# A Gradient Optimization Approach to Adaptive Multi-Robot Control

by

Mac Schwager

B.S., Stanford University (2000)
M.S., Massachusetts Institute of Technology (2005)

Submitted to the Department of Mechanical Engineering
in partial fulfillment of the requirements for the degree of

Doctor of Philosophy

at the

MASSACHUSETTS INSTITUTE OF TECHNOLOGY

September 2009

Author .................... [illegible] ...............
Department of Mechanical Engineering
August 26, 2009

Certified by............. [illegible] ...............
Daniela Rus
Professor of Electrical Engineering and Computer Science
Thesis Supervisor

Accepted by.............. [illegible] ...............
David Hardt
Chairman, Department Committee on Graduate Theses

# A Gradient Optimization Approach to Adaptive Multi-Robot Control

by

Mac Schwager

Submitted to the Department of Mechanical Engineering
on August 26, 2009, in partial fulfillment of the
requirements for the degree of
Doctor of Philosophy

## Abstract

This thesis proposes a unified approach for controlling a group of robots to reach a goal configuration in a decentralized fashion. As a motivating example, robots are controlled to spread out over an environment to provide sensor coverage. This example gives rise to a cost function that is shown to be of a surprisingly general nature. By changing a single free parameter, the cost function captures a variety of different multi-robot objectives which were previously seen as unrelated. Stable, distributed controllers are generated by taking the gradient of this cost function. Two fundamental classes of multi-robot behaviors are delineated based on the convexity of the underlying cost function. Convex cost functions lead to consensus (all robots move to the same position), while any other behavior requires a nonconvex cost function.

The multi-robot controllers are then augmented with a stable on-line learning mechanism to adapt to unknown features in the environment. In a sensor coverage application, this allows robots to learn where in the environment they are most needed, and to aggregate in those areas. The learning mechanism uses communication between neighboring robots to enable distributed learning over the multi-robot system in a provably convergent way.

Three multi-robot controllers are then implemented on three different robot platforms. Firstly, a controller for deploying robots in an environment to provide sensor coverage is implemented on a group of 16 mobile robots. They learn to aggregate around a light source while covering the environment. Secondly, a controller is implemented for deploying a group of three flying robots with downward facing cameras to monitor an environment on the ground. Thirdly, the multi-robot model is used as a basis for modeling the behavior of a herd of cows using a system identification approach. The controllers in this thesis are distributed, theoretically proven, and implemented on multi-robot platforms.

Thesis Supervisor: Daniela Rus
Title: Professor of Electrical Engineering and Computer Science

# Acknowledgments

This thesis owes its existence to the support, encouragement, and feedback of a number of people. I thank my advisor, Professor Daniela Rus, whose vision and guidance have been invaluable. She taught me the intangibles: how to sense a good problem, how to know when it is solved, and how to persuade people that your solution is the right one. She has been a tireless advocate for my work from its conception; and although her faith was sometimes undeserved, it was never unappreciated. I am also grateful to Professor Jean-Jacques Slotine, for his perceptive advice, mathematical insight, and practical approach to difficult mathematical analyses. The mathematics in this thesis certainly bears his mark. I also thank Professor Harry Asada for giving gentle yet objective criticism. His encouragement to unite my different research interests has made this a more coherent thesis than it otherwise might have been.

I am also deeply indebted to my co-authors and collaborators. I thank Professor James McLurkin whose technical mastery made the SwarmBot experiments successful, and whose good humor made them enjoyable. I thank Brian Julian, a magician who can turn incoherent Matlab code into extraordinary flying machines. The quadrotor experiments would have been impossible without him. My thanks also go to Dr. Dean Anderson. His keen knowledge of cow behavior played no small part in the formulation of a mathematical model of the motion of animal groups. I also thank Professor Vijay Kumar, Professor Francesco Bullo, Professor David Skelly, Professor Nikolaus Correll, and Dr. Luciano Pimenta for lively and thought provoking discussions about multi-agent systems and the mathematics that underpins their control. Furthermore, I thank my friends and colleagues in the Distributed Robotics Lab who have filled my days with colorful camaraderie.

For financial support, I am very grateful to the MURI SWARMS project (grant number W911NF-05-1-0219), the NSF (grant numbers IIS-0513755, IIS-0426838, CNS-0520305, CNS-0707601, EFRI-0735953), the MAST project, and The Boeing Company.

Finally, I thank my family and friends whose love, encouragement, and realism kept me on the right trajectory, just as a good feedback controller ought to do. Above all, I owe a special debt of gratitude to my mother, in whose memory I dedicate this thesis. She left me a most valuable inheritance: her love for research has become my love for research.

# Contents

# List of Figures

# List of Tables

# Chapter 1

# Introduction

The robots of the future will be numerous and talkative. This is an inevitable result of the decreasing cost of electronics, combined with the increasing ubiquity of networking technology. Large networks of robots have far reaching technological potential. They will change the way we grow food, manufacture products, and wage wars; they will help us to look after the environment, to collect scientific data, and will allow us to explore unfamiliar places. More generally, multi-robot systems will enable direct human influence over large scale natural phenomena. However, before we can see the benefits of multi-robot technology, we must first answer a basic question: how do we control all of these gregarious robots? What should we make them say to one another, and how should they act upon the information they get from their neighbors?

In this thesis we define, analyze, and implement a multi-robot control approach based on distributed optimization of a cost function to stably and adaptively control the movement of a group of robots towards a goal. We define a cost function that can lead to various different multi-robot behaviors by changing a single free parameter. Using the cost function, we design gradient descent controllers so that each robot moves to decrease the value of the cost function. Our controllers allow for simple, general stability and convergence results, and lead to robust, practical control strategies that can be implemented on robots with limited computational resources. Our controllers are adaptive to failures. If one robot fails, the others will automatically reconfigure to compensate for it. They also are adaptive to slowly changing environments. The robots will move to try to maintain an optimal configuration in response to changing environmental conditions. The control approach in this thesis can be used, for example, to deploy a group of hovering robots (i.e. autonomous helicopters) with downward facing cameras to collectively monitor an area on the ground. With knowledge of its own position and the positions of its neighbors, each robot moves to maximize its own field of view, while not overlapping its field of view too much with neighboring robots. The robots deploy themselves over the area, spreading outward and upward until the whole area is covered by the group. In many applications, as in this example, the use of a multi-robot system offers superior robustness, speed, and sensor resolution over a single robot solution.

The design, analysis, and implementation of multi-robot controllers are accompanied by difficulties peculiar to the multi-robot setting. Firstly, in designing multi-robot controllers, one must always consider the constraint that each robot has only partial information, yet the whole group must converge to a desired configuration. This demands a careful composition of communication, control, and sensing. Secondly, proving analytical properties of multi-robot systems is difficult because their dynamics are often nonlinear, and they are coupled through a network which changes over time. Thirdly, implementing multi-robot controllers requires maintaining multiple robot platforms simultaneously, and implementing algorithms over real ad hoc wireless networks. We overcome each of these difficulties in this thesis. We design controllers by focusing our attention to multi-robot tasks that can be quantified by a cost function, and by using gradient descent controllers to minimize the cost function. We analyze the performance of our controllers using a combination of analysis tools from optimization, Lyapunov stability theory, and graph theory to prove theorems concerning the asymptotic convergence of the robots to their final configurations, the rate of their convergence, and the optimality of their final configurations. Finally, we implement multi-robot algorithms on three different hardware platforms (shown in Figure 1-1), a group of ground robots, a group of flying quad-rotor robots, and sensor/actuator boxes mounted to the heads of cows in a herd.

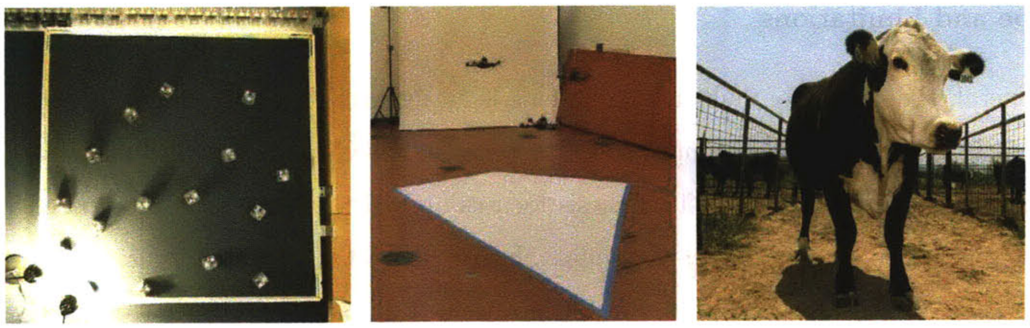

(a) Ground Robots (b) Quad-Rotors (c) Cows

Figure 1-1: The robot platforms used in the three case studies are shown.

## 1.1 Approach

As an archetypal problem, we consider the deployment of multiple robots over an environment for distributed surveillance and sensing, a task we call *coverage*. The example described above with hovering robots is a typical instance of coverage. We show that coverage is closely related to a number of other multi-robot tasks, including *consensus* (all the robots move to the same position), and *herding* (the robots aggregate in group without collisions). The link between these tasks is elucidated by showing that they all result from variations on the same optimization problem, and controllers for each of these tasks are shown to be obtained by taking the negative gradient of the cost function. We then consider the situation in which robots must learn a function in the environment to carry out their control task. For example, in a sensor coverage task, the robots learn the areas that require the most dense sensor coverage and move to aggregate in those areas. Learning is incorporated in a distributed way with provable stability and performance guarantees. Building upon this foundation, we pursue three detailed multi-robot case studies. The controllers for these case studies are implemented on three different kinds of multi-robot platforms: a swarm of ground robots (Figure 1-1(a)), a group of flying quad-rotor robots (Figure 1-1(b)), and a herd of cows outfitted with sensing and control boxes (Figure 1-1(c)).

## Scope and Limitations

In general, the control of multi-robot systems is an intractably large problem space. One must specify the kind of multi-robot systems and the class of multi-robot tasks to make meaningful headway. This thesis focuses on multi-robot systems composed of identical robots with simple dynamics. Furthermore the tasks we consider are those that can be formulated as the optimization of a cost function that depends upon the positions of the robots. In other words, our problem space is to drive the robots to a final goal configuration (or to a set of possible goal configurations) and remain fixed unless a robot fails or the environment changes, in which case the robots adjust to find a new goal configuration. This precludes, for example, algorithmic tasks involving temporal logic specifications, such as, "move in formation to area A, explore area A for T hours, then return in formation, while avoiding areas B and C." It also precludes tasks in which not only the final positions of the robots are important, but also their trajectories, for example, moving the robots to cover an environment using the shortest possible paths. These kinds of tasks might also be phrased as optimizations, though not with cost functions that only depend on the positions of the robots.

Notice that our problem space naturally divides into regimes based upon aggregation. The goal configuration of the robots can either be spread apart over the environment, in which case we say they are doing coverage. Otherwise they are grouped together, in which case we say they are doing consensus (if they are all occupying the same position), or herding (if they are not all occupying the same position). This exhausts the possible species of goal configurations in our problem space. Despite this simple characterization, we will see that rather complex controllers and tasks are possible in this problem space.

## Gradient Based Control

The main feature that defines our class of multi-robot problems is that they can be cast in terms of the optimization of a cost function that depends on the positions of the robots. We derive distributed controllers by taking the negative gradient of the cost function so that the robots are always moving to decrease the cost of their positions. Thus the closed-loop system is a gradient system, which is a dynamical system whose dynamics are given by the negative gradient of a cost function.

Restricting our scope to gradient systems still allows for considerable complexity in behavior, but has two pronounced benefits. Firstly, it ties the closed loop-system to an optimization problem. Optimization problems have a powerful set of mathematical tools that can be used to prove fundamental properties of the closed-loop system. Secondly, gradient systems carry with them particularly strong and simple stability guarantees which can be deduced from the properties of the underlying cost function alone. This leads to goal directed behavior, because gradient controllers move the robots to local minima of the cost function which represent goal states.

### Incorporating Learning

This thesis puts a heavy emphasis on the role of learning and adaptation on multi-robot control, which distinguishes it from other works in this area. In any group of biological agents, learning and adaptation plays a key role in the group's behavior. Agents learn to specialize their roles within the group, and they adapt their behavior using knowledge they acquire about the environment. Taking inspiration from biological agents, we integrate learning within multi-robot controllers with rigorous stability guarantees, so as to enable more complex behaviors. Control and learning are both represented under the umbrella of optimization and are analyzed with the tools of Lyapunov stability theory. We use the term learning to specifically mean tuning the parameters of a parametric model, as in system identification or adaptive control. This is a more narrow notion of learning than that used in the statistical learning and machine learning communities.

In our applications learning is important for the robots to integrate sensor information into their behavior and thereby accommodate uncertainty about the desired task. For example, in a coverage control application in which robots spread out over an environment to do sensing, it is useful for the robots to concentrate in areas where their sensors are most needed. Our controllers allow the robots to learn on-line where they are most needed and to automatically concentrate in those areas. In this thesis we also use learning in the form of parameter tuning to build models of cow herding. GPS data from cows are used to tune the parameters of a model to fit the behavior of the cows, including their affinity for one another and their preferred paths over the environment.

## Basic Assumptions

In this thesis, the robots are assumed to have simple integrator dynamics, unless otherwise explicitly stated, so that the control input is the robot's velocity. This kind of kinematic model is common in the multi-robot control literature, and we have found in repeated experimental implementations that it is a good approximation provided that the robots have a fast inner control loop to regulate the velocity. Indeed, from a design point of view it is desirable to separate the control of the high-level multi-robot behavior from the low-level dynamical behavior of the robots. Commercially available robot platforms typically have fast low-level control loops to track a commanded velocity or to drive to given way points.

Also, in this thesis the robots interact with one another over a network and are assumed to be able to exchange state information, such as position, only with their immediate neighbors in the network. We do not assume the presence of a point-to-point routing protocol to move messages between two specific robots in the network. In an implementation, this means that robots simply broadcast their states at each time step, and any other robot within communication range uses the broadcasted state as required by its controller. Also, we do not explicitly consider network delays, bandwidth constraints, packet losses, and other real but difficult-to-model limitations on network performance. However, in multiple hardware implementations our controllers are shown to perform well despite the limitations of real networks.

## 1.2 Applications

The multi-robot control strategies in this thesis have two key qualities that make them superior to single robots systems for many applications. Firstly, they are inherently tolerant to the failure of any individual in the group, whereas in a single robot system, if the robot fails the mission fails. Secondly, multi-robot systems can carry out tasks over large geographical areas more quickly and with greater resolution than single robots. The robots can move in parallel to cover an environment, while a single robot must traverse the environment in a serial fashion. These qualities make our multi-robot controllers useful in a broad range of applications, as described below.

### Large Scale Scientific Data Collection

Scientists who study large scale environmental systems such as ecologists, geologists, archaeologists, oceanographers, and meteorologists spend an inordinate amount of time and effort to collect data over large geographical areas. This laborious activity could be automated by the use of a multi-robot system. To that end, the controllers considered in this thesis can be used to deploy underwater robots over a coral reef to monitor coral health, or to deploy flying robots to take measurements over geological formations, forests, or archaeological sites. Such controllers could also be used by collaborating rovers to explore the surface of other planets or moons. The ability to collect scientific data over large geographical areas will accelerate scientific progress and facilitate the study of physical and biological processes that take place beyond the scale of current measurement apparatus.

### Distributed Surveillance and Servicing

The control strategies in this thesis are also useful for distributed surveillance or servicing. In many scenarios we want to automatically monitor an area for security, to identify irregular activity, or to provide some service to users in the area. For example, in a military context a group of robots (flying robots, ground robots, or a mixed group) could use our controllers to position themselves over a battle field. They could be used to monitor the movement and concentration of enemy combatants, or to provide mapping information and navigation information about the battlefield. The robots could also be used to provide network support to troops and vehicles on the battlefield, acting as self-positioning network routers. This is useful in civilian applications as well. For example, they could be used in disaster relief to provide mapping information or to act as an ad hoc communication network for rescue workers. Robots can also be deployed using our controllers over an urban area to monitor human activity, or to provide wireless connectivity for computers and mobile phones. They could also be used to position flying robots with cameras over, for example, sporting events or parades to provide media coverage. The same controllers can be used also for multi-robot servicing tasks, for example, to position oil clean-up robots over an oil spill so that they clean up the spill in minimum time, or to position de-mining robots to service a mine field in minimum time.

#### Formations for Traveling in Groups

The controllers in this thesis are also relevant to formation flying applications. Maintaining a formation is useful for groups of aerial and ground robots as a means of traveling in a team. This is particularly useful in a semi-automated scenario in which a human operator controls a group of vehicles. Directly controlling each vehicle in the group is too complicated a task for a single operator, but if each vehicle is equipped with a formation controller, the pilot can control the group abstracted as a single entity.

#### Modeling Biological Group Behavior

The models and tools described in this work can be seen through engineer's eyes as controllers for groups of robots, but they can also be seen through scientist's eyes as mathematical models of the dynamics of groups of biological agents. They can be used to predict and describe foraging, predation, herding, and other group behaviors in nature. For example, one of the detailed case studies in this thesis deals with modeling the herding behavior of cows. The modeling technique can also be used to generate dynamical models of groups of other animals, crowds of people, or even traffic flow. Dynamical models of biological group behavior can also be used to drive groups of robots to mimic the behavior of natural groups. This may be useful in reproducing collaborative behaviors exhibited in natural systems, or in producing decoy robots to participate with natural or engineered groups, and even to influence the behavior of natural groups [39]. Therefore, in this work the word *robot* should be seen as including natural autonomous agents such as people, cells, or cows.

## 1.3 State of the Art

The study of controlling multiple robots with the ability to communicate over a network has become a particularly important part of the controls and robotics research communities. Most research has focused on prototypical tasks. These prototypical tasks can be stated as follows.[1]

1. *Coverage*– the deployment of a group of agents over an environment, or, more generally, the dispersion of agents' states over a state space.

2. *Consensus*– the convergence of a group of agents to a common point, or, more generally, convergence of the states of a group of agents to a common final vector or manifold. Consensus can be seen as in opposition to coverage, since in the former agents come together and in the later they spread apart. This phenomenon is often called by other names including rendezvous, agreement, and flocking, and is closely related to gossip algorithms in distributed computation, and oscillator synchronization.

3. *Herding*– the aggregation of a group of agents in such a way that they do not get too far from, nor too close to, one another. This can be seen as a composition of coverage and consensus.

[1]These categories, including their names, are my convention and are necessarily somewhat arbitrary.

In the literature, these tasks have emerged separately from one another and are usually treated as entirely different problems. This thesis shows that they all derive from the same cost function and are in fact all different regimes in a continuum of behaviors. The main difference between them is in the convexity of the underlying cost function. We discuss the relevant previous work from these three areas in detail in Chapter 2, Section 2.2.

The state of the art currently is to analyze a variation on one of these three tasks, or to implement an instantiation of one of them on a multi-robot platform. There are few works which combine these tasks to design or implement more complex robot behaviors in a multi-robot setting, as is done in this thesis. We combine coverage with learning, which requires consensus, thereby composing two of these behaviors in a provably stable way to create a more complex behavior. We also combine herding with learning to produce models of complex cow herd motion.

## 1.4 Contributions

The contributions of this thesis are as follows.

1. *Unified Optimization Formulation–* An optimization problem is presented which is general enough to represent all of the three categories of multi-robot problems described above. This illuminates the common nature behind these multi-robot phenomena and allows for their treatment under a unified gradient optimization setting.

2. *Convexity and Consensus–* We prove that if the cost function representing a multi-robot problem is convex, then one of its global optima is consensus (i.e. all robots occupying the same state). Conversely, if we wish to solve a problem for which consensus is not optimal, for example coverage or herding, we know that the underlying cost function must be nonconvex. This has important ramifications for reaching global optima using gradient based controllers.

3. *On-Line Learning–* Learning in the form of on-line parameter adaptation is

incorporated into our multi-robot controller. We use consensus in a novel way to enable learning in the distributed multi-robot setting, so that each robot in the group learns asymptotically as well as if it had global information. Stability and convergence properties are proved using a Lyapunov approach. Learning allows for provably stable multi-robot behavior that can adapt to uncertain and slowly changing environments.

4. *Implementation of Learning Coverage Controller on Mobile Robots*– A coverage controller with on-line learning is implemented on a group of SwarmBots (Figure 1-1(a)). A group of 16 SwarmBots learns the distribution of sensor information in the environment while spreading out to cover the environment.

5. *Implementation of Camera Coverage Controller on Flying Quad-Rotor Robots*– Using the unified optimization formulation, a coverage controller is designed for controlling a group of flying robots with downward facing cameras. The controller is implemented and tested on a group of three flying quad-rotor robots (Figure 1-1(b)).

6. *Implementation of Model Learning for a Herd of Cows*– The multi-robot model is used for modeling the herding behavior of cows. System identification techniques are used to tune the parameters of the model using GPS data collected from 3–10 actual cows (Figure 1-1(c)).

## 1.5 Organization

This thesis begins by reviewing an existing multi-robot coverage controller that uses the geometric notion of a Voronoi tessellation. After stating and proving the basic qualities of this existing controller, we propose an optimization problem which is shown to incorporate the Voronoi controller as a special limiting case. This optimization problem is shown to be of a surprisingly general nature, specializing to give controller designs for a number of different multi-robot problems. After posing the basic optimization problem, we consider an important extension to the case where

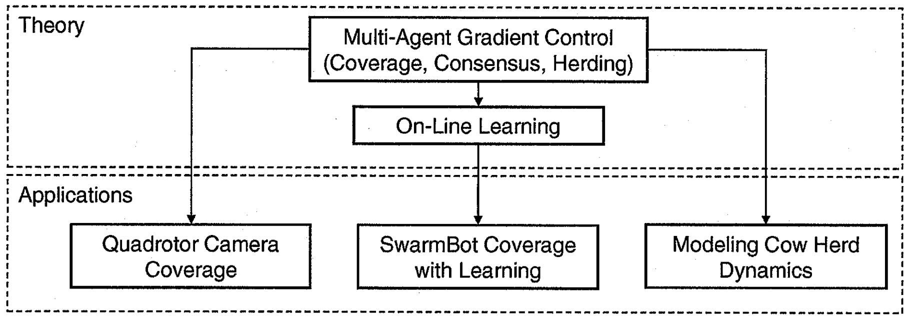

Theory
Multi-Agent Gradient Control (Coverage, Consensus, Herding)
On-Line Learning
Applications
Quadrotor Camera Coverage
SwarmBot Coverage with Learning
Modeling Cow Herd Dynamics

Figure 1-2: This figure shows the relation of the main optimization problem to each of our three case studies.

some information in the environment is lacking and must be learned through sensor measurements. We proceed by specializing the general optimization problem to three specific applications. Each application involves formulating the correct optimization problem, deriving a gradient controller, verifying stability properties, building a numerical simulation, and finally, implementing the controller on a multi-robot platform. The relation among the main parts of the thesis is shown in Figure 1-2. Thus the thesis begins from a theoretical point of view and proceeds towards more practical matters. After a short introduction, each chapter has an itemized summary of contributions and a previous work section, and concludes with a synopsis of the important points in the chapter.

Chapter 2 gives the background necessary for the rest of the thesis. It starts by reviewing the previous work relevant to the thesis and describing the contributions of the thesis in the context of the existing literature. It then defines the mathematical notation that is used throughout the thesis, and states two theorems relating to the convergence and stability of gradient systems that will be used repeatedly in later chapters. We then derive a well-known Voronoi based controller and prove its convergence properties. The material in this chapter is not a novel research contribution.

Chapter 3 provides the main theoretical foundation of the thesis by posing a general optimization problem. A cost function is formulated using the motivating example of coverage control. Controllers are obtained by taking the gradient of the cost function, resulting in nonlinear controllers which involve computing an integral of some quantity over the environment. It is shown that the coverage cost function can be seen in a more general light as being relevant to consensus, herding, and other multi-robot control tasks. The chapter draws attention to the way in which sensor measurements are combined from different robots, and in so doing poses the idea of a *mixing function*. It is shown that the choice of mixing function roughly dictates how tightly the robots aggregate. A parameterized class of mixing functions is proposed which is shown to unify and extend several different multi-robot control strategies, including ones with geometric interpretations, probabilistic interpretations, and potential field interpretations. This chapter also shows that the Voronoi based controller can be approximated arbitrarily well by a smooth controller that does not require the computation of a Voronoi tessellation. The chapter concludes by formally delineating two classes of multi-robot problems: consensus problems and non-consensus problems. Coverage control is shown to be a non-consensus problem, which therefore requires the optimization of a nonconvex cost function.

Chapter 4 completes the theoretical part of the thesis by considering how to augment the multi-robot controllers from Chapter 3 to include learning. Learning is first incorporated into the standard Voronoi based controller from Chapter 2. We then apply the learning architecture to augment the more general class of gradient controllers seen in Chapter 3. We frame the learning algorithm as a matter of adapting parameters of a parametric model on-line as data is collected. Additionally, the algorithm leverages communication among neighboring robots to facilitate distributed learning. A consensus method propagates the learning parameters from any one robot throughout the network so that every robot learns asymptotically as well as if it had global information. The controller and learning algorithm are then analyzed in the same mathematical context using Lyapunov stability theory. We address questions of convergence of the learning algorithm and convergence of the robots positions to their goal configuration. Rates of convergence and conditions for asymptotically perfect learning performance are also investigated and a number of different stable learning algorithms are proposed.

Chapter 5 uses the theory from both Chapters 3 and 4 to implement a Voronoi based controller that incorporates learning. The controller drives a group of robots to spread over an environment while aggregating in areas of high sensory interest. The controller learns the areas of interest from sensor measurements while simultaneously driving the robots to minimize a cost function representing the surveillance cost of the group. The algorithm is implemented on a team of 16 mobile robots. In experiments, the robots repeatably and effectively learned the distribution of sensory interest while covering the environment.

