$$

where Σ is a positive-definite two-dimensional matrix that determines the “shape” for the the Gaussian functions (e.g., $\Sigma = \sigma I_2$, $\sigma > 0$, provides a circular “shape”), R is the number of fingers the user is tapping with, ϵ_0 is a small positive constant so that $\phi > 0$ even if there are no fingers presents, and $\mu_i(t)$ is the solution to

$$\dot{\mu}_i = -\kappa(\mu_i - r_i(t))$$

for some $\kappa > 0$ and where $r_i(t) \in D$ is the location of the i^{th} desired location in D (provided by the number of “taps” on the tablet-interface) which directly influences the centroid of the individual Gaussian functions, $i = 1, \ldots, R$. The Gaussian terms are weighted by transition functions in order to smoothly introduce and remove them from the domain of interest. These are of the form

$$\ell_i(t) = \frac{1}{1 + e^{-\alpha((t-\tau_{i,1})-\frac{1}{2})}} - \frac{1}{1 + e^{-\alpha((t-\tau_{i,2})-\frac{1}{2})}}$$

$$\dot{\ell}_i(t) = \frac{\alpha e^{-\alpha((t-\tau_{i,1})-\frac{1}{2})}}{\left(1 + e^{-\alpha((t-\tau_{i,1})-\frac{1}{2})}\right)^2} - \frac{\alpha e^{-\alpha((t-\tau_{i,2})-\frac{1}{2})}}{\left(1 + e^{-\alpha((t-\tau_{i,2})-\frac{1}{2})}\right)^2}$$

where $\tau_{i,1}$ corresponds to the time at which the i^{th} Gaussian term is added, $\tau_{i,2}$ corresponds to the time at which the i^{th} Gaussian term is removed ($\tau_{i,1} < \tau_{i,2}$), and $\alpha > 0$ (e.g., $\alpha = 10$) determines the rate of transition.

3.5 Robotic Experiments

Both design methods where implemented on robotic platforms to validate the approaches to HSI. For every agent, only it and its neighbors’ state and density information in their respective Voronoi cells were used to compute the update laws. Line integrals were approximated by Riemann sums, whereas Gaussian quadrature rule was used for area integrals.

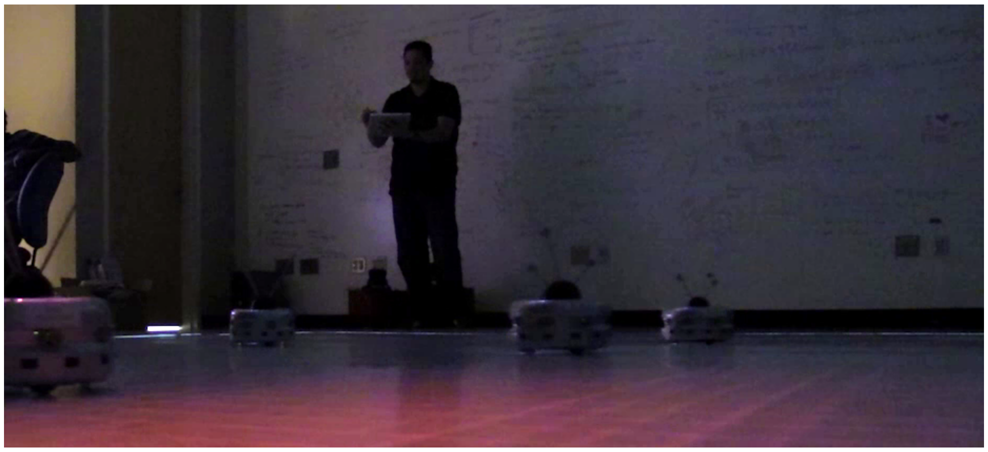
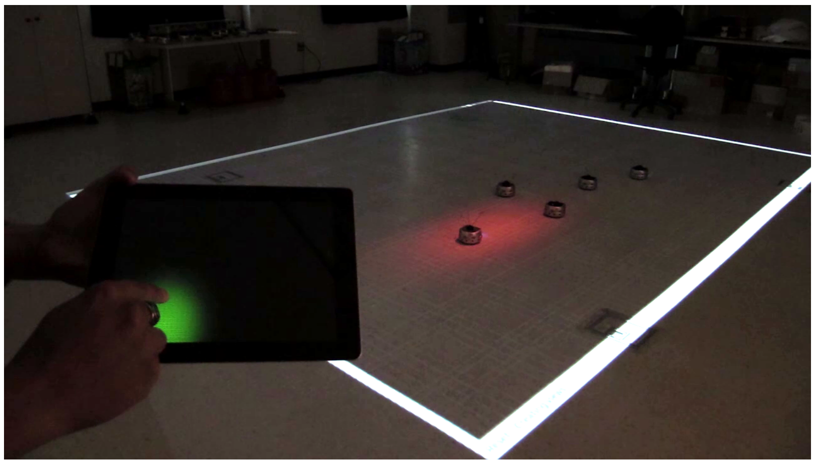
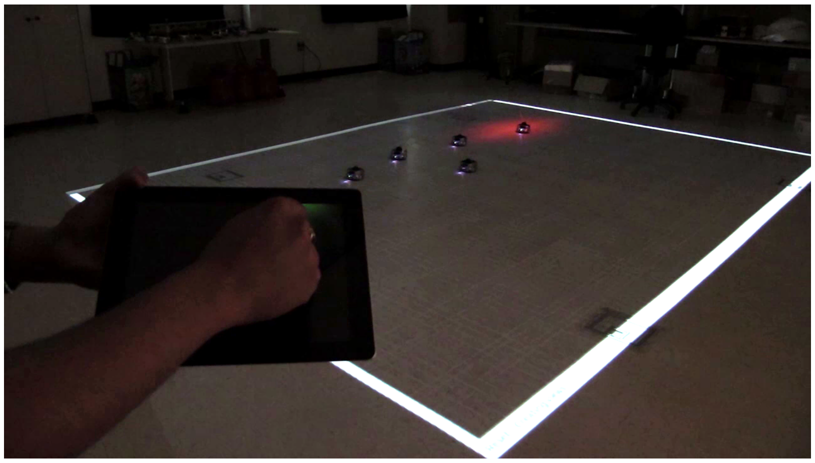


Figure 4. User observes position of swarm and redirects it with the use of a tablet.

The human operator used an iPad to input the density locations. Figure 4 illustrates the user observing the swarm and redirecting it with the use of the tablet interface. These were transmitted over WiFi and UDP to a central computer which calculated the update law for each agent. Even though a central computer is used for the computation, the control is computed for every agent only using adjacency information, and then it is transmitted to the pertinent robot via WiFi and UDP.

The density design method described in Section 3.4.2 was implemented on an Ubuntu computer with an Intel dual core CPU 2.13GHz and 4GB of memory, running ROS (Robot Operating System, version Diamondback). This computer also received state information from ten OptiTrack S250e motion capture cameras that were used to provide position and orientation data for the state information of an agent and its neighbors was used to compute the Voronoi tessellation. The robotic platforms used for the experiments were Khepera III robots from K-team. The Khepera III robots each have a 600MHz ARM processor with 128Mb RAM, embedded Linux, differential drive wheeled robots equipped with a wireless card for communication over a wireless router. The rviz package in ROS was used for visualization of the user-provided density function. The visualization was overlapped with the real physical environment to give a real-time visual representation of the user-provided density function. This is shown in Figure 5.


(a) User influencing swarm to center of domain.


(b) User influencing swarm to corner of domain.

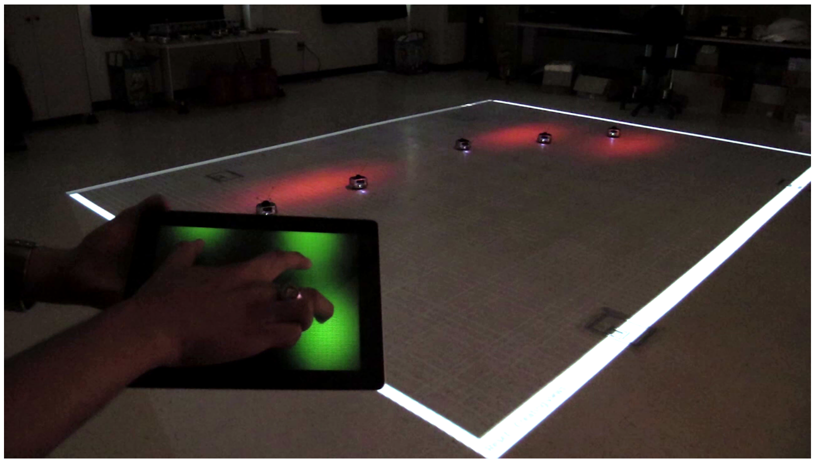

(c) User splits the swarm by introducing multiple locations of interest.

Figure 5. Robotic implementation of HSI via coverage of Gaussian functions. A tablet is used to directly influence the location of the swarm formation. An overhead projector is used to visualize the user-provided density function in real-time over the robot workspace.

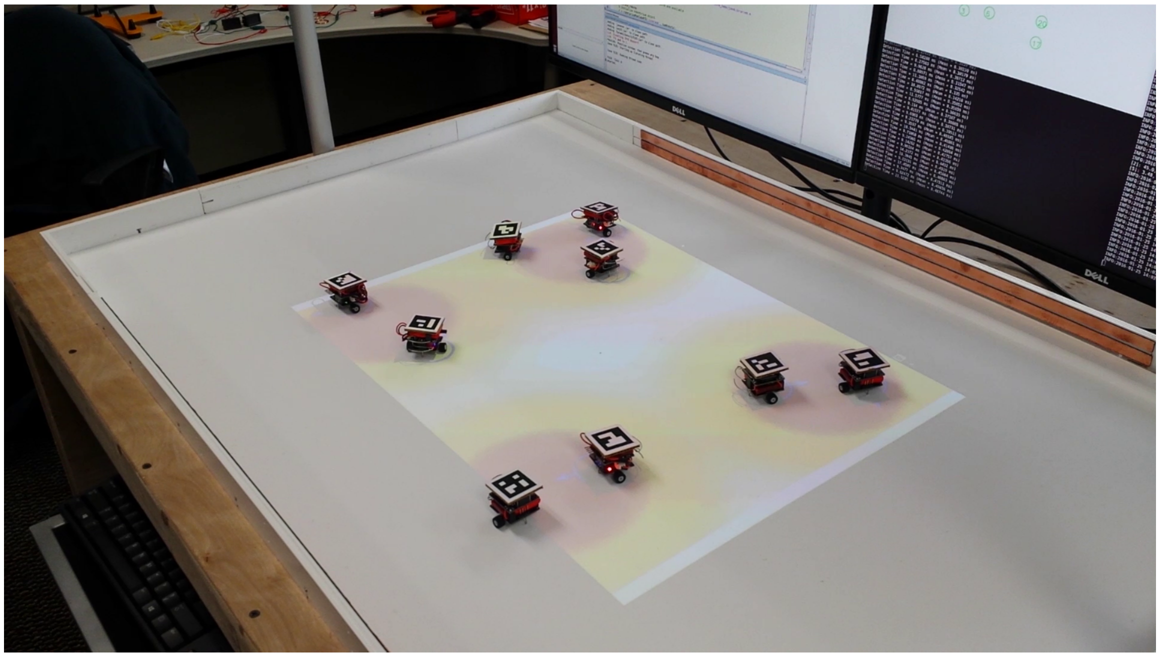
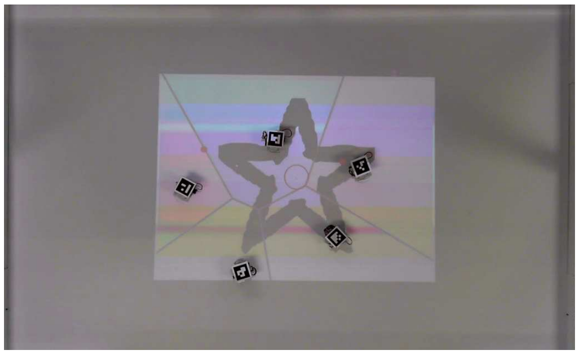


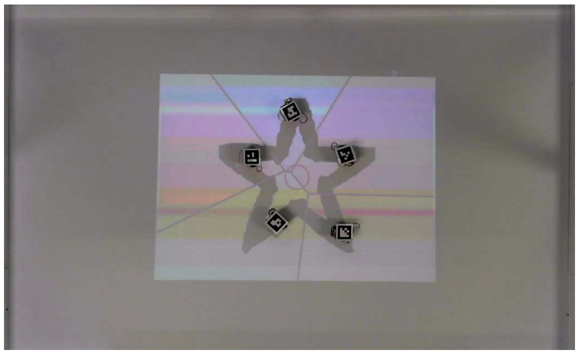
Figure 6. Nine *GRITSBots* being influenced by the *control of Gaussian functions* HSI scheme.

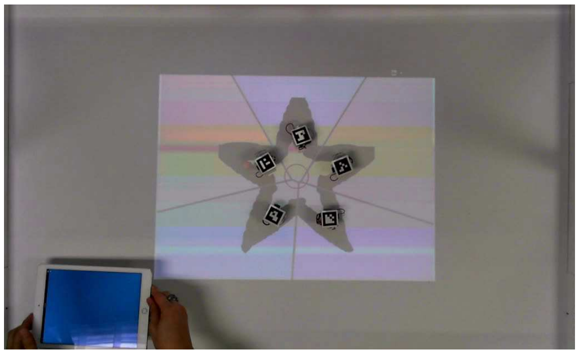
Both density design methods described in Section 3.4.2 and Section 3.4.1 were implemented on a smaller and lower-cost multi-robot system to compare the effectiveness of the methodology across platforms. For these experiments, the computer handles tracking of the robots based on standard webcams (1280 x 720 resolution) that detect ArUco tags for tracking (also used for virtual reality applications), and that runs at approximately 25 Hz. The tracking system was used to provide position and orientation data to the agents, and was also used to compute the Voronoi tessellation. The robotic platforms used for these experiments were GRITSBots[2] [39]. The GRITSBots possess a main processor that runs at 80 MHz and handles user code and WiFi data processing, and a co-processor for handling motor control running at 8 MHz, stepper motors for differential drive, odometry that does not require wheel encoders, and WiFi-based communication IEEE 802.11 B/G/N (up to 54 MBit/s) that uses UDP-based sockets. Matlab is used for visualization of the user-provided density function, the Voronoi tessellation, and the location of the density centroid. An overhead projector is used to project the visualization onto the real physical environment of the

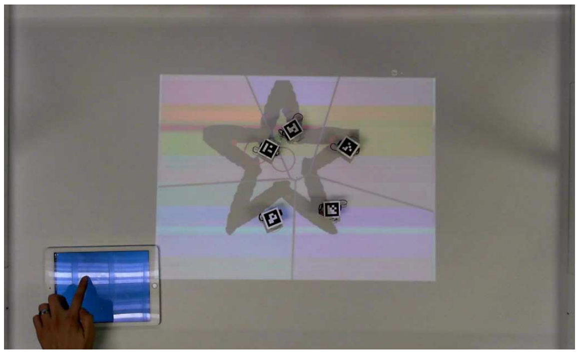
[2]For more information on the GRITSBots and their testbed, visit `www.robotarium.org`.

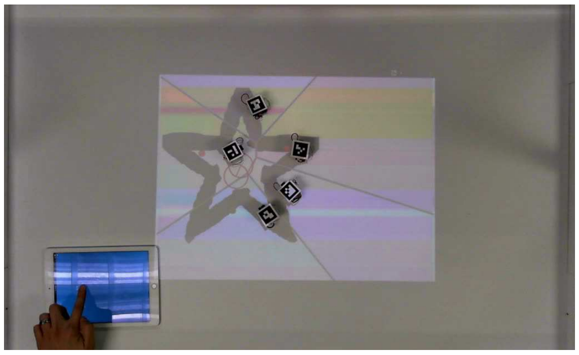
robots to give a real-time visual representation. Figure 6 illustrates the control of Gaussian functions HSI scheme described in 3.4.2, whereas Figure 7 illustrates the diffusion of drawn geometric configurations HSI scheme described in 3.4.1.


(a) $t = 0\ s$


(b) $t = 5\ s$


(c) $t = 20\ s$


(d) $t = 25\ s$


(e) $t = 30\ s$

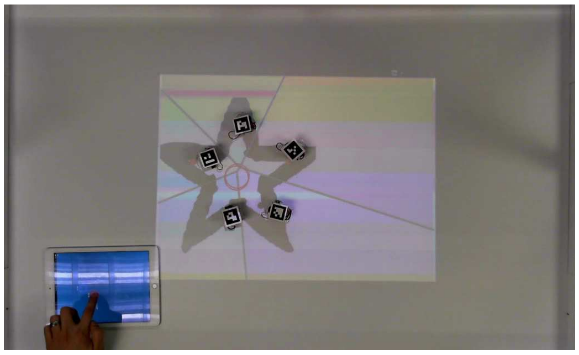

(f) $t = 35\ s$

Figure 7. Robotic implementation of a drawn star-shaped density on a team of GRITSBots. A tablet is later used (Figure 7c-7f) to directly influence the location of the swarm formation. An overhead projector is used to visualize in real-time the broadcasted density function, the Voronoi tessellation, and the current desired location of the swarm provided by the user through the tablet.

3.6 Concluding Remarks

In this chapter we constructed a framework for human-swarm interactions in which the human operator abstracts away the number of agents in the robot swarm and manipulates it as a whole by directly imposing the desired density of robots in the environment. In order to pursue this approach, contributions were made to the problem of coverage of time-varying functions – for which there was a dearth in the literature. In particular, important spectral properties are provided which are used to provide convergence guarantees for the presented control laws. In addition to these, a family of distributed approximations is presented which allow the agents to provide optimal coverage of the human-generated density functions while only relying on local information.

Although collision avoidance is not explicitly considered in this chapter, collisions are rarely observed during the robotic experiments. This is in part due to the nature of the coverage problem, in which the goal (to an extent) is for the agents to spread out. In fact, if agents are point-particles, agents will never collide given that they remain in the interior of their Voronoi cells which are non-overlapping. Practically, since agents are not point-particles, and depending on how concentrated the density function is, it is possible for the agents to get close enough to the boundary of their Voronoi cells to collide. A common way to handle collisions is to introduce artificial repulsive force terms to the agents' control law. Although this can be a simple modification to the system, it does not provide formal guarantees. There has been recent work on using *barrier certificates* to modify swarm controllers in a minimally invasive way that results in provably collision-free controllers that try to be as close as safely possible to the original controllers [81].

We are, however, not only interested in obtaining collision-free motion from the multi-robot system but also characterizing these interactions between agents. In some applications, it might even be of interest to enforce that agents interact with each other (e.g., coordinated sensing, collaborative assembly). The next chapter presents a framework for multi-robot motion planning that allows us to accomplish these objectives.

CHAPTER 4
MULTI-ROBOT MOTION PLANNING

In the previous chapter, we explored *human-swarm interactions* where a human operator is able to manipulate a large team of robots by imposing desired densities on the environment. As agents execute rich motion patterns, collision avoidance begins to play a bigger role the multi-robot system. In particular, we are interested in addressing and characterizing *inter-agent interactions* that can occur in multi-robot systems as an effect of rich motion patterns in a shared space (in which case collision avoidance is required) or due to mission objectives (e.g., for coordinated sensing or information exchange). In this chapter, we construct a framework for multi-robot motion planning for the sake of interactions which we call *multi-robot mixing* [82–84].

4.1 Inter-Robot Interactions in Multi-Robot Systems Using Braids

As discussed in Section 2.1, many applications have been proposed using multi-robot systems, e.g., multi-robot foraging, cooperative assembly, Multi-robot simultaneous localization and mapping, transportation systems, convoy protection. In many of these applications, the overall objectives can be stated in terms of making a team of robots follow a physical path, such as a road or the movements of a ground convoy, while ensuring that particular search patterns are executed [85–87]. These patterns should be selected in such a way that certain secondary geometric objectives are met, including ensuring that an area along the path is covered, that multiple views of the same objects are achieved, that an aerial vehicle is always on top of the convoy, or that a sufficient number of vehicle-to-vehicle interactions take place for the purpose of information sharing [88, 89]. In this work, we collect all of these different secondary objectives under one unified banner, namely *multi-robot mixing*. In particular, we specify interaction patterns with certain desired levels of mixing (i.e., interactions between agents), and then proceed to generate the actual cooperative movements that realize these mixing levels.

In this chapter, we study the problem of characterizing inter-robot interactions for the sake of coordination and collision avoidance. We specify the mixing patterns through elements of the so-called braid group [59, 60], where each element corresponds to a particular interaction pattern. We do not focus on a particular pattern *per se*, but rather on the problem of being able to execute a whole class of patterns. The result from such a construction is a hybrid system driven by symbolic inputs [57], i.e., the braids, that must be mapped onto actual paths that both obtain the mixing level specified through the braid, and remain safe in the sense that collisions are avoided.

As stated in Section 2.3, the use of braids for robot motion planning has been considered before. Using the notion of configuration space [90] for robot motion planning, [91] studies the problem of construction and classification of configuration spaces for graphs, e.g., robots on a manufacturing floor constrained on rails or paths. By studying the topological data associated with these graphs, such as the braid groups, he is able to provide a measure of the complexity of the control problem (e.g., the construction of potential field controllers on homeomorphic spaces).

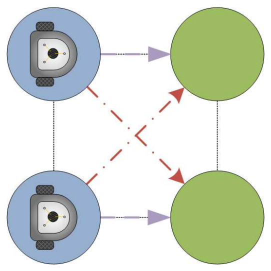
In [61], the *graph braid group* is used as the fundamental group of the configuration space of graphs that describe robot motion. There, each graph in the configuration space represents discretized collision-free robotic motion plan (e.g. road maps), where at each discrete time the graph vertices represent the positions of robots and (possibly moving) obstacles, and the edges represent fixed tracks connecting these vertices. They present an algorithm to construct the presentation of the graph braid group of N agents, where the group generators (i.e., the braids) represent actual paths between configurations of robots on the graph, i.e., the motion plan to transition from one configuration to another. However, they only consider zero-size robots where they rely on a "one edge separation" between points at all times to avoid collisions, and as such, their approach is purely academic and mainly focuses on the combinatorics of ideal robots moving on a graph.

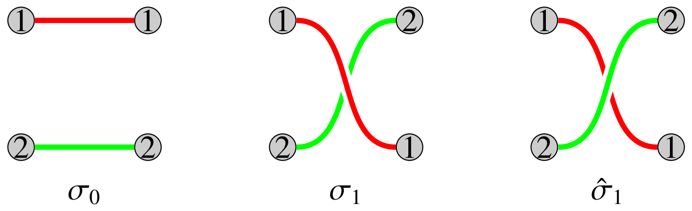
The research presented in this chapter is the first approach that uses braids in multi-robot motion planning to symbolically characterize and enforce rich interaction patterns, implementable on actual robotic platforms. In particular, we extend the notions presented in [82–84]. The definition on planar braids and their geometric interpretations are generalized from their original form presented in [82]. A controller was designed for nonholonomic vehicles to track a geometric path in [83]. In this chapter, we modify this controller to instead produce optimal trajectories that are provably safe in the collision-free sense, satisfy a set of spatio-temporal constraints, and follow desired geometric paths (Section 4.2.2). This also allows us to obtain tighter bounds on the amount of interaction patterns that are achievable in a space than the one found in [83]. We also provide novel contributions in Section 4.4, where these trajectory-generating controllers are extended to non-rectangular regions, and in Section 4.5.1 we provide an optimal trajectory-tracking controller with formal guarantees on optimality and spatio-temporal constraint satisfaction.

The outline of this chapter is as follows: in Section 4.1.1, the braid group is introduced as a way of specifying mixing levels, and in Section 4.1.2, the corresponding symbolic braid objects are given a geometric interpretation in terms of planar robot paths; controllers are then presented so that mixing strategies satisfying a given specification can be executed by a class of robots as described in Section 4.2, together with bounds on the highest achievable mixing levels for a given space. Section 4.3 introduces a new specification language in order to specify rich, temporally-layer tasks. In Section 4.4, the trajectory generating controllers are extended to non-rectangular regions. In Section 4.5, implementation of these controllers on actual robotic platforms is addressed, and these ideas are deployed on a team of actual mobile robots.

4.1.1 Planar Braids

Consider two agents on a square, initially located at the two left vertices of the square as in Figure 8a. The agents' task is to move to the two right vertices of the square. There are two ways in which these target vertices can be assigned. The first is to simply let the


(a) Two robots, originally at one side of a square, map to the other.



(b) Three braids for two planar agents. Note that "strand" 1 goes over "strand" 2 in σ_1 and vice versa in $\hat{\sigma}_1$.

Figure 8. Two-robot interactions and corresponding symbolic braids.

robots move along a straight line while the second is to have them cross paths and move to vertices diagonally across from their initial placement. If the robots are not to collide with each other, one agent can cross the intersection of the two paths first, and then the other (or vice versa). In the braid group, these two options correspond to different "braids," and we have thus identified three planar braids for two agents, as shown in Figure 8b. Let us momentarily denote these three braids, $\sigma_0, \sigma_1, \hat{\sigma}_1$.