Chapter 6 uses the theory presented in Chapter 3 to design a controller to deploy hovering robots with downward facing cameras to collectively monitor an environment. Information per pixel is proposed as a general optimization criterion for multi-camera placement problems. This metric is used to derive a specific cost function for multiple downward facing cameras mounted on hovering robot platforms. A controller is derived by taking the negative gradient of this cost function, and convergence is proved with the theorems from Chapter 2. The controller is implemented on three flying quad-rotor robots. Results of the robot experiments are presented and compared with simulation results.

In Chapter 7 the multi-robot model is adapted to model the behavior of cows in a herd. Least Squares system identification is applied to tune the parameters of the model to fit the behavior of an actual herd of cows. The herd model describes the interaction between agents using a parameterized nonlinear force law and captures the animals' preference for certain paths over the environment as a parameterized vector-field. To demonstrate the method, GPS data collected from three cows in one instance, and ten cows in another are used to tune the model parameters. Conclusions, lessons learned, and future work are given in Chapter 8.

# Chapter 2

# Background

## 2.1 Introduction

This chapter accomplishes four goals: 1) it situates our work in the research literature of robotics and controls, 2) it formulates the main multi-robot system model that is referred to repeatedly in the thesis, 3) it proves two theorems about the convergence and stability of gradient systems that are used repeatedly in the thesis, and 4) it develops in detail a previously existing multi-robot coverage controller that will serve as a baseline for comparison and elaboration throughout the thesis. The material in this chapter does not constitute novel research contributions.

## 2.2 Previous Work

As described in Chapter 1, most research in the control of multi-robot systems has focused on the following prototypical problems.

1. *Coverage*– the deployment of a group of agents over an environment, or, more generally, the dispersion of agents' states over a state space.

2. *Consensus*– the convergence of a group of agents to a common point, or, more generally, convergence of the states of a group of agents to a common final vector or manifold. Consensus can be seen as in opposition to coverage, since in

the former agents come together and in the later they spread apart. This phenomenon is often called by other names including rendezvous, agreement, and flocking, and is closely related to gossip algorithms in distributed computation, and oscillator synchronization.

3. *Herding*– the aggregation of a group of agents in such a way that they do not get too far from, or too close to one another. This can be seen as a composition of coverage and consensus.

In this thesis we show that these three problem arise from the same basic optimization problem with different parameters or weightings on different terms. These three areas have emerged in the research literature as separate problems, however. We will look at the relevant literature foe each of tehse area and situate this thesis with respect to it.

The kind of coverage control considered in this thesis owes its beginning to the optimization formulation introduced in [26] which uses the geometrical notion of a Voronoi partition to divide up the environment among the robots. This work itself adapted concepts from locational optimization [30, 114], which is the study of optimally placing industrial facilities. This, in turn, derives from a classical problem of finding geometric median points, which has been attributed to Fermat. The coverage controller in [26] drives the robots to reach a centroidal Voronoi configuration [70]. There are a number of other notions of coverage including the notion of painting a sensor footprint over an environment as in [17, 20, 56], or of introducing sensors sequentially in a centralized way to optimize a probabilistic quantity, as in [52,53]. This thesis adopts the locational optimization approach for its interesting possibilities for analysis, its connection to distributed optimization, and the resulting potential for integrating it with graph theory (to model communication networks) and learning. The basic idea introduced in [26] has been extended and elaborated upon considerably. For example, [87] used a deterministic annealing technique to improve final robot configurations, [25] extended the controller to robots with finite sensor footprints and other realistic complications, [77] extended the controller to heterogeneous groups of robots and nonconvex environments, [75] generalized the Voronoi partition by introducing the Power Diagram to achieve equitable mass partitions, and [78] treated the problem of coverage in time varying environments. Probablistic scenarios that use the same kind of controller have also been considered in, for example [57], [4], and [76]. The article [63] and the book [14] provide an excellent consolidation of much of this research.

One common thread in all of these coverage control works is that the distribution of sensory information in the environment is required to be known *a priori* by all robots. This *a priori* requirement was first relaxed in [95] by introducing a controller with a simple memoryless approximation from sensor measurements. The controller was demonstrated in hardware experiments, though a stability proof was not found. One of the contributions in this thesis is to incorporate learning to enable optimal coverage of an unfamiliar environment. We formulate the problem of learning about an unknown environment as an adaptive control problem. Adaptive control is usually applied to the control of dynamical systems that are unknown, or only partially known. Some of the standard text books on adaptive control are [67, 89, 103] and a well-known paper that deals with an adaptive control architecture that accommodates more general function approximation techniques is [88]. This thesis leverages the proof techniques used in this body of work to obtain Lyapunov stability results for coverage controllers that incorporate learning, not of dynamics, but of some aspect of the environment itself.

The second multi-robot control problem relevant to the work in this thesis is consensus. Consensus phenomena have been studied in many fields, and appear ubiquitously in biological systems of all scales. However, they have only recently yielded to rigorous mathematical treatment; first in the distributed and parallel computing community [9, 10, 107, 108] in discrete time, and more recently in the controls community in continuous time. One of the foundational works on consensus in the controls community is [46], which analyzes the well-known flocking model presented in [109], and presents general consensus conditions for multi-agent systems with switching sets of neighbors. Another foundational work of this genre is [71] which deals with directed communication networks, switching sets of neighbors, and communication time-delays. Other works in this area include [111,112] which uses contraction theory to analyze synchronization of oscillators (which can be seen as consensus on the phase of multiple periodic systems), [13] which looked at asynchronous communication and unbounded communication delays, [27] investigated a model in which the influence of one agent over another decays as a function of the distance between them, drawing parallels with the emergence of language in isolated human populations, and [66] considered agents that communicate only limited state information with agents within their line-of-sight to model flocking behavior in birds.

In this thesis, consensus plays a key role in distributed learning. Unknown factors in the environment, such as where the most informative data can be found, is learned on line in a distributed way by propagating sensor measurements gathered by each robot around the network. The robots essentially reach a consensus on the function they are trying to learn, each robot getting the benefit of the senor measurements of all the other robots in the network. This is similar to distributed filtering techniques that have recently been introduced, for example in [62,117], though in contrast to those works, the controllers in this thesis are concerned with maintaining provable stability of the combined learning and control system.

Herding, the third multi-robot control problem relevant to this thesis, has been studied under many variations, and is usually seen as a modification to the consensus problem. Herding is often carried out using potential field formulation in which agents attract each other if they are too far and repel each other if they are too close, as in [35]. Situations in which agents only effect each other when they are within a certain distance are treated in [104] and extensions which attempt to maintain connectivity of the underlying communication graph in these situations are considered in [29,116]. These systems require nonsmooth analysis techniques, for example those described in [24,86]. Results pertaining to the distributed computation of graph connectivity are also an important part of the latest work on herding and consensus, as in [115]. Graph theory more generally has long history in pure mathematics, and a well known text on graph theory is [37]. Our application in Chapter 7 uses system identification to tune the parameters of a herding model. Both systems identification and herd models have a rich literature separately, though there appears to little prior work in combining the two. One exception is the work of Correll et al. [22,23] which uses system identification to learn the parameters of a rate equation describing the behavior of a multi-robot system used for turbine blade inspection. Many of the results in this thesis are based on results that have been published in [91, 93, 94, 96–100].

## 2.3 Mathematical Preliminaries

This section gives the mathematical notation and definitions used throughout the thesis and states two basic theorems that will come in handy in later chapters. We will use $\mathbb{R}^d$ to denote the $d$-dimensional Euclidean space, and $\mathbb{R}^d_{\geq 0}$ and $\mathbb{R}_{>0}$ to be the non-negative and strictly positive quadrants of $d$-dimensional Euclidean space, respectively. We mean the symbols $>, <, \geq$ and $\leq$ to apply element-wise for vectors. An open interval on the real line with end points $x$ and $y$ is denoted $(x, y)$ and the closed interval $[x, y]$, with $[x, y)$ and $(x, y]$ being the intervals closed on the left, open and the right, and open on the left, closed on the right, respectively. The symbol $\partial \cdot$ will be used to refer to the boundary of a set and $\partial \cdot / \partial \cdot$ to refer to a partial derivative. Real vectors will not be differentiated from scalars with a bold font, but wherever it is not obvious we will explicitly state $v \in \mathbb{R}^d$ or $s \in \mathbb{R}$ for a vector $v$ and scalar $s$. The vector of ones is denoted $\mathbf{1}$ and the $n \times n$ identity matrix is denoted $I_n$. The derivative of a function $v$ with respect to time will be denoted either with the conventional $dv/dt$, or with the shorter $\dot{v}$ notation where convenient. The $\ell^2$ norm is denoted $\|\cdot\|$ and $\|\cdot\|_p$ gives the $\ell^p$ norm. A function $f : \Omega \mapsto \mathbb{R}$ is called Lipschitz on $\Omega$ if there exists a constant $L$ such that $|f(x_2) - f(x_1)| \leq \beta \|x_2 - x_1\|$, for all points $x_1, x_2 \in \Omega$. The a function is called locally Lipschitz if, for any point $x \in \Omega$, there exists a ball $\mathcal{B}(x)$ centered at $x$ such that the function is Lipschitz on $\mathcal{B}(x)$ with a constant $\beta(x)$ that depends upon the point $x$. A sufficient condition for a differential equation $\dot{x} = f(x)$ to have a unique solution for a given initial condition is that the function $f$ is locally Lipschitz [14, 43, 49].

### 2.3.1 Convex Optimization

We now state some basic definitions from convex optimization which can be found in any standard text on the topic, for example [8]. A set $\Omega \subset \mathbb{R}^n$ is called convex if, for any two points in $\Omega$, all points along the line segment joining them are also in $\Omega$. Formally,

$$\alpha x + (1-\alpha)y \in \Omega \quad \forall x, y \in \Omega \quad \text{and} \quad \forall \alpha \in [0,1]. \tag{2.1}$$

An important consequence of the convexity of $\Omega$ is that any convex combination of points in $\Omega$ is also in $\Omega$. A convex combination of $m$ points $x_i \in \Omega$ is one of the form

$$x = \sum_{i=1}^{m} \alpha_i x_i \quad \text{where} \quad \sum_{i=1}^{m} \alpha_i = 1 \quad \text{and} \quad \alpha_i \geq 0 \quad \forall i. \tag{2.2}$$

A function $f : \Omega \mapsto \mathbb{R}$ is called convex if

$$f(\alpha x + (1-\alpha)y) \leq \alpha f(x) + (1-\alpha) f(y) \quad \forall x, y \in \Omega \quad \text{and} \quad \forall \alpha \in [0,1]. \tag{2.3}$$

This is equivalent to saying that the set of all points lying on or above the function $f$ is a convex set (this set is known as the epigraph of $f$). A function is called strictly convex if the '$\leq$' can be replaced with a '$<$' in the above relation. Also, with regards to optimization, we will use the word minimum to mean minimum or infimum if no minimum exists.

We now state a theorem concerning the convexity of the set of minima of a convex function. The theorem follows from Weierstrass' Theorem and some well-known properties of convex functions.

**Theorem 2.1 (Minima of Convex Functions)** *For a continuous, convex function $f : \Omega \mapsto \mathbb{R}$, where the domain $\Omega \subset \mathbb{R}^n$ is convex, if any of the following are true:*

*1. $\Omega$ is bounded*

2. *There exists a scalar $\gamma$ such that the level set $\{x \in \Omega \mid f(x) \leq \gamma\}$ is nonempty and bounded*

3. *$f$ is such that $lim_{||x|| \to \infty} f(x) = \infty$*

*then the set of global minima of $f$ is non-empty and convex.*

**Proof 2.1** *Please refer to [8].*

### 2.3.2 Graph Laplacians

An undirected graph[1] $\mathcal{G} = (\mathcal{I}, \mathcal{E})$ is defined by a set of indexed vertices $\mathcal{I} = \{1, \ldots, n\}$ and a set of edges $\mathcal{E} = \{e_1, \ldots, e_{n_\mathcal{E}}\}$, where $e_i = \{j, k\}$ and $j, k \in \mathcal{I}$. In the context of our application, a graph is induced in which each agent is identified with a vertex, and an edge exists between any two agents that are in communication with one another. Consider a function $w : \mathcal{I} \times \mathcal{I} \mapsto \mathbb{R}_{\geq 0}$ such that $w_{ij} = 0 \; \forall \{i, j\} \notin \mathcal{E}$ and $w_{ij} > 0$ $\forall \{v_i, v_j\} \in \mathcal{E}$. We call $w_{ij}$ a weighting over the graph $\mathcal{G}$. Next consider the weighted graph Laplacian matrix $L$, whose terms are given by

$$L(i,j) = \begin{cases} -w_{ij} & \text{for } i \neq j \\ \sum_{j=1}^{n} w_{ij} & \text{for } i = j. \end{cases} \tag{2.4}$$

A graph is connected if, for any two vertices, there exists a set of edges that defines a path between them. The following result is well known in graph theory and will be useful in proving properties of our distributed, on-line learning algorithm in Chapter 4.

**Theorem 2.2** *(Graph Laplacians) For a connected, undirected graph, the weighted graph Laplacian is symmetric, positive semi-definite, $L \geq 0$, and $L$ has exactly one zero eigenvalue, with the associated eigenvector $\mathbf{1} = [1, \ldots, 1]^T$. In particular, $L\mathbf{1} = \mathbf{1}^T L = 0$, and $x^T L x > 0$, $\forall x \neq c\mathbf{1}, c \in \mathbb{R}$.*

**Proof 2.2** *Please refer to [37].*

[1]We will only be dealing with undirected graphs in this thesis. When we refer to a graph it is assumed to be undirected.

## 2.4 Multi-Robot System Model

In this section we introduce the basic multi-robot system model that will be used throughout the thesis and provide a condensed review of gradient systems and their convergence and stability properties.

Let there be $n$ robots, where robot $i$ has a position $p_i \in \mathcal{P} \subset \mathbb{R}^{d_p}$. The state space for a single robot is $\mathcal{P}$ and the $d_p$ is the dimension of the state space. Consider a vector $P \in \mathcal{P}^n \subset \mathbb{R}^{nd_p}$ which is the vector obtained by stacking all of the robot positions together, $P = [p_1^T \cdots p_n^T]^T$. We will refer to the vector $P$ as the *configuration* of the robots since a single point $P$ in the high dimensional space $\mathcal{P}^n$ represents the positions of all the $n$ robots in the low dimensional space $\mathcal{P}$. Now consider a cost function $\mathcal{H} : \mathcal{P}^n \mapsto \mathbb{R}$ that represents the suitability of a given configuration of robots for a given task. We will alternately write $\mathcal{H}(P)$ or $\mathcal{H}(p_1, \ldots, p_n)$ in referring to a specific value of the function for a configuration $P$. Let the cost function be differentiable everywhere on $\mathcal{P}^n$ so that its partial derivative with respect the each robot's position is well-defined, $\partial\mathcal{H}/\partial p_i$. Also, let $\partial\mathcal{H}/\partial p_i$ be locally Lipschitz on $\mathbb{P}^n$ to guarantee the existence of solutions of our system.

Let the robots have simple integrator dynamics

$$\dot{p}_i = u_i, \tag{2.5}$$

where $u_i$ is the control input to robot $i$. This assumption will be used throughout the thesis unless otherwise explicitly stated. We have found in experiments that a fast low level control loop is sufficient in many cases to approximate integrator dynamics. The standard form for the controllers used in this thesis is given by

$$u_i = -k\frac{\partial\mathcal{H}(P)}{\partial p_i}, \tag{2.6}$$

where $k \in \mathbb{R}_{>0}$ is a positive control gain. The closed loop multi-robot system is then

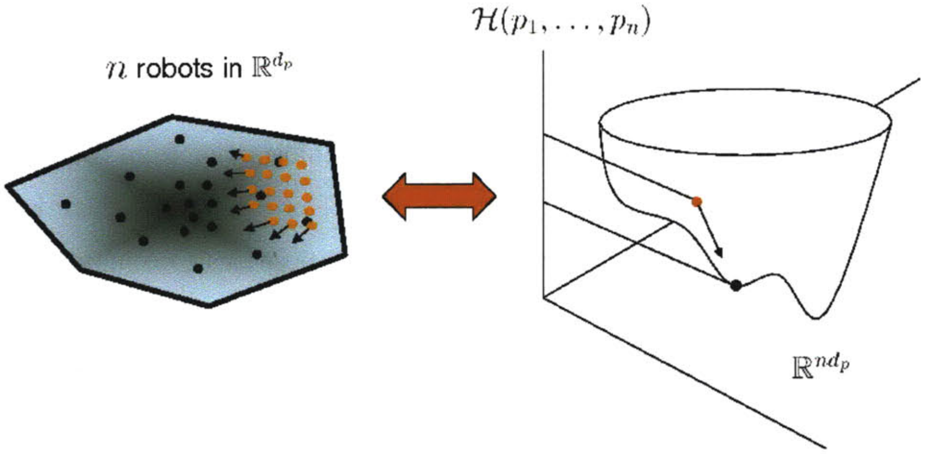

$n$ robots in $\mathbb{R}^{d_p}$
$\mathcal{H}(p_1, \ldots, p_n)$
$\mathbb{R}^{nd_p}$

Figure 2-1: This figure shows the relationship between the motions of individual robots in their environment and the trajectory of a single point in a high dimensional configuration space.

represented by the $n$ coupled differential equations

$$\boxed{\dot{p}_i = -k \frac{\partial \mathcal{H}(P)}{\partial p_i}.} \tag{2.7}$$

Equation (2.7) is the subject of this thesis. We will refer to it repeatedly as the "multi-robot system," the "closed-loop dynamics," or the the "gradient system," and we display it in a box for emphasis. The relationship between the motion of the group of robots in their environment and the motion of a single point in the high dimensional configuration space is shown in Figure 2-1.

### 2.4.1 The Induced Graph

The multi-robot system (2.7) induces a graph in the following way. The gradient component $\partial \mathcal{H}/\partial p_i$ may only depend upon some of the other robots in the network. Let $\mathcal{N}_i$ be the set of indices of the other robots upon which the gradient component

$\partial \mathcal{H}/\partial p_i$ depends, and let $P_{\mathcal{N}_i}$ be the vector of the positions of those robots. We have

$$\dot{p}_i = -k\frac{\partial \mathcal{H}(P_{\mathcal{N}_i})}{\partial p_i}.$$

For robot $i$ to compute it's own controller, it only needs information from the robots $\mathcal{N}_i$. Therefore a graph is induced in which each robot is a vertex, and there is an edge between any two robots that require one another's positions to compute their gradient component. The controller is then said to be distributed over this graph. For example, in the next section we will describe a controller that is distributed over the Delaunay graph.

Of course it be the case that a particular hardware platform and particular environmental conditions render a controller infeasible given the communication graph. In this case an approximation must be used. We recommend two possible strategies for approximating the controller on a given communication graph: 1) each robot computes its controller using only the robots with which it is in communication, and 2) each robot maintains an estimate of the positions of the robots in $\mathcal{N}_i$ and uses these estimates to compute its controller. In this thesis we assume, unless otherwise stated, that the communication graph is sufficient for each robot to compute its controller, since our emphasis is more on dynamical properties of the controllers then on network properties.

### 2.4.2 Properties of Gradient Systems

We can equivalently express the $n$ coupled equations in (2.7) as a single equation using the configuration vector $P$ as

$$\dot{P} = -k\frac{d\mathcal{H}(P)}{dP}. \tag{2.8}$$

From (2.8) it is clear that our multi-robot system is a gradient system, meaning the right hand side of the governing differential equation is proportional to the negative gradient of the scalar valued cost function $\mathcal{H}$. Gradient systems have particularly simple and powerful convergence and stability properties, the most important of which will be given here.

**Theorem 2.3 (Global Convergence of Gradient Systems)** *Let $\Omega = \{P^* \mid d\mathcal{H}/dP \mid_{P^*} = 0\}$ be the set of all critical points of $\mathcal{H}$. If $\mathcal{H}$ is radially unbounded, or if all trajectories of the system are bounded, then all trajectories of the system $\dot{P} = -kd\mathcal{H}/dP$ converge asymptotically to $\Omega$.*

**Proof 2.3** *The theorem follows as a corollary to LaSalle's Invariance Principle [49, 55, 103]. Let $\mathcal{H}$ be the Lyapunov function candidate. Then $\dot{\mathcal{H}} = -k\|d\mathcal{H}/dP\|^2 \leq 0$, and if $\mathcal{H}$ is radially unbounded, the trajectories of the system are bounded, therefore by LaSalle's Invariance Principle all trajectories converge to the largest invariant set contained in $\Omega$. By the definition of the dynamics, $\Omega$ itself is an invariant set, therefore all trajectories converge to $\Omega$.*

**Remark 2.1** *This result does not necessarily imply that the trajectories converge to a single point in $\Omega$. However, this is true if $\Omega$ is a set of isolated points. Furthermore, if the system ever reaches a point $P^* \in \Omega$, it will stay at that point for all time, whether or not it is an isolated critical point, since $\dot{P} = 0$ $\forall t \geq 0$ at such a point.*

The following useful result pertains to the local stability of critical points of $\mathcal{H}$.

**Theorem 2.4 (Local Stability of Equilibria of Gradient Systems)** *Let $P^*$ be a critical point of $\mathcal{H}$. Then $P^*$ is a locally asymptotically stable equilibrium of the gradient system $\dot{P} = -kd\mathcal{H}/dP$ if and only if $P^*$ is an isolated minimum of $\mathcal{H}$.*

**Proof 2.4** *Please see [43] Chapter 9, Section 4, corollary to Theorem 1.*

**Remark 2.2** *Theorem 2.3 is concerned with all critical points of $\mathcal{H}$—maxima, minima, and saddle points. However, it is intuitively clear that the system ought to prefer minima. This intuition is made precise in Theorem 2.4. There are initial conditions for which the system will converge to a saddle point or a maximum, but these critical points are not locally stable. That is, a perturbation will cause the system to leave the critical point. Minima, on the other hand, are locally stable. They are robust to perturbations.*

We will use Theorems 2.3 and 2.4 to prove convergence and stability for many particular controllers throughout the thesis.

## 2.5 Voronoi Coverage Control

Cortés et al. [26] proposed a controller for deploying a group of robots over an environment to provide sensor coverage of the environment. The controller uses a Voronoi tessellation to divide up the environment among the robots, each robot being in charge of sensing over its Voronoi cell. In this thesis we use this controller, which we call the Voronoi controller, as a baseline strategy. We will demonstrate that the Voronoi controller is a special case of a more general class of coverage controllers. It is necessary therefore to provide a motivation and derivation for the Voronoi controller before proceeding to our contributions in the following chapters. In this section we review the Voronoi controller and prove some crucial details about it's derivation.

### 2.5.1 Voronoi Cost Function

The cost function upon which the Voronoi controller is based is adapted from the field of locational optimization, which addresses how to optimally place retail of industrial facilities [30, 114]. The canonical example is placing retail facilities to minimize the aggregate travel time of a population of customers.

Consider a multi robot system as in Section 2.7 in which the robots are positioned in a convex bounded environment $Q$. An arbitrary point in $Q$ is denoted $q$ and the robot positions $p_i \in Q = \mathcal{P}$. Define the *sensory function*, $\phi : Q \mapsto \mathbb{R}_{>0}$, and let it be known to all of the robots. The sensory function should be thought of as a weighting of importance over $Q$. We want to have many robots where $\phi(q)$ is large, and few where it is small. For now we will assume that the function $\phi(q)$ is known by the robots in the network. In Chapter 4 we will relax this requirement with on-line learning.

The precise definition of the sensory function depends on the desired application. In an application in which a team of robots are used to clean up an oil spill, an appropriate choice for the sensory function would be the concentration of the oil as a function of position in the environment. For a human surveillance application in which robots use audio sensors, $\phi(q)$ may be chosen to be the intensity of the frequency range corresponding to the human voice. The sensory function $\phi(q)$ may also be the probability density function relating to the occurrence of events in $Q$.

The robots are equipped with sensors and the quality of sensing is assumed to decrease according to a differentiable, strictly increasing function $f : \mathbb{R}_{\geq 0} \to \mathbb{R}$. Specifically, $f(\|q - p_i\|)$ describes how costly is the measurement of the information at $q$ by a sensor at $p_i$. This form of $f(x)$ is physically appealing since it is reasonable that sensing will become more unreliable farther from the sensor. Then the standard Voronoi cost function can be written

$$\mathcal{H}(P) = \int_Q \min_{i \in \{1,\ldots,n\}} f(\|q - p_i\|)\phi(q)\, dq. \tag{2.9}$$

The minimum over sensors reflects the fact that a point $q$ should be the responsibility of the sensor that has the best sensing performance at $q$. The problem of covering the environment $Q$ is now formulated as moving the robots to a configuration $P^*$ to minimize $\mathcal{H}$.

The cost function (2.9) has been used in a wide variety of applications including data compression, allocation of resources, and placement of industrial and commercial facilities. In the following section we consider computing the gradient of (2.9) in order to design a gradient descent controller.

Consider the minimization of (2.9)

$$\min_P \mathcal{H}(P) = \min_P \int_Q \min_i f(\|q - p_i\|)\phi(q)\, dq\,.$$

The minimum inside the integral induces a partition of $Q$ into non-overlapping cells, $V_i$, to give

$$\min_P \mathcal{H}(P) = \min_P \sum_{i=1}^{n} \int_{V_i} f(\|q - p_i\|)\phi(q)\, dq\,, \tag{2.10}$$

where $V_i = \{q \in Q \mid f(\|q-p_i\| \leq f(\|q-p_j\|) \quad \forall j \neq i\}$. Since $f$ is strictly increasing, this is equivalent to

$$V_i = \{q \in Q \mid \|q-p_i\| \leq \|q-p_j\| \quad \forall j \neq i\}. \tag{2.11}$$

The region $V_i$ is the Voronoi cell of $p_i$. The collection of Voronoi cells is called the *Voronoi tessellation*[2] [70] of $Q$, an example of which is shown in Figure 2-2(a).

## 2.5.2 Computations with Voronoi Tessellations

We now define a number of quantities relating to Voronoi cells. Let $\partial V_i$ and $\partial Q$ be the boundary of $V_i$ and $Q$, respectively. By $q_{\partial V_i}(P)$ we mean a point $q \in \partial V_i$, and $n_{\partial V_i}$ is the outward facing unit normal of $\partial V_i$. Given a robot $i$, we define $\mathcal{N}_i$ as the index set of robots that share Voronoi boundaries with $V_i$, $\mathcal{N}_i = \{j \mid V_i \cap V_j \neq \emptyset\}$. We denote the set of points on the Voronoi boundary shared by agents $i$ and $j$ as $l_{ij} = V_i \cap V_j$ as shown in Fig. 2-2. Then $q_{l_{ij}}(p_i, p_j)$ is a point on that shared boundary, and $n_{l_{ij}}$ is the unit normal of $l_{ij}$ from $p_i$ to $p_j$. By the definition of the Voronoi cell (2.11), we know the following facts:

$$\begin{aligned} \partial V_i &= (\cup_{j \in \mathcal{N}_i} l_{ij}) \cup (\partial V_i \cap \partial Q), && (2.12) \\ l_{ij} &= l_{ji}, && (2.13) \\ n_{l_{ij}} &= -n_{l_{ji}}. && (2.14) \end{aligned}$$

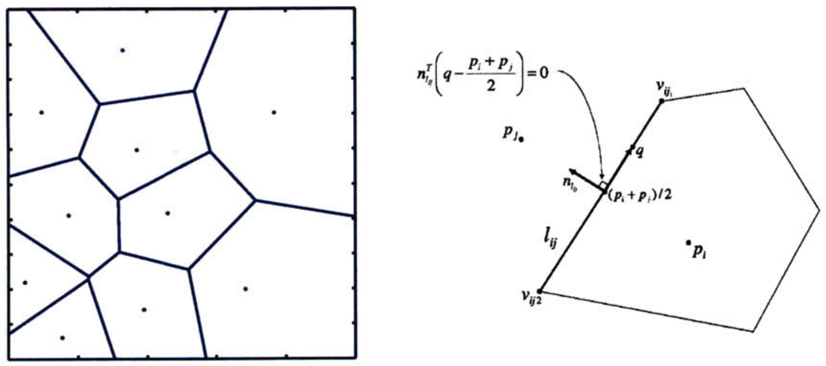
It can also be proved that the shared boundaries $l_{ij}$ are hyperplanes, and the Voronoi cells are convex.

The following lemma states an important fact about the cost function (2.10).

**Lemma 2.1 (Cancellation of Boundary Terms)** *The gradient of $\mathcal{H}(P)$ is given by*

$$\frac{\partial \mathcal{H}}{\partial p_i} = \int_{V_i} \frac{\partial}{\partial p_i} f(\|q-p_i\|)\phi(q)\, dq. \tag{2.15}$$

[2]It is convenient for us to use $\leq$ in the definition of Voronoi cell, rather than the more common definition with $<$



(a) Voronoi Tessellation (b) Detail of Voronoi Cell Boundary

Figure 2-2: An example of a Voronoi Tessellation is shown on the left and the quantities and constraints associated with the Voronoi boundary shared between neighboring agents is shown on the right.

**Proof 2.5** *Differentiating under the integral sign [32], we have*

$$\begin{aligned}\frac{\partial \mathcal{H}}{\partial p_i} &= \int_{V_i} \frac{\partial}{\partial p_i} f(\|q-p_i\|)\phi(q)\,dq \\ &+ \int_{\partial V_i} f(\|q-p_i\|)\phi(q)\frac{\partial q_{\partial V_i}(P)}{\partial p_i} n_{\partial V_i}\,dq \\ &+ \sum_{j\in\mathcal{N}_i}\int_{l_{ji}} f(\|q-p_j\|)\phi(q)\frac{\partial q_{l_{ji}}(p_i,p_j)}{\partial p_i} n_{l_{ji}}\,dq\,,\end{aligned}$$

*where* $\frac{\partial q_{\partial V_i}(P)}{\partial p_i}$ *and* $\frac{\partial q_{l_{ji}}(p_i,p_j)}{\partial p_i}$ *are* $d\times d$ *matrices. Using (2.12), (2.13), and (2.14)*

$$\begin{aligned}\frac{\partial \mathcal{H}}{\partial p_i} &= \int_{V_i} \frac{\partial}{\partial p_i} f(\|q,p_i\|)\phi(q)\,dq \\ &+ \int_{\partial V_i\cap\partial Q} f(\|q,p_i\|)\phi(q)\frac{\partial q_{\partial V_i}(P)}{\partial p_i} n_{\partial V_i}\,dq \\ &+ \sum_{j\in\mathcal{N}_i}\int_{l_{ij}} (f(\|q-p_i\| - f(\|q-p_j\|))\phi(q)\frac{\partial q_{l_{ij}}(p_i,p_j)}{\partial p_i} n_{l_{ij}}\,dq\,.\end{aligned}$$

*By definition of* $l_{ij}$, $\|q-p_i\| = \|q-p_j\|\ \forall q\in l_{ij}$, *so the last sum vanishes. Since points on the boundary of the environment do not change position as a function of* $p_i$, *we have*

$$\frac{\partial q_{\partial V_i}}{\partial p_i} = 0 \quad \forall q \in \partial V_i \cap \partial Q$$

*and the second term vanishes.* □

**Remark 2.3** *Lemma 2.1 is surprising for its simplicity. One might expect the gradient of $\mathcal{H}$ to be more complicated due to the fact that the Voronoi tessellation which defines the boundary of the integrals is a function of the robots' positions. Essentially, all of the complicated boundary terms cancel out leaving a simple expression for the gradient.*

**Remark 2.4** *For an agent to compute its gradient component (2.15) it must be able to compute its Voronoi cell, which means it must know the positions of its Voronoi neighbors. It is a common assumption in the literature [26, 77, 87] that a robot knows the positions of its Voronoi neighbors either by sensing or communication. Unfortunately, this assumption presents a practical conundrum: one does not know beforehand how far away the farthest Voronoi neighbor will be, thus this assumption cannot be translated into a communication range constraint (aside from the conservative requirement for each robot to have a communication range as large as the diameter of $Q$). In practice, only Voronoi neighbors within a certain distance will be in communication, in which case results can be derived, though with considerable complication [25]. We will take this assumption as implicit. Indeed, our experimental and numerical results suggest that performance degrades gracefully with decreasing communication range among robots.*

We now restrict ourselves to the case in which $f(x) = 1/2x^2$. This form of $f(x)$ is appropriate for light-based sensors, for example cameras, infrared detectors, or laser scanners, since the intensity from a light source drops of with the square of the distance to the source. The cost function becomes

$$\mathcal{H} = \sum_{i=1}^{n} \int_{V_i} 1/2\|q - p_i\|^2 \phi(q)\, dq\,. \tag{2.16}$$

Define the mass, first moment, and centroid of the Voronoi cell $V_i$ as

$$M_{V_i} = \int_{V_i} \phi(q)\, dq\,, \quad L_{V_i} = \int_{V_i} q\phi(q)\, dq\,, \quad \text{and} \quad C_{V_i} = L_{V_i}/M_{V_i}, \tag{2.17}$$

respectively. Note that $f(x)$ strictly increasing and $\phi(q)$ strictly positive imply both $M_{V_i} > 0 \;\forall\; V_i \neq \emptyset$ and $C_{V_i} \in V_i \backslash \partial V_i$ ($C_{V_i}$ is in the interior of $V_i$). Thus $M_{V_i}$ and $C_{V_i}$ have properties intrinsic to physical masses and centroids.

Using this notation, $\partial\mathcal{H}/\partial p_i$ simplifies to

$$\frac{\partial \mathcal{H}}{\partial p_i} = -\int_{V_i} (q - p_i)\phi(q)dq = -M_{V_i}(C_{V_i} - p_i)\,. \tag{2.18}$$

Critical points of $\mathcal{H}$ (configurations where the gradient is zero) are those in which every agent is at the centroid of its Voronoi cell, $p_i = C_{V_i}\ \forall i$. The resulting partition of the environment is commonly called a *Centroidal Voronoi Configuration* (CVC). It is known that CVC's can correspond to local maxima, minima, or saddle points of $\mathcal{H}$. Finding global minima of $\mathcal{H}$ is known to be difficult (NP-hard for a given discretization of $Q$) even in the fully centralized case. Next, we present a distributed control law that is used in [26] to make the robots converge to a CVC.

### 2.5.3 Voronoi Controller

A classic discrete-time method to compute CVC's is Lloyd's algorithm [60]. In each iteration this method executes three steps: (i) compute the Voronoi regions; (ii) compute the centroids; (iii) move each $p_i$ to its corresponding centroid.

In [26] continuous-time version of this approach is proposed for robots with simple integrator dynamics as in (2.5). The control law is given by

$$u_i = k(C_{V_i} - p_i) \tag{2.19}$$

and it guarantees that the system converges to a CVC. This control law gives a variation on the multi-robot system (2.7), with

$$\dot{p}_i = -\frac{k}{M_{V_i}}\frac{\partial \mathcal{H}}{\partial p_i}. \tag{2.20}$$

The proof of convergence is similar to that of Theorem 2.3. Firstly, we must restrict the state space $\mathcal{S} \subset \mathcal{P}^n$ to those configurations for which $p_i \neq p_j \ \forall i \neq j$, that is $\mathcal{S} = \{P = [p_1^T \cdots p_n^T]^T \mid p_i \neq p_j \quad \forall i \neq j\}$. It is known from [26] that $\mathcal{S}$ is invariant under the control law (2.19) (this relies upon $Q$ being convex). Now define the set of CVCs $\Omega = \{P \mid p_i = C_{V_i} \quad \forall i\}$. Notice from $\partial \mathcal{H}/\partial p_i$ that $\Omega$ is precisely the set of critical points of $\mathcal{H}$ over $\mathcal{P}$. This leads to the following convergence result, which is similar to Theorem 2.3, but has a slightly more subtle proof.

**Theorem 2.5 (Voronoi Convergence)** *The system with dynamics $\dot{p}_i = k(C_{V_i} - p_i)$ $i = 1, 2, \ldots, n$ converges asymptotically to the set of centroidal Voronoi configurations $\Omega$.*

**Proof 2.6** *The theorem follows from LaSalle's Invariance Principle [49, 55, 103]. By assumption, the domain of the robots $Q$ is bounded, therefore $\mathcal{P}$ is bounded. $\mathcal{P}$ is also invariant under the control law (2.19), therefore all trajectories are bounded. The control law is locally Lipschitz (this can be verified directly from the definition of locally Lipschitz). Computing $\dot{\mathcal{H}}$ along the trajectories of the system we find $\dot{\mathcal{H}} = -\sum_i^n M_{V_i}\|C_{V_i} - p_i\|^2 \leq 0$, so all the conditions of LaSalle's Invariance Principle are satisfied, and the system converges to the largest invariant set in $\Omega$, but $\Omega$ itself is invariant, so the system converges to $\Omega$.*

**Remark 2.5** *Theorem 2.4 applies directly to this controller as well. Unfortunately, it is difficult to determine for this cost function which, if any, critical points are isolated minima. Indeed, this appears to be strongly dependent upon the specific geometry of $Q$ and $\phi(q)$. For example, if $Q \subset \mathbb{R}^2$ is a circular disc and $\phi(q)$ is constant, then the set of critical points can be shown to be a closed orbit in $\mathbb{R}^{2n}$ (this is not difficult to visualize considering the symmetry of the problem).*

## 2.6 Synopsis

In this chapter we laid the groundwork for the remainder of the thesis. Firstly, we situated our work in the research literature. Next, we fixed the mathematical notation that will be used throughout the thesis. We then formulated the main multi-robot system model to be used throughout the thesis and stated two foundational theorems, one concerning the convergence of gradient systems to the set of critical points of their associated cost function, and one stating that only isolated local minima are stable. Finally, we described in detail an existing multi-robot coverage control strategy that relies upon a Voronoi tessellation, and we proved the convergence of that strategy to a centroidal Voronoi configuration.

The Voronoi controller (2.19) serves as a baseline throughout the thesis. We show in Chapter 3 that it is, in fact, a limiting instance of a more general class of coverage controllers that model a broad range of sensing and actuation modalities. We add stable on-line learning to learn the sensory function $\phi(q)$ in Chapter 4, and we look at three detailed case studies with experiments in Chapters 5, 6, and 7.

# Chapter 3

# Generalized Coverage Control

## 3.1 Introduction

In this chapter we introduce two of the main theoretical contributions of the thesis. Firstly we propose a general multi-robot cost function which can be specialized to a number of different multi-robot tasks. The negative gradient of the cost function is used as a multi-robot controller as in equation (2.7). The cost function has three main components: a sensory function, a sensor cost, and a mixing function. We put special emphasis on the role of the mixing function, and show that a particular family of mixing functions act as a smooth approximation to the Voronoi controller seen in Chapter 2. The Voronoi controller is recovered exactly in the limit as a parameter goes to $-\infty$, while a new probabilistic interpretation is achieved with a parameter value of $-1$. Herding and consensus controllers are also derived with different values of the mixing function, sensor cost, and sensory function.

Secondly, we prove a result that justifies the fact that our controllers are proven to converge only to local minima of the cost function, rather than global minima. It is known that gradient descent controllers can be proven to find global minima of convex functions, but in general they can only be proven to find local minima for nonconvex functions. It is tempting, therefore, to try to find convex cost functions for multi-robot problems. We prove in this chapter that any multi-robot task other than consensus must be characterized by a nonconvex cost function, and therefore, in general, one cannot expect better than convergence to a local minima. This does not close the door on the problem, but rather, it focuses attention on finding special classes of nonconvex cost functions that may allow for specialized convergence results.

### 3.1.1 Related Work

Cortés et al. [26] introduced a controller for multi-robot coverage that works by continually driving the robots toward the centroids of their Voronoi cells, as described in Section 2.19. This inherently geometric strategy has seen many recent extensions to robots with a limited sensing radius in [25], to heterogeneous groups of robots and nonconvex environments in [77], and to incorporate learning of unknown environments in [97]. A recent text that presents much of this work in a cohesive fashion is [14] and an excellent overview is given in [63]. Coverage controllers also have been successfully implemented on robotic systems in [94, 96]. In this work we adopt notational conventions from the Voronoi based coverage control literature. Other common methods for coverage control take a probabilistic perspective. For example [57] proposes an algorithm for positioning robots to maximize the probability of detecting an event that occurs in the environment. Distributed dynamic vehicle routing scenarios are considered in [4, 76], in which events occur according to a random process and are serviced by the robot closest to them. Another common coverage control method is for robots to drive away from one another using artificial potential fields [44]. Despite the rather different models and objectives in these works, there are two common points which motivate us to find a unifying principle: 1) they all rely upon an optimization, and 2) they all use controllers that solve this optimization through the evolution of a dynamical system.

Some existing approaches do not fit under the framework we propose in this chapter. A significant body of work has looked at coverage control as a motion planning problem. A survey of this work can be found in [20], and some significant contributions can be found in, for example, [16, 56] and the citations therein. Other authors have proposed information theoretic algorithms which consider placing sensors sequentially rather than driving them with a controller. Works such as [42, 52] position sensor nodes to maximize information for the sake of estimating a Gaussian random process in the environment.

### 3.1.2 Contributions

The optimization approach in this chapter ties together much of the existing literature on coverage control. Specifically, our contributions are:

1. We propose a cost function, putting particular emphasis on the role of a *mixing function*, a previously unrecognized component that captures critical assumptions about the coverage task. We introduce a family of mixing functions with a free parameter, $\alpha$, and show that different values of the parameter correspond to different assumptions about the coverage task, specifically showing that a minimum variance solution (i.e. a probabilistic strategy) is obtained with a parameter value of $\alpha = -1$, Voronoi coverage (a geometric strategy) is recovered in the limit $\alpha \to -\infty$, and a broad family of potential field based herding and consensus controllers are recovered for positive values of $\alpha$.

2. We prove a new result linking the convexity of a cost function to the multi-agent phenomenon of consensus. We show that coverage tasks are fundamentally different from consensus, and that they require the optimization of a nonconvex cost function. This suggests inherent limitations to gradient descent controller designs, which are pervasive in the coverage control literature.

The chapter is organized as follows. In Section 3.2 we introduce the cost function, describing the purpose of each of its parts including the mixing function. We then produce a class of provably stable distributed coverage controllers by taking the gradient of the cost function. In Section 3.3 we derive three special cases of the controller; a Voronoi controller, a minimum variance controller, and a potential field controller. Section 3.4 presents our results on the relation between the convexity of a cost function, and multi-agent consensus. Simulation results are given in Section 3.5 and a synopsis is presented in Section 3.6.

## 3.2 Generalized Coverage

In this section we introduce a general multi-agent cost function. We will use this cost function to define a new class of multi-agent controllers by introducing a *mixing function*, which describes how information from different robots should be combined. We use the cost function to derive a stable gradient descent controllers of the form (2.7).

### 3.2.1 Coverage Cost Function

In keeping with the notation from Chapter 2, let there be $n$ robots, and let robot $i$ have a position $p_i \in \mathcal{P} \subset \mathbb{R}^{d_p}$, where $\mathcal{P}$ is the state space of a robot, and $d_p$ is the dimension of the space. The configuration of the multi-robot system in denotes $P = [p_1^T \cdots p_n^T]^T \in \mathcal{P}^n$. We want our robots to cover a bounded region $Q \subset \mathbb{R}^{d_q}$, which may or may not be related to the position space $\mathcal{P}$ of the robots. For example, the robots may be constrained to move in the space that they cover, so $\mathcal{P} = Q$ as in [26], or the robots may hover over a planar region that they cover with cameras, so $\mathcal{P} \subset \mathbb{R}^3$ and $Q \subset \mathbb{R}^2$, as in [94].

For each robot, a cost of sensing, or servicing, a point $q \in Q$ is given by a function $f(p_i, q)$. For simplicity of analysis we assume that $f(p_i, q)$ takes on only non-negative values, and that it is differentiable with respect to $p_i$ (this can be generalized considerably as in [25]). The sensor measurements of the $n$ robots are combined in a function $g(f(p_1, q), \ldots, f(p_n, q))$, which we will call the *mixing function.* The mixing function embodies assumptions about the coverage task; that is, by changing the mixing function we can derive Voronoi based coverage control, probabilistic coverage control, and a variety of other kinds of distributed controllers.

Combining these elements, we propose to use a cost function of the form

$$\mathcal{H}(P) = \int_Q g(f(p_1, q), \ldots, f(p_n, q))\phi(q)\, dq. \tag{3.1}$$

where $\phi : \mathbb{R}^{d_q} \mapsto \mathbb{R}_{>0}$ (we use the notation $\mathbb{R}_{>0}$ to mean the set of positive real numbers and $\mathbb{R}^d_{>0}$ the set of vectors whose components are all positive, and likewise for $\mathbb{R}_{\geq 0}$ and $\mathbb{R}^d_{\geq 0}$) is a weighting of importance over the region $Q$. Intuitively, the cost of the group of robots sensing at a single arbitrary point $q$ is represented by the integrand $g(f(p_1, q), \ldots, f(p_n, q))$. Integrating over all points in $Q$, weighted by their importance $\phi(q)$ gives the total cost of a configuration of the robots. We want to find controllers that stabilize the robots around configurations $P^*$ that minimize $\mathcal{H}$. We will see in Section 3.4 that for coverage, and many other multi-agent problems, $\mathcal{H}$ is necessarily nonconvex, therefore gradient based controllers will yield *locally* optimal robot configurations. The cost function (3.1) will be shown to subsume several different kinds of existing coverage cost functions. Drawing out the relations between these different coverage algorithms will suggest new insights into when one algorithm should be preferred over another.

### 3.2.2 Mixing Function

The mixing function $g_\alpha : \mathbb{R}^n_{\geq 0} \mapsto \mathbb{R}$ describes how information from different robots should be combined to give an aggregate cost of the robots sensing at a point $q$. This is shown graphically in Figure 3-1 where the overlap of the two sensors is shown for illustrative purposes as the intersection of two circles. We propose a mixing function of the form

$$g_\alpha(f_1, \ldots, f_n) = \Big(\sum_{i=1}^{n} f_i^\alpha\Big)^{\frac{1}{\alpha}}, \tag{3.2}$$

with a free parameter $\alpha$. The arguments $f_i \geq 0$ are real valued, and in our context they are given by evaluating the sensor function $f(p_i, q)$, hence the notation $f_i$.

This mixing function has several important properties. Firstly, notice that for $\alpha \geq 1$ it is the $p$-norm of the vector $[f_1 \cdots f_n]^T$. Specifically, it is convex for $\alpha \geq 1$ and as $\alpha \to \infty$, $g_\alpha(\cdot) \to \max_i(\cdot)$, which is the $\ell^\infty$ norm. However, in the regime where $\alpha < 1$, $g_\alpha(\cdot)$ is not a norm because it violates the triangle inequality. In this regime it is also nonconvex, the significance of this will be explored more in Section

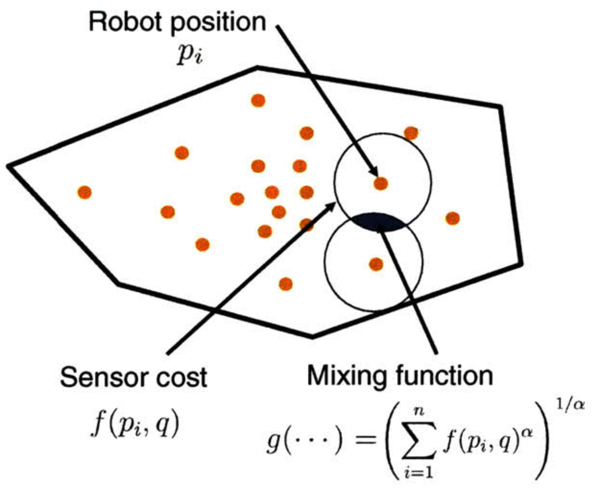
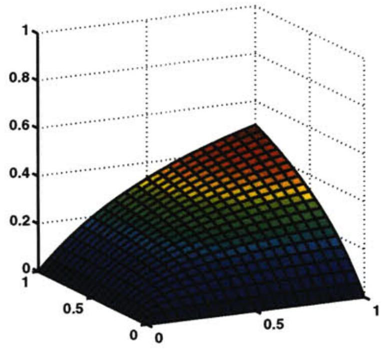
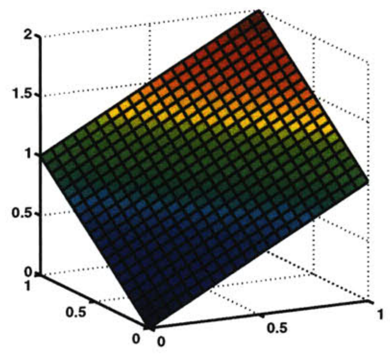

Robot position
$p_i$
Sensor cost
$f(p_i, q)$
Mixing function
$g(\cdots) = \left(\sum_{i=1}^{n} f(p_i, q)^\alpha\right)^{1/\alpha}$

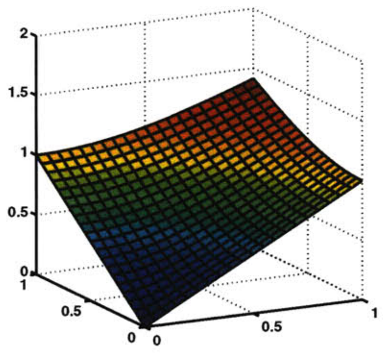
Figure 3-1: The mixing function is illustrated in this figure. The mixing function determines how information from the sensors of multiple robots is to be combined, shown graphically as the intersection of the two circles in the figure.

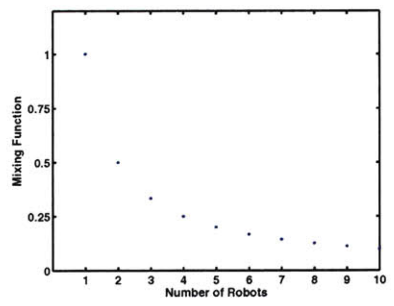
3.4. One can readily verify[1] that as $\alpha \to -\infty$, $g(\cdot) \to \min_i(\cdot)$. From an intuitive point of view, with $\alpha < 1$, $g_\alpha(\cdot)$ is smaller than any of its arguments alone. That is, the cost of sensing at a point $q$ with robots at $p_i$ and $p_j$ is smaller than the cost of sensing with either one of the robots individually. Furthermore, the decrease in $g_\alpha$ from the addition of a second robot is greater than that from the addition of a third robot, and so on. There is a successively smaller benefit to adding more robots. This property is often called supermodularity, and has been exploited in a rather different way in [52]. Surface plots of $g_\alpha(f_1, f_2)$ for $\alpha = -1$, 1, and 2 are shown in Figures 3-2(a), 3-2(b), and 3-2(c), respectively, and the decrease in $g_\alpha(\cdot)$ as the number of arguments grows is shown in Figure 3-2(d). In this work we consider the number of robots to be fixed, but it is useful to illustrate the supermodularity property of the mixing function by considering the successive addition of new robots.