Now, given these three braids, we can concatenate them together to form other braids, as seen in Figure 9. The left braid is given by $\sigma_1 \cdot \sigma_1$ and the right braid is $\sigma_1 \cdot \hat{\sigma}_1$. In the braid group, what really matters is not the geometric layout of the paths, but how the paths wrap around each other. As can be seen, if we were to "pull" the right corners in $\sigma_1 \cdot \sigma_1$, the "strands" would get "tangled" in the middle, while a "stretched-out" $\sigma_1 \cdot \hat{\sigma}_1$ is simply σ_0. Thus we let σ_0 be the identity braid, such that σ_1 and $\hat{\sigma}_1$ are each others' inverses in the sense that

$$\sigma_1 \cdot \hat{\sigma}_1 = \hat{\sigma}_1 \cdot \sigma_1 = \sigma_0.$$

In fact, every braid has an inverse and, as such, the set of braids (together with the concatenation operation) is indeed a group. And, as $\sigma_1^{-1} = \hat{\sigma}_1$ (and $\sigma_1^{-1} = \hat{\sigma}_1$), σ_0 and σ_1 (or σ_0 and $\hat{\sigma}_1$ for that matter) are the so-called generator braids for this group in that all planar braids can be written as concatenations of these two braids and their inverses [59, 60].

This notion of generator braids can be extended to the case when there are $N \geq 2$, with the only difference being that we will have N generators rather than just two, i.e., let σ_0 be

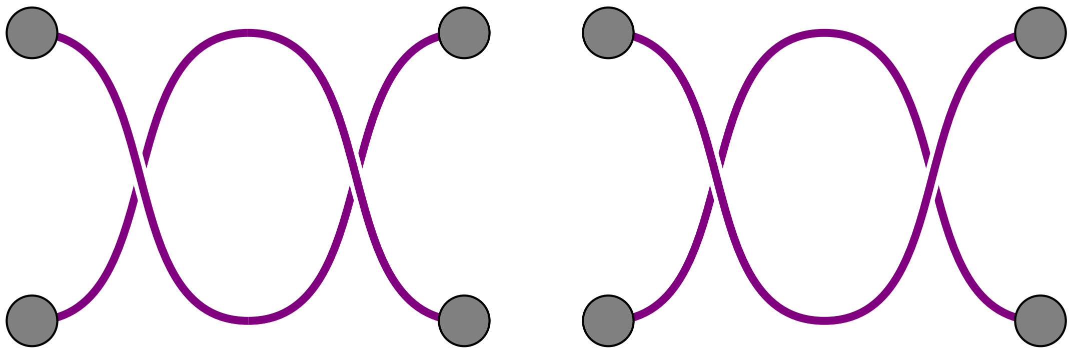


Figure 9. Two concatenated braids $\sigma_1 \cdot \sigma_1$ and $\sigma_1 \cdot \hat{\sigma}_1$. The latter of these two braids is the same as the identity braid σ_0.

the trivial generator with no interactions and σ_k denote the interactions between agents k and $k+1$, $k = 1, \ldots, N-1$. If we let Σ_N be the set of all planar generator braids over N agents, this set will serve as the alphabet over which braid strings (themselves braids) are produced from, and we let Σ_N^M denote the set of all braids of length M (i.e., braids composed of M generators) over N agents.

4.1.2 A Geometric Interpretation

Although heavily inspired by geometry in [59], the braid group is not concerned with the actual geometry of the braid strands. For the sake of describing the robot motion plans, we will associate geometric paths with the different braids. First of all, we assume that the braid is geometrically located in a rectangular area of height h and length ℓ no matter how long the braid string is. Using the particular two-agent braids discussed in the previous paragraphs, we assume that the two agents are initially located at the points $(0,0)$ and $(0,h)$, while the final locations are at $(\ell, 0)$ and (ℓ, h). If the total braid results from the use of one single-generator braid σ_0, or σ_1 then no additional points are needed. However, if the braid has length 2, then we also need to introduce intermediary *half-way* points $(\ell/2, 0)$ and $(\ell/2, h)$. As such, we let $\mathcal{P}_2^q = \left\{\left(\frac{q}{M}\ell, 0\right), \left(\frac{q}{M}\ell, h\right)\right\}$ be a set of uniformly spaced[1] positions

[1]Unless otherwise stated, the rest of this document we will assume these sets of intermediary points are uniformly spaced in h and ℓ, but the notions presented here extend to points which are not, as the application demands. More on this in Section 4.4.

for these intermediary points, where the subscript 2 denotes the two-agent case, M is the length of the braid to be executed, and $q = 0, 1, \ldots, M$.

Using this notation, we can refer back to Figure 8a and say that each of the two generator braids correspond to an assignment, i.e., a bijective map, between $\mathcal{P}_2^0$ and $\mathcal{P}_2^1$, and we use the following notation to denote this fact

$$\sigma_i : \mathcal{P}_2^0 \rightarrow_b \mathcal{P}_2^1, \;\; i = 0, 1,$$

where $\rightarrow_b$ denotes *bijection*. Note that this is a slight abuse of notation in that σ_i now denotes both an element in the braid group as well as a map – this distinction, however, should be entirely clear from the context. Further, we will refer to these points which agents are bijectively mapped to and from as *braid points*.

If we generalize this to $N \geq 2$ agents and let σ denote a string of generators of length $M \geq 2$, i.e., $\sigma \in \Sigma_N^M$, we will use the notation

$$\sigma(k) : \mathcal{P}_N^{(k-1)} \rightarrow_b \mathcal{P}_N^k, \quad k = 1, \ldots, M,$$

where $\sigma(k)$ is the k^{th} braid[2] in the string σ and

$$\mathcal{P}_N^q = \left\{ \left(\tfrac{q}{M}\ell, 0\right), \left(\tfrac{q}{M}\ell, \tfrac{1}{(N-1)}h\right), \left(\tfrac{q}{M}\ell, \tfrac{2}{(N-1)}h\right), \ldots, \left(\tfrac{q}{M}\ell, h\right) \right\},$$

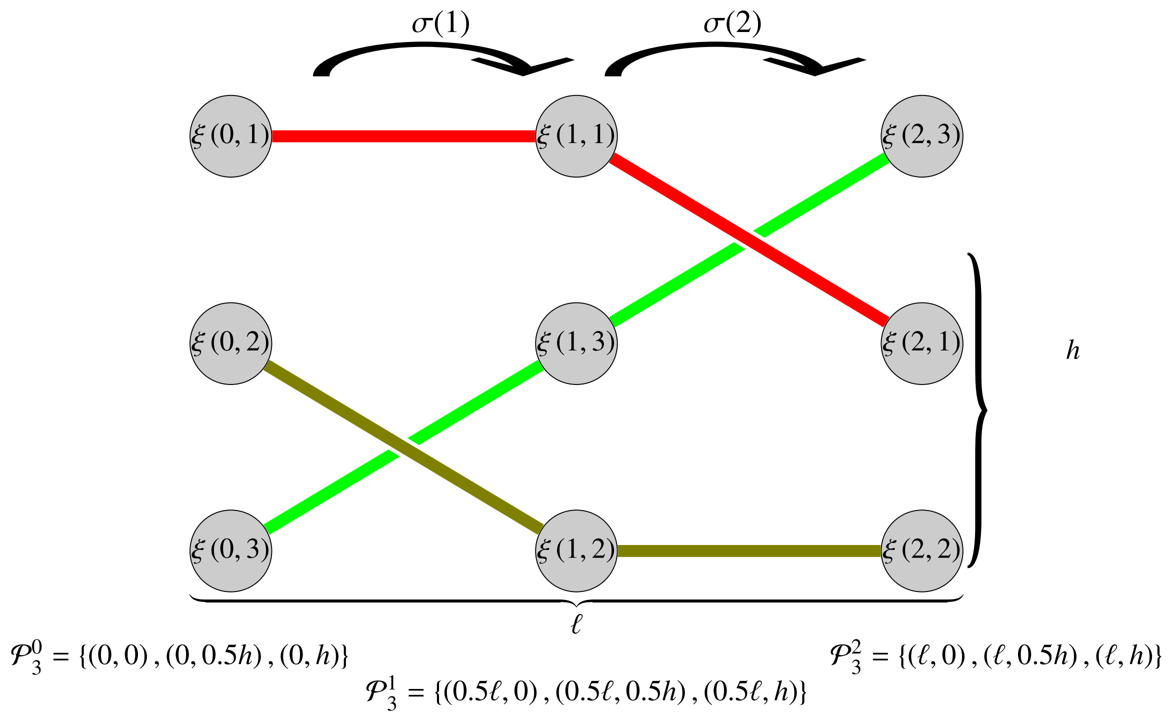
as shown for the three-agent case in Fig. 10.

We moreover use the notation $\xi(i, j) \in \mathbb{R}^2$ to denote the point agent j should go to at step i, $i = 1, \ldots, M$. We use the convention that $\xi(0, j) = (0, (j-1)h/(N-1))$, $j = 1, 2, \ldots, N$, to denote agent j's initial position. In other words,

$$\xi(1, j) = \sigma(1)\langle \xi(0, j) \rangle,$$

$$\xi(2, j) = \sigma(2)\langle \xi(1, j) \rangle = \sigma(2) \circ \sigma(1)\langle \xi(0, j) \rangle,$$

[2]Note that a *braid* here refers to a member of the braid group, e.g., a single generator (i.e., a single bijective map) or a concatenation of several generators (i.e., a composition of bijective maps).



$\sigma(1)$
$\sigma(2)$
$\xi(0,1)$
$\xi(1,1)$
$\xi(2,3)$
$\xi(0,2)$
$\xi(1,3)$
$\xi(2,1)$
h
$\xi(0,3)$
$\xi(1,2)$
$\xi(2,2)$
ℓ
$\mathcal{P}_3^0 = \{(0,0),(0,0.5h),(0,h)\}$
$\mathcal{P}_3^2 = \{(\ell,0),(\ell,0.5h),(\ell,h)\}$
$\mathcal{P}_3^1 = \{(0.5\ell,0),(0.5\ell,0.5h),(0.5\ell,h)\}$

Figure 10. The geometric interpretation of braid string $\sigma = \sigma_2 \cdot \sigma_1$ for the three-agent case. In this example, $\sigma(1) = \sigma_2$ and maps $\mathcal{P}_3^0$ to $\mathcal{P}_3^1$, while $\sigma(2) = \sigma_1$ and maps $\mathcal{P}_3^1$ to $\mathcal{P}_3^2$.

or more generally,

$$\begin{aligned}\xi(i,j) &= \sigma(i)\langle\xi(i-1,j)\rangle \\ &= \sigma(i) \circ \sigma(i-1) \circ \cdots \circ \sigma(1)\langle\xi(0,j)\rangle,\end{aligned}$$

where we use the $\langle\cdot\rangle$ notation to denote the argument to $\sigma(i)$ and $\circ$ to denote composition. This construction is also illustrated in Figure 10 for the three-agent case.

The geometric interpretation we will make of the planar braids is that the mobile agents that are to execute them must traverse through these braid points. They must moreover do so in an orderly and safe manner, which will be the topic of the next section.

4.2 Braid Controllers

Given a collection of N agents with dynamics

$$\dot{x}_j = f(x_j, u_j),$$

and planar output

$$y_j = h(x_j) \in \mathbb{R}^2, \quad j = 1, \ldots, N,$$

then it is of interest to have these agents execute a braid $\sigma \in \Sigma_N^M$. We now define what it means for this braid to be executed.

Given an input braid string σ, what each individual agent should do is "hit" the intermediary braid points $\{\xi(i, j)\}_{j=1}^N$ at specified time instances t_i. We let T denote the time it should take for the entire string to be executed. As such, the first condition for a multi-agent motion to be feasible with respect to the braid is the following:

Definition 4.2.1 (Braid-Point Feasibility)**.**
A multi-robot trajectory is braid-point feasible *if*

$$y_j(t_i) = \xi(i, j), \; i = 0, \ldots, M, \; j = 1, \ldots, N.$$

where the t_i form a partition of a given time window $[0, T]$*, i.e.,*

$$t_0 = 0 < t_1 < \cdots < t_i < \cdots < t_M = T. \qquad \diamond$$

On top of braid-point feasibility, we also insist that the robots do not collide as they maneuver. To a certain degree, this condition is what restricts the level of *mixing* that is possible, i.e., since the braid is constrained in a rectangle of fixed height and width, what length strings the multi-robot system can execute while maintaining a desired level of safety separation.

Definition 4.2.2 (Collision-Free)**.**
A multi-robot trajectory is collision-free *if*

$$\|y_i(t) - y_j(t)\| \geq \delta_{ij}, \quad \forall i \neq j, \quad t \in [0, T],$$

where $\delta_{ij} > 0$ is the desired level of safety separation between agents i and j. $\diamond$

For convenience, we will refer to $\bar{\delta} = \max \delta_{ij}$ as the *maximum safety separation* such that no agent collides. This will come up in Theorem 4. We are missing a notion to describe what the multi-robot mixing problem is, that is, what constitutes a *braid controller*.

Definition 4.2.3 (Braid Controller)**.**
A multi-robot controller is a braid controller *if the resulting trajectories are both braid-point feasible and collision free, for all collision-free initial conditions such that*

$$y_j(0) = \xi(0, j), \;\; j = 1, \ldots, N. \qquad \diamond$$

As a final notion, we are interested in how much mixing a particular system can support.

Definition 4.2.4 (Mixing Limit)**.**
The mixing limit $M^\star$ *is the largest integer* M *such that there exists a braid controller for every string in* Σ_N^M. $\diamond$

Whenever two strands of the braid associated with a given braid string cross, the two associated agents will have to interact. The mixing limit therefore serves as an input-independent bound on how much mixing is achievable for a given team of agents operating in a given environment. The mixing limit is in general quite hard to compute; it needs to consider every permutation of strings of varying lengths up to some number, the geometry assigned to each string, and is dependent on the kinematical response of the multi-robot system. However, under certain assumptions it is possible to find bounds on $M^\star$ for a given braid controller.

4.2.1 Braid Controllers: Stop-Go-Stop Hybrid Strategy

Our first attempt at executing braids will be a hybrid control strategy called the *Stop-Go-Stop*. We will assume that agent dynamics are given by single integrator dynamics, i.e., $\dot{x}_j = u_j \in \mathbb{R}^2$ with $y_j = x_j$, $j = 1, \ldots, N$. Further, for practical considerations, assume that there is a cap on the maximum velocity achievable by the agents, i.e., $\|u_j\| \in [0, v_{\max}]$. The idea is then at each braid step to send agents off straight to their next braid point in order of the distance they'll need to travel, waiting just long enough to avoid collisions before sending an agent off. To that end, we define $s_i : \{1, \ldots, N\} \to \{1, \ldots, N\}$ to be a bijective mapping that denotes the farthest distance ordering at step i, that is

$$s_i(p) < s_i(q) \qquad \Longrightarrow \qquad \|\xi(i,p) - \xi(i-1,p)\| \geq \|\xi(i,q) - \xi(i-1,q)\|$$

where ties are arbitrarily broken such that s_i remains a bijection. We let the agents heading angle be given by $\theta_{i,j} = \tan^{-1}\left(\frac{(\xi(i,j)-\xi(i-1,j))_2}{(\xi(i,j)-\xi(i-1,j))_1}\right)$ where the subscript indicates the first or second component, and the unit heading vector be $\hat{\rho}_{i,j} = \left[\cos(\theta_{i,j}), \sin(\theta_{i,j})\right]^T$. Lastly, the time the agents will wait before entering *GO* mode will be given by their ordering as $(s_i(j) - 1)\tau$ where

$$\tau = \frac{\delta}{v_{\max}\cos(\theta^*)}$$

is the time required to be δ apart horizontally[3], with $\cos(\theta^\star) = \frac{\ell/M}{\sqrt{\ell^2/M^2+h^2}}$ being the maximum horizontal distance an agent could travel given $\sigma \in \Sigma_N^M$. To ensure that the agents do not overtake the first agent (horizontally), the speed of the agents should be scaled by the velocity of the first agent, i.e., $u_j = v_{\max}\cos(\theta_{i,s_i^{-1}(1)})/\cos(\theta_{i,s_i(j)})$. The hybrid automaton describing the stop-go-stop controller is given in Figure 11.

Theorem 3. *[82] The STOP-GO-STOP controller in Figure 11 is a braid controller if the braid points themselves are sufficiently separated and*

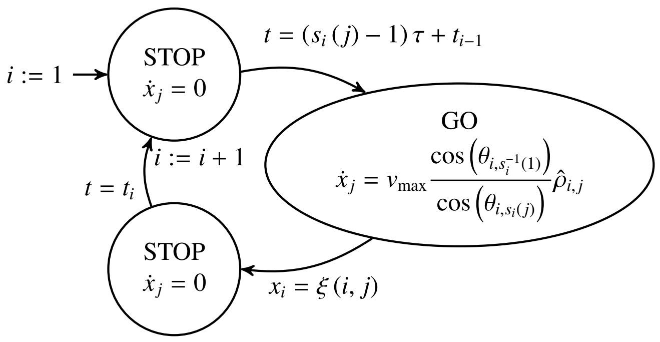
$$\cos(\theta^\star)v_{\max}\left(\min_i (t_i - t_{i-1}) - (N-1)\tau\right) \geq \sqrt{\ell^2/M^2 + h^2}.$$

Proof:

The STOP-GO-STOP controller ensures that the agents are never within δ of each other by virtue of the fact that they have to wait until they are indeed at least that far apart (horizontally) before entering GO mode. As such, the trajectories are collision-free.

What remains to show is that they are also braid-point feasible. Consider the agent that has to wait the longest before it can move, i.e., the agent that has to wait a total of

[3]Since we are interested in the mixing limit, the analysis is done with the horizontal direction being the limiting factor for safe execution of the braids.



$i := 1$
STOP
$\dot{x}_j = 0$
$t = (s_i(j) - 1)\tau + t_{i-1}$
GO
$\dot{x}_j = v_{\max} \frac{\cos\left(\theta_{i,s_i^{-1}(1)}\right)}{\cos\left(\theta_{i,s_i(j)}\right)} \hat{\rho}_{i,j}$
$i := i + 1$
$t = t_i$
STOP
$\dot{x}_j = 0$
$x_i = \xi(i, j)$

Figure 11. Hybrid STOP-GO-STOP braid controller.

$(N-1)\tau$, and at its worst has $\min_i (t_i - t_{i-1}) - (N-1)\tau$ time left to reach the next braid point. In other words, we need that the distance traveled in that amount of time at the speed $v_{\max} \cos(\theta_{i,s_i^{-1}(1)}) / \cos(\theta_{i,s_i(j)})$ is greater than the distance required. But, we note that

$$v_{\max} \cos(\theta_{i,s_i^{-1}(1)}) / \cos(\theta_{i,s_i(j)}) \geq v_{\max} \cos(\theta^\star)$$

and, as we are only looking for a bound, we assume that we use this lower speed and that the distance required to travel is the largest distance possible (which it really is not). In other words, we need

$$\cos(\theta^\star) v_{\max} \left(\min_i (t_i - t_{i-1}) - (N-1)\tau \right) \geq \sqrt{\ell^2/M^2 + h^2},$$

and the proof follows. □

Note that Theorem 3 implicitly provides a lower bound on the mixing limit as long as the agents' paths are straight lines. In Figure 12 we can see the mixing limit as a function of the number of agents in the team for parameters $v_{\max} = 5$, $T = 20$, $\ell = 5$, $h = 10$, and $\Delta = 0.2$. The problem with this strategy is that it does not allow for more general geometric paths (which could be interpreted as feasible trajectories under a given robot dynamical model), nor does it ensure that agents get within a specified distance from each other (which could be necessary when collaboration is required). The next section we present a new strategy called *braid reparameterization* which will explicitly consider more general strand geometries, and allow us to obtain analytical bounds on the mixing limit.

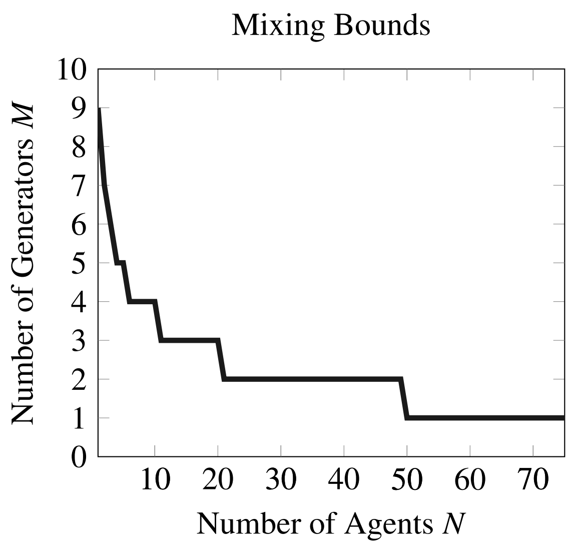


Figure 12. Lower bound on the mixing limit using the Stop-Go-Stop braid controller.

4.2.2 Braid Controller: Braid Reparameterization

We are now seeking a strategy that will allow us to follow a given geometry while achieving a mixing strategy encoded as a braid string $\sigma \in \Sigma_N^M$. Further, we wish to enforce inter-agent interaction as dictated in σ by having agents get as close as the safety separation δ_{jk}. Before moving forward, consider the following lemma.

Lemma 5. *If at any braid step, the generators in the braid substring $\sigma(k) = \sigma \subseteq \Sigma_N^m$, $m \leq M$, have indices that are two or more apart, then any agent interacts with at most one other agent at this step.*

Proof:

Let $\mathcal{B}_N^k \in \mathbb{R}^{N\times 2}$ be a matrix that contains the set of braid points at time k such that $\mathcal{B}_N^0 = [\xi(0,1)\cdots\xi(0,N)]^T$. Consider the two-generator concatenation $\sigma_i \cdot \sigma_j$. As a bijective map, if $\sigma_i = \sigma_0$, then the agents do not interact, and the agent in braid point position $[\mathcal{B}_N^{k-1}]_n$ gets mapped to braid point position $[\mathcal{B}_N^k]_n$, where $[\mathcal{B}]_n$ corresponds to the n^{th} row of $\mathcal{B}$, $n = 1,\ldots,N$. If $\sigma_i \neq \sigma_0$, then σ_i will swap the position of the two agents occupying the braid point positions i and $i+1$ at step $k-1$, i.e., it maps the agent occupying $[\mathcal{B}_N^{k-1}]_i$ to $[\mathcal{B}_N^k]_{i+1}$ and the agent occupying $[\mathcal{B}_N^{k-1}]_{i+1}$ to $[\mathcal{B}_N^k]_i$. Similarly, σ_j swaps the position of the agents occupying the braid point positions j and $j+1$.

The two agents in positions i and $i+1$ at $k-1$ would only interact with the two agents in positions j and $j+1$ at $k-1$ if i (or $i+1$ for that matter) is equal to either j or $j+1$. But if we let $|i-j| \geq 2$, then we get that

$$\begin{aligned} |i-j| \geq 2 &\Leftrightarrow 2 \leq i-j \text{ or } 2 \leq j-i \\ &\Rightarrow j < j+1 < j+2 \leq i < i+1 \\ &\text{or } \; i < i+1 < i+2 \leq j < j+1 \\ &\Rightarrow \begin{cases} i \neq j \\ i \neq j+1 \\ i+1 \neq j \\ i+1 \neq j+1, \end{cases} \end{aligned}$$

and as such these two-generator concatenation maps the agents from one set of braid points to the next with at most two interaction between at most two agents per interaction.

One of the two braid group relations [59, 60] tells us that if $|i-j| \geq 2$ then $\sigma_i \cdot \sigma_j = \sigma_j \cdot \sigma_i$. More generally, if we let

$$h : \{1, \ldots, m\} \to \{1, \ldots, N-1\}$$

be a surjective map such that $|h(i) - h(j)| \geq 2$ for all $i, j \in \{1, \ldots, m\}$ with $i \neq j$, then for *any* bijective map $g : \{1, \ldots, m\} \to \{1, \ldots, m\}$ we have that

$$\sigma(k) = \sigma_{h(1)} \cdot \sigma_{h(2)} \cdot \cdots \cdot \sigma_{h(m)} = \sigma_{h(g(1))} \cdot \sigma_{h(g(2))} \cdot \cdots \cdot \sigma_{h(g(m))}.$$

As such, the braid generators can be rearranged to obtain any permutation of two-generator concatenations from generators in $\sigma(k)$. Since for all permutations of two-generator concatenations we will have indices that are two or more apart, these will map the agents at braid step k from the set of braid points $\mathcal{B}_N^{k-1}$ to the next set of braid points $\mathcal{B}_N^k$ with at most m interactions total, and at most one interaction per agent. $\square$

If the geometric interpretation of the braid string is restricted to the case of only pairwise interactions at every braid step, then it is possible to devise a hybrid strategy with the desired properties, which we can then compose together to achieve the desired interaction patterns. As such, we will restrict the geometric interpretation of braid strings to those satisfying pairwise interactions as in Lemma 5.