[1]We know $\lim_{\beta\to\infty}[\sum_i h_i^\beta]^{1/\beta} = \max_i h_i$. Write $\lim_{\alpha\to-\infty}[\sum_i f_i^\alpha]^{1/\alpha}$ as $\lim_{\beta\to\infty}[[\sum_i h_i^\beta]^{1/\beta}]^{-1}$ with $h_i = 1/f_i$ and $\beta = -\alpha$. We have $\lim_{\beta\to\infty}[[\sum_i h_i^\beta]^{1/\beta}]^{-1} = [\max_i h_i]^{-1} = [\frac{1}{\min_i f_i}]^{-1} = \min_i f_i$.


(a) Mixing Function Surface ($\alpha = -1$)


(b) Mixing Function Surface ($\alpha = 1$)


(c) Mixing Function Surface ($\alpha = 2$)



(d) Mixing Function Supermodularity ($\alpha = -1$)

Figure 3-2: The proposed mixing function with $\alpha = -1$, 1, and 2 is shown in 3-2(a), 3-2(b), and 3-2(c), respectively. The function is convex for $\alpha \geq 1$ and nonconvex otherwise. The nonlinear decrease in the function as more sensors are added, a property known as supermodularity, is shown in Figure 3-2(d).

Including this mixing function in the cost function from (3.1) gives

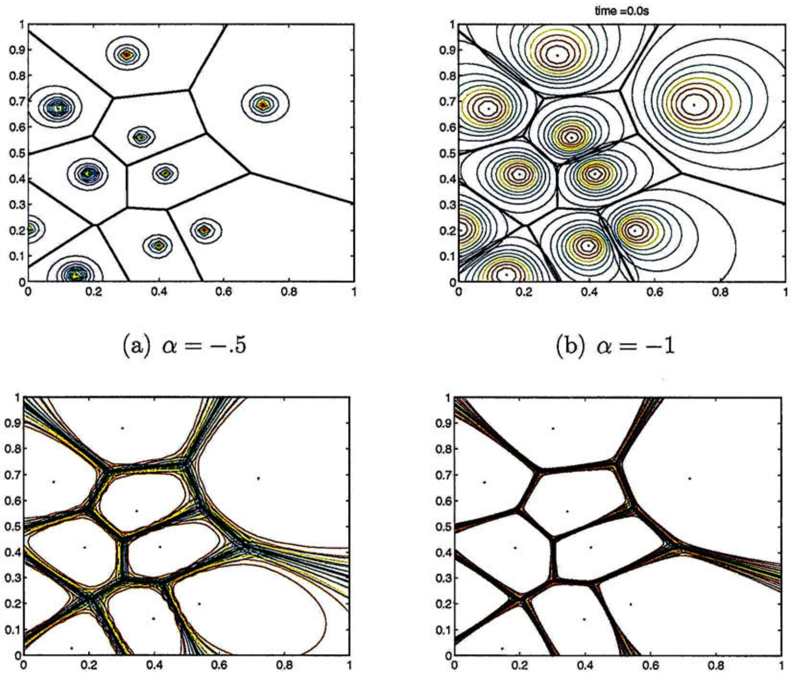
$$\mathcal{H}_\alpha = \int_Q \big(\sum_{i=1}^{n} f(p_i, q)^\alpha\big)^{\frac{1}{\alpha}} \phi(q)\, dq. \tag{3.3}$$

## 3.2.3 Gradient Control

We use the multi-robot system equation (2.7) to derive the gradient controller

$$\dot{p}_i = -k\frac{\partial \mathcal{H}_\alpha}{\partial p_i} = -k \int_Q \left(\frac{f(p_i, q)}{g_\alpha}\right)^{\alpha-1} \frac{\partial f(p_i, q)}{\partial p_i} \phi(q)\, dq. \tag{3.4}$$

To provide some intuition about the meaning of this function, notice that in the case that $f(p_i, q)$ is strictly increasing, the function inside the integral $(f(p_i, q)/g_\alpha)^{\alpha-1}$ gives an approximation to the indicator function[2] of the Voronoi cell of agent $i$, the approximation improving as $\alpha \to -\infty$. This is shown graphically in Figure 3-3. For



(a) $\alpha = -.5$ (b) $\alpha = -1$

(c) $\alpha = -5$ (d) $\alpha = -10$

Figure 3-3: Contour plots of $(f(p_i, q)/g_\alpha)^{\alpha-1}$ are shown for a configuration of ten agent positions. The Voronoi tessellation is shown as well for comparison. As the parameter $\alpha$ approaches $-\infty$, $(f(p_i, q)/g_\alpha)^{\alpha-1}$ becomes closer to the indicator function of the Voronoi cell $V_i$.

simplicity, we choose the function $f(p_i, q)$ to be

$$f(p_i, q) = \frac{1}{2}\|q - p_i\|^2, \quad \text{so that} \quad \frac{\partial f(p_i, q)}{\partial p_i} = -(q - p_i). \tag{3.5}$$

Other choices of $f(p_i, q)$ were investigated in [25] and could be used here as well. This function represents the cost of a single robot $i$ sensing at the position $q$. The quadratic form is appropriate for a variety of sensors including light based sensors,

[2]The indicator function for a set $S \subset Q$ returns 1 for $q \in S$, and 0 otherwise.

such as cameras or laser scanners, chemical sensors, and others. For tasks in which robots have to drive to a point $q$ for servicing, and we want the cost to be proportional to the distance travelled, it would be more appropriate to use $f(p_i, q) = \|q - p_i\|$, for example. Convergence to critical points of $\mathcal{H}_\alpha$ and stability of isolated local minima can be proved by a direct application Theorem 2.3 and Theorem 2.4, respectively, from Chapter 2.

**Theorem 3.1 (Convergence of Coverage Control)** *For a group of robots with closed-loop dynamics (3.4) the robots converges to the set of critical points of* $\mathcal{H}_\alpha$.

**Proof 3.1** *The theorem is immediate from Theorem 2.3.*

**Theorem 3.2 (Stability of Local Minima)** *For a group of robots with closed-loop dynamics (3.4) only configurations* $P^*$ *for which* $\mathcal{H}_\alpha(P^*)$ *is an isolated local minimum are locally asymptotically stable.*

**Proof 3.2** *The theorem follows directly from Theorem 2.4.*

**Remark 3.1 (Stability Vs. Convergence)** *Although we already brought attention to this distinction in Chapter 2, it is useful to reiterate the point. Theorem 3.1 says that the system will converge to the set of critical points of* $\mathcal{H}_\alpha$*, while Theorem 3.2 says that only local minima of* $\mathcal{H}_\alpha$ *are stable. It is a basic fact of dynamical systems that, given a special set of initial conditions, trajectories may converge to unstable equilibria. However small perturbations will cause them to leave an unstable equilibrium, while stable equilibria are robust to small perturbations. In our experience cost functions such as* $\mathcal{H}_\alpha$ *have many saddle points and local maxima, so it is meaningful to specify that the system prefers local minima of* $\mathcal{H}_\alpha$.

**Remark 3.2 (Network Requirements)** *The computation of the controller requires that robot i knows the states of all the robots in the network. For this to be feasible there must either be a global supervisor or a fully connected network communication topology. It would be more useful if the controller depended only upon the states of robots with which it communicates. We suggest two methods to accomplish this, but* *we do not analyze them in detail in this thesis. First, robot $i$ can approximate its control law simply by computing (3.4) using only the states of the robots with which it is in communication. We expect this to give a good approximation because the function $(f(p_j,q)/g_\alpha)^{\alpha-1}$ depends weakly upon the states of agents that are not Voronoi neighbors, especially for small values of $\alpha$, as evident from Figure 3-3 . A rigorous stability analysis of this approximation scheme is difficult, however. A second option is for a robot $i$ to use an estimated configuration vector, $\hat{P}$, in its calculation of the control law. The estimated configuration can be updated online using a standard distributed consensus algorithm (a so called "consensus estimator"). We expect that such a scheme may be amenable to a rigorous stability proof as its architecture is similar to adaptive control architectures. The investigation of these matters is left for future work.*

## 3.3 Deriving Special Cases

In this section we show how the cost function 3.1 can be specialized to give three common kinds of coverage controllers, a Voronoi controller, which is geometric in nature, a minimum variance controller, which has a probabilistic interpretation, and a potential field controller. We conjecture that other coverage objectives beyond these three can be achieved with different choices of the mixing function parameter $\alpha$.

### 3.3.1 Voronoi Coverage

The Voronoi-based coverage controller described in Section 2.19 is based on a gradient descent of the cost function

$$\mathcal{H}_\mathcal{V} = \sum_{i=1}^{n} \int_{V_i} \frac{1}{2}\|q - p_i\|^2 \phi(q)\, dq, \tag{3.6}$$

where $V_i = \{q \in Q \mid \|q - p_i\| \leq \|q - p_j\|, \forall j \neq i\}$ is the Voronoi cell of robot $i$ and the use of the subscript $\mathcal{V}$ is to distinguish it from $\mathcal{H}$ and $\mathcal{H}_\alpha$. The Voronoi partition can equivalently be written using the min function as

$$\mathcal{H}_{\mathcal{V}} = \int_Q \min_i(\frac{1}{2}\|q - p_i\|^2)\phi(q)\, dq, \tag{3.7}$$

because a point $q$ is in the Voronoi cell $V_i$ if and only if $\|q - p_j\|$ is minimized for $j = i$. As noted in Section 3.2.2, $\lim_{\alpha\to-\infty} g_\alpha(f_1, \ldots, f_n) = \min_i f_i$. Therefore $\mathcal{H}_{\mathcal{V}}$ is a special instance of (3.3) with the mixing function $g_{-\infty} = lim_{\alpha\to-\infty} g_\alpha$ and $f(p_i, q) = 1/2\|q - p_i\|^2$.

The choice of the min function for a mixing function now warrants some reflection. Consider a distributed actuation scenario in which we want to position robots so as to service an event that occurs randomly at some point in the environment $q$. Suppose any robot is equally capable of rendering the service, robots have to physically travel to the event to render the service, and our objective is to service an event as quickly as possible. Naturally, an event should be serviced by the robot that is closest to it, as it will reach the event the most quickly. In this case, the min function is the appropriate choice for a mixing function. By using the min function we are saying that the cost incurred by all the robots due to the event at $q$ is the same as that incurred by the robot that is closest to $q$.

On the other hand, consider a sensing task in which an event of interest occurs randomly at a point $q$ and is sensed at a distance by sensors located on the robots. In this case the use of the min function is more difficult to justify. Using the min function in this instance would imply that even though both $p_i$ and $p_j$ have some sensory information about the event, the cost function only counts the information from the one that is closest to $q$. This seems to be a poor choice of cost function for sensing, since in such cases we would want to capture the intuition that two sensors are better than one. The mixing function (3.2) captures this intuition. Furthermore, even in distributed actuation tasks, using a continuous approximation to the Voronoi cell improves the robustness of the controller. The discrete, geometric nature of the Voronoi computation combined with the continuous controller can lead to chattering, and small sensing errors can result in large changes in the control input. Fortunately, the Voronoi tessellation can be approximated arbitrarily well by choosing a small value of $\alpha$, thereby preserving the Voronoi controller behavior while improving robustness.

### 3.3.2 Minimum Variance Coverage

We show in this section that setting the mixing function parameter to $\alpha = -1$ causes the robots to minimize the expected variance of their measurement of the location of a target of interest. As a side effect, we will formulate an optimal Bayesian estimator for the location of the target given the measurements of the agents.

Suppose our agents are equipped with sensors that give a noisy measurement of the position of a target in the environment. Let the target position be given by a random variable $q$ that takes on values in $Q$, and agent $i$ gives a measurement $y_i = q + w$, where $w \sim N(0, I_2\sqrt{f(p_i, q)})$ is a bi-variate normally distributed random variable, and where $I_2$ is the $2{\times}2$ identity matrix. The variance of the measurement, $f(p_i, q)$, is a function of the position of the sensor and the target. Intuitively one would expect a sensor to localize a target with more precision the closer the target is to the sensor. Then the measurement likelihood of agent $i$ is $\mathbb{P}(y_i \mid q : p_i) = 1/(2\pi f(p_i, q)) \exp\{-\|y_i - q\|^2/(2f(p_i, q))\}$, and the notation $\mathbb{P}(\cdot : p_i)$ is to emphasize that the distribution is a function of the agent position. Assume the measurements of different agents conditioned on the target position are independent. Also, let $\phi(q)$ be the prior distribution of the target's position. Then Bayes rule gives the posterior distribution,

$$\mathbb{P}(q \mid y_1, \ldots, y_n) = \frac{\prod_{i=1}^{n} \mathbb{P}(y_i \mid q : p_i)\phi(q)}{\int_Q \prod_{i=1}^{n} \mathbb{P}(y_i \mid q : p_i)\phi(q)\, dq}. \tag{3.8}$$

One can use the posterior to obtain a Bayesian estimate of the position of the event $q$ given the measurements. For example, one may choose to estimate $q$ using the mean, the median, or the maximum of the posterior in (3.8).

Our interest here, however, is not in estimating $q$. Instead we are interested in positioning the robots so that whatever estimate of $q$ is obtained is the best possible one. To this end, we seek to position the robots to minimize the variance of their combined sensor measurements. The product of measurement likelihoods in the numerator of (3.8) can be simplified to a single likelihood function, which takes the form of an un-normalized Gaussian

$$\prod_{i=1}^{n} \mathbb{P}(y_i \mid q : p_i) = A \exp\left\{-\frac{\|\bar{y} - q\|^2}{2g_{-1}(\cdot)}\right\}, \tag{3.9}$$

whose variance is equivalent to our mixing function $g_{-1}(\cdot) = (\sum_{i=1}^{n} f(p_i, q)^{-1})^{-1}$. The values of $A$ and $\bar{y}$ are not important in this context, though we state them for completeness:

$$\bar{y} = g_{-1}(\cdot)\sum_{i=1}^{n} f(p_i, q)^{-1} y_i, \quad \text{and}$$

$$A = \frac{1}{(2\pi)^n \prod_{i=1}^{n} f(p_i, q)} \exp\left\{\frac{1}{2}\|\bar{y}\|^2 g_{-1}(\cdot) - \frac{1}{2}\sum_{i=1}^{n} \|y_i\|^2 f(p_i, q)\right\}.$$

If we want to position the robots so as to obtain the most decisive information from their sensors, we should move them to minimize this variance. Notice, however, that $g_{-1}(f(p_1, q), \dots, f(p_n, q))$ is a random variable since it is a function of $q$. Taking the expectation over $q$ of the likelihood variance gives our original cost function,

$$\mathcal{H}_{-1} = \mathbb{E}_q[g_{-1}(f(p_1, q), \dots, f(p_n, q))] = \int_Q g_{-1}(f(p_1, q), \dots, f(p_n, q))\phi(q)\, dq. \tag{3.10}$$

Thus we can interpret the coverage control optimization as finding the agent positions that minimize the expected variance of the likelihood function for an optimal Bayes estimator of the position of the target.

A more theoretically appealing criterion would be to position the agents to minimize the variance of the *posterior* distribution in (3.8). This gives a considerably more complicated cost function.

$$\mathrm{Var}[q \mid y_1, \dots, y_n] = \frac{\int_Q \prod_{i=1}^{n} \mathbb{P}(y_i \mid q : p_i)\phi(q) q q^T\, dq}{\int_Q \prod_{i=1}^{n} \mathbb{P}(y_i \mid q : p_i)\phi(q)\, dq} - \bar{q}\bar{q}^T, \tag{3.11}$$

where

$$\bar{q} = \mathbb{E}[q \mid y_1, \dots, y_n] = \frac{\int_Q \prod_{i=1}^n \mathbb{P}(y_i \mid q : p_i)\phi(q) q \, dq}{\int_Q \prod_{i=1}^n \mathbb{P}(y_i \mid q : p_i)\phi(q) \, dq}. \tag{3.12}$$

The complication of this cost function and the fact that gradients can not be easily computed makes it a less practical option.

## 3.3.3 Potential Field Coverage

The third type of coverage controller we consider is significantly different from the previous two in that it does not involve an integral over the environment. Instead it relies on the idea that robots should push away from one another to spread out over an environment, but should not move too far from one another or else they will become disconnected. Surprisingly, however, we will show that this rather different coverage philosophy can be reconciled with our generalized coverage cost function $\mathcal{H}_\alpha$ in (3.3).

Let the importance function, $\phi(q)$, be given as a sum of delta-Dirac functions centered at each of the robot positions

$$\phi(q) = \sum_{i=1}^n \delta(\|q - p_i\|). \tag{3.13}$$

Substituting this for $\phi(q)$ in (3.3), the integral in $\mathcal{H}_\alpha$ can then be evaluated analytically to give $\mathcal{H}_{\text{pot}} = \sum_{i=1}^n g_\alpha(f(p_1, p_i), \dots, f(p_n, p_i))$, and with $\alpha = 1$ we get

$$\mathcal{H}_{\text{pot}} = \sum_{i=1}^n \sum_{j=1, j\neq i}^n f(p_j, p_i), \tag{3.14}$$

which is a common cost function for potential field based models for herding and consensus, where $f(p_j, p_i)$ can be interpreted as an inter-agent potential function. One choice for $f(p_j, p_i)$ is

$$f(p_j, p_i) = \frac{1}{6}\|p_j - p_i\|^{-2} - \|p_j - p_i\|^{-1} \tag{3.15}$$

which, taking the gradient of (3.14), yields the controller

$$\dot{p}_i = k \sum_{j=1, j\neq i}^{n} \left( \|p_j - p_i\|^{-2} - \frac{1}{3}\|p_j - p_i\|^{-3} \right) \frac{p_j - p_i}{\|p_j - p_i\|}. \tag{3.16}$$

Controllers similar to this one have been studied in a number of works, for example [29,35,44,104]. There are numerous variations on this simple theme in the literature.

### 3.3.4 Computational Complexity

The gradient controllers described in this work must inevitably be discretized and implemented in a discrete time control loop. A common criticism of the Voronoi based coverage controller is that it is computationally intensive. At each iteration of the loop, a robot must re-compute its Voronoi cell. Additionally, the controller must compute spatial integrals over a region. In general, a discretized approximation must be used to compute the integral of $\phi(q)$ over the Voronoi cell, which is again computationally intensive. The two parameters that are important for computation time are the number of robots $n$ and the number of grid squares in the integral computation, which we will call $m$. The typical decentralized algorithm for a single robot to compute its Voronoi cell (from [26]) runs in $O(n)$ time. The time complexity for computing a discretized integral is linear in the number of grid squares, and at each grid square requires a check if the center point is in the Voronoi cell, which is an $O(n)$ operation. Therefore the time complexity of the integral is in $O(nm)$. The Voronoi cell must be computed first, followed by the discretized integral, therefore the standard Voronoi controller has time complexity $O(n(m + 1))$ at each step of the control loop.

Our controller in (3.4) does not require the computation of a Voronoi cell, but it does require the discretized spatial integral over the environment. We do not have to check if a point is in a polygon, but the integrand we evaluate, namely $g_\alpha$ is linear in $n$. Therefore the integral computation still has time complexity $O(nm)$, which is the time complexity of the controller at each step of the control loop. Yet as $\alpha$ decreases, the behavior of the controller approaches that of the Voronoi controller. The controller we propose in this chapter is therefore significantly simper in implementation (since it does not require the Voronoi computation), and it is faster computationally.

## 3.4 Convexity and Consensus

Since we treat the multi-agent coverage problem as an optimization, it is natural to ask what sort of optimization we are dealing with, and what optimization tools can be brought to bear to solve it. We show in this section that the cost function in (3.3) is nonconvex, and that nonconvexity is a required feature of a large class of multi-agent problems, however undesirable this may be from an optimization perspective. Specifically, we demonstrate a link between the *convexity* of a cost function and the multi-agent phenomena known as *consensus.* For our purposes, consensus describes a multi-agent configuration in which all agents take on the same state, $p_1 = p_2 = \ldots = p_n$. Consensus is geometrically represented in the state space $\mathcal{P}^n$ as a $d_p$-dimensional hyperplane that passes through the origin (from the $d_p(n-1)$ independent equality constraints). This is illustrated by the diagonal line in Figure 3-5 in a simplified 2D setting. We will prove, with some technical assumptions, that a multi-agent problem with a *convex* cost function admits at least one globally optimal consensus solution. Figure 3-4 shows a graphical schematic illustrating the meaning of the theorem.

Consider a general multi-agent cost function $\mathcal{H} : \mathcal{P}^n \mapsto \mathbb{R}$. As before, an agent $i$ has a state $p_i \in \mathcal{P} \subset \mathbb{R}^d$. It will be more convenient in this section to refer to a configuration of agents as a tuple $(p_1, \ldots, p_n) \in \mathcal{P}^n$, rather than the column vector notation used previously. Let us assume that agents are anonymous with respect to the cost function, by which we mean that the positions of any two agents can be interchanged without affecting the value of the cost function. This is formalized by the following assumption.

**Assumption 3.1 (Anonymity of Agents)** *The cost function $\mathcal{H}$ is such that*

$$\mathcal{H}(\ldots, p_i, \ldots, p_j, \ldots) = \mathcal{H}(\ldots, p_j, \ldots, p_i, \ldots) \quad \forall i, j \in \{1, \ldots, n\}. \tag{3.17}$$

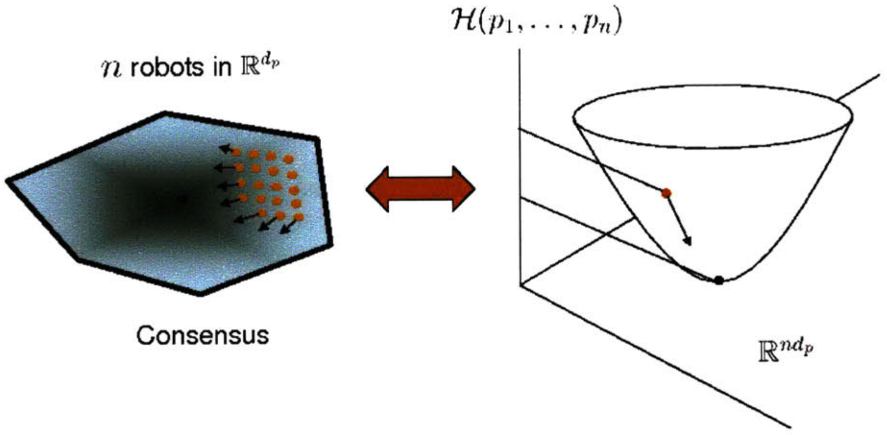

$n$ robots in $\mathbb{R}^{d_p}$
Consensus
$\mathcal{H}(p_1, \dots, p_n)$
$\mathbb{R}^{nd_p}$

Figure 3-4: This schematic illustrates the meaning of Theorem 3.3 and Corollary 3.1. If the cost function $\mathcal{H}$ is convex, the robots will all move to the same position, a behavior called consensus.

Assumption 3.1 is in keeping with the ethos of multi-robot systems, where the emphasis is on the global patterns that result from the interactions of many identical robots. Furthermore, let us assume that $\mathcal{H}$ and $\mathcal{P}^n$ satisfy at least one of the three properties in Theorem 2.1, which is simply to say that the set of minima of $\mathcal{H}$ over $\mathcal{P}^n$ is non-empty. Now we give the main result of this section.

**Theorem 3.3 (Convexity and Consensus)** *Under Assumption 3.1, if the cost function $\mathcal{H}(p_1, \dots, p_n)$ is convex, $\mathcal{P}^n$ is convex, and one of the conditions in Theorem 2.1 is satisfied, then $\mathcal{H}(p_1, \dots, p_n)$ has a global minimum such that $p_i = p_j \; \forall i, j \in \{1, \dots, n\}$.*

**Proof 3.3** *Our argument rests upon Assumption 3.1 and the fact from Theorem 2.1 that the set of minima of the convex function $\mathcal{H}$ is a convex set. Let $\Omega$ be the set of minima, and let $(\dots, p_i^*, \dots, p_j^*, \dots)$ be an optimal solution in that set. By Assumption 3.1, $(\dots, p_j^*, \dots, p_i^*, \dots)$ is also an optimal solution for any $i$ and $j$. Therefore all permutations of components in $(p_1^*, \dots, p_n^*)$ are optima. Then by convexity of $h^*$, all convex combinations of points in $h^*$ are in $h^*$. In particular, the point $(\bar{p}, \dots, \bar{p})$, where $\bar{p} = 1/n \sum_{i=1}^{n} p_i$ is an optimal solution (since it is a convex combination of permutations of $(p_1, \dots, p_n)$).*

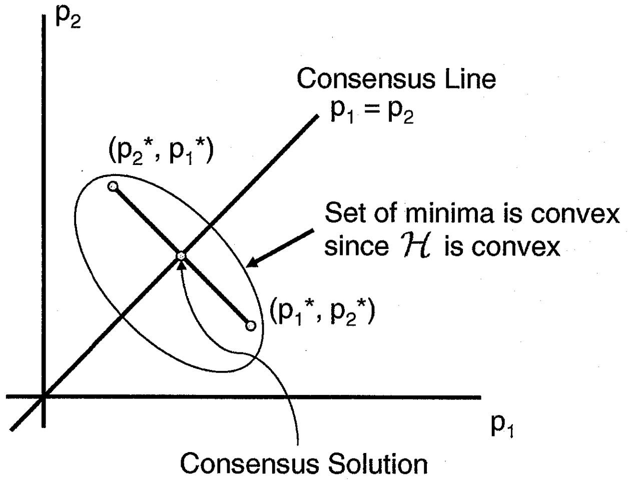


Figure 3-5: This schematic shows the geometrical intuition behind the proof of Theorem 3.3 in a simplified 2D setting. Corollary 3.1 is proved by noticing that the set of minima is a single point (the consensus solution) if $\mathcal{H}$ is strictly convex.

We show a geometric schematic of the proof argument in Figure 3-5. The proof uses the fact that the convex set of minima must intersect the consensus hyperplane (the hyperplane where $p_i = p_j \ \forall i, j$) at at least one point. A simple corollary follows.

**Corollary 3.1 (Strict Convexity)** *If the conditions of Theorem 3.3 are met and the cost function $\mathcal{H}(p_1, \dots, p_n)$ is strictly convex, then the minimum is unique and is such that $p_i = p_j \ \forall i, j \in \{1, \dots, n\}$.*

**Proof 3.4** *A strictly convex function has at most one minimum over a convex domain.*

**Remark 3.3 (Consensus vs. Non-consensus)** *Theorem 3.3 suggests that it is futile to search for convex cost functions for multi-robot problems other than consensus. It delineates two classes of multi-agent behaviors reminiscent of complexity classes in the theory of computation. One class, which we will call consensus behaviors, can be described as optimizing a convex cost function. The other class, which we will call non-consensus behaviors, is fundamentally different in that it can only be described* *with nonconvex cost functions. This is important because if we wish to design an optimization to solve a multi-agent problem, and we know that the problem cannot be solved satisfactorily by all the agents taking the same state, then we must use a nonconvex cost function. Likewise if we observe a multi-agent behavior in nature which cannot be described by all agents reaching the same state (the construction of a termite nest, for example), then an optimization-based explanation of this behavior must be nonconvex.*

**Remark 3.4 (Coverage is Nonconvex)** *This is directly applicable to coverage problems. Indeed, coverage cannot be achieved with all agents moving to the same place, therefore coverage problems must involve the optimization of a nonconvex cost function. Our parameterized cost function* $\mathcal{H}_\alpha$ *from (3.3) is nonconvex for* $\alpha < 1$*, in which regime it corresponds to a coverage task (e.g.* $\alpha \to -\infty$ *for Voronoi and* $\alpha = -1$ *for minimum variance). It becomes convex (assuming* $f$ *is convex) for* $\alpha > 1$ *in which regime it results in consensus. Theorem 3.3 explains why this is the case.*

**Remark 3.5 (Future Directions)** *From an algorithmic point of view, this is unfortunate. Convex optimization has a powerful and well characterized tool set guaranteed to reach global minima, but nonconvex optimization requires searching out special cases and special cost function properties. Often one must be satisfied with local minima. Distributed coverage controllers that use gradient methods (such as those in this chapter) guarantee convergence to local minima, which is all one can expect in a general nonconvex setting. This points towards at least two open questions for future work: can we find nonconvex multi-robot cost functions that have special properties that guarantee global results? For example can we find multi-robot cost functions for which all minima are global (all minima have the same cost) or for which there is only one minimum even though the function is nonconvex? Alternately, are there nonconvex optimization methods not based on gradient descent that can be implemented in a multi-agent setting?*

## 3.5 Simulation Results

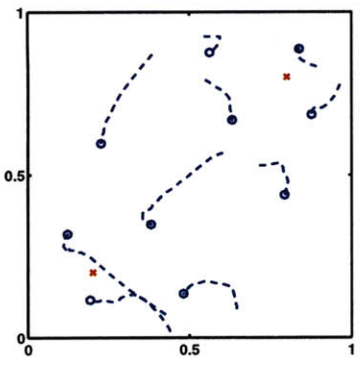
The controller for the three scenarios described in Section 3.3 were simulated in a Matlab environment. The environment $Q$ was taken to be a unit square, and the function $\phi(q)$ was set to be the sum of two Gaussian functions, one centered at $(.2, .2)$ and the other at $(.8, .8)$, both with variance .2. We expect to see a higher density of robots around areas of large $\phi(q)$. In our case, the robots group around the Gaussian centers.

The results of a simulation with ten robots using the Voronoi based controller, which corresponds to $\alpha \to -\infty$, is shown in Figs. 3-6(a) and 3-6(b). Similar plots are shown for the minimum variance controller, with $\alpha = -1$, in Figs. 3-6(c) and 3-6(d). Comparison of the two controllers shows that the Voronoi based controller causes the robots to spread out more, while as $\alpha$ increases, the robots group more closely together. When $\alpha > 1$, the cost function becomes convex, and the robots all move to the same position, which corroborates our results relating convexity to consensus (this is not shown in the plots).

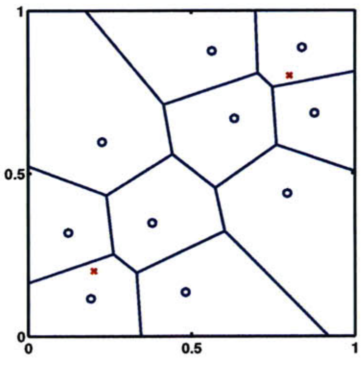
The third scenario shown in Figs. 3-6(e) and 3-6(f) uses the potential field controller from (3.16). This controller uses a sum of delta-Dirac functions for $\phi(q)$ rather than a sum of Gaussians, which causes the robots to arrange themselves in the close-packed lattice pattern.

## 3.6 Synopsis

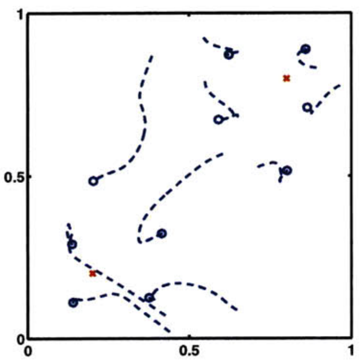
In this chapter we introduced a unifying optimization framework for multi-robot control that brings together several different existing algorithms. We point out that important properties of the underlying objective are embodied in the way sensor information or actuator capabilities are combined from different robots. We propose a parameterized function to accomplish this combination, where different parameter values are shown to lead to different kinds multi-robot algorithms. Finally, we prove that for all multi-robot problems other than consensus, the underlying optimization is necessarily nonconvex, making global optimization an unrealistic objective in general, especially for gradient descent controllers. Looking towards future research, this motivates a search for special classes of nonconvex functions that admit stronger convergence guarantees, and a search for distributed controllers other than those based on gradient methods that may lead to stronger convergence guarantees.



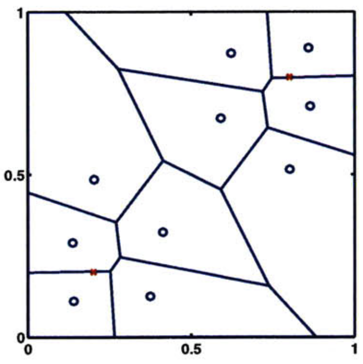
(a) Trajectory Voronoi



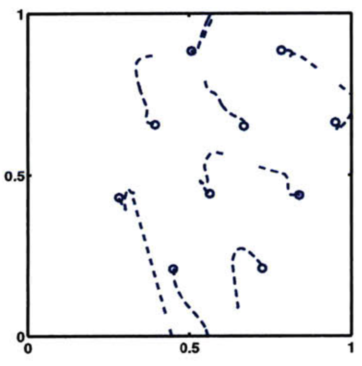
(b) Final Config. Voronoi



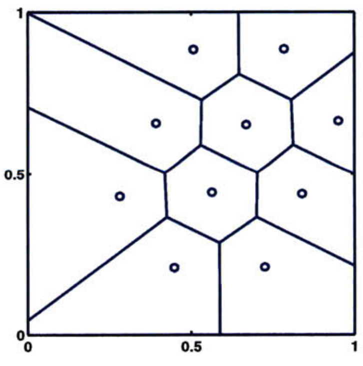
(c) Trajectory $\alpha = -1$



(d) Final Config. $\alpha = -1$



(e) Trajectory Pot. Field



(f) Final Config. Pot. Field

Figure 3-6: Trajectories and final configurations are shown for ten robots using the gradient control law with the Voronoi controller (3-6(a), 3-6(b)), the minimum variance controller (3-6(c), 3-6(d)), and a potential field controller (3-6(e), 3-6(f)). The Voronoi tessellation is shown for all scenarios for comparison, even though the right two controllers do not use the Voronoi cells for control.

# Chapter 4

# Incorporating Learning

## 4.1 Introduction

In this chapter we address the question of how to control the multi-robot system when the sensory function, $\phi(q)$, which represents the weighting of importance over the environment, is not known before hand. We build upon the gradient controller developed in Chapter 3, incorporating a parameter tuning mechanism to learn the sensory function in a provably stable way. The control strategy can be thought of as proceeding simultaneously in two spaces. In the space of robot positions, the robots move to minimize the cost function representing the collective sensing cost of the network. At the same time, in a high-dimensional parameter space, each robot adapts a parameter vector to learn[1] the distribution of sensory information in the environment. We prove that the robots eventually reach a near-optimal configuration, and if their paths are sufficiently rich, they reach an optimal configuration. An overview of the control strategy is shown in Figures 4-1.

We first describe a learning law in which each robot uses only its own sensor measurements. We then include a consensus term in the learning law to couple the learning among neighboring robots. The main effect of this coupling is that sensor measurements from any one robot propagate around the network to be used by all

[1]We will use the words learning and adaptation interchangeably. Learning and adaptation are specifically meant in the sense of parameter tuning, as in adaptive control, rather than the broader meaning often used in Biology and Bio-inspired applications.

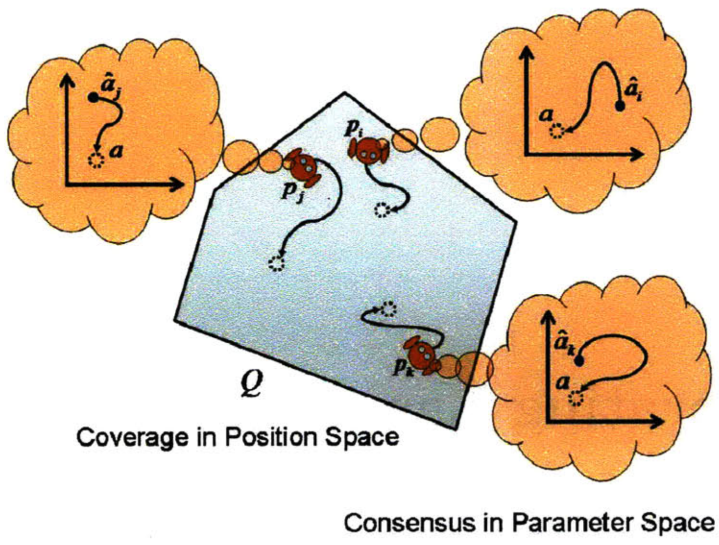


Figure 4-1: An overview of the decentralized control scheme is shown. The robots, at positions $p_i$, $p_j$, and $p_k$, spread out over the area, $Q$, to reach optimal final positions. Simultaneously, each robot adapts a parameter vector ($\hat{a}_i$, $\hat{a}_j$, and $\hat{a}_k$) to build an approximation of the sensory environment. The parameter vectors for neighboring robots are coupled in such a way that their final value, $a$, is the same for all robots in the network.

robots. All robots eventually learn the same function incorporating all the sensor measurements collected by all the robots.

### 4.1.1 Related Work

We use the notion of an optimal sensing configuration developed in [26] and build upon it a parameter adaptation mechanism similar to what is used in adaptive control [67, 89, 103]. Our emphasis in this chapter is on incorporating learning to enable optimal coverage of an unfamiliar environment. This is in contrast to [26] and other papers that use the same optimization framework (e.g. [25, 77, 87]) in which the distribution of sensory information in the environment is required to be known *a priori* by all robots. This *a priori* requirement was first relaxed in [95] by introducing a controller with a simple memoryless approximation from sensor measurements. The controller was demonstrated in hardware experiments, though a stability proof was not found. In the present work we remove this *a priori* requirement by introducing an adaptive controller inspired by the architecture in [88]. The results in this chapter elaborate and improve upon our previous works [99, 100].

It is found that when each robot uses only its own sensor measurements to learn the distribution of sensory information, learning performance can be sluggish. We address this problem by including a consensus term[2] in the parameter adaptation law. Consensus phenomena have been studied in many fields, and appear ubiquitously in biological systems of all scales. However, they have only recently yielded to rigorous mathematical treatment; first in the distributed and parallel computing community [9, 10, 107, 108] in discrete time, and more recently in the controls community in continuous time [13, 27, 46, 71, 111, 112]. In the present work, consensus is used to learn the distribution of sensory information in the environment in a decentralized way by propagating sensor measurements gathered by each robot around the network. This is similar to distributed filtering techniques that have recently been introduced, for example in [62, 117], though in contrast to those works, we are concerned with maintaining provable stability of the combined learning and control system. Consensus improves the quality and speed of learning, which in turn causes the robots to converge more quickly to their optimal positions.

### 4.1.2 Contributions

In short, the main contribution of this chapter is:

1. To provide a controller that uses parameter adaptation to accomplish coverage without *a priori* knowledge of the sensory environment. A consensus term is used within the parameter adaptation law to propagate sensory information among the robots in the network. Using a Lyapunov-like proof, we show that the control law causes the network to converge to a near-optimal sensing configuration, and if the robots' paths are sufficiently rich, the network will converge to an optimal configuration.

[2]The phenomenon of decentralized consensus is known by many names including flocking, herding, swarming, agreement, rendezvous, gossip algorithms, and oscillator synchronization. All of these are, at root, the same phenomenon—convergence of the states of a group of dynamical systems to a common final vector (or manifold) through local coupling.

This chapter is organized as follows. In Section 4.2 we state the main assumptions and definitions to set up the problem. Section 4.3 introduces the learning controller in its simplest form, in which each robot learns an approximation of the sensor function independent from the other robots. The learning controller is refined in Section 4.4 by coupling the learning between neighboring robots using a consensus term in the parameter adaptation law. Section 4.5 generalizes the controller to the broadest context: that of a distributed gradient controller whose cost function has a linearly parameterized gradient. We analyze the convergence rate of the learning law in Section 4.6, and consider several alternative stable learning laws in Section 4.7. Finally, numerical simulations are given in Section 4.8 and a synopsis is provide in Section 4.9.

## 4.2 Problem Formulation

We will use the Voronoi based controller described in Chapter 2.19 as the foundation for building the adaptive architecture and for proving convergence qualities. Recall the Voronoi cost function

$$\mathcal{H}(p_1, \dots, p_n) = \sum_{i=1}^{n} \int_{V_i} \frac{1}{2} \|q - p_i\|^2 \phi(q)\, dq, \tag{4.1}$$

and recall the mass, first moment, and centroid of a Voronoi region $V_i$ are given by

$$M_{V_i} = \textstyle\int_{V_i} \phi(q)\, dq, \quad L_{V_i} = \textstyle\int_{V_i} q\phi(q)\, dq, \quad \text{and } C_{V_i} = L_{V_i}/M_{V_i}, \tag{4.2}$$

respectively. Finally, recall the result proved from Lemma 2.1,

$$\frac{\partial \mathcal{H}}{\partial p_i} = -\int_{V_i} (q - p_i)\phi(q)\, dq = -M_{V_i}(C_{V_i} - p_i). \tag{4.3}$$

Equation (4.3) implies that critical points of $\mathcal{H}$ correspond to the configurations such that $p_i = C_{V_i}\ \forall i$, that is, each agent is located at the centroid of its Voronoi region.

### 4.2.1 Assumptions and Definitions

This brings us to the concept of optimal coverage summarized in the following definition.

**Definition 4.1 (Optimal Coverage Configuration)** *A robot network is said to be in a (locally) optimal coverage configuration if every robot is positioned at the centroid of its Voronoi region, $p_i = C_{V_i}\ \forall i$.*

We emphasize again that global optimization of (4.1) is known to be difficult (NP-hard for a given discrete representation of $\phi(q)$) even in the centralized case with full information. Thus when we refer to an optimal coverage configuration we mean a locally optimal one. Variations on the control law which attempt to find global minima through exploration are discussed in [87, 92].

We also make the assumptions, already discussed in Chapter 2, that the robots have dynamics

$$\dot{p}_i = u_i, \tag{4.4}$$

where $u_i$ is the control input, and that they are able to compute their own Voronoi cell, $V_i = \{q \mid \|q - p_i\| \leq \|q - p_j\|\}$.

More importantly for this chapter, we use a basis function approximation scheme to learn the sensory function $\phi(q)$. Let $\mathcal{K} : Q \mapsto \mathbb{R}^m_{>0}$ be a vector of bounded, continuous basis functions. Each robot has these functions available for computation. The sensory function approximation for robot $i$ is given by $\hat{\phi}_i(q,t) = \mathcal{K}(q)^T \hat{a}_i(t)$, where $\hat{a}_i(t)$ is a parameter vector that is tuned according to an adaptation law which we will describe in Section 4.3. Figure 4-2 shows a graphical representation of this function approximation scheme. For our analysis, we require that the following assumption holds.

**Assumption 4.1 (Matching Conditions)** *There exists and ideal parameter vector $a \in \mathbb{R}^m$ such that*

$$\phi(q) = \mathcal{K}(q)^T a, \tag{4.5}$$

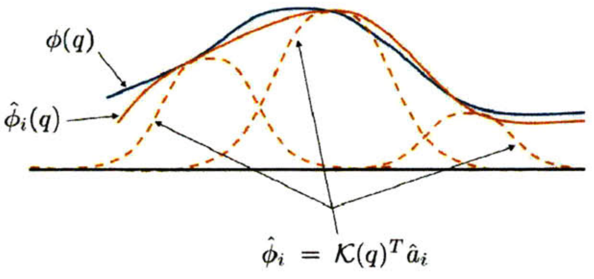


Figure 4-2: The sensory function approximation is illustrated in this simplified 2-D schematic. The true sensory function is represented by $\phi(q)$ and robot $i$'s approximation of the sensory function is $\hat{\phi}_i(q)$. The basis function vector $\mathcal{K}(q)$ is shown as three Gaussians (dashed curves), and the parameter vector $\hat{a}_i$ denotes the weighting of each Gaussian.

*and $a$ is unknown to the robots. Furthermore,*

$$a \geq \mathbf{1}a_{\min} \tag{4.6}$$

*where $a_{\min} \in \mathbb{R}_{>0}$ is a lower bound known by each robot.*

Requirements such as Assumption 4.1 are common for adaptive controllers. In theory, the assumption is not limiting since any function (with some smoothness requirements) over a bounded domain can be approximated arbitrarily well by some set of basis functions [88]. In practice, however, designing a suitable set of basis functions requires application-specific expertise.

There is a variety of basis function families to chose from for $\mathcal{K}(q)$. We use Gaussians in our simulations, but other options include wavelets, sigmoids, and splines. Gaussian basis functions have a computational advantage over non-local basis functions because, in any discrete representation, they have compact support. To compute the value of the network at a location $\hat{\phi}_i(q)$, or to tune the weights of the network $\hat{a}_i$ with new data, one has only to consider Gaussians in a region around the point of interest.

Define the moment approximations using $\hat{\phi}_i(q,t)$ as

$$\hat{M}_{V_i}(t) = \int_{V_i} \hat{\phi}_i(q,t)\, dq, \quad \hat{L}_{V_i}(t) = \int_{V_i} q\hat{\phi}_i(q,t)\, dq, \quad \text{and } \hat{C}_{V_i}(t) = \hat{L}_{V_i}(t)/\hat{M}_{V_i}(t). \tag{4.7}$$

Next, define the parameter error

$$\tilde{a}_i(t) = \hat{a}_i(t) - a, \tag{4.8}$$

and notice the relation

$$\hat{\phi}_i(q,t) - \phi(q) = \mathcal{K}(q)^T \tilde{a}_i(t). \tag{4.9}$$

In order to compress the notation, we introduce the shorthand $\mathcal{K}_i(t) = \mathcal{K}(p_i(t))$ for the value of the basis function vector at the position of robot $i$, and $\phi_i(t) = \phi(p_i(t))$ for the value of the sensory function at the position of robot $i$. As previously stated, robot $i$ can measure $\phi_i$ with its sensors. We will also commonly refer to quantities without explicitly writing their arguments. However, we may include arguments in some instances to avoid ambiguity.

The function approximation framework described above brings us to another concept of optimality for coverage.

**Definition 4.2 (Near-Optimal Coverage Configuration)** *A robot network is said to be in a near-optimal coverage configuration if each robot is positioned at the estimated centroid of its Voronoi region, $p_i = \hat{C}_{V_i}$ $\forall i$.*

Finally, we distinguish between two qualities of function approximations.

**Definition 4.3 (Globally True Approximation)** *A robot is said to have a globally true (or just true) approximation of the sensory function if its approximation is equal to the actual sensory function at every point of its domain, $\hat{\phi}_i(q) = \phi(q)$ $\forall q \in Q$.*

**Definition 4.4 (Locally True Approximation)** *A robot is said to have a locally true approximation of the sensory function over a subset $\Omega \subset Q$ if its approximation is equal to the true function at every point in the subset, $\hat{\phi}_i(q) = \phi(q)$ $\forall q \in \Omega$.*

In light of the above definitions, if the parameter error is zero, $\tilde{a}_i = 0$, then robot $i$ has a *true* approximation of the sensory function. Also, if $\tilde{a}_i = 0$ $\forall i$, then a *near-optimal* coverage configuration is also *optimal*. An overview of the geometrical objects involved in our set-up is shown in Figure 4-3.

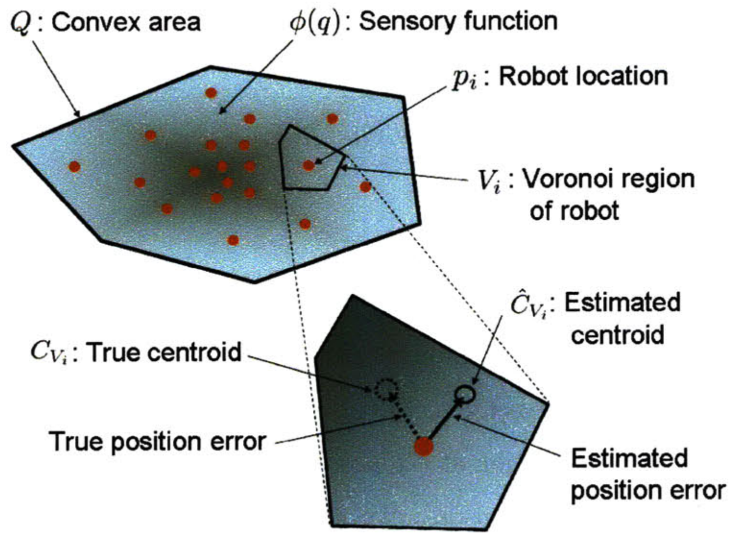


Figure 4-3: A graphical overview of the quantities involved in the controller and environment is shown. The robots move to cover a bounded, convex area $Q$ their positions are $p_i$, and they each have a Voronoi region $V_i$ with a true centroid $C_{V_i}$ and an estimated centroid $\hat{C}_{V_i}$. The true centroid is determined using a sensory function $\phi(q)$, which indicates the relative importance of points $q$ in $Q$. The robots do not know $\phi(q)$, so they calculate an estimated centroid using an approximation $\hat{\phi}_i(q)$ learned from sensor measurements of $\phi(q)$.

## 4.3 Decentralized Adaptive Control Law

We want a controller to drive the robots to an *optimal* configuration, that is, we want to position them at their Voronoi centroids. We emphasize that it is not easy to position a robot at its Voronoi centroid because (1) the robot does not know the sensory function $\phi(q)$ which is required to calculate its centroid, and (2) the centroid moves as a nonlinear function of the robot's position. To overcome the first problem, our controller learns an approximation of the centroid on-line. To overcome the second problem, our controller causes each robot to pursue its estimated centroid. We will prove that the robots achieve a *near-optimal* configuration, and that every robot learns a *locally true* approximation of the sensory function. Furthermore, if a robot's path is sufficiently rich, it achieves a *globally true* approximation, and if every robots' path is sufficiently rich, the robots reach an *optimal* configuration.

We propose to use the control law

$$u_i = K(\hat{C}_{V_i} - p_i), \tag{4.10}$$

where $K$ is a (potentially time-varying) uniformly positive definite control gain matrix, which may have a skew-symmetric component to encourage exploration as in [92]. The area $Q$ is required to be convex so that the control law is feasible, that is, the robots never attempt to cross the boundaries of $Q$. Since $\hat{C}_{V_i} \in V_i \subset Q$ and $p_i \in Q$, by convexity, the segment connecting the two is in $Q$, and the control law is feasible.

The parameters $\hat{a}_i$ used to calculate $\hat{C}_{V_i}$ are adjusted according to a set of adaptation laws which are introduced below. First, we define two quantities,

$$\Lambda_i(t) = \int_0^t \omega(\tau)\mathcal{K}_i(\tau)\mathcal{K}_i(\tau)^T \, d\tau, \quad \text{and} \quad \lambda_i(t) = \int_0^t \omega(\tau)\mathcal{K}_i(\tau)\phi_i(\tau)\, d\tau. \tag{4.11}$$

These can be calculated differentially by robot $i$ using $\dot{\Lambda}_i = \omega(t)\mathcal{K}_i\mathcal{K}_i^T$, and $\dot{\lambda}_i = \omega(t)\mathcal{K}_i\phi_i$, with zero initial conditions. The function $\omega(t) \geq 0$ determines a data collection weighting. We require that it is integrable (belongs to $L^1$), and continuous (belongs to $C^0$). Define another quantity

$$F_i = \frac{\int_{V_i} \mathcal{K}(q)(q-p_i)^T \, dq K \int_{V_i} (q-p_i)\mathcal{K}(q)^T \, dq}{\int_{V_i} \hat{\phi}_i(q)\, dq}. \tag{4.12}$$

Notice that $F_i$ is a positive semi-definite matrix. It can also be computed by robot $i$ as it does not require any knowledge of the true parameter vector, $a$. The adaptation law for $\hat{a}_i$ is now defined as

$$\dot{\hat{a}}_{\text{pre}_i} = -F_i\hat{a}_i - \gamma(\Lambda_i\hat{a}_i - \lambda_i), \tag{4.13}$$

The two terms in (4.13) have an intuitive interpretation. The first term compensates for uncertainty in the centroid position estimate. The second term carries out a gradient descent to minimize the sensory function error $\tilde{\phi}_i(p_i)$ integrated over time. The gradient descent interpretation is explored more in Section 4.7. We stress that a decentralized implementation requires that each robot adapts its own parameter vector using local information available to it. If one were interested, instead, in designing a centralized adaptation law, one could simply use a common parameter vector that is adapted using the information from all robots.