Restriction 1 (Geometric interpretation of a given braid). *As a bijective map from one set of braid points to another, the braid $\sigma(k)$ will only contain generators whose indices are two or more apart, i.e., for some $h : \mathbb{N} \to \{0, \ldots, N-1\}$*

$$\sigma(k) = \sigma_{h(1)} \cdot \sigma_{h(2)} \cdot \cdots, \quad \text{where } |h(i) - h(j)| \geq 2 \; \forall i \neq j$$

Note that this is not a restriction on which braids strings we will consider, but on how many braid steps we will need to execute the interaction pattern encoded in the braid string.

As an example, suppose that a braid string contains the substring of three concatenated generators $\sigma_1 \cdot \sigma_3 \cdot \sigma_2$. The first two generators, $\sigma_1 \cdot \sigma_3$, may take place simultaneously at braid step 1 without incurring in more than one interaction per agent with another agent, but we would require an additional braid step for σ_2 in order to avoid multiple interactions in the same step. This is illustrated geometrically in Figure 13. In (a), an agent interacts with more than one other agent since all the generators are not at least two indices apart and take place at the same braid step. Note that in the remaining cases (b)-(d), we introduce intermediate braid points while retaining the desired level of interaction, final configuration of the agents, and reducing to pairwise interactions.

Braid strings that satisfy Restriction 1 will have at most interactions involving two agents at any given braid step. So in order to safely execute braids, one need only consider the case when two agents interact. We will devise a strategy for reparameterizing the geometry such that should two agents interact, the resulting parameters are at least δ from each other. By tracking the parameterized paths, the agents' trajectories will be collision-free and braid-point feasible. The resulting controllers can be combined to satisfy a given braid

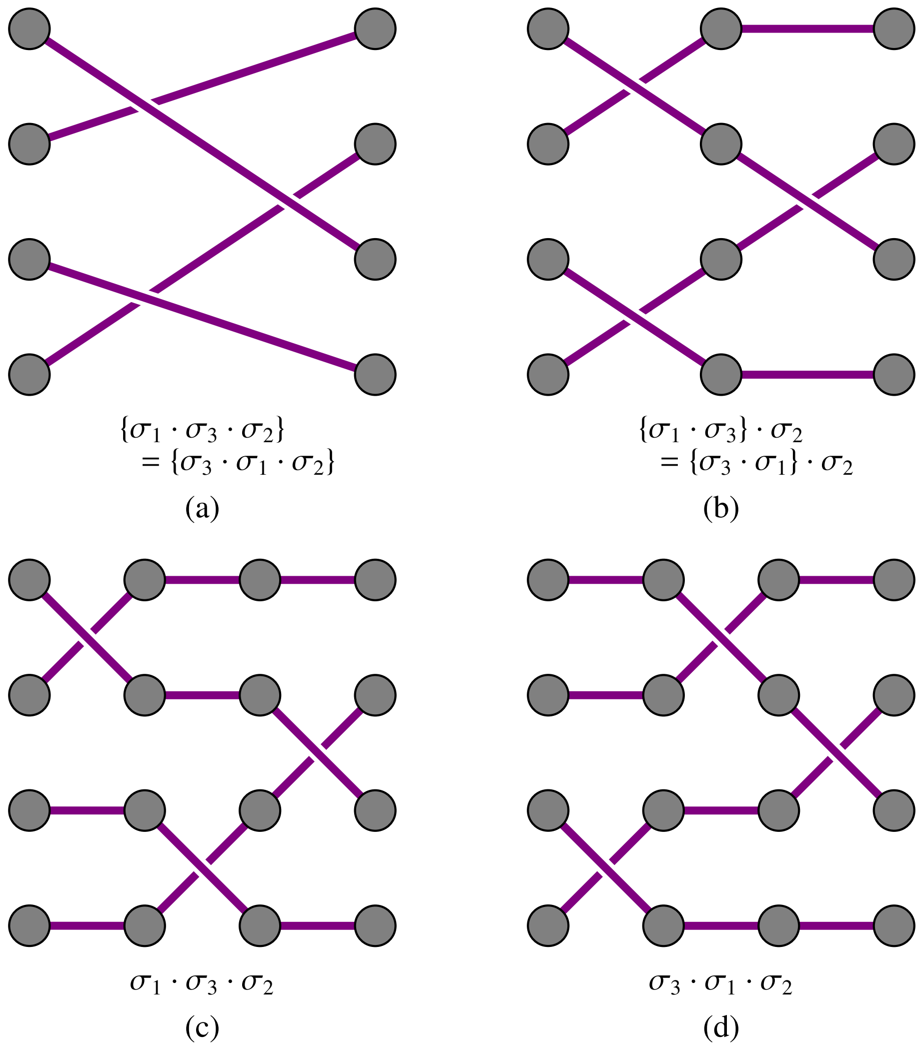

$\{\sigma_1 \cdot \sigma_3 \cdot \sigma_2\}$
$= \{\sigma_3 \cdot \sigma_1 \cdot \sigma_2\}$
(a)
$\{\sigma_1 \cdot \sigma_3\} \cdot \sigma_2$
$= \{\sigma_3 \cdot \sigma_1\} \cdot \sigma_2$
(b)
$\sigma_1 \cdot \sigma_3 \cdot \sigma_2$
(c)
$\sigma_3 \cdot \sigma_1 \cdot \sigma_2$
(d)

Figure 13. A braid string $\sigma = \sigma_1 \cdot \sigma_3 \cdot \sigma_2$ taking place in a varying number of braid steps.

string.

Consider the geometric path agent j must follow at step i given by $\gamma_i^j : [0, 1] \to \mathbb{R}^2$ with $\gamma_i^j(0) = \xi(i-1, j)$ and $\gamma_i^j(1) = \xi(i, j)$. Let Δ be the arclength of this path, given by

$$\Delta = \int_0^1 \sqrt{\left(\dot{\gamma}_i^j(p)\right)^T \dot{\gamma}_i^j(p)}\, \mathrm{d}p.$$

We wish to find a parameterization of the strand geometries such that the parameters of two intersecting strands, thought of as virtual vehicles that live on their geometries, are collision free. The strategy to do so will be to impose constraints on the agents' separation from the path intersection at a specified time. To ensure braid-point feasibility, the strategy will be to impose constraints on the time in which the agents must get from the beginning of one braid step to the end of that step. To that end, we construct the following constrained optimization problem

$$(v_j^\star, v_k^\star) = \underset{(v_j,v_k)}{\arg\min}\, J(v_j, v_k) = \underset{(v_j,v_k)}{\arg\min}\, \frac{1}{2}\int_{t_{i-1}}^{t_i} v_j^2 + v_k^2 \,\mathrm{d}\tau \tag{21}$$

subject to

$$\begin{aligned} &\dot{p}_j = v_j, && p_j(t_{i-1}) = 0, \\ &p_j\left(\frac{t_{i-1}+t_i}{2}\right) = \frac{\Delta+\delta}{2\Delta}, && p_j(t_i) = 1, \end{aligned}$$

and

$$\begin{aligned} &\dot{p}_k = v_k, && p_k(t_{i-1}) = 0, \\ &p_k\left(\frac{t_{i-1}+t_i}{2}\right) = \frac{\Delta-\delta}{2\Delta}, && p_k(t_i) = 1. \end{aligned}$$

The constraints at t_{i-1} and t_i will ensure that the reparameterization is braid-point feasible. For collision avoidance we will define a safety separation region, as depicted in Figure 14, to be the region from the intersection point to the point where it is possible for the two agents to be within δ_{jk} of each other, i.e., the set $\left[\frac{\Delta-\delta}{2\Delta}, \frac{\Delta+\delta}{2\Delta}\right]$ for $\delta \in [0, \Delta]$ such that $\left\|\gamma_i^j(p_j) - \gamma_i^k(p_k)\right\| \geq \delta_{jk}$ for all $p_j, p_k \in [0, 1]$, where we let δ be the distance from the intersection point to the boundary of the safety separation region along the path. Note that if the geometry of the braid strands are straight lines, the distance δ can be easily computed by $\delta = \delta_{jk}\csc(\theta)$, where θ is the angle between the two intersecting lines, and this would also imply the parameters get as close as $\delta_{jk}\csc(\theta/2)$. Similarly, when the geometry is *city-block*-like, we can find $\delta = \delta_{jk} + \frac{h}{2(N-1)}$ and the agent get as close as $2\delta_{jk}$. For more general geometries, δ may be conservatively selected. At $\bar{t}_i := \frac{t_{i-1}+t_i}{2}$, i.e., half-time along the braid step, we will require one parameter to exit the safety separation region as the other one is about to enter it. All these cases are illustrated in Figure 14. This way we guarantee that the two parameters are never inside the region simultaneously and thus their separation will always be of at least δ_{jk}.

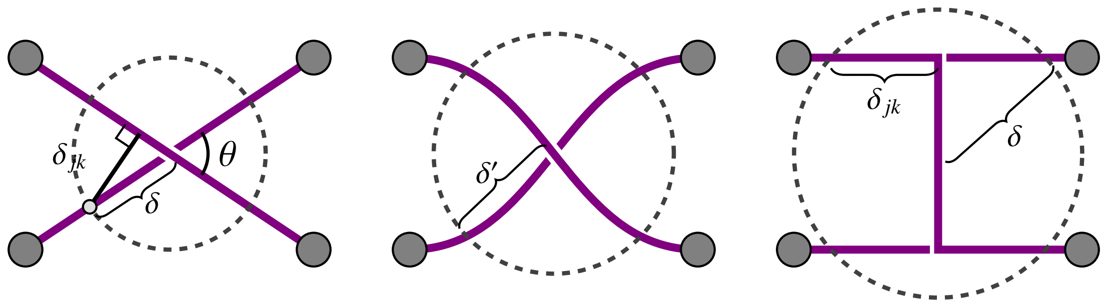


Figure 14. Safety separation region for three different geometry, two-agent braids. For straight lines (left) or *city-block*-like (right), the distance δ can be computed exactly. For arbitrary curves (center), the distance δ' can be selected conservatively.

The optimality conditions for equation (21) are given by

$$\dot{\lambda}_j = \dot{\lambda}_k = 0, \qquad \dot{p}_j = -\lambda_j, \qquad \dot{p}_k = -\lambda_k,$$

$$p_j(\bar{t}_i) = \tfrac{\Delta+\delta}{2\Delta}, \qquad p_k(\bar{t}_i) = \tfrac{\Delta-\delta}{2\Delta}, \qquad p_j(t_i) = p_k(t_i) = 1.$$

Since $\dot{\lambda}_j = \dot{\lambda}_k = 0$, we will get that λ_j and λ_k are piecewise constant. By using the midway condition, we have that for $t \in (t_{i-1}, \bar{t}_i]$

$$p_j(\bar{t}_i) = \int_{t_{i-1}}^{\bar{t}_i} v_j \, \mathrm{d}\tau = \frac{\Delta+\delta}{2\Delta}$$
$$\Rightarrow \qquad v_j(\bar{t}_i - t_{i-1}) = \frac{\Delta+\delta}{2\Delta}$$
$$\Rightarrow \qquad v_j = \frac{1}{\Delta}\frac{\Delta+\delta}{(t_i - t_{i-1})}$$

and similarly for $t \in (\bar{t}_i, t_i]$, the terminal condition tells us that

$$p_j(t_i) = \int_{\bar{t}_i}^{t_i} v_j \, \mathrm{d}\tau + \frac{\Delta+\delta}{2\Delta} = \frac{\Delta}{\Delta}$$
$$\Rightarrow \qquad v_j(t_i - \bar{t}_i) = \frac{1}{2\Delta}(2\Delta - \Delta + \delta)$$
$$\Rightarrow \qquad v_j = \frac{1}{\Delta}\frac{\Delta-\delta}{(t_i - t_{i-1})}$$

Note that for agent k, the signs are reversed on the numerator. As such, the resulting braid parameterization will have velocities given by

$$\dot{p}_j(t) = \begin{cases} \frac{1}{\Delta}\frac{\Delta\pm\delta}{(t_i - t_{i-1})} & t \in (t_{i-1}, \bar{t}_i] \\ \frac{1}{\Delta}\frac{\Delta\mp\delta}{(t_i - t_{i-1})} & t \in (\bar{t}_i, t_i] \end{cases} \tag{22}$$

where the sign in the numerator is determined by the interpretation given to the braid strand going "under" (e.g., the agent crosses the intersection point first) or "over" (e.g., the agent crosses the intersection point second), and in cases where there are no intersections we set $\delta = 0$ in the numerator.

It is possible to come up with an upper bound on the length of the braid attainable under this mixing scheme. Under Restriction 1, the following theorem provides an upper bound on the mixing limit.

Theorem 4. *Given the maximum safety separation $\bar{\delta}$ and bounds on the agents' velocities such that $v_j(t) \in [0, v_{\max}]\ \forall t, j$, the mixing limit $M^\star$ for N-agent braids that can be performed in a space of height h and length ℓ in time T is bounded above by*

$$M^\star \leq \min\left\{\frac{\ell\sqrt{4h^2 - \bar{\delta}^2 (N-1)^2}}{\bar{\delta} h}, \frac{(N-1)\left(v_{\max}T - \left(\ell + \bar{\delta}\right)\right)}{h} - \frac{1}{2}\right\}.$$

Proof. Consider $\sigma \in \Sigma_N^M$. Since at any step an agent can either interact with another agent or move straight ahead, the maximum mixing will be achieved if an agent interacts with another agent at every step. Thus, at every step it must be enforced that there are no collisions. Assuming a uniform partition of the time window, the arclength-normalized parameter velocity will be given by

$$v_j = \begin{cases} \left(\frac{M\Delta \pm M\delta}{T\Delta}\right) & \text{if } t \in \left(\frac{i-1}{M}T, \frac{2i-1}{2M}T\right] \\ \left(\frac{M\Delta \mp M\delta}{T\Delta}\right) & \text{if } t \in \left(\frac{2i-1}{2M}T, \frac{i}{M}T\right] \end{cases}$$

for $i = 1, 2, \ldots, M$, where the sign on the numerator depends on the interpretation of whether the strand goes over or under, Δ is the arclength of the strand geometry connecting two braid point, assumed equal at each braid step and for both agents due to symmetry, and δ is the safety separation distance along the braid as described above. The total braid path length Δ heavily depends on the geometry of path. However, for the sake of obtaining bounds on the mixing limit, we may assume that the braid points are uniformly distributed in height and length, such that the length of sufficiently regular paths will be bounded by

$$\sqrt{\left(\frac{h}{N-1}\right)^2+\left(\frac{\ell}{M}\right)^2} \leq \Delta \leq \frac{h}{N-1}+\frac{\ell}{M} \tag{23}$$

where the lower bound assumes straight lines connecting the braid points and the upper bound assumes *city-block*-like paths stepping midway between the two points. But after normalizing the bounds on the parameter velocity, we see that

$$0 \leq \frac{M(\Delta-\delta)}{T\Delta} \leq \frac{M(\Delta+\delta)}{T\Delta} \leq \frac{v_{\max}}{\Delta}. \tag{24}$$

The lower bound of (24) tells us that for the parameter to not go backwards we need $\delta \leq \Delta$. To ensure this, we set $\delta = \bar{\delta}\csc\theta \leq \sqrt{\left(\frac{h}{N-1}\right)^2+\left(\frac{\ell}{M}\right)^2} \leq \Delta$, and since we require $\bar{\delta} \leq \frac{h}{N-1}$ for collision-free braid points, using the geometric relationships to solve for M yields

$$M \leq \frac{\ell\sqrt{4h^2-\bar{\delta}^2(N-1)^2}}{\bar{\delta}h}.$$

Similarly, the right-hand side inequality of (24) tells us that $\Delta \leq \frac{v_{\max}T-\delta}{M} = \frac{v_{\max}T-\left(\frac{1}{2}\left(\frac{h}{N-1}\right)+\bar{\delta}\right)}{M}$, and to ensure this we set $\Delta \leq \frac{h}{N-1}+\frac{\ell}{M} \leq \frac{v_{\max}T-\left(\frac{1}{2}\left(\frac{h}{N-1}\right)+\bar{\delta}\right)}{M}$. Solving for M yields

$$M \leq \frac{(N-1)\left(v_{\max}T-\left(\ell+\bar{\delta}\right)\right)}{h}-\frac{1}{2}.$$

Thus

$$M^\star \leq \min\left\{\frac{\ell\sqrt{4h^2-\bar{\delta}^2(N-1)^2}}{\bar{\delta}h}, \frac{(N-1)\left(v_{\max}T-\left(\ell+\bar{\delta}\right)\right)}{h}-\frac{1}{2}\right\}.$$

□

Theorem 4 provides a compact expression to obtain an upper bound on the mixing limit that abstracts away strand geometry. It provides a notion of the whether or not desirable mixing levels are achievable in the space, regardless of what the actual movement patterns to achieve these mixing levels are (encoded in the braid string of length $M \leq M^\star$). Figure 15 includes a plot of the bound on the mixing limit for varying number of agents and time window size.

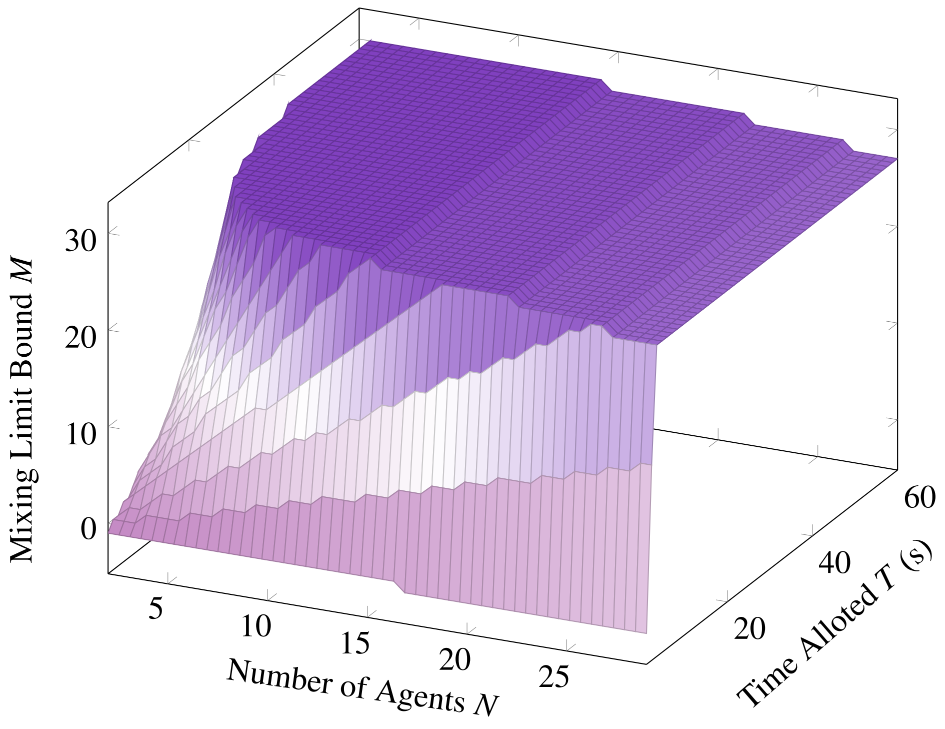


Figure 15. Upper bound on the *Mixing Limit* presented in Theorem 4 for parameters $\ell = 2\ m$**,** $h = 4\ m$**,** $\delta = 0.13\ m$**,** $v_{\max} = 2\ m/s$

4.3 Formal Synthesis of Braid Strings

In this section, we address the question of *how to determine which motion pattern to execute* given a desired interaction specification. Note that for a multi-robot system with N agents attempting to execute a particular motion pattern encoded in a braid string of M generators, there are N^M possible braid strings. Searching the space of interactions patterns in a naïve way could result in costly searches that do not afford real time implementation on multi-robot systems. Instead, we use the results from Theorem 4 together with the closure property of the braid group to construct an algorithm that significantly reduces the size of the search space while guaranteeing exactness of the solution. This is one of the two main contributions of this section. The second contribution is the introduction of a formal framework to specify rich, temporally layered multi-robot mixing requirements. We define a special class of deterministic transition systems, called braid transition systems (BTSs), to encapsulate how braid strings affect the mapping between braid points. In Section 4.3.3

we define a new logic, called *Braid Temporal Logic*, that is interpreted over runs of BTSs. In order to define this new logic, in the next section we provide some common notation on temporal logic and automata.

4.3.1 Temporal Logic and Automata

The set of all finite and set of all infinite words over alphabet Ω are denoted by Ω^* and Ω^∞, respectively.

A *deterministic transition system* [92] (DTS) is a tuple $TS = (Q, q_0, Act, Trans)$, where Q is a set of states, $q_0 \in Q$ is the initial state, Act is a set of actions, and $Trans \subseteq Q \times Act \times Q$ is a transition relation. A *labeled DTS* is a tuple $TS = (Q, q_0, Act, Trans, AP, L)$ where AP is a set of atomic propositions, and the labeling function $L : Q \to 2^{AP}$ maps states to propositions. An input sequence $a = a^0 a^1 \ldots \in Act^*$ induces a *run* $r = q^0 q^1 q^2 \ldots \in Q^*$ such that $q^0 = q_0$ and $(q^i, a^i, q^{i+1}) \in Trans$. The *trace* of a run of a labeled transition system is a word $w = w^0 w^1 \ldots \in (2^{AP})^*$ such that $w^i = L(q^i)$.

A *syntactically co-safe linear temporal logic* (scLTL) formula over a set AP is inductively defined as [93]:

$$\phi := p | \neg p | \phi \vee \phi | \phi \wedge \phi | \phi \;\mathcal{U}\; \phi | \bigcirc \phi | \diamond \phi, \tag{25}$$

where $p \in AP$ and ϕ is an scLTL formula. The logical operators $\vee$, $\wedge$, and $\neg$ are disjunction, conjunction, and negation, respectively, and the temporal operators $\mathcal{U}$, $\bigcirc$, and $\diamond$ are until, next, and eventually, respectively. We also use Boolean implication $\Rightarrow$, where $(\phi_1 \Rightarrow \phi_2) = (\neg\phi_1 \vee \phi_2)$. The logic scLTL is defined over words $w = w^0 w^1 \ldots \in (2^{AP})^*$. The notation $w \models \phi$ is used to mean that w *satisfies* an scLTL formula ϕ. The *language* of ϕ is $\mathcal{L}(\phi) = \{w | w \models \phi\}$.

A *(deterministic) finite state automaton* (FSA) is a tuple $FSA = (\Omega, \Pi, \Omega_0, F, \Delta_{FSA})$ where Ω is a finite set of states, Π is an input alphabet, $\Omega_0 \subseteq \Omega$ is a set of initial states, $F \subseteq \Omega$ is a set of final (accepting) states, and $\Delta_{FSA} \subseteq \Omega \times \Pi \times \Omega$ is a deterministic transition relation. An *accepting run* r_{FSA} of an automaton FSA is a sequence of states $\omega^0 \omega^1 \ldots \omega^{j+1}$

such that $\omega^{j+1} \in F$ and $(\omega^i, \pi^i, \omega^{i+1}) \in \Delta_{FSA}$ $\forall i \in [0, j]$. The *language* of FSA, denoted $\mathscr{L}(FSA)$, is the set of words $w \in \Pi^*$ that lead to an accepting run. Given an scLTL formula ϕ over AP, there exist off-the-shelf algorithms [94] for creating an FSA FSA_ϕ with input alphabet 2^{AP} such that $\mathscr{L}(FSA_\phi) = \mathscr{L}(\phi)$.

The *product automaton* between a labeled deterministic transition system TS and an FSA FSA_ϕ is an FSA $\mathscr{P}_\phi = TS \times FSA_\phi = (\Omega_{\mathscr{P}}, \chi_0, Act, F_{\mathscr{P}}, \Delta_{\mathscr{P}})$ [92], where $\Omega_{\mathscr{P}} \subseteq Q \times \Omega$, $\chi_0 = (q_0, \omega_0)$, $F_{\mathscr{P}} \subseteq Q \times F$, and $\Delta_{\mathscr{P}} = \{(q, \omega), p, (q', \omega') | (q, p, q') \in Trans, (\omega, L(q), \omega') \in \Delta_{FSA}\}$. The state of the automaton at time k is denoted as $\chi^k = (q^k, \omega^k)$. Any accepting word $a = a^0 a^1 \ldots \in Act^*$ over $\mathscr{P}$ induces a trace w over TS such that $w \models \phi$. Thus, finding a path on TS that satisfies ϕ corresponds to a reachability problem on $\mathscr{P}_\phi$.

The *distance to acceptance* [95, 96] $W : \Omega_{\mathscr{P}} \to \mathbb{Z}^+$ is defined such that $W(\chi)$ is the minimum number of actions required to drive $\mathscr{P}$ from χ to a state $\chi_f \in F_{\mathscr{P}}$.

4.3.2 Temporal Logic and the Braid Group: Braid Transition System

In the BTS, we model the set of braid points abstractly as configurations.