Equation (4.13) is the main adaptation law, however the controller (4.10) has a singularity at $\hat{a}_i = 0$ (since $\hat{M}_{V_i}$ is in the denominator of $\hat{C}_{V_i}$). For this reason we prevent the parameters from dropping below $a_{\min} > 0$ using a parameter projection [101]

$$\dot{\hat{a}}_i = \Gamma(\dot{\hat{a}}_{\mathrm{pre}_i} - I_{\mathrm{proj}_i}\dot{\hat{a}}_{\mathrm{pre}_i}), \tag{4.14}$$

where $\Gamma \in \mathbb{R}^{m \times m}$ is a diagonal, positive definite adaptation gain matrix, and the diagonal matrix $I_{\mathrm{proj}_i}$ is defined element-wise as

$$I_{\mathrm{proj}_i}(j) = \begin{cases} 0 & \text{for } \hat{a}_i(j) > a_{\min} \\ 0 & \text{for } \hat{a}_i(j) = a_{\min} \text{ and } \dot{\hat{a}}_{\mathrm{pre}_i}(j) \geq 0 \\ 1 & \text{otherwise,} \end{cases} \tag{4.15}$$

where $(j)$ denotes the $j^{th}$ element for a vector and the $j^{th}$ diagonal element for a matrix.

The controller described above will be referred to as the basic controller, and its behavior is formalized in the following theorem.

**Theorem 4.1 (Basic Convergence)** *Under Assumption 4.1, the network of robots with dynamics (4.4), control law (4.10), and adaptation law (4.13,4.14) converges to a* near-optimal *coverage configuration. Furthermore, each robot converges to a* locally true *approximation of the sensory function over the set $\Omega_i = \{p_i(\tau) \mid \tau \geq 0, \omega(\tau) > 0\}$, made up of all points on the robot's trajectory with positive weighting.*

**Proof 4.1** *We will define a lower-bounded, Lyapunov-like function and show that its time derivative is non-increasing. This will imply that it reaches a limit. Furthermore, the time derivative is uniformly continuous, so by Barbalat's lemma*[3] *[5, 80] it*

[3]We cannot use the more typical LaSalle invariance theorem because our system is time-varying

*approaches zero. The quantities $\|\hat{C}_{V_i}(t) - p_i(t)\|$ and $\omega(\tau)\tilde{\phi}_i(p_i(\tau), t)^2$, $0 \geq \tau \geq t$, will be included in the time derivative of this function, thereby implying $p_i(t) \rightarrow \hat{C}_{V_i}(t)$, and $\hat{\phi}_i(q, t) \rightarrow \phi(q)$ $\forall q \in \Omega_i$ for all $i$.*

*Define a Lyapunov-like function*

$$\mathcal{V} = \mathcal{H} + \sum_{i=1}^{n} \frac{1}{2}\tilde{a}_i^T \Gamma^{-1} \tilde{a}_i, \tag{4.16}$$

*which incorporates the sensing cost $\mathcal{H}$, and is quadratic in the parameter errors $\tilde{a}_i$. Note that the sensing cost $\mathcal{H}$ is computed with the* actual *sensory function $\phi(q)$, so it inherently incorporates function approximation errors as well. $\mathcal{V}$ is bounded below by zero since $\mathcal{H}$ is a sum of integrals of strictly positive functions, and the quadratic parameter error terms are each bounded below by zero.*

*Taking the time derivative of $\mathcal{V}$ along the trajectories of the system gives*

$$\dot{\mathcal{V}} = \sum_{i=1}^{n} \left[ \frac{\partial \mathcal{H}}{\partial p_i}^T \dot{p}_i + \tilde{a}_i^T \Gamma^{-1} \dot{\tilde{a}}_i \right], \tag{4.17}$$

*and substituting from (4.3) and noticing that $\dot{\tilde{a}}_i = \dot{\hat{a}}_i$ yields*

$$\dot{\mathcal{V}} = \sum_{i=1}^{n} \left[ -\int_{V_i} (q - p_i)^T \phi(q)\, dq \dot{p}_i + \tilde{a}_i^T \Gamma^{-1} \dot{\hat{a}}_i \right]. \tag{4.18}$$

*Using (4.9) to substitute for $\phi(q)$ gives*

$$\dot{\mathcal{V}} = \sum_{i=1}^{n} \left[ -\int_{V_i} (q - p_i)^T \hat{\phi}_i \, dq \dot{p}_i + \int_{V_i} \tilde{a}_i^T \mathcal{K}(q)(q - p_i)^T \, dq \dot{p}_i + \tilde{a}_i^T \Gamma^{-1} \dot{\hat{a}}_i \right].$$

*Substituting for $\dot{p}_i$ with (4.4) and (4.10), and moving $\tilde{a}_i$ out of the second integral*

due to the data weighting function $\omega(t)$.

*(since it is not a function of q) leads to*

$$\dot{\mathcal{V}} = \sum_{i=1}^{n} \Big[ -\hat{M}_{V_i}(\hat{C}_{V_i} - p_i)^T K(\hat{C}_{V_i} - p_i)$$
$$+\tilde{a}_i^T \int_{V_i} \mathcal{K}(q)(q - p_i)^T \, dq(\hat{C}_{V_i} - p_i) + \tilde{a}_i^T \Gamma^{-1} \dot{\hat{a}}_i \Big].$$

*Expanding $(\hat{C}_{V_i} - p_i)$ in the second term, and substituting for $\dot{\hat{a}}_i$ with (4.14) gives*

$$\dot{\mathcal{V}} = \sum_{i=1}^{n} \Big[ -\hat{M}_{V_i}(\hat{C}_{V_i} - p_i)^T K(\hat{C}_{V_i} - p_i) + \tilde{a}_i^T F_i \hat{a}_i - \tilde{a}_i^T F_i \hat{a}_i$$
$$-\tilde{a}_i^T \gamma(\Lambda_i \hat{a}_i - \lambda_i) - \tilde{a}_i^T I_{proj_i} \dot{\hat{a}}_{pre_i} \Big].$$

*Now we can expand $(\Lambda_i \hat{a}_i - \lambda_i)$, noting that $\lambda_i = \int_0^t \omega(\tau)\mathcal{K}_i(\mathcal{K}_i^T a) \, d\tau$, to get*

$$\dot{\mathcal{V}} = -\sum_{i=1}^{n} \Big[ \hat{M}_{V_i}(\hat{C}_{V_i} - p_i)^T K(\hat{C}_{V_i} - p_i)$$
$$+\tilde{a}_i^T \gamma \int_0^t \omega(\tau)\mathcal{K}_i\mathcal{K}_i^T \tilde{a}_i(t) \, d\tau + \tilde{a}_i^T I_{proj_i} \dot{\hat{a}}_{pre_i} \Big],$$

*and, finally, bringing $\tilde{a}_i^T$ inside the integral (it is not a function of $\tau$, though it is a function of $t$) results in*

$$\dot{\mathcal{V}} = -\sum_{i=1}^{n} \Big[ \hat{M}_{V_i}(\hat{C}_{V_i} - p_i)^T K(\hat{C}_{V_i} - p_i)$$
$$+\gamma \int_0^t \omega(\tau)(\mathcal{K}_i(\tau)^T \tilde{a}_i(t))^2 \, d\tau + \tilde{a}_i^T I_{proj_i} \dot{\hat{a}}_{pre_i} \Big], \tag{4.19}$$

*Inside the sum, the first and second terms are clearly non-negative. We focus momentarily on the third term. Expanding it as a sum of scalar terms, we see that the $j^{th}$ scalar term is of the form*

$$\tilde{a}_i(j) I_{proj_i}(j) \dot{\hat{a}}_{pre_i}(j). \tag{4.20}$$

*From (4.15), if $\hat{a}_i(j) > a_{\min}$, or $\hat{a}_i(j) = a_{\min}$ and $\dot{\hat{a}}_{pre_i}(j) \geq 0$, then $I_{proj_i}(j) = 0$*

*and the term vanishes. Now, in the case $\hat{a}_i(j) = a_{\min}$ and $\dot{\hat{a}}_{pre_i}(j) < 0$, we have $\tilde{a}_i(j) = \hat{a}_i(j) - a(j) \leq 0$ (from Assumption 4.1). Furthermore, $I_{proj_i}(j) = 1$ and $\dot{\hat{a}}_{pre_i}(j) < 0$ implies that the term is non-negative. In all cases, then, each term of the form (4.20) is non-negative, and all three terms inside the sum in (4.19) are non-negative. Thus $\dot{\mathcal{V}} \leq 0$.*

*We have that $\mathcal{V}$ is lower bounded and $\dot{\mathcal{V}} \leq 0$, so $\mathcal{V}$ approaches a limit. We establish the uniform continuity of $\dot{\mathcal{V}}$ in Lemma A.1 in Appendix A, so by Barbalat's lemma $\lim_{t\to\infty} \dot{\mathcal{V}} = 0$. From (4.19), this implies $\lim_{t\to\infty} \|p_i(t) - \hat{C}_{V_i}(t)\| = 0\ \forall i$ from the first term in the sum, so the network converges to a* near-optimal *coverage configuration.*

*Furthermore, from $\mathcal{K}_i(\tau)^T \tilde{a}_i(t) = \hat{\phi}_i(p_i(\tau), t) - \phi(p_i(\tau))$, we have from the second term of(4.19)*

$$\lim_{t\to\infty} \int_0^t \omega(\tau)(\hat{\phi}_i(p_i(\tau), t) - \phi(p_i(\tau)))^2\, d\tau = 0 \quad \forall i = 1, \ldots, n. \tag{4.21}$$

*Now notice that the integrand in (4.21) is non-negative, therefore it must converge to zero for all $\tau$ except on a set of Lesbegue measure zero. Suppose the integrand is greater than zero at some point $\tau$. The integrand is continuous (since $\mathcal{K}_i(t)$, $\hat{a}_i(t)$, and $\phi_i(t)$ are), so if it is greater than zero at $\tau$, it is greater than zero in a neighborhood of non-zero measure around it, $(\tau - \epsilon, \tau + \epsilon)$, for some $\epsilon > 0$, which is a contradiction. Thus, we have $\hat{\phi}_i(q, t) \to \phi(q)\ \forall q \in \Omega_i$ and $\forall i$.*

In [92] the following extension to the above theorem was derived. We restate it here to give a more thorough characterization the controller's behavior.

**Corollary 4.1 (Sufficient Richness for Basic Controller)** *In addition to the conditions for Theorem 4.1, if the robots' paths are such that the matrix $\lim_{t\to\infty} \Lambda_i(t)$ is positive definite $\forall i$, the network converges to an* optimal *coverage configuration, and each robot converges to a* globally true *approximation of the sensory function, $\phi(q)$.*

**Proof 4.2** *Consider the second term in (4.19). Move the two $\tilde{a}_i(t)$ outside of the* *integral (since they are not a function of $\tau$) to get*

$$\gamma\tilde{a}_i(t)^T \left[\int_0^t \omega(\tau)\mathcal{K}_i\mathcal{K}_i^T \, d\tau\right] \tilde{a}_i(t) = \gamma\tilde{a}_i(t)^T\Lambda_i(t)\tilde{a}_i(t).$$

*Since $\dot{\mathcal{V}} \to 0$, if $\lim_{t\to\infty}\Lambda_i(t)$ is positive definite (we know the limit exists because $\mathcal{K}(q)$ is bounded and $\omega(t) \in L^1$), then $\tilde{a}_i(t) \to 0$. This implies that robot $i$ converges to a* globally true *approximation of the sensory function, $\phi(q)$. Furthermore, if $\lim_{t\to\infty}\Lambda_i(t) > 0 \ \forall i$, then $\hat{C}_{V_i} = C_{V_i} \ \forall i$, so the network converges to an* optimal *coverage configuration.*

**Remark 4.1** *One may wonder how the controller will behave if Assumption 4.1 fails, so that there is no ideal parameter vector $a$ that will* exactly *reconstruct $\phi(q)$ from the basis functions. Indeed, this will be the case in any real-world scenario. Such a question requires a robustness analysis that is beyond the scope of this thesis, but analyses of robustness for centralized adaptive controllers can be found, for example, in [88] and most texts on adaptive control (e.g. [67,89,103]). It is observed in numerical simulations that the adaptation law finds a parameter to make $\hat{\phi}_i(q)$ as close as possible to $\phi(q)$, where closeness is measured by the integral of the squared difference, as described in Section 4.7.*

**Remark 4.2** *One may also wonder how the controller behaves with time varying sensory functions $\phi(q,t)$. It can be expected from existing results for centralized adaptive controllers, that our controller will track sensory functions that change slowly with respect to the rate of adaptation of the parameters. The ability to track a time varying sensory function can be enhanced by using a forgetting factor in the data weighting function $\omega(t)$ as described in Section 4.7.3.*

## 4.4 Parameter Consensus

In this section we use the properties of graph Laplacians from Section 2.3.2 to prove convergence and consensus of a modified adaptive control law. The controller from

(4.3) is modified so that the adaptation laws among Voronoi neighbors are coupled with a weighting proportional to the length of their shared Voronoi edge. Adaptation and consensus were also combined in [68] and [112], however in those works consensus was used to align the velocities of agents, not to help in the parameter adaptation process itself. Our use of consensus is more related to the recent algorithms for distributed filtering described in [62] and [117].

### 4.4.1 Consensus Learning Law

We add a term to the parameter adaptation law in (4.13) to couple the adaptation of parameters between neighboring agents. Let the new adaptation law be given by

$$\dot{\hat{a}}_{\text{pre}_i} = -F_i\hat{a}_i - \gamma\left(\Lambda_i\hat{a}_i - \lambda_i\right) - \zeta\sum_{j=1}^{n} w_{ij}(\hat{a}_i - \hat{a}_j), \tag{4.22}$$

where $w_{ij}$ is a weighting over the Delaunay graph (see Section 2.3.2) between two robots $i$ and $j$ and $\zeta \in \mathbb{R}$, $\zeta > 0$, is a positive gain. The projection remains the same as in (4.14), namely

$$\dot{\hat{a}}_i = \Gamma(\dot{\hat{a}}_{\text{pre}_i} - I_{\text{proj}_i}\dot{\hat{a}}_{\text{pre}_i}).$$

A number of different weightings $w_{ij}$ are conceivable, but here we propose that $w_{ij}$ be equal to the length (area for $N = 3$, or volume for $N > 3$) of the shared Voronoi edge of robots $i$ and $j$,

$$w_{ij} = \int_{V_i \cap V_j} dq. \tag{4.23}$$

Notice that $w_{ij} \geq 0$ and $w_{ij} = 0$ if and only if $i$ and $j$ are not Voronoi neighbors, so $w_{ij}$ is a valid weighting over the Delaunay communication graph as described in Section 2.3.2. This weighting is natural since one would want a robot to be influenced by its neighbor in proportion to its neighbor's proximity. This form of $w_{ij}$ will also provide for a simple analysis since it maintains the continuity of the right hand side of (4.22), which is required for using Barbalat's lemma.

**Theorem 4.2 (Convergence with Parameter Consensus)** *Under the conditions of Theorem 4.1, using the parameter adaptation law (4.22), the network of robots converge to a* near-optimal *coverage configuration. Furthermore, each robot converges to a* locally true *approximation of the sensory function over the set all points on every robot's trajectory with positive weighting, $\Omega = \cup_{j=1}^{n} \Omega_j$. Additionally,*

$$\lim_{t\to\infty} (\hat{a}_i - \hat{a}_j) = 0 \quad \forall i, j \in \{1, \ldots, n\}. \tag{4.24}$$

**Proof 4.3** *We will use the same method as in the proof of Theorem 4.1, adding the extra term for parameter coupling. It will be shown that this term is non-positive. The claims of the proof follow as before from Barbalat's lemma.*

*Define $\mathcal{V}$ to be (4.16), which leads to*

$$\dot{\mathcal{V}} = -\sum_{i=1}^{n} \Big[ \hat{M}_{V_i} (\hat{C}_{V_i} - p_i)^T K (\hat{C}_{V_i} - p_i) + \gamma \int_0^t \omega(\tau) (\mathcal{K}_i(\tau)^T \tilde{a}_i(t))^2 \, d\tau \\ + \tilde{a}_i^T I_{proj_i} \dot{\hat{a}}_{pre_i} \Big] - \sum_{i=1}^{n} \tilde{a}_i^T \zeta \sum_{j=1}^{n} w_{ij} (\hat{a}_i - \hat{a}_j). \tag{4.25}$$

*We have already shown that the three terms inside the first sum are non-negative. Now consider the parameter coupling term. We can rewrite this term using the graph Laplacian defined in Section 2.3.2 as*

$$\sum_{i=1}^{n} \tilde{a}_i^T \zeta \sum_{j=1}^{n} w_{ij} (\hat{a}_i - \hat{a}_j) = \zeta \sum_{j=1}^{m} \tilde{\alpha}_j^T L \hat{\alpha}_j, \tag{4.26}$$

*where $\alpha_j = a(j)\mathbf{1}$, $\hat{\alpha}_j = [\hat{a}_1(j) \ \cdots \ \hat{a}_n(j)]^T$, and $\tilde{\alpha}_j = \hat{\alpha}_j - \alpha_j$. Recall the ideal parameter vector $a = [a(1) \ \cdots \ a(j) \ \cdots \ a(m)]^T$, and the parameter estimate for each agent*
$\hat{a}_i = [\hat{a}_i(1) \ \cdots \ \hat{a}_i(j) \ \cdots \ \hat{a}_i(m)]^T$. *We have simply regrouped the parameters by introducing the $\alpha_j$ notation. From Section 2.3.2 we saw that $\alpha_j^T L = a(j)\mathbf{1}^T L = 0$.*

*This gives*

$$\zeta \sum_{j=1}^{m} \tilde{\alpha}_j^T L \hat{\alpha}_j = \zeta \sum_{j=1}^{m} \hat{\alpha}_j^T L \hat{\alpha}_j \geq 0, \tag{4.27}$$

*since* $L \geq 0$. *Thus* $\dot{\mathcal{V}} \leq 0$.

*Lemma A.2 establishes the uniform continuity of* $\dot{\mathcal{V}}$ *for this controller. We can therefore use Barbalat's lemma to conclude that* $\dot{\mathcal{V}} \to 0$. *As before this implies the two claims of Theorem 4.1. Since the graph Laplacian is positive semi-definite, and* $\hat{a}_i(j) \geq a_{\min}$, $\lim_{t\to\infty} \hat{\alpha}_j^T L \hat{\alpha}_j = 0 \Rightarrow \lim_{t\to\infty} \hat{\alpha}_j = a_{final}(j)\mathbf{1}\ \forall j \in \{1, \ldots, m\}$, *where* $a_{final} \in \mathbb{R}^m$ *is some undetermined vector, which is the common final value of the parameters for all of the agents. The consensus assertion (4.24) follows.*

*Finally, recall the fact that for robot* $j$, $\hat{\phi}_j(q) \to \phi(q)$ *over* $\Omega_j$, *but* $\hat{a}_i \to \hat{a}_j$, *therefore* $\hat{\phi}_i(q) \to \phi(q)$ *over* $\Omega_j$. *This is true for all robots* $i$ *and* $j$, *therefore* $\hat{\phi}_i \to \phi(q)$ *over* $\Omega = \cup_{j=1}^{n} \Omega_j$ *for all* $i$.

**Corollary 4.2 (Sufficient Richness for Consensus Controller)** *In addition to the conditions for Theorem 4.2, if the robots' paths are such that* $\int_\Omega \mathcal{K}(q)\mathcal{K}(q)^T dq$ *is positive definite, the network converges to an* optimal *coverage configuration, and each robot converges to a* globally true *approximation of the sensory function,* $\phi(q)$.

**Proof 4.4** *Since* $\hat{\phi}_i(q,t) \to \phi(q)$ *over* $\Omega$, *we have* $\tilde{a}_i(\infty)^T \mathcal{K}(q)\mathcal{K}(q)^T \tilde{a}_i(\infty) = 0$ *over* $\Omega$, *where* $\tilde{a}_i(\infty)$ *is shorthand for* $\lim_{t\to\infty} \tilde{a}_i(t)$. *Then*

$$0 = \int_\Omega \tilde{a}_i(\infty)^T \mathcal{K}(q)\mathcal{K}(q)^T \tilde{a}_i(\infty) dq = \tilde{a}_i(\infty)^T \int_\Omega \mathcal{K}(q)\mathcal{K}(q)^T dq\, \tilde{a}_i(\infty) \tag{4.28}$$

*Therefore if* $\int_\Omega \mathcal{K}(q)\mathcal{K}(q)^T dq > 0$, *then* $\tilde{a}_i(\infty) = 0$. *This is true for all* $i$.

**Remark 4.3** *The condition of Corollary 4.2 is less strict than that of Corollary 4.1 because only the union of all the robots' paths has to be sufficiently rich, not each path individually. This means it is easier to achieve an* optimal *configuration with the consensus controller.*

**Remark 4.4** *Another commonly used weighting for algorithms over communication graphs is*

$$w_{ij} = \begin{cases} 1 & \textit{for } j \in \mathcal{N}_i \\ 0 & \textit{for } j \notin \mathcal{N}_i, \end{cases}$$

*where $\mathcal{N}_i$ is the set of indices of neighbors of $i$, as was proposed in [100]. In this case, stability can be proved, but with considerable complication in the analysis, since $\dot{\mathcal{V}}$ is not continuous. Even so, recent extensions of Barbalat's lemma to differential inclusions from [61, 86] (and applied to flocking systems in [104]) can be used to prove the same result as in Theorem 4.2.*

**Remark 4.5** *Introducing parameter coupling increases parameter convergence rates and makes the controller equations better conditioned for numerical integration, as will be discussed in Section 4.8. However there is a cost in increased communication overhead. In a discrete-time implementation of the controller in which parameters and robot positions are represented finitely with $b$ bits, a robot will have to transmit $(m+2)b$ bits and receive $|\mathcal{N}_i|(m+2)b$ bits per time step. While for the basic controller, each robot must transmit $2b$ and receive $2|\mathcal{N}_i|b$ bits per time step. This may or may not represent a significant communication overhead, depending upon $b$ and the speed of the control loop. In hardware experiments we have found this to be a negligible communication cost. Note that although discretization is necessary for a practical implementation, it does not affect the essential phenomenon of consensus, as shown in [33, 48].*

## 4.5 Adaptive Gradient Controller

The parameter adaptation architecture developed thus far for the Voronoi controller can be analogously constructed for any distributed gradient controller with an unknown cost function if the gradient of that cost function can be linearly parameterized. Specifically, the distributed gradient controllers from Chapter 3 can all support adaptation in the same way as the Voronoi controller. In this section we summarize the most general case, which can then be specialized to a specific gradient controller. As before, let $\mathcal{H}(P)$ be the unknown cost of a network of robots and let its gradient be linearly parameterized by

$$\left.\frac{\partial \mathcal{H}}{\partial p_i}\right|_P = \kappa_i(P)^T a, \tag{4.29}$$

where $\kappa_i : \mathcal{P}^n \mapsto \mathbb{R}^{m \times d}$ is known to agent $p_i$, but $a$ is unknown. Also, suppose each robot has a sensor with which it can measure $\partial \mathcal{H}/\partial p_i \mid_{P(t)}$ at the current robot configuration $P(t)$, and let the robots communicate over a communication graph with a weighting function $w_{ij}$. The Voronoi controller fits this description with

$$\kappa_i(P) = -\int_{V_i(P)} \mathcal{K}(q)(q - p_i)^T \, dq. \tag{4.30}$$

Let the robots' dynamics be given by

$$\dot{p}_i = -K\kappa_i(P)^T \hat{a}_i, \tag{4.31}$$

and the adaptation of robot $i$'s estimated parameters be given by

$$\dot{\hat{a}}_i = -\Gamma\left(\kappa_i(P)K\kappa_i(P)^T\hat{a}_i + \gamma(\Lambda_i\hat{a}_i + \lambda_i) + \zeta\sum_{j=1}^{n} w_{ij}(\hat{a}_i - \hat{a}_j)\right), \tag{4.32}$$

where

$$\Lambda_i(t) = \int_0^t \omega(\tau)\kappa_i(P(\tau))\kappa_i(P(\tau))^T \, d\tau \tag{4.33}$$

and

$$\lambda_i(t) = \int_0^t \omega(\tau) \left.\frac{\partial \mathcal{H}}{\partial p_i}\right|_{P(\tau)} d\tau. \tag{4.34}$$

To be precise, $\Lambda_i$ and $\lambda_i$ as defined here are slightly different from their definition for the Voronoi controller. Convergence and consensus results analogous to Theorem 4.2

and Corollary 4.2 follow directly for this controller using the same proof arguments.

## 4.6 Parameter Convergence Analysis

As a separate matter from the asymptotic convergence in Theorem 4.1 and Theorem 4.2, one may wonder *how quickly* parameters converge to their final values. In this section we show that parameter convergence is not exponential, though given sufficiently rich trajectories it can be shown to converge exponentially to an arbitrarily small error. The rate of this convergence is shown to be faster for the controller with parameter consensus than for the basic controller. We neglect the projection operation, as the non-smooth switching considerably complicates the convergence analysis.

From (4.13) and (4.14), neglecting the projection, but including the adaptation gain matrix $\Gamma$, we have

$$\dot{\hat{a}}_i = -\Gamma(F_i\hat{a}_i + \gamma(\Lambda_i\hat{a}_i - \lambda_i)), \tag{4.35}$$

which can be written as

$$\dot{\tilde{a}}_i = -\Gamma\gamma\Lambda_i(t)\tilde{a}_i - \Gamma F_i\hat{a}_i, \tag{4.36}$$

leading to

$$\frac{d}{dt}\|\tilde{a}_i\| = -\frac{\gamma\tilde{a}_i^T\Gamma\Lambda_i(t)\tilde{a}_i}{\|\tilde{a}_i\|} - \frac{\tilde{a}_i^T\Gamma F_i\hat{a}_i}{\|\tilde{a}_i\|}. \tag{4.37}$$

Let $\lambda_{\min_i}(t) \geq 0$ be the minimum eigenvalue of $\Gamma\Lambda_i(t)$ (we know it is real-valued and non-negative since $\Lambda_i(t)$ is symmetric positive semi-definite). Then we have

$$\frac{d}{dt}\|\tilde{a}_i\| \leq -\gamma\lambda_{\min_i}(t)\|\tilde{a}_i\| + \|\Gamma F_i\hat{a}_i\|. \tag{4.38}$$

Now consider the signal $\|\Gamma F_i\hat{a}_i\|$. We proved in Theorem 4.1 that $\|\hat{C}_{V_i} - p_i\| \to 0$ and all other quantities in $\Gamma F_i\hat{a}_i$ are bounded for all $i$, therefore $\|\Gamma F_i\hat{a}_i\| \to 0$. Also,

$\lambda_{\min_i}(0) = 0$, and $\lambda_{\min_i}(t)$ is a nondecreasing function of time. Suppose at some time $T$, robot $i$ has a sufficiently rich trajectory (so that $\Lambda_i(T)$ is positive definite, as in Corollary 4.1), then $\lambda_{\min_i}(t) > \lambda_{\min_i}(T) > 0 \; \forall t \geq T$. Then from (4.38), $\|\tilde{a}_i\|$ will decay faster than an exponentially stable first order system driven by $\|\Gamma F_i \hat{a}_i\|$. Finally, the gains $\Gamma$ and $\gamma$ can be set so that $\|\Gamma F_i \hat{a}_i\|$ is arbitrarily small compared to $\gamma \lambda_{\min_i}$ without affecting stability. Thus, if the robot's trajectory is sufficiently rich, exponentially fast convergence to an arbitrarily small parameter error can be achieved.

Now we consider a similar rate analysis for the controller with parameter consensus. In this case, because the parameters are coupled among robots, we must consider the evolution of all the robots' parameters together. Let

$$\tilde{A} = [\tilde{a}_1^T \quad \cdots \quad \tilde{a}_n^T]^T. \tag{4.39}$$

be a concatenated vector consisting of all the robots' parameter errors. Also, define the block diagonal matrices $F = \mathrm{diag}_{i=1}^n(\Gamma F_i)$, $\Lambda = \mathrm{diag}_{i=1}^n(\Gamma \Lambda_i)$, and the generalized graph Laplacian matrix

$$\mathcal{L} = \begin{bmatrix} \Gamma(1)L(1,1)I_m & \cdots & L(1,n)I_m \\ \vdots & \ddots & \vdots \\ L(n,1)I_m & \cdots & \Gamma(n)L(n,n)I_m \end{bmatrix}. \tag{4.40}$$

The eigenvalues of $\mathcal{L}$ are the same as those of $\Gamma L$, but each eigenvalue has multiplicity $m$. As for a typical graph Laplacian, $\mathcal{L}$ is positive semi-definite. The coupled dynamics of the parameters over the network can be written

$$\dot{\tilde{A}} = -(\gamma \Lambda + \zeta \mathcal{L})\tilde{A} - F\hat{A}, \tag{4.41}$$

with $\hat{A}$ defined in the obvious way. Notice the similarity in form between (4.36) and

(4.41). Following the same type of derivation as before we find

$$\frac{d}{dt}\|\tilde{A}\| \leq -\lambda_{\min}(t)\|\tilde{A}\| + \|F\hat{A}\|, \tag{4.42}$$

where $\lambda_{\min}(t) \geq 0$ is the minimum eigenvalue of $\gamma\Lambda(t)+\zeta\mathcal{L}(t)$. Again, it is real-valued and non-negative since $\gamma\Lambda(t) + \zeta\mathcal{L}(t)$ is symmetric positive semi-definite.

As before, the signal $\|F\hat{A}\| \to 0$. If after some time $T$, $\text{mineig}(\Lambda(T)) > 0$ then $\lambda_{\min}(t) \geq \text{mineig}(\Lambda(t)) > 0 \ \forall t \geq T$ and the network's trajectory is sufficiently rich. Then from (4.37), $\|\tilde{A}\|$ will decay at least as fast as an exponentially stable first order system driven by $\|F\hat{A}\|$. Finally, the gains $\Gamma$, $\gamma$, and $\zeta$ can be set so that $\|F\hat{A}\|$ is arbitrarily small compared to $\gamma\Lambda(t) + \zeta\mathcal{L}(t)$ without affecting stability. Thus, if the robot network's trajectory is sufficiently rich, exponentially fast convergence to an arbitrarily small parameter error can be achieved for the whole network.

To compare with the performance of the basic controller consider that $\gamma\Lambda(t) \leq \gamma\Lambda(t) + \zeta\mathcal{L}(t)$. Therefore the minimum eigenvalue for the consensus controller is always at least as large as that for the basic controller implying convergence is at least as fast. In practice, as we will see in Section 4.8, parameter convergence is orders of magnitude faster for the consensus controller.

## 4.7 Alternative Learning Laws

The adaptation law for parameter tuning (4.13) can be written more generally as

$$\dot{\hat{a}}_i = -F_i\hat{a}_i + f_i(p_i, V_i, \hat{a}_i, t), \tag{4.43}$$

where we have dropped the projection operation for clarity. There is considerable freedom in choosing the learning function $f_i(\cdot)$. We are constrained only by our ability to find a suitable Lyapunov-like function to accommodate Barbalat's lemma.

### 4.7.1 Gradient Laws

The form of $f_i(\cdot)$ chosen in Section 4.3 can be called a gradient law, since

$$f_i = -\frac{\partial}{\partial \hat{a}_i}\left[\frac{1}{2}\gamma \int_0^t \omega(\tau)(\hat{\phi}_i - \phi_i)^2\, d\tau\right]. \tag{4.44}$$

The parameter vector follows the negative gradient of the Least Squares cost function, seeking a minimum.

Another possible learning law is to follow the gradient, given by

$$\begin{aligned} f_i &= -\frac{\partial}{\partial \hat{a}_i}\left[\frac{1}{2}\gamma\omega(\tau)(\hat{\phi}_i - \phi_i)^2\right] \\ &= -\gamma\omega(t)\mathcal{K}_i(\mathcal{K}_i^T\hat{a}_i - \phi_i). \end{aligned} \tag{4.45}$$

Using the same Lyapunov function as before, it can be verified that this learning law results in a *near-optimal* coverage configuration.

These two gradient laws can be combined to give

$$f_i = -\gamma\left[\omega(t)\mathcal{K}_i(\mathcal{K}_i^T\hat{a}_i - \phi_i) + (\Lambda_i\hat{a}_i - \lambda_i)\right], \tag{4.46}$$

which is, in fact, equivalent to the first law with a weighting function $w_c(t,\tau) = \delta(t-\tau)\omega(t) + \omega(\tau)$, where $\delta(t-\tau)$ is the delta-Dirac function (we can make $\omega(\cdot)$ a function of $t$, and $\tau$ with minimal consequences to the convergence proof). The same Lyapunov-like function can be used, such that the resulting time derivative is

$$\begin{aligned} \dot{\mathcal{V}} = -\sum_{i=1}^{n}\Big[\hat{M}_{V_i}(\hat{C}_{V_i} - p_i)^T K(\hat{C}_{V_i} - p_i) + \tilde{a}_i^T I_{\mathrm{proj}_i}\dot{\hat{a}}_{\mathrm{pre}_i} + \\ \gamma\tilde{a}_i^T\left[\omega(t)\mathcal{K}_i\mathcal{K}_i^T + \Lambda_i\right]\tilde{a}_i\Big], \end{aligned}$$

leading to the same convergence claims as in Theorem 4.1 and Corollary 4.1.

### 4.7.2 Recursive Least Squares Laws

Another interesting possibility for a learning law is the continuous-time Recursive Least Squares method. This law can be interpreted as continuously solving the Least Squares minimization problem recursively as new data is acquired. Let

$$J = \frac{1}{2}\int_0^t \omega(\tau)(\hat{\phi}_i - \phi_i)^2 \, d\tau \tag{4.47}$$

be the standard Least Squares cost function with a data weighting function $\omega(\tau)$. Then, taking the gradient with respect to $\hat{a}_i$ and setting to zero we find

$$\Lambda_i(t)\hat{a}_i = \lambda_i(t). \tag{4.48}$$

If the matrix $\Lambda_i(t)$ is full rank, we can pre-multiply both sides by its inverse to solve the Least Squares problem. However, we seek a recursive expression, so taking the time derivative we obtain

$$\dot{\hat{a}}_i = -P_i(t)\omega(t)\mathcal{K}_i(\mathcal{K}_i^T\hat{a}_i - \phi_i), \text{ where } P_i(t) = \Lambda_i(t)^{-1}. \tag{4.49}$$

Using an identity from vector calculus, $P_i$ can be computed differentially by $\dot{P}_i = -P_i\omega(t)\mathcal{K}_i\mathcal{K}_i^T P_i$, but the initial conditions are ill defined. Instead, we must use some nonzero initial condition, $P_{i0}$, with the differential equation $\dot{P}_i = -P_i\omega(t)\mathcal{K}_i\mathcal{K}_i^T P_i$, to give the approximation

$$P_i = \Lambda_i^{-1} + P_{i0}. \tag{4.50}$$

The initial condition can be interpreted as the inverse covariance of our prior knowledge of the parameter values. We should choose this to be small if we have no idea of the ideal parameter values when setting initial conditions.

Before we can apply the Recursive Least Squares law to our controller, there is one additional complication that must be dealt with. We can no longer use the same projection operator to prevent the singularity when $\hat{M}_{V_i} = 0$. However, it is possible to formulate a different stable controller that eliminates this singularity altogether. This formulation also has the advantage that it no longer requires $a(j) > a_{\min}\ \forall j$ in Assumption 4.1. We can use the controller

$$u_i = K(\hat{L}_{V_i} - \hat{M}_{V_i} p_i), \tag{4.51}$$

with the adaptation law

$$\dot{\hat{a}}_i = -P_i \left[ \hat{M}_{V_i} F_i \hat{a}_i + \omega(t) \mathcal{K}_i (\mathcal{K}_i^T \hat{a}_i - \phi_i) \right] \tag{4.52}$$

to approximate the Recursive Least Squares law. Asymptotic convergence can be proven for this case by using the Lyapunov function

$$\mathcal{V} = \mathcal{H} + \sum_{i=1}^{n} \frac{1}{2} \tilde{a}_i^T P_i^{-1} \tilde{a}_i, \tag{4.53}$$

which leads to

$$\dot{\mathcal{V}} = -\sum_{i=1}^{n} \left[ k \hat{M}_{V_i}^2 (\hat{C}_{V_i} - p_i)^T K (\hat{C}_{V_i} - p_i) + \frac{1}{2} \tilde{a}_i^T \left[ \omega(t) \mathcal{K}_i \mathcal{K}_i^T \right] \tilde{a}_i \right], \tag{4.54}$$

Note that the only difference in the Lyapunov function is that $\Gamma$ has been replaced with the time-varying quantity $P_i$.

We can also formulate a learning law analogous to the combined gradient law (4.46) as

$$\dot{\hat{a}}_i = -P_i \left( \hat{M}_{V_i} F_i \hat{a}_i + \frac{1}{2} \omega(t) \mathcal{K}_i (\mathcal{K}_i^T \hat{a}_i - \phi_i) + (\Lambda_i \hat{a}_i - \lambda_i) \right), \tag{4.55}$$

with $\Lambda_i$ and $\lambda_i$ defined as before. The same Lyapunov function can be used (4.53), resulting in

$$\dot{\mathcal{V}} = -\sum_{i=1}^{n} \left[ k \hat{M}_{V_i}^2 (\hat{C}_{V_i} - p_i)^T K (\hat{C}_{V_i} - p_i) + \tilde{a}_i^T \Lambda_i \tilde{a}_i \right].$$

Interestingly, the integral terms (those involving $\Lambda_i$ and $\lambda_i$) of the learning law in (4.55) have a gradient interpretation. Taking just those terms we have

$$\begin{aligned} f_i &= -P_i(\Lambda_i \hat{a}_i - \lambda_i) \\ &= -\tilde{a}_i + P_{i0}\Lambda_i\tilde{a}_i \\ &= -\frac{\partial}{\partial \hat{a}_i}\left(\frac{1}{2}\tilde{a}_i^T\tilde{a}_i\right) + P_{i0}\Lambda_i\tilde{a}_i, \end{aligned} \tag{4.56}$$

so the law approximates the gradient of the squared parameter error. The last term on the right hand side arises from the mismatch in initial conditions between $P_i$ and $\Lambda_i$.

The combination of Least Squares and gradient learning apparent in this law is quite similar to the Composite Adaptation described in [102, 103]. In fact, if one identifies the prediction error as $\mathcal{K}_i^T\hat{a}_i - \phi_i$ and the tracking error as $\Lambda_i - \lambda_i\phi_i$ we have composite adaptation (except, of course, for the term containing $F_i$, which is required for the stability proof).

Unfortunately, it is found that the equations resulting from the Least Squares formulation are difficult to solve numerically, often causing robots to jump outside of the area $Q$, which then corrupts the Voronoi calculation. Alleviating this problem is a matter of ongoing research.

## 4.7.3 Data Weighting Functions

The form of the function $\omega(\cdot)$ can be designed to encourage parameter convergence. One obvious choice is to make $\omega(\tau)$ a square wave, such that data is not incorporated into $\int_0^t \omega(\tau)\mathcal{K}_i\mathcal{K}_i^T\, d\tau$ after some fixed time. This can be generalized to an exponential decay, $\omega(\tau) = \exp(-\tau)$, or a decaying sigmoid $\omega(\tau) = 1/2(\text{erf}(c-t)+1)$. Many other options exist.

One intuitive option for $\omega(\cdot)$ is $\omega(\tau) = \|\dot{p}_i\|^2$, since the rate at which new data is collected is directly dependent upon the rate of travel of the robot. This weighting, in a sense, normalizes the effects of the rate of travel so that all new data is incorporated with equal weighting. Likewise, when the robot comes to a stop, the value of $\phi(p_i)$ at the stopped position does not overwhelm the learning law. This seems to make good sense, but there is an analytical technicality: to ensure that $\Lambda_i$ and $\lambda_i$ remain bounded we have to prove that $\dot{p}_i \in L^2$. In practice, we can set $\omega(\tau) = \|\dot{p}_i\|^2$ up to some fixed time, after which it is zero.

We can also set $\omega(t, \tau) = \exp\{-(t - \tau)\}$, which turns the integrators $\Lambda_i$, $P_i$, and $\lambda_i$ into first order systems. This essentially introduces a forgetting factor into the learning law which has the advantage of being able to track slowly varying sensory distributions. Forgetting factors can have other significant benefits such as improving parameter convergence rates and allowing the flexibility to reject certain frequencies of noise in the error signal. A thorough discussion of forgetting factors can be found in [103], Section 8.7.

## 4.8 Numerical Simulations

Simulations were carried out in a Matlab environment. The dynamics in (4.4) with the control law in (4.10), and the adaptation laws in (4.14) (with (4.13) for the basic controller and (4.22) for the consensus controller) for a group of $n = 20$ robots were integrated forward in time. A numerical solver with a fixed-time-step of .01s was used to integrate the equations. The area $Q$ was taken to be the unit square. The sensory function, $\phi(q)$, was parameterized as a linear combination of nine Gaussians. In particular, for $\mathcal{K} = [\ \mathcal{K}(1) \ \cdots \ \mathcal{K}(9)\ ]^T$, each component, $\mathcal{K}(j)$, was implemented as

$$\mathcal{K}(j) = \frac{1}{2\pi\sigma_j^2} \exp -\frac{(q - \mu_j)^2}{2\sigma_j^2}, \tag{4.57}$$

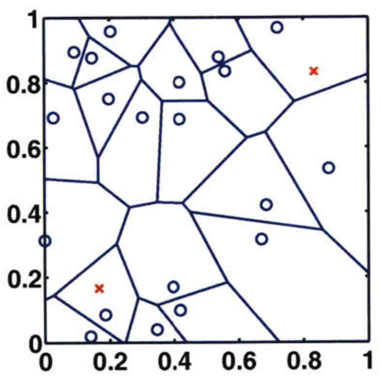
where $\sigma_j = .18$. The unit square was divided into an even $3 \times 3$ grid and each $\mu_j$ was chosen so that one of the nine Gaussians was centered at the middle of each grid square. The parameters were chosen as $a = [100 \quad a_{\min} \quad \cdots \quad a_{\min} \quad 100]^T$, with $a_{\min} = .1$ so that only the lower left and upper right Gaussians contributed significantly to the value of $\phi(q)$, producing a bimodal distribution.

The robots in the network were started from random initial positions. Each robot used a copy of the Gaussians described above for $\mathcal{K}(q)$. The estimated parameters $\hat{a}_i$ for each robot were started at a value of $a_{\min}$, and $\Lambda_i$ and $\lambda_i$ were each started at zero. The gains used by the robots were $K = 3I_2$, $\Gamma = I_9$, $\gamma = 300$ and $\zeta = 0$ for the basic controller, and $\gamma = 100$ and $\zeta = 50$ for the consensus controller. In practice, the first integral term in the adaptive law (4.13) seems to have little effect on the performance of the controller. Choosing $\Gamma$ small and $\gamma$ comparatively large puts more weight on the second term, which is responsible for integrating measurements of $\phi(p_i)$ into the parameters. The spatial integrals in (4.7) and (4.13) required for the control law were computed by discretizing each Voronoi region $V_i$ into a $7 \times 7$ grid and summing contributions of the integrand over the grid. Voronoi regions were computed using a decentralized algorithm similar to the one in [26].

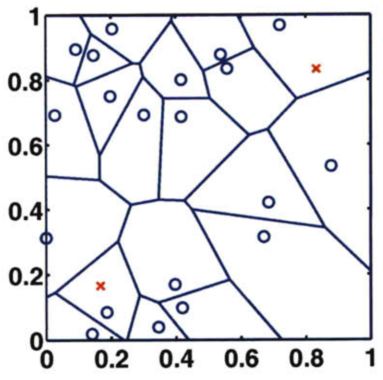
### 4.8.1 Simulation Results

Figure 4-4 shows the positions of the robots in the network over the course of a simulation run for the parameter consensus controller (left column) and the basic controller (right column). The centers of the two contributing Gaussian functions are marked with $\times$s. It is apparent from the final configurations that the consensus controller caused the robots to group more tightly around the Gaussian peaks than the basic controller. The somewhat jagged trajectories are caused by the discrete nature of the spatial integration procedure used to compute the control law.

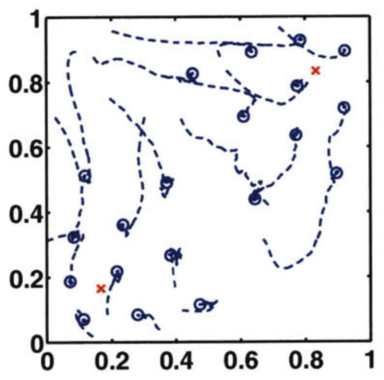
Figure 4-5(a) shows that both controllers converge to a *near-optimal* configuration—one in which every robot is located at the estimated centroid of its Voronoi region, in accordance with Theorem 4.1. However, the true position error also converged to zero for the consensus controller, indicating that it achieved an *optimal* coverage configuration, as shown in Figure 4-5(b). The basic controller did not reach an *optimal* coverage configuration. Furthermore, convergence was so much faster for the consensus controller that we have to use a logarithmic time scale to display both curves on the same plot. Again, the somewhat jagged time history is a result of the discretized spatial integral computation over the Voronoi region.



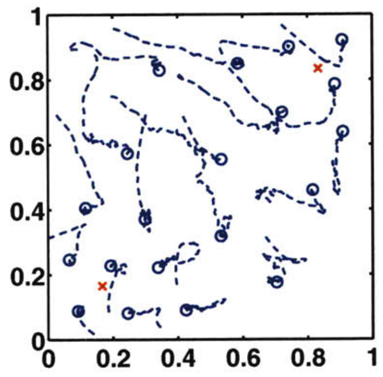
(a) Consensus Initial Config.



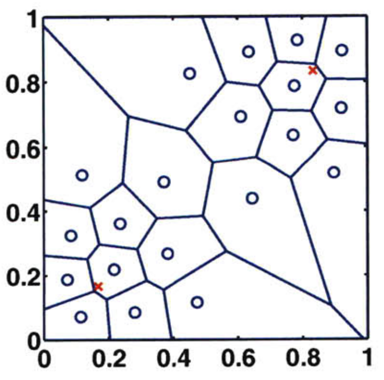
(b) Basic Initial Config.



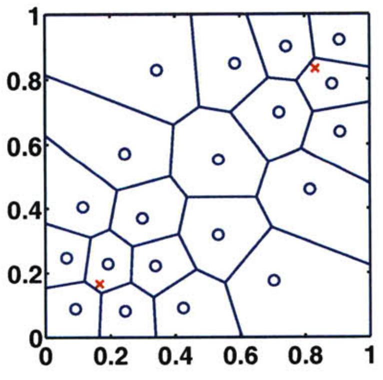
(c) Consensus Trajectories



(d) Basic Trajectories

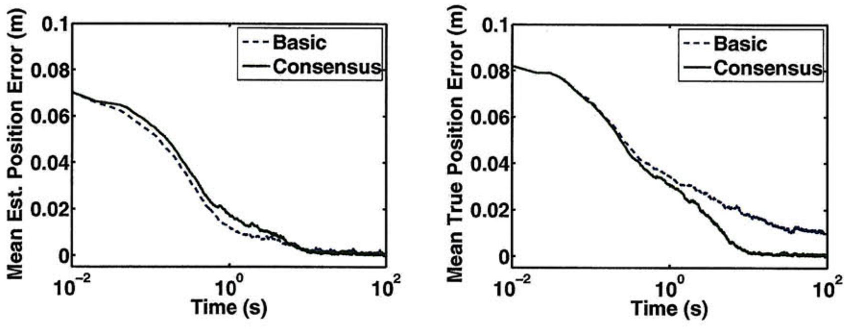


(e) Consensus Final Config.



(f) Basic Final Config.

Figure 4-4: Simulation results for the parameter consensus controller are shown in the left column (4-4(a), 4-4(c), and 4-4(e)), and for the basic controller in the right column (4-4(b), 4-4(d), and 4-4(f)). The Gaussian centers of $\phi(q)$ are marked by the red x's.

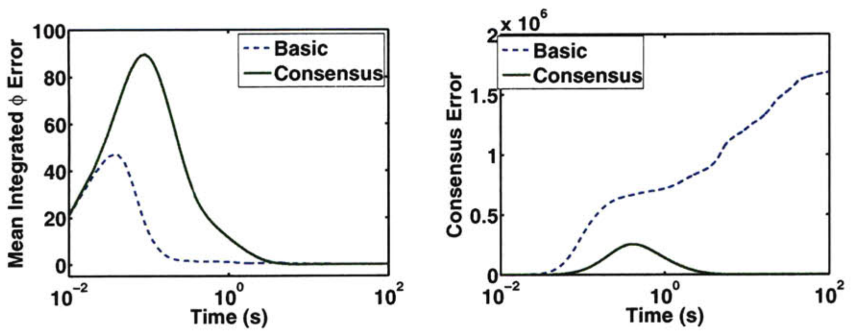


(a) Mean Estimated Position Error (b) Mean True Position Error

Figure 4-5: The estimated position error, $\|\hat{C}_{V_i} - p_i\|$, and the true position error, $\|C_{V_i} - p_i\|$ averaged over all the robots in the network is shown for the network of 20 robots for both the basic and parameter consensus controllers. The true position error converges to zero only for the parameter consensus controller, 4-5(b). However, in accordance with Theorem 4.1, the estimated error converges to zero in both cases, 4-5(a). Note the logarithmic time scale.

The Figure 4-6(a) demonstrates that a *locally true* sensory function approximation is achieved for each robot over $\Omega_i = \{p_i(\tau) \mid \tau \geq 0, \omega(\tau) > 0\}$, the set of points along the robot's trajectory with positive weighting. The plot shows the integral in (4.21) as a function of time averaged over all the robots in the network converging asymptotically to zero. The disagreement among the parameter values of robots is shown in the right of Figure 4-6(b). The parameters were initialized to $a_{\min}$ for all robots, so this value starts from zero in both cases. However, the consensus controller causes the parameters to reach consensus, while for the basic controller the parameters do not converge to a common value.

Figure 4-7(a) shows that the consensus controller obtained a lower value of the Lyapunov function at a faster rate than the basic controller, indicating both a lower-cost configuration and a better function approximation. In fact, Figure 4-7(b) shows that the parameter errors $\|\tilde{a}_i\|$ actually converged to zero for the consensus controller, so the conditions for Corollary 4.2 were met. This was also evidenced in Figure 4-5(b) since the true position error converged to zero. For the basic controller, on the other hand, the parameters did not converge to the true parameters.