Definition 4.3.1 (Configuration space)**.** *Let* $v_N = [1 \ \ldots \ N]^T$. *The* (mixing) configuration space *for a team of N agents is* $Perm(v_N)$ *where* $Perm(\cdot)$ *denotes the set of permutations of the elements of the vectors.*

The configuration associated with $\mathcal{P}_0$ is by definition v_N. A column vector $c \in Perm(v_N)$ corresponds to a configuration of the braid points such that $c(k) = j$ if and only if agent j is mapped to the braid point $[\mathcal{P}_i]_k$ at step i.

Definition 4.3.2 (Braid Transition System (BTS))**.** *The* braid transition system *of size N is a deterministic transition system described by the tuple*

$$BTS_N = \left(v_N \cup Perm(v_N)^2, v_N, \Sigma_N, Trans_N\right),$$

where $Trans_N \subseteq C_N \times \Sigma_N \times C_N$ *is the smallest transition relation that satisfies*

$$(v_N, \sigma_0, (v_N, v_N)) \in Trans_N \tag{26a}$$

$$\left.\begin{array}{c} (v_N, \sigma_i, (v_N, c_{22})) \in Trans_N \Leftrightarrow \\ c_{22}(i) = i + 1 \wedge c_{22}(i+1) = i \\ \forall i \in 1, \ldots, N-1 \end{array}\right\} \tag{26b}$$

$$((c_{11}, c_{12}), \sigma_0, (c_{12}, c_{12})) \in Trans_N \tag{26c}$$

$$\left.\begin{array}{c} ((c_{11}, c_{12}), \sigma_i, (c_{21}, c_{21})) \in Trans_N \Leftrightarrow \\ c_{21} = c_{12} \wedge c_{21}(i) = c_{22}(i+1) \\ \wedge c_{21}(i+1) = c_{22}(i) \\ \forall i \in 1, \ldots, N-1 \end{array}\right\} \quad \diamond \tag{26d}$$

Example 1. *The braid transition system for two agents, BTS_2, is illustrated in Figure 16a.* ◇

A BTS stores one time unit of history, i.e., if the BTS is in state (c_1, c_2), $c_i \in Perm(v_N)$, at time k, then the robots were in configuration c_1 at time $k-1$ and are in configuration c_2 at time k. This allows us to check properties that explicitly involve interactions between agents. Given a run of the braid transition system $r = v_N, (v_N, c^1), (c^1, c^2) \ldots \in (Perm(v_N) \cup Perm(v_N)^2)^*$ we define its *configuration trace* as $r_C = v_N, c^1, c^2, \ldots$. For a finite N, the number of states in BTS_N is $(N!)N + 1$ and the number of transitions in $Trans_N$ is $N^2(N!) + N$.

4.3.3 Braid Temporal Logic

In order to describe rich, temporally layered requirements on the agents' mixing, we define a new predicate temporal logic, called Braid Temporal Logic (BTL).

Definition 4.3.3 (BTL Syntax)**.** *The* syntax *of BTL is defined inductively as*

$$\begin{array}{c} \phi := \; A_m p_g | d(A_m, A_\ell) \sim x | A_m A_\ell | \neg Aipj | \neg AiAj \\ | \phi \vee \phi | \phi \wedge \phi | \phi \; \mathscr{U} \; \phi | \bigcirc \phi | \diamond \phi, \end{array} \tag{27}$$

where ϕ is a BTL formula, $\sim \in \{<, >\}$, $x \in \mathbb{N}$, and the Boolean and temporal operators are as defined for scLTL in Section 4.3.1. The predicate $A_m p_g$ means agent m is in position g; $d(A_a, A_\ell) \sim x$ means that the distance between agents m and ℓ is less than (or greater than) x; $A_m A_\ell$ means that agents m and ℓ interact. $\diamond$

Definition 4.3.4 (BTL semantics)**.** *The* semantics *of BTL is defined recursively as*

$$\begin{aligned}
c^i &\models A_m p_g && \Leftrightarrow \quad c^i(g) = m \\
c^i &\models \neg A_m p_g && \Leftrightarrow \quad c^i \not\models A_m p_g \\
c^i &\models d(A_m, A_\ell) \sim x && \Leftrightarrow \quad |f - g| \sim x \text{ where} \\
& && \qquad c^i(f) = m \text{ and } c^i(f) = \ell \\
c^i &\models A_m A_\ell && \Leftrightarrow \quad m \text{ and } \ell \text{ swap between} \\
& && \qquad c^{i-1} \text{ and } c^i. \\
c^i &\models \neg A_m A_\ell && \Leftrightarrow \quad c^i \not\models A_m A_\ell \\
c^i &\models \phi_1 \vee \phi_2 && \Leftrightarrow \quad c^i \models \phi_1 \text{ or } c^i \models \phi_2 \\
c^i &\models \phi_1 \wedge \phi_2 && \Leftrightarrow \quad c^i \models \phi_1 \text{ and } c^i \models \phi_2 \\
c^i &\models \phi_1 \,\mathcal{U}\, \phi_2 && \Leftrightarrow \quad \exists j \geq i \text{ s.t. } c^j \models \phi_2 \\
& && \qquad \text{and } c^k \models \phi_1 \ \forall i \leq k < j \\
c^i &\models \bigcirc \phi && \Leftrightarrow \quad c^{i+1} \models \phi \\
c^i &\models \diamond \phi && \Leftrightarrow \quad \exists j \geq i \text{ s.t. } c^j \models \phi. \qquad \diamond
\end{aligned} \tag{28}$$

Example 2 (cont'd)**.** *For the case of two robots interacting, we can use the BTL formula*

$$\phi_{n=2} = \diamond (A_1 p_2 \wedge \bigcirc A_1 A_2) \tag{29}$$

to describe the property: eventually, Agent 1 is in position 2 and in the next step, Agent 1 and Agent 2 interact.

Example 3. *Consider the specification*

$$\begin{aligned}\phi_c = &\diamond (A_3A_4 \vee A_2A_4)\\ &\wedge (\neg(A_3A_4 \vee A_2A_4) \,\mathscr{U}\, A_1A_4)\\ &\wedge ((A_3A_4 \Rightarrow\ \diamond A_3p_5) \wedge (\neg A_3p_5 \,\mathscr{U}\, A_3A_4)\\ &\vee (A_2A_4 \Rightarrow\ \diamond A_2p_5) \wedge (\neg A_2p_5 \,\mathscr{U}\, A_2A_4))\end{aligned} \tag{30}$$

In plain English, this is: agent 4 communicates with agent 2 or 3 after it has communicated with Agent 1. Whichever agent 4 communicates with reports to position 5.

4.3.4 Synthesis of Braid Strings from BTL Formulae

In this work, we are both interested verifying rich temporally layered behaviors of interacting robots, and in developing braid controllers that enforce a given BTL specification. We encode this in the following problem.

Problem 1 (Braid String Synthesis)**.** *Given a set of N agents and a BTL formula ϕ, find a word $\sigma \in \Sigma_N^M$ such that applying σ to BTS_N will lead to a configuration trace that satisfies ϕ and σ has fewer generators than the mixing limit M^*.* ◇

There are potentially many words σ that satisfy Problem 1. Here, we synthesize the shortest satisfying word.

The standard approach to solving Problem 1 is to convert it to the problem of scLTL-based synthesis for labeled transition systems. Briefly, we convert a given BTL formula ϕ to an scLTL formula ϕ' by applying a mapping π that maps every predicate in ϕ to a unique atomic proposition. The product automaton $\mathscr{P} = BTS_N \times FSA_{\phi'}$ is constructed and then Djikstra's algorithm is used to produce the shortest accepting word. We ensure that the length of the resulting word is less than the mixing limit. Applying this word to BTS_N will result in a configuration trace r_C that satisfies ϕ.

Example 4 (Cont'd)**.** *Figure 16b shows the FSA constructed from* (29)*. Figure 16c shows the product automaton between BTS_2 and the FSA from* (29)*. In Figure 16d, we see the path that results from finding the shortest accepting word on $\mathscr{P}$.*

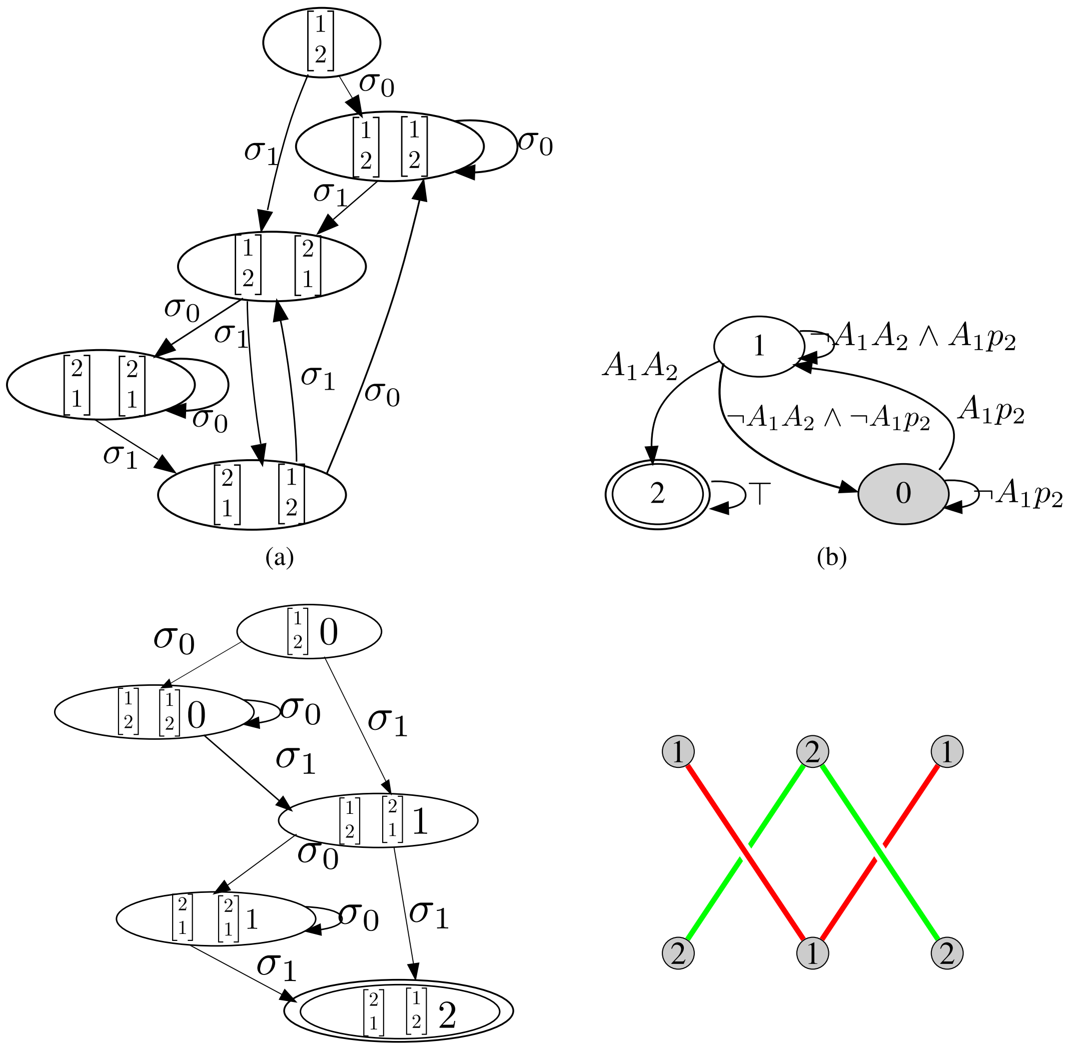


Figure 16. (a) The braid transition system constructed when two agents interact. (b) FSA constructed from (29)**. The initial state ω_0 is indicated in grey and the accepting state is indicated by the double outline. The edges are annotated with the BTL subformulae whose language is the set of inputs that enable the indicated transition. (c) Product automaton between (a) and (b). Again, the accepting state is indicated with double circles. (d) The braid resulting from finding the shortest accepting path on (c).**

4.3.5 Language-Guided Synthesis

The number of states in BTS_N scales exponentially with N. We present a procedure, outlined in Algorithm 1, that constructs the part of the product automaton between BTS_N and $FSA_{\phi'}$ necessary to solve Problem 1, denoted $\mathscr{P}_{LG}$, that does not require explicitly constructing BTS_N.

After constructing $FSA_{\phi'}$, we use its accepting states and the set of predicates that enable transitions to these states to enumerate the accepting states F_P of $\mathscr{P}$. Next, we construct $\mathscr{P}_{LG}$ backwards. At each iteration j, the procedure BackStep constructs the set of states K_j such that $W(\chi) = j \; \forall \chi \in K_j$ and connects K_j to $\mathscr{P}_{LG}$. Since $Trans_N$ can be represented by (26), we can enumerate all transitions in BTS_N that would result in a state $(c_1, c_2) \in K_j$. The inputs of the transitions in $FSA_{\phi'}$ can be used to determine whether paths originating from these enumerated states can reach an accepting state in j steps. Finally, after executing BackStep $M^* - 1$ times, we connect the initial state (v_N, ω_0) to $\mathscr{P}_{LG}$ and then trim any states in the graph that are not reachable from (v_N, ω_0).

Algorithm 1: Language-guided product automaton construction.

```
function LanguageGuidedConstruction($N, \phi, M^*$)
    $FSA_{\phi'}$ = BuildFSA($\phi$)
    $F_P$ = ComputeAcceptingState($FSA_{\phi'}$)
    $\Omega_P = F_P$; $\Delta_P = \emptyset$; $K_0 = F_P$;
    for $j = 1$ to $(M^* - 1)$ do
        $(K_j, \Delta_{FSA}, \Omega_P, \Delta_P)$ = BackStep($K_{j-1}, \Delta_{FSA}, \Omega_P, \Delta_P$)
    end
    $(\Omega_P, \Delta_P)$ = ConnectInitialState($\Omega_P, \Delta_P$)
    $\mathscr{P}_{LG} = (\Omega_P, (v_n, c, \omega_0), Act, F_P, \Delta_P)$
return Trim($\mathscr{P}_{LG}$)
```

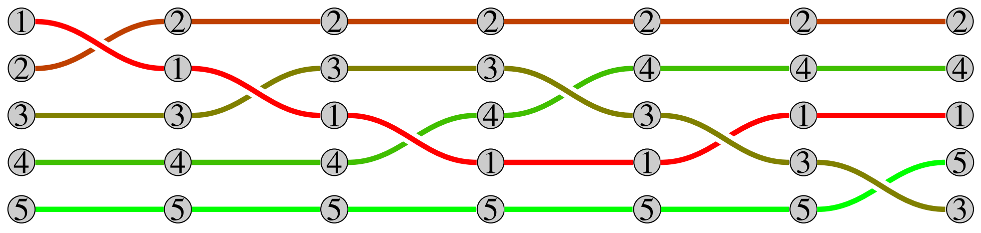


Figure 17. Shortest braid that satisfies (30)**.**

Proposition 1 (Exactness)**.** *The language of $\mathscr{P}_{LG}$ is the set of all paths that will induce BTS_N to satisfy ϕ and respect the mixing limit.*

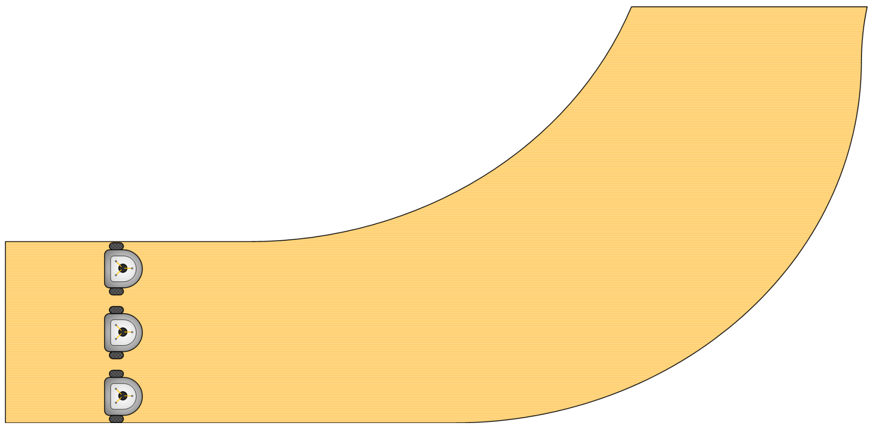
Proof. (Sketch) The result is guaranteed by enforcing the loop invariant $W(\chi) = j \; \forall \chi \in K_j$. □

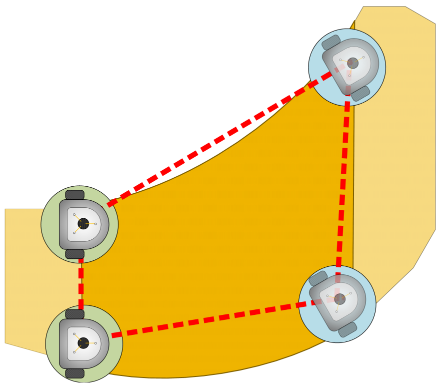
Example 5 (Cont'd.)**.** *The braid in Figure 17 satisfies* (30) *and respects the mixing limit of 8. This braid was generated by Algorithm 1 in 3.7s from an automaton with 845 states. Applying the standard approach (Section 4.3.4) calculated the solution in 939s from an automaton with 5724 states. All calculations were performed on a PC with a 2.6 GHz processor with 7.8 GB RAM.*

4.4 Non-Rectangular Regions

Up to this point, only the problem of braiding on a rectangular region of height h and length l has been considered. On this region, the braid points were uniformly distributed along both dimensions and bounds on the mixing limit were provided through the use of the presented braid controller. In this section, the scheme is extended to more generally shaped regions, e.g., the road on Figure 18. As has been done previously, discussion begins by first considering the two agent case.

Consider the two agents attempting to perform a braid of length one on the arbitrarily curved region on Figure 18b. Connecting the braid points together results in the quadrilateral depicted in the red dotted line. Let this quadrilateral be considered as the space where the agent needs to perform a braid of length one, rather than the curved region itself. Note


(a) Three agents braiding on a curved region.


(b) Two agents need to move from the left-most circles to the right-most circles along the curved region.

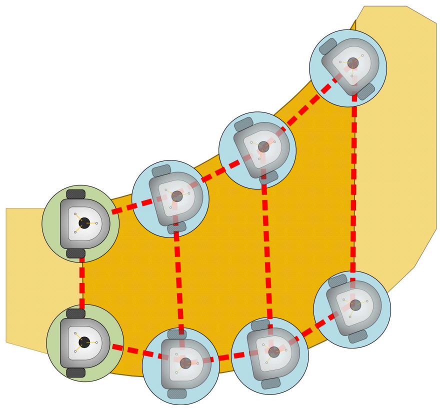
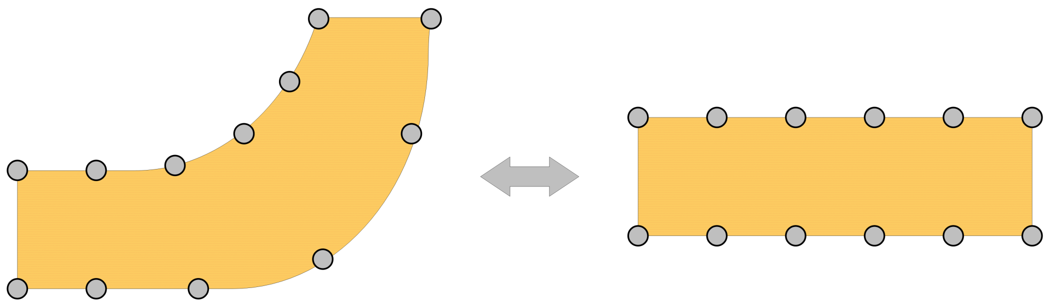
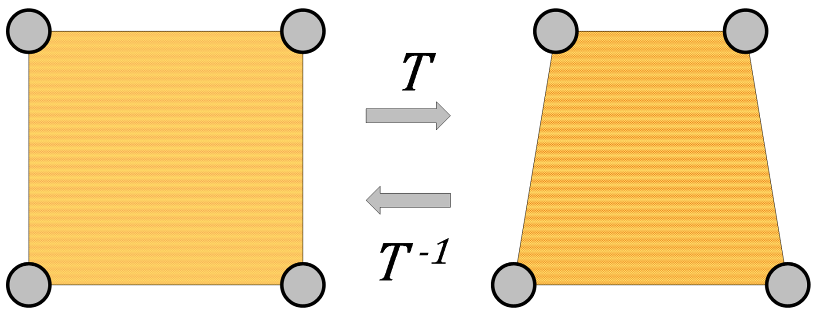

(c) The longer the braid length in this curved road segment, the closer the quadrilaterals resemble the curved region.

Figure 18. Agents braiding on a region that curves.

that as longer length braids are included in this road segment, other quadrilaterals appended together will be obtained which approximate the road slightly better. Since it is of interest to obtain mixing strategies near the mixing limit, as longer length braids are included in this road segment, better approximations of the curved region will be obtained by these composition of quadrilaterals. This is depicted in Figure 18c.

The strategy for performing a mixing strategy in curved region will be to transform the curved region into a straightened rectangular region of known height and length, as illustrated in Figure 19a. In this way the braid controller can be fashioned as in previous sections and the resulting braid controller can be transformed back into the actual curved region. After distributing the braid points on both the curved region and the rectangular


(a) The curved region will be mapped to the rectangular region of known width and height where control design will take place.



(b) The bijective transformation $\mathcal{T}$ maps points in the rectangle to points in the quadrilateral. Both shapes are defined by the braid points which determine the corners.

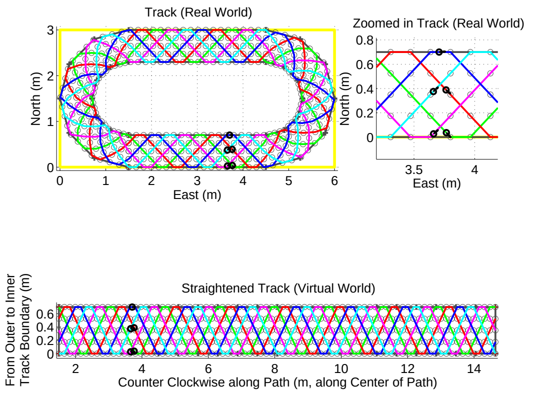
Figure 19. Rectangular and non-rectangular regions.

region, as in Figure 19a, the next step is to find a transformation to map between these two regions.

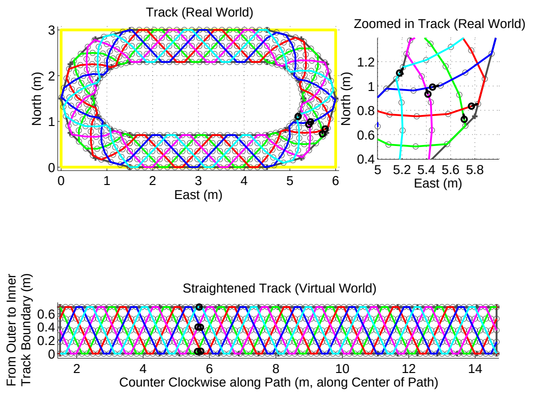
Let agents j and k interact in at braid step i. Denote $\mathcal{S}^q_{i,j}$ to be the quadrilateral formed by connecting together the braid points $\xi(i-1, j), \xi(i-1, k), \xi(i, j)$, and $\xi(i, k)$, like the one depicted in Figure 18b. Let $\mathcal{S}^r_{i,j}$ be a rectangular plane of specified height and length whose corners are given by $\xi_r(i-1, j), \xi_r(i-1, k), \xi_r(i, j)$, and $\xi_r(i, k)$.

With knowledge of these braid points and through the use of a projective transform as in [97], it is possible to obtain a local diffeomorphism that maps from a rectangle of specified height and length to the convex arbitrarily shaped quadrilateral, i.e., $\mathcal{T}_{i,j} : \mathcal{S}^r_{i,j} \to \mathcal{S}^q_{i,j}$.

Note that by selecting transforms based on the corners of these quadrilaterals, a continuous curve that spans across the boundary between $\mathcal{S}^r_{i,j}$ and $\mathcal{S}^r_{i+1,j}$ might be mapped to a discontinuous curve that spans across the boundary between $\mathcal{S}^q_{i,j}$ and $\mathcal{S}^q_{i+1,j}$ when transformed using $\mathcal{T}_{i,j}$ and $\mathcal{T}_{i+1,j}$ in their respective spaces. However, there will certainly be



(a) Simulation Time 5.04 seconds.



(b) Simulation Time 9.57 seconds.

Figure 20. Five agents performing a mixing strategy given by a braid of length 80. The top left plot represents the actual curved region the agents are mixing in and the top right is a close-up of the agents. The lower plot is the virtual "straightened" rectangular region where the design of the braid took place.

continuity in the mapping of curves passing through the braid points, since these points are shared by the quadrilaterals and are used to compute the transforms, i.e., $\mathcal{T}_{i,j}\left(\xi_r(i,j)\right) = \xi(i,j) = \mathcal{T}_{i+1,j}\left(\xi_r(i,j)\right)$.

Recall the braid controller presented for the rectangular region. For agent j, at braid step i, this was given by

$$v_j(t) = \begin{cases} \frac{1}{\Delta}\left(\frac{\Delta-\delta}{t_i-t_{i-1}}\right) & \text{if } t \in (t_{i-1}, \bar{t}_i]\,, \\ \frac{1}{\Delta}\left(\frac{\Delta+\delta}{t_i-t_{i-1}}\right) & \text{if } t \in (\bar{t}_i, t_i]\,. \end{cases}$$

where it is expected that $\mathcal{T}_{i,j}^{-1}\left(x\left(t_{i-1}\right)\right) = \xi_r(i-1,j)$ and $\mathcal{T}_{i,j}^{-1}\left(x\left(t_i\right)\right) = \xi_r(i,j)$. In the expression, Δ corresponds to the length of the geometric path agent j must follow to move between $\xi_r(i-1,j)$ and $\xi_r(i,j)$, while δ corresponds to the distance along the path agent j switches velocities in order to avoid collisions. Note that for a given parameterization of the geometric path $\gamma_{i,r}^j(p)$ in $\mathcal{S}_{i,j}^r$ with parameter $p \in [0,1]$, the arclength Δ may be computed as follows

$$\Delta = \int_0^1 \sqrt{\dot{\gamma}_{i,r}^{j\top}(p)\dot{\gamma}_{i,r}^j(p)}\,\mathrm{d}p.$$

If the geometric curve is directly given in $\mathcal{S}_{i,j}^q$ as $\gamma_i^j(p)$ such that the curve in $\mathcal{S}_{i,j}^r$ may be parameterized as $\gamma^j_{i,r}(p) = \mathcal{T}^{-1}_{i,j}(\gamma^j_i(p))$, then the arclength may be computed by

$$\Delta = \int_0^1 \sqrt{\left(\dot{\gamma}^j_i(p)\right)^\top \mathcal{M}(\gamma^j_i(p))\dot{\gamma}^j_i(p)}\,\mathrm{d}p.$$

where $\mathcal{M}(\gamma^j_i(p)) = \left(D\mathcal{T}^{-1}_{i,j}\left(\gamma^j_i(p)\right)\right)^\top D\mathcal{T}^{-1}_{i,j}\left(\gamma^j_i(p)\right)$ and $D\mathcal{T}^{-1}_{i,j}$ is the Jacobian of $\mathcal{T}^{-1}_{i,j}$. In the special case where the geometry is given by straight lines, then $\Delta = \|\xi_r(i,j) - \xi_r(i-1,j)\|$.

Finally, it is of interest to find δ in order to avoid collisions. If the parameterization $\gamma^j_i(p)$ of the curve in $\mathcal{S}^q_{i,j}$ is known, then the safety separation ball may be set in $\mathcal{S}^q_{i,j}$ as before and the safety separation distance may be computed as

$$\delta = \int_a^b \sqrt{\left(\dot{\gamma}^j_i(p)\right)^\top \mathcal{M}(\gamma^j_i(p))\dot{\gamma}^j_i(p)}\,\mathrm{d}p.$$

where $[a,b] \subset [0,1]$, and if the agent is meant to braid *over* (resp. *under*) then $\gamma^j_i(a)$ corresponds to the point along the curve where the agent enters the safety separation region (resp. where the two curves intersect) and $\gamma^j_i(b)$ corresponds to the point along the curve where the two curves intersect (resp. where the agent exits the safety separation region).

In the special case where the geometry is given by straight lines, then by setting

$$\gamma^n_i(p) = (1-p)\xi(i-1,n) + p\xi(i,n),\ p \in [0,1],\ n = j,k.$$

as the path agent j and k must follow, the intersection point s may be found by setting $s = \gamma^j_i(\pi_j) = \gamma^k_i(\pi_k)$ where

$$\begin{bmatrix}\pi_j \\ \pi_k\end{bmatrix} = A^{-1}\left(\xi(i-1,k) - \xi(i-1,j)\right)$$

with $A = \left[\left(\xi(i,j) - \xi(i-1,j)\right), -\left(\xi(i,k) - \xi(i-1,k)\right)\right]$. Recall that the distance in $\mathcal{S}^q_{i,j}$ from the intersection point s, for the special case of the geometry being straight lines, was given by $\delta = \delta_{jk}\csc(\theta)$ with θ being the angle between these two lines, i.e., $\theta = \cos^{-1}\left(\hat{x}^\top_j \hat{x}_k\right)$ where $\hat{x}_n$ is the unit vector pointing towards the next point, i.e.,

$$\hat{x}_n = \frac{(\xi(i,n) - \xi(i-1,n))}{\|\xi(i,n) - \xi(i-1,n)\|}, \qquad n = j,k.$$

By setting $\hat{\gamma}(p) = (1-p)s \pm p\delta\hat{x}_j$, where sign depends on whether the braid goes *over* or *under*, it is possible to determine δ in the non-rectangular plane directly as

$$\delta_r = \int_0^1 \sqrt{\left(\pm\delta\hat{x}_j - s\right)^\top \mathcal{M}(\hat{\gamma}(p)) \left(\pm\delta\hat{x}_j - s\right)}\,\mathrm{d}p.$$

With this information, v_j can be found in $\mathcal{S}^r_{i,j}$ for interactions between agents j and k. Note that a reparameterization of the path can now be set equivalent to the desired trajectory of agent j by setting $\gamma^j_{i,d}(t) = \gamma^j_{i,r}(p_j(t))$ (see (31) below). Thus, the braid controller parameter velocity $v^q_j(t)$ for agent j in $\mathcal{S}^q_{i,j}$ will be given by $v^q_j(t) = \left\| D\mathcal{T}_{i,j}(\gamma^j_{i,d}(t))\dot{\gamma}^j_{i,d}(t) \right\|$.

This strategy was implemented in simulation over the curved region illustrated in Figure 20. In the figure, agents are performing the mixing strategy given to them by a braid of length 80 on the curved region (top) and simultaneously on the straightened rectangular region (bottom). The parameters used for this simulation were $\delta_{jk} = 7.7\ cm$ and $v_{max} = 1.5\ m/s\ \forall j, k$, and $T = 30\ s$.

4.5 Implementing Braids

Section 4.1 approached the problem of multi-robot mixing from an execution level, where given a symbolic input it is possible to find a braid controller that generates trajectories to satisfy the input. In Section 4.3, we addressed the specification level where given a specification it is possible to synthesize a braid string that satisfies the specification. In this section we consider the implementation level, that is, we address how to find controllers that are implementable on actual robotic systems that follow the braid controllers generated trajectories in previous section. We then validate the framework by implementing a mixing strategy on a team of six robots.

4.5.1 Optimal Tracking Controller

We will now utilize the braid parameter velocity to find the braid controller for the agents. By integrating (22) we obtain the *braid parameterization* of the path γ_i^j

$$p_j(t) = \begin{cases} \frac{t-t_{i-1}}{(t_i-t_{i-1})}\frac{\Delta\pm\delta}{\Delta} & t \in (t_{i-1}, \bar{t}_i] \\ \frac{t-\bar{t}_i}{(t_i-t_{i-1})}\frac{\Delta\mp\delta}{\Delta} + \frac{\Delta\pm\delta}{2\Delta} & t \in (\bar{t}_i, t_i]\,. \end{cases} \tag{31}$$

Assume that agents have single integrator dynamics, i.e., $\dot{x}_j = u_j$ with $y_j = x_j \in \mathbb{R}^2$. The agent's controller will be found by optimally tracking the reparameterized path $\gamma_i^j(p_j(t))$ to minimize the cost

$$J(u_j) = \frac{1}{2}\int_{t_{i-1}}^{t_i} \left(x_j - \gamma_i^j\right)^T Q\left(x_j - \gamma_i^j\right) + u_j^T R u_j \,\mathrm{d}\tau \tag{32}$$

for $Q = Q^T \succ 0$ and $R = R^T \succ 0$, with constraints $\dot{x}_j = u_j$, $x_j(t_{i-1}) = \gamma_i^j(0)$, and $x_j(t_i) = \gamma_i^j(1)$. Using the standard variational argument together with Pontryagin's minimum principle, the first order necessary conditions for optimality tell us that the optimal tracking controller u_j^* is given by

$$u_j^* = -R^{-1}\lambda_j$$

where λ is the so-called costate and satisfies

$$\dot{\lambda}_j = -Q\left(x_j - \gamma_i^j\right)$$

with unknown terminal condition $\lambda_j(t_i)$. Suppose that similarly to [98, Chapter 5.3] we can construct λ_j as an affine combination of the unknown $\lambda_j(t_i)$ and the state, i.e.,

$$\lambda_j(t) = H(t)x_j(t) + K(t)\lambda_j(t_i) + E(t)$$

and similarly, the terminal state as

$$x_j(t_i) = \xi(i, j) = F(t)x_j(t) + G(t)\lambda_j(t_i) + D(t)$$

for some yet unknown functions H, K, E, F, G, and D. One can differentiate these equations and manipulate the equations to obtain

$$\left(HR^{-1}H - Q - \dot{H}\right)x_j + \left(HR^{-1}K - \dot{K}\right)\lambda_j(t_i) + \left(HR^{-1}E + Q\gamma_i^j - \dot{E}\right) = 0$$

and

$$\left(FR^{-1}H - \dot{F}\right)x_j + \left(FR^{-1}K - \dot{G}\right)\lambda_j(t_i) + \left(FR^{-1}E - \dot{D}\right) = 0.$$

In order to satisfy these equations for any value of $x_j(t)$ and $\lambda_j(t_i)$, the terminal conditions, and after noticing that $F = K^T$, we obtain that

$$\begin{aligned}
\dot{H} &= HR^{-1}H - Q, & H(t_i) &= 0_{2\times 2} \\
\dot{K} &= HR^{-1}K, & K(t_i) &= I_2 \\
\dot{G} &= K^T R^{-1}K, & G(t_i) &= 0_{2\times 2} \\
\dot{E} &= HR^{-1}E + Q\gamma_i^j, & E(t_i) &= 0_{2\times 1} \\
\dot{D} &= K^T R^{-1}E, & D(t_i) &= 0_{2\times 1}.
\end{aligned} \tag{33}$$

Note that $G(t_i) = 0$ and $\dot{G}(t) \geq 0$ for all t, which suggests that $G(t) \leq 0$ for $t < t_i$. If the problem is not *abnormal*, i.e., there exists a neighboring minimum solution, then G will be invertible at some $t < t_i$. In particular, by solving backwards in time in the sequence $H \to K \to E \to G \to D$ up to $t = t_{i-1}$, we can find that

$$\lambda_j(t_i) = G^{-1}(t_{i-1})\left(\xi(i, j) - K^T(t_{i-1})\xi(i-1, j) - D(t_{i-1})\right)$$

resulting in the feedback optimal trajectory tracking control law

$$\begin{aligned}
&u_j^*(x_j, t) = \\
&\quad - R^{-1}\Big[H(t)x_j(t) + K(t)G^{-1}(t_{i-1})\Big(\xi(i, j) - K^T(t_{i-1})\xi(i-1, j) - D(t_{i-1})\Big) + E(t)\Big]
\end{aligned} \tag{34}$$

where H, K, E, G and D are the solutions to terminal value problems in (33). which can be solved numerically backwards from t_i. As it turns out, these conditions are also sufficient for optimality as presented in the following theorem.

Theorem 5. *The tracking controller in* (34) *is a minimizer to the cost functional* (32) *whose optimal value is given by*

$$J(u^*) =$$
$$\left.\left[\xi(i-1,j)^T\left(\frac{1}{2}H - KG^{-1}K^T\right)\xi(i-1,j) + \xi(i-1,j)^T\left(KG^{-1}\left(\xi(i,j)-D\right)+E\right)+\varphi\right]\right|_{t=t_{i-1}}$$

where $\varphi(t)$ is the solution to the terminal boundary problem

$$\dot{\varphi} = \frac{1}{2}\left(\Lambda_j(t)\right)^T R^{-1}\Lambda_j(t) - \frac{1}{2}\left(\gamma_i^j(t)\right)^T Q\gamma_i^j(t)$$
$$\varphi(t_i) = -\xi(i,j)^T\Lambda_j(t_i)\xi(i,j)$$

with

$$\Lambda_j(t) = E(t) + K(t)G^{-1}(t_{i-1})(\xi(i,j) - K^T(t_{i-1})\xi(i-1.j) - D(t_{i-1})).$$

Proof. As the control law was derived from the necessary conditions for optimality, we only need to show that it is sufficient for optimality. We will do so by leveraging the Hamilton-Jacobi-Bellman theorem [99, Chapter 2].

Note that the optimization problem is regular as there exists a u_j that allows the Hamiltonian to achieve a minimum with respect to it, i.e.,

$$\begin{aligned}\mathcal{H}(x_j,u_j,\lambda_j) &= \frac{1}{2}\left[\left(x_j-\gamma_i^j\right)^T Q\left(x_j-\gamma_i^j\right) + u_j^T R u_j\right] + \lambda_j^T u_j \\ &= \frac{1}{2}\left(u_j + R^{-1}\lambda_j\right)^T R\left(u_j+R^{-1}\lambda_j\right) - \frac{1}{2}\lambda_j^T R^{-1}\lambda_j + \frac{1}{2}\left(x_j-\gamma_i^j\right)^T Q\left(x_j-\gamma_i^j\right)\end{aligned}$$

which attains a minimum with respect to u_j when

$$u_j^* = -R^{-1}\lambda_j.$$

Define $V(z,t)$ as

$$\begin{aligned}V(z,t) = \frac{1}{2}z^T H(t)z + z^T K(t)G^{-1}(t_{i-1})\xi(i,j)& \\ - z^T\left(K(t)G^{-1}(t_{i-1})(K^T(t_{i-1})\xi(i-1,j) - D(t_{i-1})) + E(t)\right) + \varphi(t)&\end{aligned}$$

where $\varphi(t)$ is as defined above. It can be verified that $V(z,t)$ satisfies the terminal condition $V(\xi(i,j),t_i) = 0$, that $\frac{\partial V(z,t)}{\partial z}\Big|_{z=x_j}^T = \lambda_j$, and that is satisfies the Hamilton-Jacobi-Bellman equation, i.e.,

$$0 = \frac{\partial V(z,t)}{\partial t}\bigg|_{z=x_j} + \mathcal{H}\left(x_j, u_j^*\left(x_j,t\right), \frac{\partial V(z,t)}{\partial z}\bigg|_{z=x_j}^T\right).$$

As a consequence, u_j^* is a minimizer to the cost functional and

$$J(u^*) = V(\xi(i-1,j),t_{i-1}).$$

□

As a final note, the terminal costate value $\lambda_j(t_i)$ was computed using the initial conditions for the problem. However, as the gains involved are solved from terminal conditions, the choice of initial conditions is arbitrary, and evaluating at $t = t_{i-1}$ results in control law (34) being open-loop in the terminal costate value. This could yield undesired results under the influence of disturbances and errors. To alleviate this, we can rewrite the terminal costate as a function of the current state value instead, i.e.,

$$\begin{aligned}\lambda_j(t_i) &= G^{-1}(t_{i-1})\left(\xi(i,j) - K^T(t_{i-1})\xi(i-1,j) - D(t_{i-1})\right)\\ &= G^{-1}(t)\left(\xi(i,j) - K^T(t)x_j(t) - D(t)\right)\end{aligned}$$

in order to obtain the fully closed-loop optimal tracking controller

$$u_j^*(x_j,t) = -R^{-1}\left[\left(H(t) - K(t)G^{-1}(t)K^T(t)\right)x_j(t) + K(t)G^{-1}(t)\left(\xi(i,j) - D(t)\right) + E(t)\right]. \quad (35)$$

4.5.2 Robotic Implementation

We consider a team of 6 agents with heterogeneous sensors that is tasked with the high-level mission: *Agent 3 visits location 5 and communicates with agent 1. After agent 3's mission is complete, agent 1 goes to location 6 and agent 6 goes to location 1. Agents 1 and 2 are never more than 3 locations apart for the duration of the mission.* The corresponding BTL formula is given by

$$\begin{aligned}\phi_s = \; & (\Diamond A_3 p_5) \wedge (\Diamond A_3 A_1) \\ & \wedge ((\neg A_1 p_6 \vee \neg A_6 p_1) \; \mathcal{U} \; A_3 A_1) \\ & \wedge ((\neg A_1 p_6 \vee \neg A_6 p_1) \; \mathcal{U} \; A_3 p_5) \\ & \wedge (\Diamond (A_1 p_6 \wedge A_6 p_1)) \\ & \wedge (d(A_1, A_2) < 4 \; \mathcal{U} \; (A_1 p_6 \wedge A_6 p_1)).\end{aligned} \tag{36}$$

This mission represents part of a much longer mission that has been decomposed into sequential BTL specifications over different windows. This would be the case if the team of agents are moving along a path and have variable requirements at different points along this path.

An interpretation of this specification is that over this time-window, Agent 3 is carrying an infrared camera that needs to measure an algal bloom in a pond in location 5. Agent 1 will be tasked with updating a base station along the path during the next time window, so it needs an update from Agent 3 about its recent measurements and needs to be in position 6 during the next time window. Agent 6 needs to be in location 1 so it is ready to measure tree density with its LIDAR in the next time window. Agents 1 and 2 use downward-facing cameras to sense cooperatively, so their proximity requirement is permanent.

We used Algorithm 1 to generate a braid string that satisfies the given specification and meets the mixing limit of 15. The braid string was calculated in 43s from an automaton with 5749 states. This resulted in the string

$$\sigma_{spec} = \{\sigma_1 \cdot \sigma_3 \cdot \sigma_5\} \cdot \sigma_2 \cdot \sigma_3 \cdot \sigma_4 \cdot \{\sigma_3 \cdot \sigma_5\} \cdot \{\sigma_2 \cdot \sigma_4\} \cdot \sigma_1 \tag{37}$$

where the grouped substrings may occur simultaneously without violating the specification.

To validate the above results in a practical setting, the braid controllers were implemented on a team of six Khepera III differential-drive robots, which may be modeled using unicycle dynamics, i.e.,

$$\dot{x}_j = \begin{bmatrix} v_j \cos\theta_j & v_j \sin\theta_j \end{bmatrix}^T, \qquad \dot{\theta}_j = \omega_j.$$

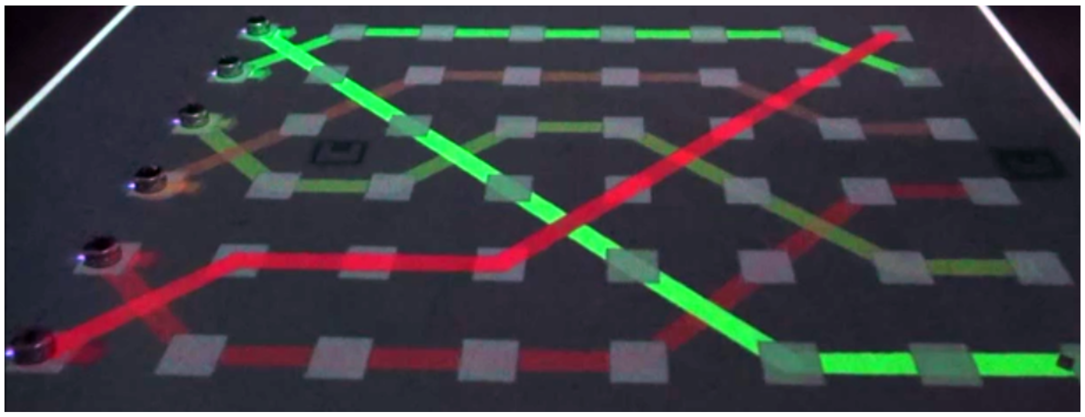
where $x_j \in \mathbb{R}^2$ is the robot's planar position and θ_j its heading. The single integrator control u_j from (35) can be mapped to unicycle dynamics as

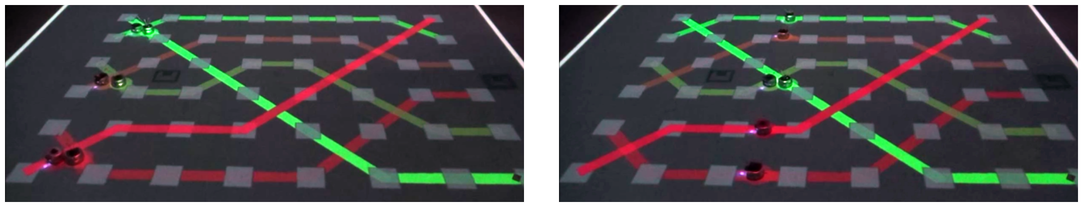
$$v_j = \begin{bmatrix} \cos\theta_j & \sin\theta_j \end{bmatrix} \cdot u_j, \qquad \omega_j = \begin{cases} \kappa \left[-\sin\theta_j \;\; \cos\theta_j \right] \cdot \frac{u_j}{\|u_j\|}, & \text{if } \|u_j\| > 1 \\ \kappa \left[-\sin\theta_j \;\; \cos\theta_j \right] \cdot u_j, & \text{otherwise} \end{cases}$$

for tuning gain $\kappa > 0$.

The robots were controlled over WiFi UDP from an Ubuntu (version 14.04LTS) computer with a 2.8GHz processor and 5.8GB RAM, running ROS (Robot Operating System, Indigo distribution). This computer also received robot state information from ten OptiTrack S250e motion capture cameras. Figure 21 shows the actual execution of the mixing strategy on the robots at two stages of a $60s$ total mission time-window, where a visual representation of the braid string geometry is being projected onto the robot workspace via an overhead projector mounted on the ceiling of the lab space. The braid points were uniformly distributed on a rectangular space of length $\ell = 3.4m$ and height $h = 2.5m$. Straight lines were chosen as the braid strands' geometric interpretation. Optimal trajectory tracking controllers are used to minimize the error between the robots actual trajectories and the desired specification-satisfying trajectories, as described in Section 4.5.

Figure 21b shows the agents at a stage of interaction, where an agent exits the safety separation region before the other one enters. Figure 22a illustrates the instantaneous minimum inter-agent distance throughout the execution. It can be seen that the minimum inter-robot distance achieved is approximately $0.132m$ — since the Khepera's have a diameter of $0.13m$, no collisions were observed during execution. Figure 21c illustrates the robots simultaneously arriving at a set of braid points. Figure 22b illustrates the robot trajectories in the plane. The optimal tracking controller compensates for deviations due to velocity saturation and the robots' dynamics, thus ensuring the controller remains collision-free and braid points are reached while successfully satisfying the mission specification.


(a) The robots start at the beginning of the braid.


(b) *Collision-free* – robots get as close as δ.

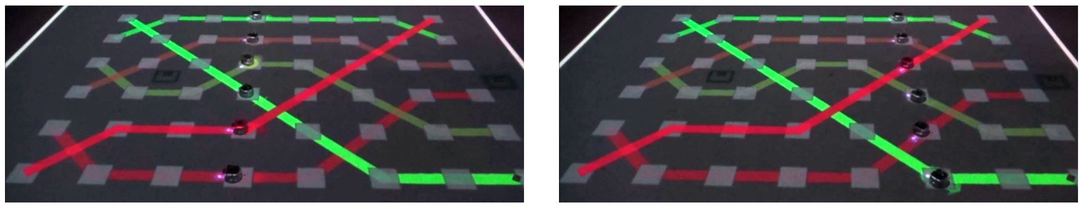
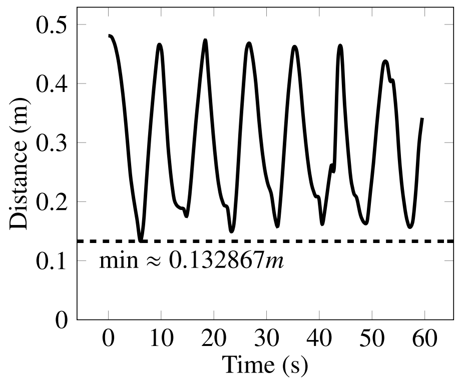
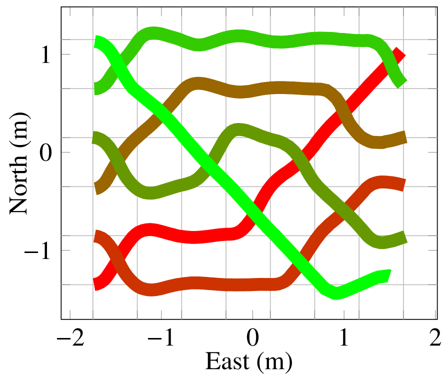

(c) The controller is *braid point feasible* – braid points are reached simultaneously.

Figure 21. Robots executing the mixing strategy in (37)**. The geometric paths and spatio-temporal constraints are being projected on the workspace with an overhead projector for the sake of visualization.**



(a) Instantaneous minimum inter-robot distance.



(b) Robot trajectories in the plane.

Figure 22. Data associated with robotic implementation in Figure 21.

4.6 Concluding Remarks

In this chapter, we constructed a framework for multi-robot mixing – a multi-robot motion planning scheme that allows users to provide rich, temporally-layered specification to characterize and specify desired levels of interactions and spatio-temporal constraints, e.g., for the sake of coordinated sensing, information exchange, and collision avoidance. We use elements of the braid group, which allows us to abstractly characterize classes of interaction patterns (as opposed to particular ones) and construct controllers to generate trajectories that satisfy these symbolic inputs. This results in a hybrid control system that is driven by strings of these symbols, which are then mapped onto actual motion patterns that both realize the desired interaction levels and remain safe in the sense that collisions are avoided. We further construct a new specification language with syntax similar to linear temporal logic. This new language allows us to provide rich, temporally-layered specifications to the multi-robot mixing framework for specifying frequency on interactions among agents, desired position configurations, and other pairwise distance criteria. Algorithms are presented to significantly reduce the search space of specification satisfying symbolic inputs with exactness guarantees. This approach is also tested on robotic platforms.

It should be noted that one of the benefits to having these layers of abstraction is that the characterization of interactions may extend to more than agents coming in close proximity of each other. It is conceivable that one encodes interactions in braid temporal logic referring to more abstract notions, such as those pertaining to routing and queueing systems. Further, the motion planning aspect presented in here focused on reference path tracking and spatio-temporal constraint satisfaction, however other notions of motion planning could be used to construct braid controllers (e.g., potential fields, Lyapunov controlled functions, barrier functions) – as long as they accept a symbolic braid as an input.

As stated above, the motion planning aspect presented in this chapter focused on reference path tracking and spatio-temporal constraint satisfaction. In the next chapter, we present research on these subjects from a robot navigation and control point of view.

CHAPTER 5
ROBOT NAVIGATION

In this chapter, we consider two robot navigation and control subjects. In Section 5.1, we explore the problem of finding the (globally) shortest reference path in a cluttered environment between two points in three dimensions. An algorithm is presented that is complete and has one of the best complexities found in the literature. This algorithm is also implementable online as is demonstrated by a robotic implementation on a quadcopter robot navigating through a field with obstacles.

In Section 5.2, research is presented on generating optimal trajectories for tracking a given reference path while satisfying given required navigation performance criteria. The novelty comes in two fronts: introducing a reparameterization of the reference path as part of a trajectory-tracking optimization problem in order to better accommodate vehicle dynamics; and the application of these optimal control techniques in flight management systems for the sake of enabling next generation air traffic control system concepts.

5.1 Shortest Path through 3D Cluttered Environments

In this section we consider a single agent attempting to find the shortest feasible path between two points in $\mathbb{R}^3$ in a cluttered environment such that a robot does not collide with any obstacles. It is our goal to do so in an online and computationally tractable fashion. Taking advantage of certain properties about paths in Euclidean space and recently developed tools, namely intermittent diffusion, we will obtain a finite dimensional stochastic differential equation which can be efficiently solved numerically forward in time to produce, with certain probability guarantees, the shortest path between these points. The scheme itself is quite intuitive: gradient descent will be used to find a locally short paths, then a certain amount of noise will be injected into the system to "kick" the solution out of local traps, followed by another application of gradient descent to find (potentially) different local minima. This process is repeated for a certain number of iterations, and the shortest path found will be the (globally) shortest path with prescribed probability δ.

5.1.1 Problem Formulation

Formally, we'll consider the problem of N obstacles in $\mathbb{R}^3$, which we'll denote by a connected and compact set $P_k \subset \mathbb{R}^3$, $k = 1, \ldots, N$. We are interested in the least-length rectifiable and feasible path (in that it does not intersect with the interior of any obstacles) described by a curve γ in $\mathbb{R}^3$, which is a continuous map $\gamma\colon [0,1] \to \mathbb{R}^3$. A path with finite length is said to be rectifiable, and feasible if

$$\phi_k(\gamma(t)) \geq 0, \quad t \in [0,1], \quad 1 \leq k \leq N. \tag{38}$$

where ϕ_k is the signed distance function to the k^{th} obstacle, i.e., negative if inside the obstacle, minimum Euclidean distance to the obstacle if outside of it, or the shortest geodesic if the two points lie on the boundary. As we wish to avoid collisions with the obstacles, we need to accommodate for the robot's footprint. Fortunately, the level set representation of obstacles in (38) enables us to deal with the robot's footprint in a straightforward manner. If we assume that the robot can be approximated by a ball of radius r (see figure 23), then all that one needs to do is enlarge all obstacles by this radius r, which boils down to a uniform decrease in the level set function. We may restate the definition of feasibility to: a path is said to be feasible for a "ball robot" with radius r if

$$\phi_k(\gamma(t)) - r \geq 0, \quad t \in [0,1], \quad 1 \leq k \leq N. \tag{39}$$

We can now formally state the problem at hand: if we let F be the set of all feasible rectifiable paths which start at X and end at Y (i.e., $\gamma(0) = X, \gamma(1) = Y$), we're interested in γ_{opt} given by

$$\gamma_{opt} = \arg\min_{\gamma \in F} L(\gamma).$$

The shortest path possesses a special structure that can be harnessed for the purpose of path planning: the optimal path γ_{opt} consists only of straight line segments and geodesics

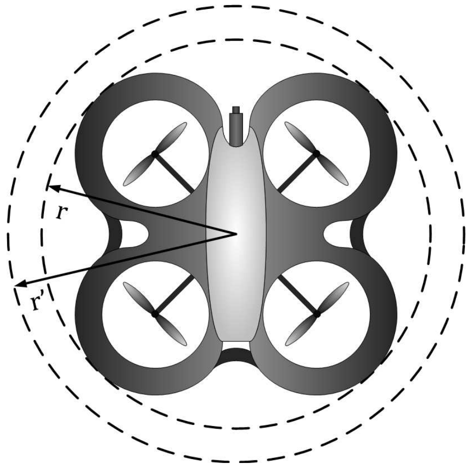


Figure 23. A robot's footprint is approximated by a ball of radius r. This ball can be enlarged to r' in order to provide a safety margin.

on the boundaries of obstacles. This implies that we can represent γ_{opt} by a sequence of junctions (special points) that connect different line segments or obstacle boundary

$$(x_0, x_1, \dots, x_n, x_{n+1}), \quad x_0 = X, \quad x_{n+1} = Y.$$

Each x_i connects to its neighbors x_{i-1} and x_{i+1} either by a line segment or a curve on the boundary. Let x_i^c be the adjacent junction that connects to x_i by a curve and x_i^s the junction that connects to x_i by line segment ($x_0^c = x_0$, $x_{n+1}^c = x_{n+1}$), as shown in figure 24, where in the example $x_{i+1}^s = x_i$, $x_{i+1}^c = x_{i+2}$, $x_{i+2}^s = x_{i+3}$, and $x_{i+2}^c = x_{i+1}$.

This structure enables us to search for the optimal path through a sequence of junctions $(x_0, x_1, \dots, x_n, x_{n+1})$. Each x_i (except x_0 and x_n) is the ending point of exactly one line

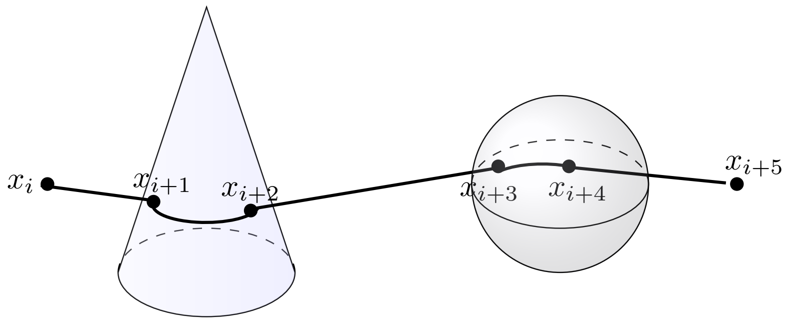


Figure 24. The shortest feasible path with obstacles and the junction points that represent it.

segment and one geodesic on the boundary. The sum of their length is given by

$$J(x_i) = \left\| x_i - x_i^s \right\| + d_{n_i}(x_i, x_i^c), \tag{40}$$

where n_i is the index such that $x_i \in P_{n_i}$ and $d_{n_i}(x_i, x_i^c)$ is the shortest geodesic on P_{n_i} connecting x_i and x_i^c. The length of γ is then

$$L(\gamma) = L(x_1, \ldots, x_n) = \frac{1}{2} \sum_{i=0}^{n+1} J(x_i). \tag{41}$$

The above formulation converts the original infinite dimensional problem of finding a path into the a finite dimensional problem of finding the optimal junctions. Therefore, we can apply finite dimensional optimization techniques such as gradient descent to find a local minimizer.

To overcome the challenge that gradient descent is only able to find the local minimizer, we adopt a recently developed global optimization strategy called intermittent diffusion. The key idea of intermittent diffusion is that when the gradient flow gets stuck at a local minimizer, we introduce a certain amount of noise to the flow, which will kick the flow out of the local trap. The perturbed gradient flow becomes the following stochastic differential equation

$$dx_i = -\nabla J(x_i)dt + \sigma(t)\mathbf{T}(x_i, x_i^c)dW(t), \tag{42}$$

where $W(t)$ is the standard Brownian motion and $\sigma(t)$ is a step function representing the magnitude of the noise we add at time t. Here

$$\mathbf{T}(x_i, x_i^c) = -\nabla d_{n_i}(x_i, x_i^c),$$

i.e., $\mathbf{T}(x_i, x_i^c)$ is the tangent direction at x_i on the curve belonging to the shortest path connecting x_i and x_i^c. The tangent on this curve exists, even when x_i is a critical point of ∂P_{n_i}, for example, at the tip of the cylinder in figure 25, as long as $x_i \neq x_i^c$.

We also have that $\sigma(t)$ is given by

$$\sigma(t) = \sum_{i=1}^{m} \sigma_i \mathbf{1}_{[S_i, T_i]}(t).$$

Here $t \in [0, T]$ and $\mathbf{1}_{[S_i,T_i]}(t)$ is the indicator function of the interval $[S_i, T_i]$, i.e. turns the noise "on" if $t \in [S_i, T_i]$, and "off" when not in these intervals. The parameters S_i and T_i form a partition of $[0, T]$, i.e., $0 = S_1 < T_1 < S_2 < T_2 < \cdots < S_m < T_m < S_{m+1} = T$, and represent the duration of the injected noise. The parameters σ_i represent the magnitude of the injected noise. An example of $\sigma(t)$ is depicted in figure 26. Given a desired minimum probability of obtaining the global optimizer, the parameters m and T are then selected in order to comply with this probability.

The idea behind the choice of σ is that when $\sigma(t) = 0$, x_i converges to a local minimizer following the negative gradient flow and jumps out of a local trap when $\sigma(t) > 0$ with certain probability. Moreover, by storing the solutions obtained when $\sigma(t) = 0$, we will be able to obtain not only the global optimizer but also a series of local minimizers. This is useful when limited time is allowed for the algorithm to run.

The following theorem captures how the intermittent diffusion algorithm works:

Theorem 6. *Given any real number $\delta > 0$, there exist $\tau > 0$, $\sigma_0 > 0$ and integer $m_0 > 0$ such that if $T_i - S_i > \tau$, $\sigma_i < \sigma_0$ (for $i = 1, 2, \cdots, m$), then equation* (42) *converges to the global minimizer of equation* (41) *with probability at least* $1 - \delta$.

This theorem is a direct application of intermittent diffusion and a detailed description, including details on how to select τ, σ_0 and m_0, can be found in [100].

To solve (42), we can discretize it by many well established schemes. For example,

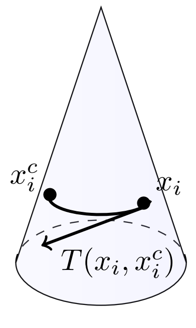


Figure 25. An illustration of T(x_i, x_i^c).

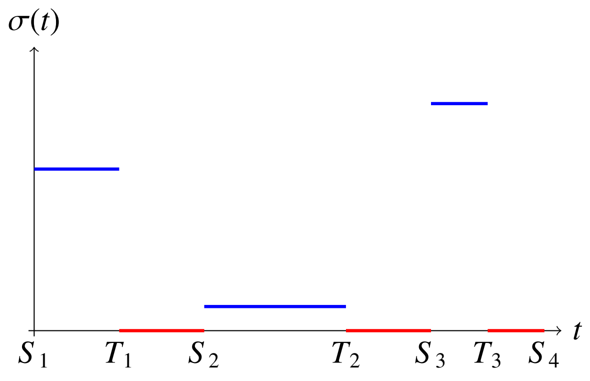


Figure 26. Step function for the magnitude of noise to be injected at time t.

we can consider the Euler scheme, in which the noise term $dW(t)$ is discretized in time as $dW(t) = \sqrt{\Delta t}\,\xi$, where $\xi \sim N(0,1)$ is a standard, normal random variable and Δt is the step size. Plugging in (40) into (42) results in

$$dx_i = -\frac{x_i - x_i^s}{\left\|x_i - x_i^s\right\|}dt + (\sigma(t)dW(t) + dt)\mathbf{T}(x_i, x_i^c),$$

which when combined with the Euler scheme yields the discretization

$$x_i^{k+1} = x_i^k - \frac{x_i^k - (x_i^k)^s}{\left\|x_i^k - (x_i^k)^s\right\|}\Delta t + (\sigma(k\Delta t)\sqrt{\Delta t}\,\xi + \Delta t)\mathbf{T}(x_i^k, (x_i^k)^c). \tag{43}$$

5.1.2 Online Implementation

An algorithm was proposed in [53] to find the set of junctions that represents the shortest path using (43). We present this algorithm below.

```
Algorithm 2: Finding the Shortest Path.
Input          : level set function $\phi_k(x)$,
                 the distance function $d_k(x, y)$,
                 starting and ending points $X$ and $Y$,
                 number of intermittent diffusion intervals $m$.
Output         : The optimal set $U_{opt}$ of junctions.
Initialization: Find the initial set $U$ of junction points.
Select duration of diffusion $\Delta T_\ell$, $\ell \leq m$;
Select diffusion coefficients $\sigma_\ell$, $\ell \leq m$;
for $\ell = 1, \ldots, m$ do
    $U_\ell = U$;
    for $x_i \in U_\ell$ do
        for $j = 1, \ldots, \Delta T_\ell$ do
            Update $x_i$ according to (43) with $\sigma(t) = \sigma_i$;
            Update set $U_\ell$, i.e. remove junctions from or add junctions to $U_\ell$;
        end
        while $\left\|x_i^{k+1} - x_i^k\right\| > \epsilon$ do
            Update $x$ according to (43) with $\sigma(t) = 0$;
            Update set $U_\ell$;
        end
    end
end
Compare $U_\ell$'s and set $U_{opt} = \arg\min_{\ell \leq m} L(U_\ell)$. Compute $s_k$ as above for $k = 1, \ldots, N_\ell$;
```

The initial set U of junctions can be set as the intersection points of the line segment $\overline{XY}$ and the obstacles. This initialization gives initial path similar to those generated by bug algorithms [101], and in many cases is already close enough to the global optimizer. It should be noted though that a good initialization is not required for the proposed algorithm.

Given enough time to run the algorithm, the global minimizer will still be obtained even starting from a far initial path.

Both the duration of diffusion ΔT_l and diffusion coefficients σ_l are randomly selected in intervals $[0, T_{\max}]$ and $[0, \sigma_{\max}]$ respectively. Experiments show that $T_{\max} = 20$ and $\sigma_{\max} = 2$ are often adequate. Depending on whether one wants to record local minimizers, line 20 can be replaced by keeping track only of the best minimizers at current realization.

We now give a brief analysis of the algorithm:

Completeness: Since we assume all the obstacles are bounded and disjoint, and we start from a feasible path, Theorem 6 guarantees the proposed algorithm is complete.

Complexity Following [102], instead of discussing the algebraic complexity of the algorithm, we will consider the running time in order to achieve certain relative error ϵ. We will compare our result with other approaches only for polyhedral obstacles since most of the literature focus on them.

1. The initialization can be done by a bisection line search, which can be achieved in $O(\log \frac{1}{\epsilon})$ time..

2. Line 2-3 takes $O(m)$ time.

3. Inner loop line 7-12 takes $O(\Delta T_l)$ time. This is because equation (43) takes constant time, and so does adding or removing junctions.

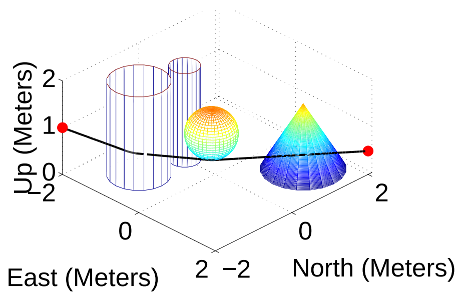
4. Inner loop line 13-17 takes $T(\epsilon)$ time where $T(\epsilon)$ denotes the number of iterations required until the error is less than ϵ. If we assume the Hessian matrix of the gradient is nondegenerate, which is the case for all polyhedral obstacles [103], then $T(\epsilon) = O(\log \frac{1}{\epsilon})$.

Let $\Delta T = \max_{i \leq l} \Delta T_i$. Then the total running time is $O(m(\Delta T + \log \frac{1}{\epsilon}))$. From [100], it can be shown that in order to obtain the desired successful probability $1 - \delta$, the number of realizations must be of order $O(\log \frac{1}{\delta})$. Therefore, the complexity is $O(\log \frac{1}{\delta} \log \frac{1}{\epsilon})$. Table 1 shows a complexity comparison with some existing methods.

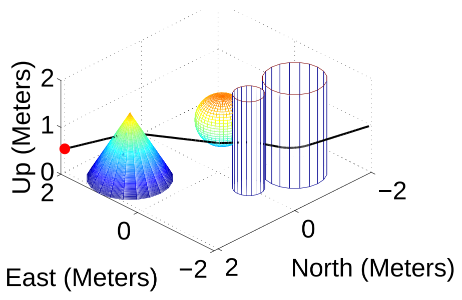
Table 1. Complexity comparison to other algorithms.

Algorithm	Complexity
A^*	$O((\frac{1}{\epsilon})^3 \log \frac{1}{\epsilon})$
Papadimitriou [102]	$O(\frac{1}{\epsilon})$
Choi, et. al. [103] (When the shortest path is not unique.)	$O(\frac{1}{\epsilon})$
Choi, et. al. [103] (When the shortest path is unique.)	$O(\log \frac{1}{\epsilon})$

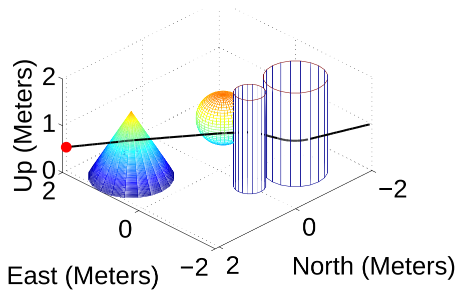
Figure 27 illustrates several local minimizers obtained from the algorithm in an environment with 4 obstacles. Note that even though their lengths are very comparable, their geometry can be very different. It is possible to then select a short path from the list in order meet other criteria, such as increasing the margin of safety, rather than following the absolute shortest.



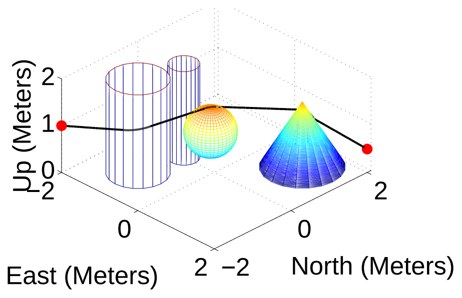
(a) Minimizer 1: Path length = 5.8660 m



(b) Minimizer 2: Path length = 5.9527 m



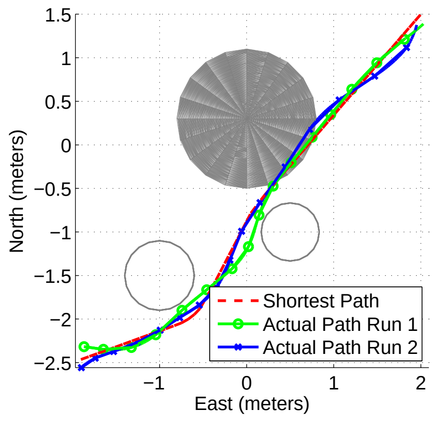
(c) Minimizer 3: Path length = 6.0403 m



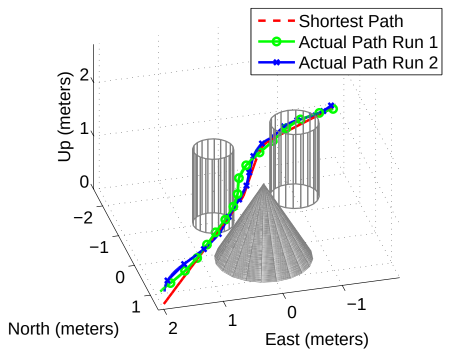
(d) Minimizer 7: Path length = 6.2305 m

Figure 27. The three shortest and the longest minimizer in an environment with four obstacles.

Further, the feasibility, in a dynamical sense, of the online algorithm is demonstrated when it is implemented on a Parrot AR.Drone quadrotor robot. Two separate runs of the quadrotor tracking the shortest path are depicted in figure 28. Figures 29a and 29b illustrate the actual robot executing these runs at two different points along their trajectory.

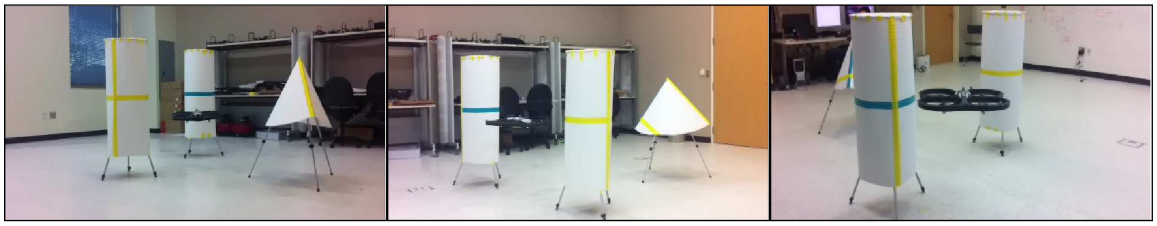


(a) Top view.

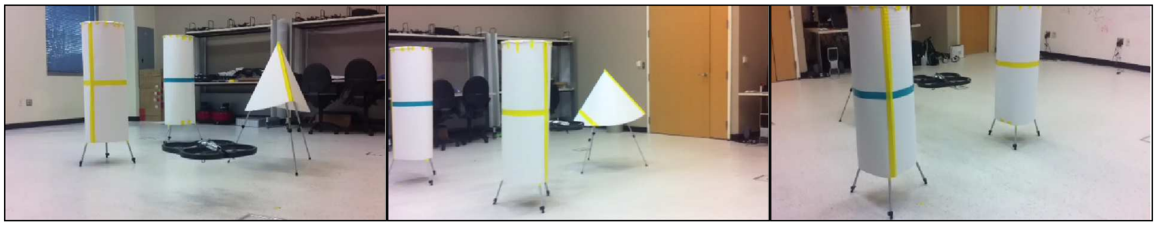


(b) Angled view.

Figure 28. Shortest reference and robot's actual paths.



(a) A third of the way: front, profile and back view.



(b) Two thirds of the way: front, profile and back view.

Figure 29. Robotic implementation of shortest paths in cluttered environments.

5.2 Trajectory Generation from Reference Paths

The previous section presented research on generating the shortest path in a cluttered environment which is then to be followed by a robot. In this section, we consider the problem of generating trajectories that satisfy a set of spatio-temporal constraints, including closely following a desired reference path such as a road or a flight path, and arriving at specific locations along the path at certain times, e.g., like toll booths or metering points. Aside from this subset of locations with required time of arrivals (RTA), the provided reference path does not have required times of arrival. This implies that the path itself can be thought of as a set of spatial constraints without temporal constraints. Conventional trajectory-tracking solutions that use the reference path parameterization as provided may result costly maneuvers, or higher fuel expenditure. One could instead find a new temporal parameterization of the path to best accommodate vehicle dynamics, effectively turning into an optimal trajectory to be tracked. In particular, the following research explores the problem of reparameterizing a reference path to meet certain spatio-temporal constraints as part of the optimization problem.

This research was the result of an attempt to facilitate the development of the Next Generation Air Transportation System (NextGen) concepts. The idea is that prototype 4D flight management system (FMS) capabilities will need to generate computationally tractable, new trajectory solutions that comply with multiple RTA constraints, e.g., spatio-temporal constraints on certain points along a reference path.

The novelty of the research found in this section comes in two fronts: introducing a reparameterization of the reference path as part of a trajectory-tracking optimization problem in order to better accommodate vehicle dynamics; and the application of these optimal control techniques in flight management systems for the sake of enabling next generation air traffic control system concepts. More details on this research can be found in [104].

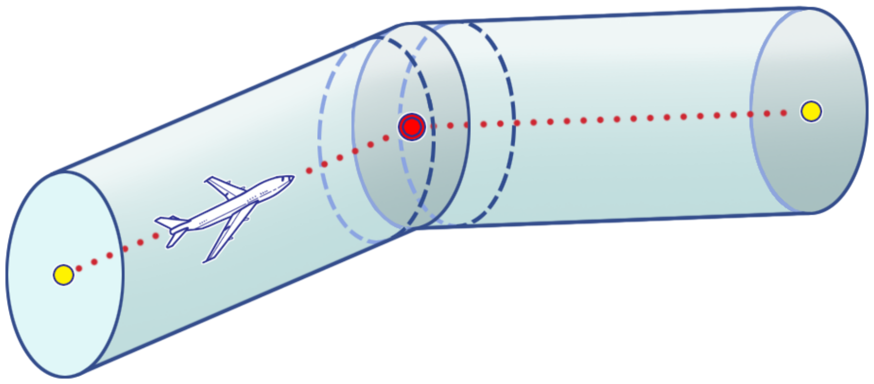


Figure 30. Representation of RNP Constraints. The dotted line represents the reference path. The tube represents the distance from the reference path that satisfies the RNP. The dots represent points with required times of arrivals. The dashed disks represent the spatial tolerance around the central spatio-temporal constraint.

5.2.1 Problem Statement

We are interested in finding a continuous trajectory that complies with required navigation performance (RNP), e.g., spatial non-temporal constraints, as well as RTA. Further, since control effort will directly tie in to the amount of fuel consumed, it is of interest to find a trajectory that will minimize the amount of energy required to arrive at all the waypoints. Using the parameterized reference path $r(t) \in \mathbb{R}^3$, one way to solve this problem is to express it as a generic tracking problem in the form of the optimal control problem

$$\begin{aligned} u^*(t) &= \arg\min_{u(t)} J(u(t)) \\ &= \arg\min_{u} \tfrac{1}{2} \int_0^T \left[u^T R u + (y - r(t))^T Q (y - r(t)) \right] dt + \tfrac{1}{2} (x(T) - x_T)^T S_T (x(T) - x_T) \end{aligned}$$

subject to the constraint $\dot{x} = f(x, u)$ with $x(0) = x_0$ and $y(t) = h(x)$, for $x \in \mathbb{R}^n$, $u \in \mathbb{R}^m$, $y \in \mathbb{R}^3$, positive definite symmetric matrices Q and R and positive semi-definite matrix S_T. The cost functional $J(u(t))$ encodes a cost on the expended control energy over the interval $[0, T]$ as weighted by cost matrix R, as well as a cost for deviating from the reference path $r(t)$ as weighted by cost matrix Q. The terminal cost penalizes any trajectory that deviates from the RTA and other terminal state requirements as weighted by cost matrix S_T. A visual representation of these constraints can be seen in Figure 30.

Note that the path specified in $r(t)$ only has RTAs at its endpoints. This means that we only need there to exist a $\tau \in (0, T)$ such that $\|y(t) - r(\tau)\| <$ RNP for all $t \in (0, T)$, as opposed to tracking $r(t)$ as it is parameterized. In order to reduce the tracking cost and accommodate for the vehicle's dynamics, we can reparameterize the path with a new parameter $p \in \mathbb{R}$ which is the solution to the differential equation $\dot{p} = v$. We will think of this new reparameterization as a virtual vehicle that moves perfectly along the reference path, and whose velocity along it we get to control directly. The real vehicle needs to track this virtual vehicle whose velocity will be designed to reduce the real vehicle's control effort. We can modify the problem to include the virtual vehicle's velocity

$$\begin{aligned}\operatorname*{arg\,min}_{u(t),v(t)} J(u(t), v(t)) &= \operatorname*{arg\,min}_{u(t),v(t)} \int_0^T L(x(t), u(t), r(p(t)), v(t))\, dt + \Psi(x(T)) \\ &:= \operatorname*{arg\,min}_{u,v} \tfrac{1}{2} \int_0^T \left[u^T R u + (y - r(p))^T Q\, (y - r(p)) + P\, (v - v_t)^2\right] dt \\ &\qquad + \tfrac{1}{2}\, (x(T) - x_T)^T\, S_T\, (x(T) - x_T) \end{aligned} \tag{44}$$

subject to the constraints

$$\begin{array}{lll} \dot{p}(t) = v(t), & p(0) = 0, & p(T) = T \\ \dot{x}(t) = f(x, u), & x(0) = x_0, & y(t) = h(x) \end{array}$$

for $P > 0$. We can interpret the cost of $y(t)$ as how closely we are tracking the virtual vehicle and we can interpret the cost of $v(t)$ as penalizing for letting $p(t)$ flow too differently from the original parameterization, e.g., if the reference path was originally parameterized by t, then $v_t = 1$ implies that $p(t)$ should be close to real time.

We let $\lambda_x(t)$ and $\lambda_p(t)$ be the so called co-states associated with $x(t)$ and $p(t)$, respectively. Using the standard variational argument from calculus of variations and Pontryagin's maximum principle, we obtain the first order necessary conditions for optimality

$$\begin{array}{ll} \dot{\lambda}_x(t) = -\frac{\partial f}{\partial x}^T \lambda_x(t) - \frac{\partial L}{\partial x}^T & \\ & 0 = \frac{\partial f}{\partial u}^T \lambda_x(t) + \frac{\partial L}{\partial u}^T \\ \dot{\lambda}_p(t) = -\frac{\partial r}{\partial p}^T \frac{\partial L}{\partial r}^T & \\ & 0 = \lambda_p(t) + \frac{\partial L}{\partial v}. \\ \lambda_x(T) = \left.\frac{\partial \Psi}{\partial x}^T \right|_{t=T} & \end{array}$$

These equations turn the original optimization problem into a two-point boundary problem, and provide enough information together with the constraint set to obtain the optimal solution $u^*(t)$ and $v^*(t)$. The two-point boundary problem could be numerically solved to obtain an open-loop solution, where the control signal could be computed offline and stored for later use when computational power is not available online. In practice though, this strategy is very fragile to noise and perturbations and the use of the open-loop optimal control signal could in fact result in great deviations from the desired results. In order to avoid this scenario we turn to a closed-loop control strategy, i.e., instead of finding the optimal control signal value for the time interval, we find the gains to form a feedback loop that results in the optimal output.

5.2.2 Feedback Form Optimal Control Signal

In autopilot design for aircrafts, there are seldom high-fidelity models available to describe the aircraft dynamics. This is due to the high complexity and level of uncertainty with aero coefficients. Instead, lookup tables with empirical aero coefficients based on current flight conditions are used to generate linearized models. If high-fidelity models were available, then one could use the optimal control signals $u^*(t)$ and $v^*(t)$ with the model to generate the nominal optimal trajectories $x^*(t)$ and $p^*(t)$. One could linearize around these optimal trajectories to obtain locally linear models for which a closed loop strategy could be formulated. Whichever the case, it is of interest to consider a linear model for the system in the form

$$\dot{x}(t) = A(t)x(t) + B(t)u(t) \qquad\qquad y(t) = C(t)x(t)$$

Additionally, we will consider the special case where the reference path r is given by a straight line connecting some initial point $y(0) = Cx_0$ and some final point $y(T) = Cx_T$, e.g., the shortest path in free space. This kind of reference path can be expressed parametrically as a function of $p \in [0, T]$ by

$$r(p) = \left(1 - \tfrac{p}{T}\right) C(t)x_0 + \tfrac{p}{T} C(t)x_T = C(t)\left(\tfrac{x_T - x_0}{T}\right) p + C(t)x_0 =: C(t)x_s p + C(t)x_0.$$

which has a derivative given by

$$\tfrac{dr}{dp} = C(t)x_s.$$

Using the optimality conditions, we can plug in the value of $u(t)$ and $v(t)$ obtained from the second column into the state and co-state dynamics to get the following (affine) two-point boundary problem

$$\dot{\hat{x}}(t) = \hat{A}\hat{x}(t) - \hat{B}\hat{R}^{-1}\hat{B}^T\hat{\lambda}(t) + \hat{B}k, \qquad \hat{x}(0) = \left[\begin{smallmatrix} x_0 \\ 0 \end{smallmatrix}\right] \tag{45}$$

$$\dot{\hat{\lambda}}(t) = -\hat{A}^T\hat{\lambda}(t) - \hat{Q}\hat{x}(t) + \hat{Q}\hat{x}(0), \qquad \hat{\lambda}(T) = \left[\begin{smallmatrix} S_T(x(T)-x_T) \\ \alpha_p \end{smallmatrix}\right] \tag{46}$$

for some yet undetermined $\alpha_p \in \mathbb{R}$, where $\hat{x} = [x^T, p]^T$ is the augmented state, $\hat{\lambda} = [\lambda_x^T, \lambda_p]^T$ is the augmented co-state, and

$$\hat{A} = \left(\begin{smallmatrix} A & 0_{n\times 1} \\ 0_{1\times n} & 0 \end{smallmatrix}\right) \quad \hat{B} = \left(\begin{smallmatrix} B & 0_{n\times 1} \\ 0_{1\times m} & 1 \end{smallmatrix}\right) \quad \hat{Q} = \left(\begin{smallmatrix} C^TQC & -C^TQCx_s \\ -x_s^TC^TQC & x_s^TC^TQCx_s \end{smallmatrix}\right) \quad \hat{R} = \left(\begin{smallmatrix} R & 0_{m\times 1} \\ 0_{1\times m} & P \end{smallmatrix}\right) \quad k = \left[\begin{smallmatrix} 0_{m\times 1} \\ v_t \end{smallmatrix}\right].$$

Note that these are affine equations on the state and co-states, and we can use the sweep method [98] to obtain a solution in feedback form. We can express the set of zero-error terminal constraints (i.e., $p(T) = T$) as a linear combination of the initial states and the terminal co-states (plus an offset due to the fact that the dynamics are affine), i.e.,

$$p(T) = F(0)\hat{x}(0) + G(0)\lambda_p(T) + D(0)$$

for some $F : [0, T] \to \mathbb{R}^{1\times(n+1)}$, $G : [0, T] \to \mathbb{R}$, $D : [0, T] \to \mathbb{R}$. Similarly, the initial conditions for the co-state equation could also be expressed as a linear combination of the initial states and the terminal co-states (plus an offset), i.e.,

$$\hat{\lambda}(0) = S(0)\hat{x}(0) + K(0)\lambda_p(T) + E(0)$$

for some $S : [0, T] \to \mathbb{R}^{(n+1)\times(n+1)}$, $K : [0, T] \to \mathbb{R}^{n+1}$, $E : [0, T] \to \mathbb{R}^{n+1}$. Since the co-state is to be solved backwards in time, any time before the terminal time can potentially be an initial point, i.e., for $t \leq T$ it is possible that

$$\hat{\lambda}(t) = S(t)\hat{x}(t) + K(t)\lambda_p(T) + E(t) \tag{47}$$

$$p(T) = F(t)\hat{x}(t) + G(t)\lambda_p(T) + D(t) \tag{48}$$

We can differentiate (47) and (48) with respect to time and plug in (45), (46) and (47) to obtain that $F(t) = K^T(t)$ and the following set of differential equations with their appropriate boundary conditions which make use of the optimality conditions in (46)

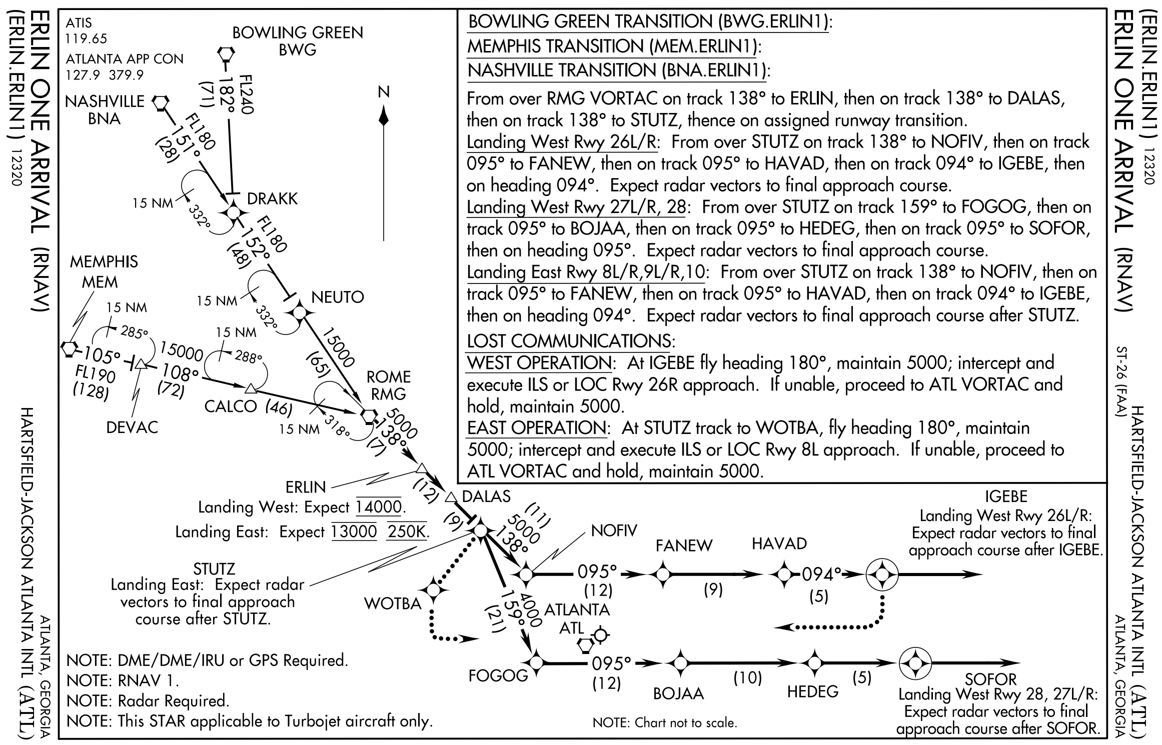
$$\begin{aligned}
&\dot{S}(t) = -S(t)\hat{A}(t) - \hat{A}^T(t)S(t) + S(t)\hat{B}(t)\hat{R}^{-1}\hat{B}^T(t)S(t) - \hat{Q}, && S(T) = \left(\begin{smallmatrix} S_T & 0_{n\times 1} \\ 0_{1\times n} & 0 \end{smallmatrix}\right) \\
&\dot{E}(t) = -\left(\hat{A}^T(t) - S(t)\hat{B}(t)\hat{R}^{-1}\hat{B}^T(t)\right)E(t) - S(t)\hat{B}(t)k + \hat{Q}\hat{x}(0), && E(T) = \left[\begin{smallmatrix} -S_T x_T \\ 0 \end{smallmatrix}\right] \\
&\dot{K}(t) = -\left(\hat{A}^T(t) - S(t)\hat{B}(t)\hat{R}^{-1}\hat{B}^T(t)\right)K(t), && K(T) = \left[\begin{smallmatrix} 0_{n\times 1} \\ 1 \end{smallmatrix}\right] \\
&\dot{D}(t) = K^T(t)\hat{B}(t)\hat{R}^{-1}\hat{B}^T(t)E(t) + K^T(t)\hat{B}(t)k, && D(T) = 0 \\
&\dot{G}(t) = K^T(t)\hat{B}(t)\hat{R}^{-1}\hat{B}^T(t)K(t), && G(T) = 0
\end{aligned}$$

where $G(t) < 0$ since $G(T) = 0$ and $\dot{G}(t) > 0$ for $t \leq T$. These differential equations are either differential Riccati equations, linear time-varying equations, or quadratures, which have been broadly studied, and efficient numerical methods exist to solve them. In particular, it is possible to sequentially solve these equations by sweeping backwards in time from the terminal condition in the order presented above. Note that we may solve for the co-state terminal condition in (48) to obtain

$$\lambda_p(T) = G^{-1}(t)\left[p(T) - K^T(t)\hat{x}(t) - D(t)\right]$$

which when combined with (47) results in the optimal $\hat{u}^*(t) = \left[u^{*T}(t), v^*(t)\right]^T$ feedback-feedforward strategy given by

$$\hat{u}^*(t) = -\hat{R}^{-1}\hat{B}^T(t)\left[\left(S(t) - K(t)G^{-1}(t)K^T(t)\right)\hat{x}(t) + K(t)G^{-1}(t)\left(p(T) - D(t)\right) + E(t)\right] + k \tag{49}$$



BOWLING GREEN TRANSITION (BWG.ERLIN1):
MEMPHIS TRANSITION (MEM.ERLIN1):
NASHVILLE TRANSITION (BNA.ERLIN1):
From over RMG VORTAC on track 138° to ERLIN, then on track 138° to DALAS, then on track 138° to STUTZ, thence on assigned runway transition.
Landing West Rwy 26L/R: From over STUTZ on track 138° to NOFIV, then on track 095° to FANEW, then on track 095° to HAVAD, then on track 094° to IGEBE, then on heading 094°. Expect radar vectors to final approach course.
Landing West Rwy 27L/R, 28: From over STUTZ on track 159° to FOGOG, then on track 095° to BOJAA, then on track 095° to HEDEG, then on track 095° to SOFOR, then on heading 095°. Expect radar vectors to final approach course.
Landing East Rwy 8L/R,9L/R,10: From over STUTZ on track 138° to NOFIV, then on track 095° to FANEW, then on track 095° to HAVAD, then on track 094° to IGEBE, then on heading 094°. Expect radar vectors to final approach course after STUTZ.
LOST COMMUNICATIONS:
WEST OPERATION: At IGEBE fly heading 180°, maintain 5000; intercept and execute ILS or LOC Rwy 26R approach. If unable, proceed to ATL VORTAC and hold, maintain 5000.
EAST OPERATION: At STUTZ track to WOTBA, fly heading 180°, maintain 5000; intercept and execute ILS or LOC Rwy 8L approach. If unable, proceed to ATL VORTAC and hold, maintain 5000.

ERLIN
Landing West: Expect 14000.
Landing East: Expect 13000 250K.

STUTZ
Landing East: Expect radar vectors to final approach course after STUTZ.

IGEBE
Landing West Rwy 26L/R: Expect radar vectors to final approach course after IGEBE.

SOFOR
Landing West Rwy 28, 27L/R: Expect radar vectors to final approach course after SOFOR.

NOTE: DME/DME/IRU or GPS Required.
NOTE: RNAV 1.
NOTE: Radar Required.
NOTE: This STAR applicable to Turbojet aircraft only.

Figure 31. ERLIN ONE Arrival chart.

5.2.3 Simulation Results

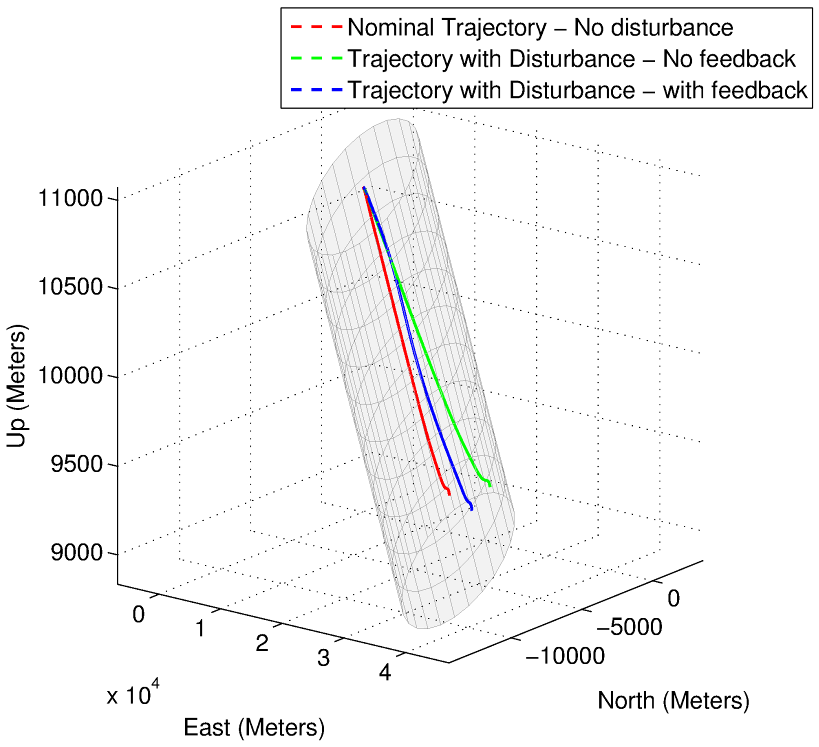
The approach described above was simulated to demonstrate its feasibility. In order to obtain results that are relevant to real-world scenarios, a descent scenario was chosen for Hartsfield-Jackson Atlanta International Airport (KATL). In particular, a descent trajectory was performed for a lateral path segment generated from the ERLIN ONE STAR chart at KATL airport (see Figure 31). The route was constructed from the consecutive connection of waypoints DEVAC-CALCO-ROME-ERLIN.

A nonlinear aircraft model was linearized around a nominal trajectory generated from open-loop numerical solutions to the two-point boundary problem. To test the robustness of the feedback form controllers, disturbances where introduced through a simple wind model obtained from a polynomial fit of random wind data samples generated from NOAA METAR data of KATL airport. In the simulation, the origin was placed at DEVAC at sea level. The x-coordinate represents East, the y-coordinate represents North, and the z-coordinate represents upward direction. It was assumed that the aircraft started at DEVAC at 35,000 feet above mean-sea level and at a true airspeed of 450 knots, oriented 105° from the North and leveled to the Earth. The aircraft is to continuously descend to 20,000 feet above sea level at CALCO, which is located 72.1 nautical miles away from DEVAC at 105° from the North. In particular, the focus was on the first 180 second segment of this flight. In order to evaluate RTA compliance performance, a virtual RTA was set up at the 180 seconds point along the continuous descent trajectory.

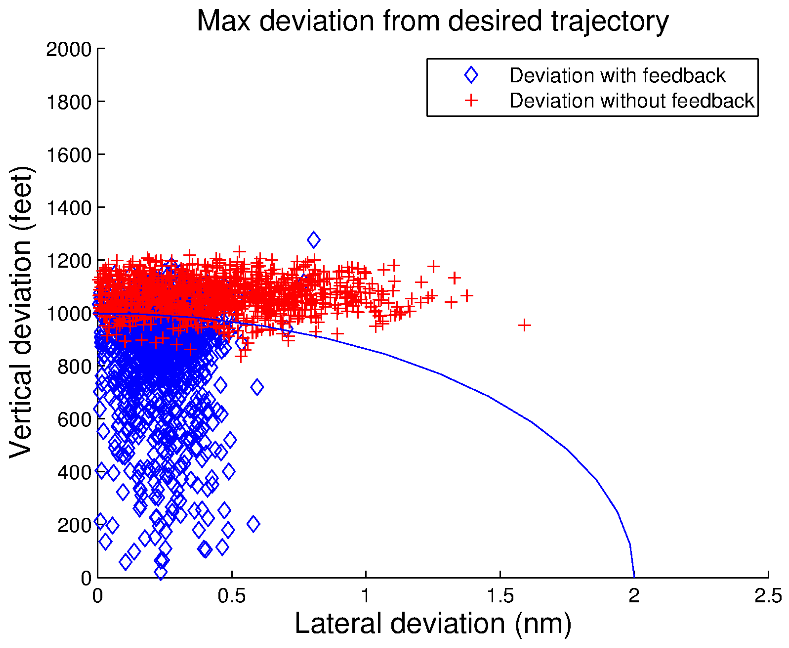
Monte Carlo simulations of the 180 second flight segment were run for 1,000 iterations where the mean and covariance for the wind sample were set to be 5 m/s and 15 m/s, respectively. The RNP value is 2 nautical miles for lateral deviation and 1,000 feet for vertical deviation. RTA compliance was defined as whether the aircraft was within 1 nautical miles of the desired point at final time. Minimum descent angle is defined as $-6°$. The nominal true airspeed is to decrease linearly as a function of altitude, 450 knots at 35,000 feet, and 300 knots at 10,000 feet. The allowable deviation from this nominal speed is defined to be 30 knots. Table 2 shows the compliance of these runs with the open loop and feedback control laws. Figure 32 show sample trajectories inside of the RNP tube representation, as well as the results from the Monte Carlo simulations.

Table 2. Constraint compliance during Monte Carlo runs.

	With Feedback	Open Loop
RNP Compliance	74.9%	9.5%
RTA Compliance	99.3%	98.8%
Minimum Descent Angle Compliance	100%	
True Airspeed Compliance	97.7%	



(a) Sample trajectories in RNP tube.



(b) Max deviation from Monte Carlo simulations. Trajectories outside the tube violate the RNP.

Figure 32. Trajectories obtained from Monte Carlo simulations.

CHAPTER 6
CONCLUSIONS

This dissertation provides contributions with respect to two topics in multi-robot systems. The first is a framework for human-swarm interactions in which the human operator abstracts away the number of agents in the robot swarm and manipulates it as a whole by directly imposing the desired density of robots in the environment. In order to pursue this approach, contributions were made to the problem of coverage of time-varying functions – for which there was a dearth in the literature. In particular, important spectral properties are provided which are used to provide convergence guarantees for the presented control laws. In addition to these, a family of distributed approximations is presented which allow the agents to provide optimal coverage of the human-generated density functions while only relying on local information.

The second major contribution comes from a framework for multi-robot mixing – a multi-robot motion planning scheme that allows users to provide rich, temporally-layered specification to characterize and specify desired levels of interactions and spatio-temporal constraints, e.g., for the sake of coordinated sensing, information exchange, and collision avoidance. We use elements of the braid group, which allows us to abstractly characterize classes of interaction patterns (as opposed to particular ones) and construct controllers to generate trajectories that satisfy these symbolic inputs. This results in a hybrid control system that is driven by strings of these symbols, which are then mapped onto actual motion patterns that both realize the desired interaction levels and remain safe in the sense that collisions are avoided. We further construct a new specification language with syntax similar to linear temporal logic. This new language allows us to provide rich, temporally-layered specifications to the multi-robot mixing for specifying frequency on interactions among agents, desired position configurations, and other pairwise distance criteria. Algorithms are presented to significantly reduce the search space of specification satisfying symbolic inputs with exactness guarantees. This approach is also tested on robotic platforms.

Novel work is also presented with respect to robot navigation and control. In particular, work is done on finding the shortest path between two points in cluttered environments. An algorithm is presented that has one of the best complexities found in the literature. This algorithm is also implementable online as demonstrated by a robotic implementation on a quadcopter robot navigating a field with obstacles. In addition to the work on finding the shortest path, work is also presented on generating optimal trajectories for tracking a given reference path while satisfying required navigation performance. The novelty comes in two fronts: introducing a reparameterization of the reference path as part of a trajectory tracking optimization problem in order to better accommodate vehicles dynamics; and the application of these optimal control techniques in flight management systems for the sake of enabling next generation air traffic control system concepts.

REFERENCES

[1] T. Balch and R. C. Arkin, “Behavior-based formation control for multirobot teams,” *Robotics and Automation, IEEE Transactions on*, vol. 14, no. 6, pp. 926–939, 1998.

[2] M. Egerstedt and X. Hu, “Formation constrained multi-agent control,” *Robotics and Automation, IEEE Transactions on*, vol. 17, pp. 947–951, Dec 2001.

[3] B. P. Gerkey and M. J. Matarić, “A formal analysis and taxonomy of task allocation in multi-robot systems,” *The International Journal of Robotics Research*, vol. 23, no. 9, pp. 939–954, 2004.

[4] L. C. Pimenta, V. Kumar, R. C. Mesquita, and G. A. Pereira, “Sensing and coverage for a network of heterogeneous robots,” in *Decision and Control, 2008. CDC 2008. 47th IEEE Conference on*, pp. 3947–3952, IEEE, 2008.

[5] B. Grocholsky, J. Keller, V. Kumar, and G. Pappas, “Cooperative air and ground surveillance,” *Robotics & Automation Magazine, IEEE*, vol. 13, no. 3, pp. 16–25, 2006.

[6] O. F. Rana and K. Stout, “What is scalability in multi-agent systems?,” in *Proceedings of the fourth international conference on Autonomous agents*, pp. 56–63, ACM, 2000.

[7] M. Georgeff, “Communication and interaction in multi-agent planning,” *Readings in distributed artificial intelligence*, vol. 313, pp. 125–129, 1988.

[8] Y. U. Cao, A. S. Fukunaga, and A. Kahng, “Cooperative mobile robotics: Antecedents and directions,” *Autonomous robots*, vol. 4, no. 1, pp. 7–27, 1997.

[9] R. Olfati-Saber, “Flocking for multi-agent dynamic systems: Algorithms and theory,” *Automatic Control, IEEE Transactions on*, vol. 51, no. 3, pp. 401–420, 2006.

[10] S. Mastellone, D. M. Stipanović, C. R. Graunke, K. A. Intlekofer, and M. W. Spong, “Formation control and collision avoidance for multi-agent non-holonomic systems: Theory and experiments,” *The International Journal of Robotics Research*, vol. 27, no. 1, pp. 107–126, 2008.

[11] T. Balch, “The impact of diversity on performance in multi-robot foraging,” in *Proceedings of the third annual conference on Autonomous Agents*, pp. 92–99, ACM, 1999.

[12] K. Sugawara, T. Kazama, and T. Watanabe, “Foraging behavior of interacting robots with virtual pheromone,” in *Intelligent Robots and Systems, 2004. (IROS 2004). Proceedings. 2004 IEEE/RSJ International Conference on*, vol. 3, pp. 3074–3079 vol.3, Sept 2004.

[13] J. F. Traniello, “Foraging strategies of ants,” *Annual review of entomology*, vol. 34, no. 1, pp. 191–210, 1989.

[14] D. H. Morse, “Ecological aspects of some mixed-species foraging flocks of birds,” *Ecological monographs*, pp. 119–168, 1970.

[15] M. Bonert, L. Shu, and B. Benhabib, “Motion planning for multi-robot assembly systems,” *International Journal of Computer Integrated Manufacturing*, vol. 13, no. 4, pp. 301–310, 2000.

[16] W. Nguyen and J. K. Mills, “Multi-robot control for flexible fixtureless assembly of flexible sheet metal auto body parts,” in *Robotics and Automation, 1996. Proceedings., 1996 IEEE International Conference on*, vol. 3, pp. 2340–2345, IEEE, 1996.

[17] R. Groß, M. Bonani, F. Mondada, and M. Dorigo, “Autonomous self-assembly in swarm-bots,” *Robotics, IEEE Transactions on*, vol. 22, no. 6, pp. 1115–1130, 2006.

[18] E. Klavins, “Programmable self-assembly,” *Control Systems, IEEE*, vol. 27, no. 4, pp. 43–56, 2007.

[19] S. Dubowsky and P. Boning, “The coordinated control of space robot teams for the on-orbit construction of large flexible space structures,” in *2007 IEEE International Conference Robotics and Automation*, 2007.

[20] K. H. Petersen, R. Nagpal, and J. K. Werfel, “Termes: An autonomous robotic system for three-dimensional collective construction,” in *proceedings of robotics: science and systems conference (RSS)*, 2011.

[21] Q. Lindsey, D. Mellinger, and V. Kumar, “Construction with quadrotor teams,” *Autonomous Robots*, vol. 33, no. 3, pp. 323–336, 2012.

[22] S. M. Pedersen, S. Fountas, H. Have, and B. Blackmore, “Agricultural robotssystem analysis and economic feasibility,” *Precision agriculture*, vol. 7, no. 4, pp. 295–308, 2006.

[23] N. Noguchi, J. Will, J. Reid, and Q. Zhang, “Development of a master–slave robot system for farm operations,” *Computers and Electronics in agriculture*, vol. 44, no. 1, pp. 1–19, 2004.

[24] P. Ögren, E. Fiorelli, and N. E. Leonard, “Cooperative control of mobile sensor networks: Adaptive gradient climbing in a distributed environment,” *Automatic Control, IEEE Transactions on*, vol. 49, no. 8, pp. 1292–1302, 2004.

[25] A. Howard, M. J. Matarić, and G. S. Sukhatme, “An incremental self-deployment algorithm for mobile sensor networks,” *Autonomous Robots*, vol. 13, no. 2, pp. 113–126, 2002.

[26] G. Wang, G. Cao, T. L. Porta, and W. Zhang, "Sensor relocation in mobile sensor networks," in *INFOCOM 2005. 24th Annual Joint Conference of the IEEE Computer and Communications Societies. Proceedings IEEE*, vol. 4, pp. 2302–2312, IEEE, 2005.

[27] Z. Butler and D. Rus, "Event-based motion control for mobile-sensor networks," *Pervasive Computing, IEEE*, vol. 2, no. 4, pp. 34–42, 2003.

[28] N. Heo and P. K. Varshney, "Energy-efficient deployment of intelligent mobile sensor networks," *Systems, Man and Cybernetics, Part A: Systems and Humans, IEEE Transactions on*, vol. 35, no. 1, pp. 78–92, 2005.

[29] R. C. Shah, S. Roy, S. Jain, and W. Brunette, "Data mules: Modeling and analysis of a three-tier architecture for sparse sensor networks," *Ad Hoc Networks*, vol. 1, no. 2, pp. 215–233, 2003.

[30] I. Vasilescu, K. Kotay, D. Rus, M. Dunbabin, and P. Corke, "Data collection, storage, and retrieval with an underwater sensor network," in *Proceedings of the 3rd international conference on Embedded networked sensor systems*, pp. 154–165, ACM, 2005.

[31] W. Zhao, M. Ammar, and E. Zegura, "Controlling the mobility of multiple data transport ferries in a delay-tolerant network," in *INFOCOM 2005. 24th annual joint conference of the IEEE computer and communications societies. Proceedings IEEE*, vol. 2, pp. 1407–1418, IEEE, 2005.

[32] A. Howard, "Multi-robot simultaneous localization and mapping using particle filters," *The International Journal of Robotics Research*, vol. 25, no. 12, pp. 1243–1256, 2006.

[33] N. E. Özkucur and H. L. Akın, "Cooperative multi-robot map merging using fast-slam," in *RoboCup 2009: Robot Soccer World Cup XIII*, pp. 449–460, Springer, 2010.

[34] R. M. Murray, "Recent research in cooperative control of multivehicle systems," *Journal of Dynamic Systems, Measurement, and Control*, vol. 129, no. 5, pp. 571–583, 2007.

[35] X. C. Ding, A. R. Rahmani, and M. Egerstedt, "Multi-uav convoy protection: an optimal approach to path planning and coordination," *Robotics, IEEE Transactions on*, vol. 26, no. 2, pp. 256–268, 2010.

[36] M. Rubenstein, C. Ahler, and R. Nagpal, "Kilobot: A low cost scalable robot system for collective behaviors," in *Robotics and Automation (ICRA), 2012 IEEE International Conference on*, pp. 3293–3298, IEEE, 2012.

[37] M. Rubenstein, A. Cornejo, and R. Nagpal, "Programmable self-assembly in a thousand-robot swarm," *Science*, vol. 345, no. 6198, pp. 795–799, 2014.

[38] A. Kushleyev, D. Mellinger, C. Powers, and V. Kumar, "Towards a swarm of agile micro quadrotors," *Autonomous Robots*, vol. 35, no. 4, pp. 287–300, 2013.

[39] D. Pickem, M. Lee, and M. Egerstedt, "The GRITSBot in its natural habitat - A multi-robot testbed," in *Robotics and Automation (ICRA), 2015 IEEE International Conference on*, pp. 4062–4067, May 2015.

[40] S. Bashyal and G. Venayagamoorthy, "Human swarm interaction for radiation source search and localization," in *Swarm Intelligence Symposium, 2008. SIS 2008. IEEE*, pp. 1–8, Sept 2008.

[41] C. Vasile, A. Pavel, and C. Buiu, "Integrating human swarm interaction in a distributed robotic control system," in *Automation Science and Engineering (CASE), 2011 IEEE Conference on*, pp. 743–748, Aug 2011.

[42] S. Nunnally, P. Walker, N. Chakraborty, M. Lewis, and K. Sycara, "Using coverage for measuring the effect of haptic feedback in human robotic swarm interaction," in *Systems, Man, and Cybernetics (SMC), 2013 IEEE International Conference on*, pp. 516–521, Oct 2013.

[43] A. Naghsh, J. Gancet, A. Tanoto, and C. Roast, "Analysis and design of human-robot swarm interaction in firefighting," in *Robot and Human Interactive Communication, 2008. RO-MAN 2008. The 17th IEEE International Symposium on*, pp. 255–260, Aug 2008.

[44] J. Nagi, H. Ngo, A. Giusti, L. Gambardella, J. Schmidhuber, and G. Di Caro, "Incremental learning using partial feedback for gesture-based human-swarm interaction," in *RO-MAN, 2012 IEEE*, pp. 898–905, Sept 2012.

[45] A. Giusti, J. Nagi, L. Gambardella, S. Bonardi, and G. Di Caro, "Human-swarm interaction through distributed cooperative gesture recognition," in *Human-Robot Interaction (HRI), 2012 7th ACM/IEEE International Conference on*, pp. 401–401, March 2012.

[46] S. Nunnally, P. Walker, A. Kolling, N. Chakraborty, M. Lewis, K. Sycara, and M. Goodrich, "Human influence of robotic swarms with bandwidth and localization issues," in *Systems, Man, and Cybernetics (SMC), 2012 IEEE International Conference on*, pp. 333–338, Oct 2012.

[47] M. Diana, J.-P. de la Croix, and M. Egerstedt, "Deformable-medium affordances for interacting with multi-robot systems," in *Intelligent Robots and Systems (IROS), 2013 IEEE/RSJ International Conference on*, pp. 5252–5257, Nov 2013.

[48] J. Cortes, S. Martinez, T. Karatas, and F. Bullo, "Coverage control for mobile sensing networks," *IEEE Transactions on Robotics and Automation*, vol. 20, pp. 243–255, Apr. 2004.

[49] S. Lloyd, "Least squares quantization in PCM," *IEEE Transactions on Information Theory*, vol. 28, pp. 129–137, March 1982.

[50] Q. Du, V. Faber, and M. Gunzburger, "Centroidal Voronoi Tessellations: Applications and Algorithms," *SIAM Review*, vol. 41, pp. 637–676, Dec. 1999.

[51] J. Cortes, S. Martinez, T. Karatas, and F. Bullo, "Coverage control for mobile sensing networks: Variations on a theme," in *Mediterranean Conference on Control and Automation*, (Lisbon, Portugal), July 2002. Electronic Proceedings.

[52] J.-C. Latombe, *Robot motion planning*, vol. 124. Springer Science & Business Media, 2012.

[53] J. Lu, Y. Diaz-Mercado, M. Egerstedt, H. Zhou, and S.-N. Chow, "Shortest paths through 3-dimensional cluttered environments," in *Robotics and Automation (ICRA), 2014 IEEE International Conference on*, pp. 6579–6585, May 2014.

[54] M. Lahijanian, J. Wasniewski, S. B. Andersson, and C. Belta, "Motion planning and control from temporal logic specifications with probabilistic satisfaction guarantees," in *Robotics and Automation (ICRA), 2010 IEEE International Conference on*, pp. 3227–3232, IEEE, 2010.

[55] F. Janabi-Sharifi and D. Vinke, "Integration of the artificial potential field approach with simulated annealing for robot path planning," in *Intelligent Control, 1993., Proceedings of the 1993 IEEE International Symposium on*, pp. 536–541, IEEE, 1993.

[56] P. Song and V. Kumar, "A potential field based approach to multi-robot manipulation," in *Robotics and Automation, 2002. Proceedings. ICRA'02. IEEE International Conference on*, vol. 2, pp. 1217–1222, IEEE, 2002.

[57] C. Belta, A. Bicchi, M. Egerstedt, E. Frazzoli, E. Klavins, and G. J. Pappas, "Symbolic planning and control of robot motion," *IEEE Robotics and Automation Magazine*, vol. 14, pp. 61–70, March 2007.

[58] M. Egerstedt, X. Hu, and A. Stotsky, "Control of mobile platforms using a virtual vehicle approach," *Automatic Control, IEEE Transactions on*, vol. 46, no. 11, pp. 1777–1782, 2001.

[59] E. Artin, "Theory of braids," *Annals of Mathematics*, vol. 48, no. 1, pp. 101–126, 1947.

[60] J. Birman, *Braids, links, and mapping class groups*. Princeton University Press, 1974.

[61] V. Kurlin, "Computing braid groups of graphs with applications to robot motion planning," *Homology, Homotopy and Applications*, vol. 14, no. 1, pp. 159–180, 2012.

[62] S. Martinez, J. Cortes, and F. Bullo, "Motion Coordination with Distributed Information," *Control Systems Magazine, IEEE*, vol. 27, no. 4, pp. 75–88, 2007.

[63] A. Ghosh and S. K. Das, “Coverage and connectivity issues in wireless sensor networks: A survey,” *Pervasive and Mobile Computing*, vol. 4, no. 3, pp. 303–334, 2008.

[64] J. Cortés and F. Bullo, “Coordination and Geometric Optimization via Distributed Dynamical Systems,” *SIAM Journal on Control and Optimization*, vol. 44, pp. 1543–1574, October 2005.

[65] L. C. A. Pimenta, M. Schwager, Q. Lindsey, V. Kumar, D. Rus, R. C. Mesquita, and G. A. S. Pereira, “Simultaneous Coverage and Tracking (SCAT) of Moving Targets with Robot Networks,” in *Algorithmic Foundation of Robotics VIII* (G. S. Chirikjian, H. Choset, M. Morales, and T. Murphey, eds.), vol. 57 of *Springer Tracts in Advanced Robotics*, pp. 85–99, Springer Berlin Heidelberg, 2010.

[66] D. Haumann, V. Willert, and K. D. Listmann, “DisCoverage: From Coverage to Distributed Multi-Robot Exploration,” in *Proceedings of the 4th IFAC Workshop on Distributed Estimation and Control in Networked Systems*, vol. 4, (Koblenz, Germany), pp. 328–335, September 2013.

[67] A. Macwan, G. Nejat, and B. Benhabib, “Target-Motion Prediction for Robotic Search and Rescue in Wilderness Environments,” *IEEE Transactions on Systems, Man, and Cybernetics, Part B*, vol. 41, no. 5, pp. 1287–1298, 2011.

[68] A. Ghaffarkhah, Y. Yan, and Y. Mostofi, “Dynamic coverage of time-varying environments using a mobile robot – A communication-aware perspective,” in *GLOBECOM Workshops (GC Wkshps), 2011 IEEE*, pp. 1297–1302, 2011.

[69] S. Lee, Y. Diaz-Mercado, and M. Egerstedt, “Multirobot control using time-varying density functions,” *Robotics, IEEE Transactions on*, vol. 31, pp. 489–493, April 2015.

[70] Y. Diaz-Mercado, S. G. Lee, and M. Egerstedt, “Distributed dynamic density coverage for human-swarm interactions,” in *American Control Conference (ACC), 2015*, pp. 353–358, July 2015.

[71] S. Meguerdichian, F. Koushanfar, G. Qu, and M. Potkonjak, “Exposure in wireless Ad-Hoc sensor networks,” in *Proceedings of the 7th annual international conference on Mobile computing and networking*, MobiCom ’01, (New York, NY, USA), pp. 139–150, ACM, 2001.

[72] S. Adlakha and M. B. Srivastava, “Critical density thresholds for coverage in wireless sensor networks,” in *Wireless Communications and Networking Conference (WCNC)*, pp. 1615–1620, IEEE, 2003.

[73] M. Iri, K. Murota, and T. Ohya, “A fast Voronoi-diagram algorithm with applications to geographical optimization problems,” in *System Modelling and Optimization* (P. Thoft-Christensen, ed.), vol. 59 of *Lecture Notes in Control and Information Sciences*, pp. 273–288, Springer Berlin Heidelberg, 1984.

[74] Q. Du, M. Emelianenko, and L. Ju, "Convergence of the Lloyd Algorithm for Computing Centroidal Voronoi Tessellations," *SIAM Journal on Numerical Analysis*, vol. 44, pp. 102–119, Jan. 2006.

[75] Y. Liu, W. Wang, B. Lévy, F. Sun, D.-M. Yan, L. Lu, and C. Yang, "On centroidal voronoi tessellation – energy smoothness and fast computation," *ACM Transactions on Graphics*, vol. 28, pp. 1–17, Sept. 2009.

[76] Q. Du and M. Emelianenko, "Acceleration schemes for computing centroidal Voronoi tessellations," *Numerical Linear Algebra with Applications*, vol. 13, no. 2-3, pp. 173–192, 2006.

[77] M. Zedek, "Continuity and location of zeros of linear combinations of polynomials," *Proceedings of the American Mathematical Society*, vol. 16, no. 1, pp. 78–84, 1965.

[78] G. Stewart, *Matrix Algorithms Volume 1: Basic Decompositions*. Society for Industrial and Applied Mathematics, 1998.

[79] P. Perona and J. Malik, "Scale-space and edge detection using anisotropic diffusion," *Pattern Analysis and Machine Intelligence, IEEE Transactions on*, vol. 12, pp. 629–639, Jul 1990.

[80] S. Calinon, F. Guenter, and A. Billard, "On Learning, Representing and Generalizing a Task in a Humanoid Robot," *IEEE Transactions on Systems, Man and Cybernetics, Part B*, vol. 37, no. 2, pp. 286–298, 2007.

[81] U. Borrmann, L. Wang, A. D. Ames, and M. Egerstedt, "Control barrier certificates for safe swarm behavior," *IFAC-PapersOnLine*, vol. 48, no. 27, pp. 68–73, 2015.

[82] Y. Diaz-Mercado and M. Egerstedt, "Multi-robot mixing using braids," in *Decision and Control (CDC), 2013 IEEE 52nd Annual Conference on*, pp. 2001–2005, Dec 2013.

[83] Y. Diaz-Mercado and M. Egerstedt, "Multi-robot mixing of nonholonomic mobile robots," in *Control Applications (CCA), 2014 IEEE Conference on*, pp. 524–529, Oct 2014.

[84] Y. Diaz-Mercado, A. Jones, C. Belta, and M. Egerstedt, "Correct-by-construction control synthesis for multi-robot mixing," in *Decision and Control (CDC), 2015 IEEE 54th Annual Conference on*, pp. 221–226, Dec 2015.

[85] R. Hall, "Efficient spiral search in bounded spaces," *IEEE Transactions on Pattern Analysis and Machine Intelligence*, vol. 4, pp. 208–215, March 1982.

[86] W. Huang, "Optimal line-sweep-based decompositions for coverage algorithms," in *IEEE International Conference on Robotics and Automation*, 2001.

[87] R. Hashemi, L. Jin, G. Anderson, E. Wilson, and M. Clark, "A comparison of search-patterns for cooperative robots operating in remote environment," in *International Conference on Information Technology: Coding and Computing*, 2001.

[88] J. Kim, F. Zhang, and M. Egerstedt, "Simultaneous cooperative exploration and networking based on Voronoi diagrams," in *IFAC Workshop on Networked Systems*, October 2009.

[89] B. Tovar, L. Freda, and S. M. LaValle, "Using a robot to learn geometric information from permutations of landmarks," in *Contemporary Mathematics*, vol. 438, pp. 33–45, American Mathematical Society, 2007.

[90] T. Lozano-Perez, "Spatial planning: A configuration space approach," *Computers, IEEE Transactions on*, vol. C-32, pp. 108–120, Feb 1983.

[91] R. Ghrist, "Configuration spaces and braid groups on graphs in robotics," *AMS IP STUDIES IN ADVANCED MATHEMATICS*, vol. 24, pp. 29–40, 2001.

[92] C. Baier and J.-P. Katoen, *Principles of Model Checking (Representation and Mind Series)*. The MIT Press, 2008.

[93] O. Kupferman and M. Y. Vardi, "Model checking of safety properties," *Formal Methods in System Design*, 2001. volume 19, pages 291-314.

[94] T. Latvala, "Efficient model checking of safety properties," in *Model Checking Software* (T. Ball and S. Rajamani, eds.), vol. 2648 of *Lecture Notes in Computer Science*, pp. 624–636, Springer Berlin / Heidelberg, 2003.

[95] E. Aydin Gol, M. Lazar, and C. Belta, "Language-guided controller synthesis for discrete-time linear systems," in *Proceedings of the 15th ACM international conference on Hybrid Systems: Computation and Control*, (New York, NY, USA), pp. 95–104, 2012.

[96] A. Jones, M. Schwager, and C. Belta, "Information-guided persistent monitoring under temporal logic constraints," in *IEEE American Control Conference (ACC)*, pp. 1911–1916, July 2015.

[97] A. Criminisi, I. Reid, and A. Zisserman, "A plane measuring device," *Image and Vision Computing*, vol. 17, no. 8, pp. 625–634, 1999.

[98] A. E. Bryson, *Applied optimal control: optimization, estimation and control*. CRC Press, 1975.

[99] A. Locatelli, *Optimal control: An introduction*. Springer Science & Business Media, 2001.

[100] S.-N. Chow, T.-S. Yang, and H. Zhou, "Global Optimizations by Intermittent Diffusion," *Chaos, CNN, Memristors and Beyond*, 2013.

[101] V. J. Lumelsky, "Algorithmic and complexity issues of robot motion in an uncertain environment," *Journal of Complexity*, vol. 3, no. 2, pp. 146–182, 1987.

[102] C. Papadimitriou, "An algorithm for shortest-path motion in three dimensions," *Information Processing Letters*, vol. 20, no. 5, pp. 259–263, 1985.

[103] J. Choi, J. Sellen, and C. Yap, “Precision-sensitive Euclidean shortest path in 3-space,” in *Proceedings of the eleventh annual symposium on Computational geometry*, pp. 350–359, ACM, 1995.

[104] Y. Diaz-Mercado, S. Lee, M. Egerstedt, and S.-Y. Young, “Optimal trajectory generation for next generation flight management systems,” in *Digital Avionics Systems Conference (DASC), 2013 IEEE/AIAA 32nd*, pp. 3C5–1–3C5–10, Oct 2013.