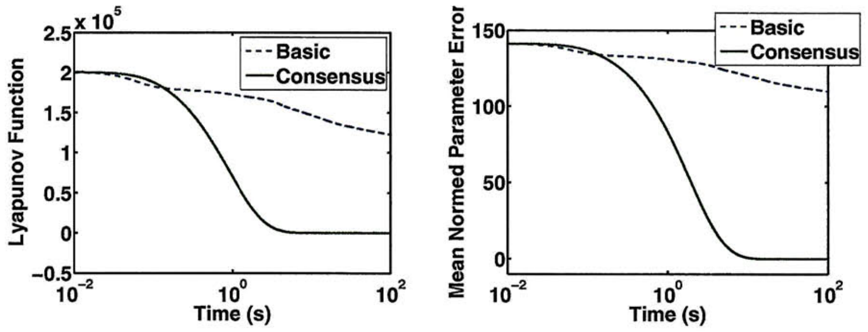
(a) Mean Int. $\phi(q)$ Error

(b) Consensus Error

Figure 4-6: The integrated sensory function error, namely $\int_0^t \omega(\tau)(\mathcal{K}_i\tilde{a}_i)^2\, d\tau$, averaged over all the robots is shown for the basic and consensus controllers in 4-6(a). The plot demonstrates that each robot converges to a locally true function approximation over all points along its trajectory with positive weighting, $\omega(\tau) > 0$, as asserted in Theorem 4.1. The quantity $\sum_{i=1}^n \hat{a}_i^T \sum_{j=1}^n(\hat{a}_i - \hat{a}_j)$ is shown in 4-6(b), representing a measure of the disagreement of parameters among robots. The disagreement converges to zero for the consensus controller, as asserted in Theorem 4.2, but does not converge for the basic controller.

## 4.9 Synopsis

In this chapter we augmented the distributed coverage controller from Chapter 3 to including learning of the sensory function. The learning controller was proven to cause the robots to move to the estimated centroids of their Voronoi regions, while also causing their estimate of the sensory distribution to improve over time. Parameter coupling was introduced in the adaptation laws to increase parameter convergence rates and cause the robots' parameters to achieve a common final value. The control law was demonstrated in numerical simulations of a group of 20 robots sensing over an area with a bimodal Gaussian distribution of sensory information.



(a) Lyapunov Function (b) Mean $\|\tilde{a}_i(t)\|$

Figure 4-7: The Lyapunov function is shown in 4-7(a) for both the basic and parameter consensus controllers. Notice that the parameter consensus controller results in a faster decrease and a lower final value of the function. The normed parameter error $\|\tilde{a}_i\|$ averaged over all robots is shown in 4-7(b). The parameter error converges to zero with the consensus controller indicating that the robot trajectories were sufficiently rich.

# Chapter 5

# From Theory to Practice: Coverage with SwarmBots

## 5.1 Introduction

In Chapter 4 we introduced a distributed, theoretically-proven controller for a group of robots to provide sensor coverage of an environment while learning the sensory function. In this chapter we describe the algorithmic and systems challenges we solved to implement this coverage controller on a group of robots. We present results of experiments with 16 robots. As described in Chapter 4 and shown graphically in Figures 4-1 and 4-2, we implement an algorithm in which robots simultaneously learn the areas of the environment which need to be covered, and move to cover those areas. The learning algorithm uses on-line parameter adaptation and a consensus algorithm to approximate the sensory function from sensor measurements.

### 5.1.1 Related Work

There is little existing experimental work on multi-robot coverage control using the Voronoi based method aside from that presented in this thesis. The first experimental results with Voronoi based coverage were obtained in [95] with a controller that approximated the sensory function, though not using learning as in this thesis. Other experiments were carried out for a time-varying sensory function in [78]. Other multi-robot coverage methods not involving Voronoi tessellations have been investigated experimentally, however. For example, [23] used both reactive and deliberative approaches to inspect turbine blades with multiple robots. Preliminary experiments with one fixed and one moving robot were described in [82], and [51] describes multi-robot experiments with a lawn mowing-type coverage algorithm. Experiments of multiple robots covering an environment using an exploration algorithm without localization information are reported in [6].

### 5.1.2 Contributions

The main contribution of this chapter is as follows:

1. The controller from Chapter 4 with on-line learning of the sensory function is implemented on a group of 16 SwarmBots. The performance of the robot group is compared to that predicted by the theoretical results of Chapter 4.

In Section 5.2 we translate the controller from 4 into an algorithm that is practical for implementation on robot platforms with limited computational resources. We also enumerate the differences between the practical algorithm and the idealized controller. In Section 5.3 we give results of two experiments and show experimental snapshots. The algorithm is shown to operate in realistic situations in the presence of noise on sensor measurements and actuator outputs. Conclusions and discussion are in Section 5.4.

## 5.2 Coverage Control Algorithm

The Coverage control algorithm has two components, corresponding to the two spaces described in Figure 4-1. In position space, the robots pursue their estimated centroids, given by

$$p_i(t+1) = \hat{C}_{V_i}(t). \tag{5.1}$$

The estimated centroid of the Voronoi region is its geometric center, weighted by the sensory function approximation. We calculate the discrete approximation of the centroid of $V_i$ by dividing it up into a set of grid squares. Let the set of center points of the grid squares be $\bar{V}_i$ and each grid square has equal area $\Delta q$. Then the estimated centroid $\hat{C}_{V_i}$ of $V_i$, weighted by $\hat{\phi}_i(q,t)$, is given by

$$\hat{C}_{V_i}(t) = \frac{\sum_{q \in \bar{V}_i} q \hat{\phi}_i(q,t) \Delta q}{\sum_{q \in \bar{V}_i} \hat{\phi}_i(q,t) \Delta q}, \tag{5.2}$$

where $\hat{\phi}_i(q,t)$ is defined by

$$\hat{\phi}_i(q,t) = \mathcal{K}(q)^T \hat{a}_i(t), \tag{5.3}$$

as in Chapter 4, and $\hat{a}_i(t)$ is the estimated parameter vector of robot $i$.

In parameter space, the robots collaboratively learn the function $\phi(q)$. They do this by iteratively integrating the values of $\phi(p_i)$ into the quantity $\lambda_i(t)$. They also integrate the value of the basis function vector at their position $\mathcal{K}(p_i(t))$ into the quantity $\Lambda_i(t)$. Specifically,

$$\lambda_i(t+1) = \lambda_i(t) + \mathcal{K}(p_i(t))\phi(p_i(t)) \text{ and,} \tag{5.4}$$

$$\Lambda_i(t+1) = \Lambda_i(t) + \mathcal{K}(p_i(t))\mathcal{K}(p_i(t))^T. \tag{5.5}$$

Here for simplicity we use a uniform time weighting function, $\omega(t) = 1$. Each robot then tunes its parameter vector using

$$\hat{a}_{i\text{pre}}(t) = \hat{a}_i(t) + \gamma\big(\lambda_i(t) - \Lambda_i(t)\hat{a}_i(t)\big) + \zeta \sum_{j \in \mathcal{N}_i(t)} \big(\hat{a}_j(t) - \hat{a}_i(t)\big). \tag{5.6}$$

where $\gamma$ and $\zeta$ are positive gains. We do not use the length of the shared Voronoi face as a weighting in the parameter tuning (5.6). Instead we use the simpler $0-1$ weighting described in Remark 4.4. As described in Chapter 4, the term $\lambda_i(t) - \Lambda_i(t)\hat{a}_i(t)$ changes the parameters to follow the negative gradient of the Least Squares cost function. The term $\sum_{j \in \mathcal{N}_i(t)} \big(\hat{a}_j(t) - \hat{a}_i(t)\big)$ has the effect of propagating every robot's parameters around the network to be used by every other robot, and ultimately causes

**Algorithm 1** Consensus-Based Coverage

```
Require: Each robot knows its position $p_i(t)$
Require: Each robot can communicate with its Voronoi neighbors
Require: Each robot can compute its Voronoi cell, $V_i$
Require: Each robot can measure $\phi(p_i)$ with its sensors
  Initialize:   $\Lambda_i(0) = 0$,   $\lambda_i(0) = 0$,   and   $\hat{a}_i(0) = [a_{\min}, \ldots, a_{\min}]^T$
  loop
    Update:
        $\lambda_i(t+1) = \lambda_i(t) + \mathcal{K}(p_i(t))\phi(p_i(t))$
        $\Lambda_i(t+1) = \Lambda_i(t) + \mathcal{K}(p_i(t))\mathcal{K}(p_i(t))^T$
        $\hat{a}_{i\text{pre}}(t) = \hat{a}_i(t) + \gamma\big(\lambda_i(t) - \Lambda_i(t)\hat{a}_i(t)\big) + \zeta \sum_{j \in \mathcal{N}_i(t)} \big(\hat{a}_j(t) - \hat{a}_i(t)\big)$
    Project $\hat{a}_{i\text{pre}}(t)$ to ensure parameters remain positive:   $\hat{a}_i(t+1) = \max(\hat{a}_{i\text{pre}}(t), a_{\min})$
    Compute the robot's Voronoi region $V_i$
    Discretize $V_i$ into grid squares with area $\Delta q$ and center points $q \in \bar{V}_i$
    Compute the centroid estimate:
        $\hat{C}_{V_i}(t) = \frac{\sum_{q \in \bar{V}_i} q \hat{\phi}_i(q,t)\Delta q}{\sum_{q \in \bar{V}_i} \hat{\phi}_i(q,t)\Delta q}$, where   $\hat{\phi}_i(q,t) = \mathcal{K}(q)^T \hat{a}_i(t)$
    Drive to the estimated centroid:   $p_i(t+1) = \hat{C}_{V_i}(t)$
  end loop
```

all robots' parameter vectors to approach a common value. Finally, parameters are maintained above a predefined minimum positive value $a_{\min} \in \mathbb{R}$, $a_{\min} > 0$, using

$$\hat{a}_i(t+1) = \max(\hat{a}_{i\text{pre}}(t), a_{\min}), \tag{5.7}$$

where the $\min(\cdot,\cdot)$ operates element-wise on the vector $\hat{a}_{i\text{pre}}(t)$. Our consensus-based coverage algorithm (as executed asynchronously by each robot) is written in Algorithm 1.

In summary, our coverage control algorithm integrates the sensor measurements and robot trajectory into $\lambda_i \in \mathbb{R}^m$ and $\Lambda_i \in \mathbb{R}^{m \times m}$, respectively. These are then used to tune the parameter vector $\hat{a}_i(t)$, which is also combined with the neighbors' parameter vectors. The parameter vector is used to calculate the sensory function estimate $\hat{\phi}_i(q,t)$, which is used to calculate the estimated Voronoi centroid $\hat{C}_{V_i}$, which the robot then moves toward. The algorithm is a discrete-time interpretation of the control law from [100], which, under mild assumptions, was proved to cause robots to converge to the centroids of their Voronoi cells.

By implementing the control algorithm on a group of robots, a number of complications are introduced that were not considered in [100], as described in Table 5.1. The presence of noise in all measurement and actuation operations is a significant change from the noiseless scenario considered in [100]. Noise on the position measurements of neighbors in particular seemed to be a large source of error in the computation of the centroid of the Voronoi regions. We find that the algorithm performs well despite the presence of these real-world complications. The robustness of the algorithm can be attributed to its closed-loop structure, which constantly incorporates position updates and new sensor measurements to naturally correct mistakes. Also, the consensus-learning law tends to smooth the effects of noise on the sensory function measurements. This is because the parameter vectors are iteratively combined with neighbors' parameter vectors, so inaccuracies that might otherwise accumulate due to measurement errors are counteracted by measurement errors from neighboring robots.

## 5.3 Results and Experimental Snapshots

The algorithm was implemented in integer arithmetic on a network of 16 SwarmBots [64] (Figure 5-5(a)). Each SwarmBot used an on-board IR system to sense relative neighbor positions (for computing its Voronoi cell) and to communicate its parameter vector to its neighbors. The robots moved in a square environment 2.44m$\times$2.44m. Each robot's global position was measured by an overhead camera and sent to it by radio. Each SwarmBot used a 40MHz 32-bit ARM Thumb microprocessor, which provided enough processing power to execute our algorithm in real-time. There was no centralized or off-line processing.

The system reliably performed numerous experiments and demonstrations. Here

Table 5.1: Algorithm 1 vs. Controller from Chapter 4

| **Algorithm 1** | **Controller from Chapter 4** |
|---|---|
| • Discrete-time difference equations | • Continuous-time differential equations |
| • Nonholonomic "unicycle" robot dynamics cause position errors and turning delays | • Holonomic "integrator" robot dynamics |
| • Asynchronous execution of instructions | • Synchronous evolution of equations |
| • Approximate Voronoi cells constructed from noisy measurements of neighbors within sensing range | • Exact Voronoi cells computed from exact positions of all Voronoi neighbors |
| • Discretized sums over the Voronoi cell | • Exact integrals over the Voronoi cell |
| • Noisy measurement of global position | • Exact knowledge of global position |
| • Noisy actuators | • Noiseless actuators |
| • Noisy measurement of sensory function | • Noiseless measurement of sensory function |
| • Basis function approximation cannot reconstruct exact sensory function | • Basis function approximation can reconstruct sensory function exactly with ideal parameter vector |

we present detailed results of two experiments. In the first experiment in Section 5.3.1, the robots were given a noiseless measurement of a *simulated* sensory function $\phi(p_i)$. This allowed us to compare the performance of the algorithm to a known ground truth. Since the function $\phi(q)$ is known, we also know the true position errors of the robots (the distances to their true centroids), as well as the true parameter errors. In the second experiment in Section 5.3.2, the robots used their on-board light sensors to sense light intensity in the environment as a sensory function. In this case we have no ground truth value for $\phi(q)$. We verify that the algorithm exhibits the behavior that one would expect given the scenario.

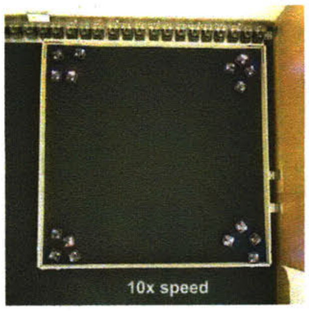
### 5.3.1 Simulated Sensory Function

The simulated sensory function, $\phi(q)$, was represented by two Gaussians, one in the lower right of the environment and one in the upper left. The set of basis functions was chosen to be 9 Gaussians arranged in a grid over the square environment. In particular, each of the nine components of $\mathcal{K}(q)$ was implemented as $1/(2\pi\sigma^2)\exp\left\{-\|q-\mu_j\|^2/(2\sigma_j^2)\right\}$, where $\sigma_j = .37m$. The 2.44m$\times$2.44m square was divided into an even $3 \times 3$ grid and each $\mu_j$ was chosen so that one of the 9 Gaussians was centered at the middle of each grid square. The parameters for the simulated sensory function were chosen as $a = [200 \quad a_{\min} \quad \cdots \quad a_{\min} \quad 200]^T$, with $a_{\min} = 1$ so that only the upper left and lower right Gaussians contributed significantly to the value of $\phi(q)$.

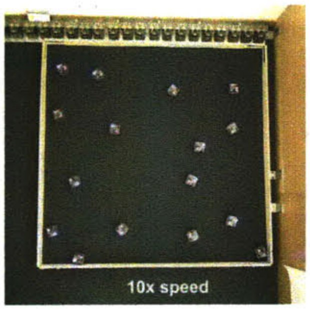
Figure 5-1 shows the positions of 16 robots over the course of an experiment. The algorithm caused the robots to group around the Gaussian peaks. The robots had no prior knowledge of the number or location of the peaks. Figure 5-2(a) shows the distance to the centroid, averaged over all the robots. The distance to the true centroid decreased over time to a steady value. The distance to the estimated centroid decreased to a value close to the pre-set dead zone of 5cm. The significant noise in the distance to the estimated centroid comes from noise in the IR system used to measure the neighbor positions. This caused the Voronoi cells to change rapidly, which in turn caused the centroid estimates to be noisy. Despite this noise, the true distance to the centroid decreased steadily, indicating that the algorithm is robust to these significant sources of error. Figure 5-2(b) shows that the normed parameter error, averaged over all of the robots, decreased over time. Figure 5-2(c) shows $\sum_{i=1}^n \hat{a}_i(t)^T \sum_{j\in\mathcal{N}_i}(\hat{a}_i(t) - \hat{a}_j(t))$, representing the disagreement among the parameter vectors of different robots. The disagreement started at zero because all parameters were initialized with the same value of $a_{\min}$. The disagreement initially grew, then decreased as the robots' parameters reached a consensus.

### 5.3.2 Measured Sensory Function

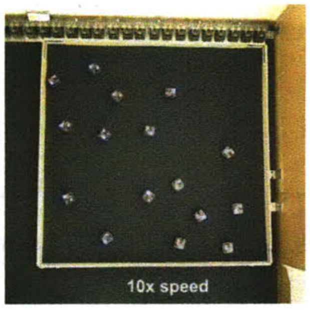
An experiment was also carried out using light intensity over the environment as the sensory function. Two incandescent office lights were placed at the lower left corner of the environment, and the robots used on-board light sensors to measure the light intensity. The same 3$\times$3 grid of basis functions as in the first experiment was used. In



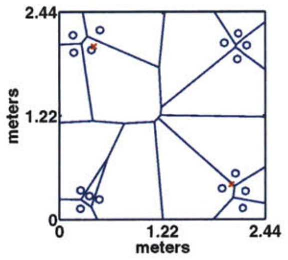
(a) Initial Snapshot



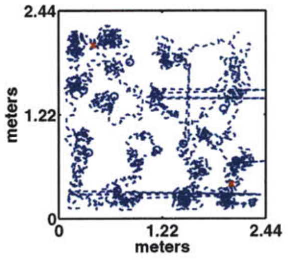
(b) Middle Snapshot



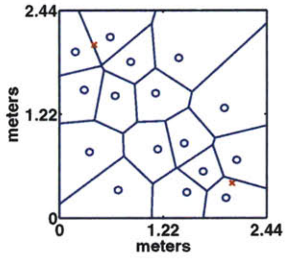
(c) Final Snapshot



(d) Initial Config.



(e) Trajectories

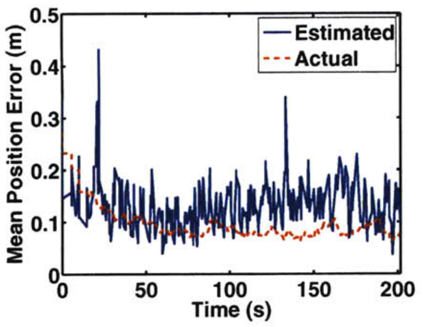


(f) Final Config.

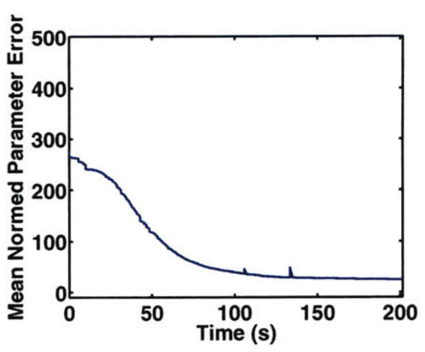
Figure 5-1: Results for the algorithm are shown in video snapshots in the left column (5-1(a), 5-1(b), and 5-1(c)). The positions collected from the overhead camera for the same experiment are plotted in the right column (5-1(d), 5-1(e), and 5-1(f)). The Gaussian centers of $\phi(q)$ are marked by red x's.

this experiment there was no ground truth against which to compare the performance of the algorithm since we did not know the "true" light intensity function over in the environment. We instead show that the algorithm caused the network to do what one would expect given the *qualitative* light intensity distribution.

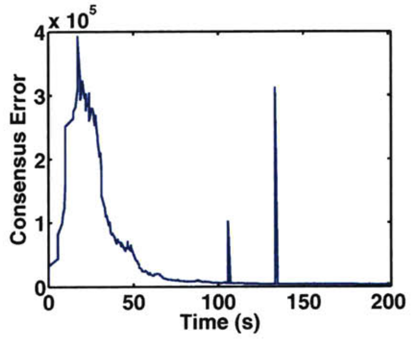
Figure 5-3 shows snapshots of the experiment taken from the overhead camera. Notice that the robots collected in higher density around the light sources while still covering the environment. Figure 5-4(a) shows that the distance to the robots' estimated centroids decreased, albeit with a significant amount of noise due to uncertainty in the neighbor position estimates, as in the previous experiment. Figure 5-4(a) also shows the distance to the estimated centroid filtered so that the decreasing trend becomes more evident. Also, Figure 5-5(b) shows that the robots learned a function with a large weight near the position of the light sources. The weights on the 9 Gaussians adjusted to find the best fit of the data. Figure 5-4(b) shows that,



(a) Mean Position Error



(b) Mean Parameter Error



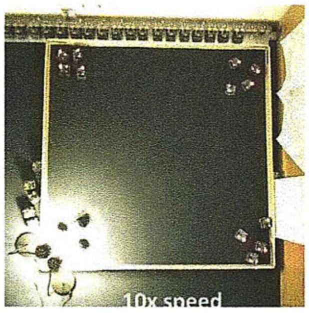
(c) Consensus Error

Figure 5-2: The distance to the actual centroid, and the distance to the estimated centroid, averaged over all the robots are shown in 5-2(a). The normed parameter error averaged over all robots is shown in 5-2(b). The plot in 5-2(c) shows a quantity representing the disagreement of parameters among robots.

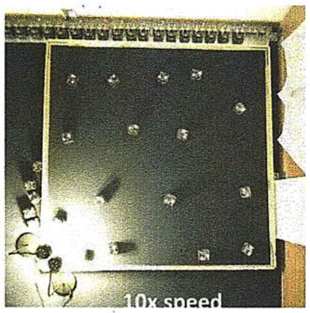
as in the previous experiment, disagreement between robot parameters initially grew, then decreased as the robots tended toward consensus. The parameters never actually reach consensus because of noise and calibration differences among the different robots' light sensors.

## 5.4 Synopsis

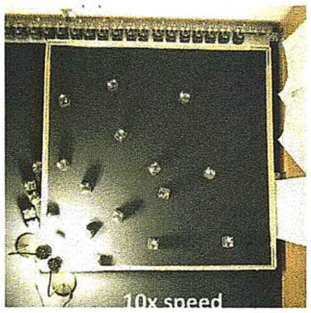
In this chapter, we implemented a control algorithm for multi-robot coverage on a minimalist robot platform. The controller was adapted to the hardware platform available, and was shown to perform robustly despite the presence of sensor and actuator noise, and other real-world complications. We presented the results of two



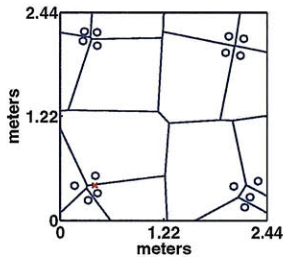
(a) Initial Snapshot



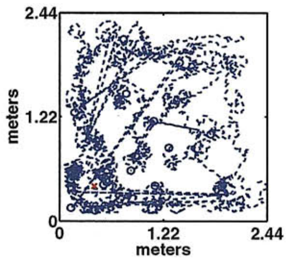
(b) Middle Snapshot



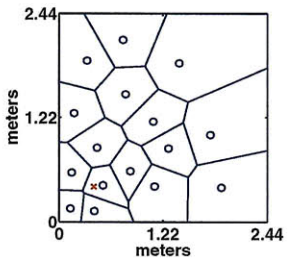
(c) Final Snapshot



(d) Initial Config.



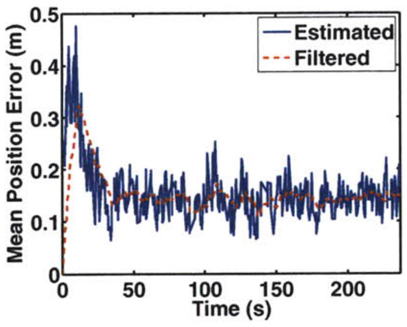
(e) Trajectories



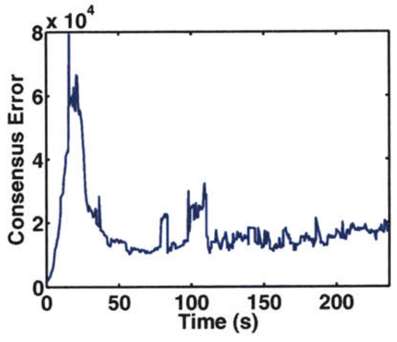
(f) Final Config.

Figure 5-3: Results for the algorithm are shown in video snapshots in the left column (5-3(a), 5-3(b), and 5-3(c)). The positions collected from the overhead camera for the same experiment are plotted in the right column (5-3(d), 5-3(e), and 5-3(f)). The robots used the light intensity measured with on board light sensors as the sensory function.

experiments with 16 robots. In the first experiment, the robots were given simulated sensory function measurements so that we could compare the results with a known ground truth. In the second experiment, the robots used measurements from light sensors as a sensory function. We hope these results represent a significant step toward the use of multi-robot coverage control algorithms in practical monitoring and surveillance applications in the future.

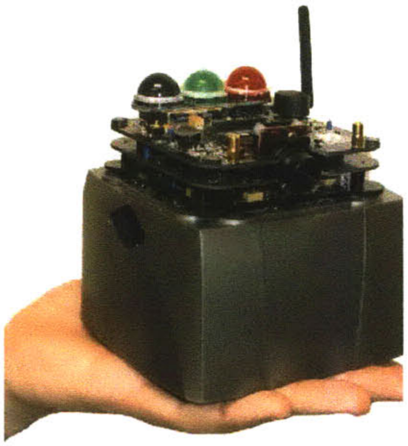


(a) Mean Position Error

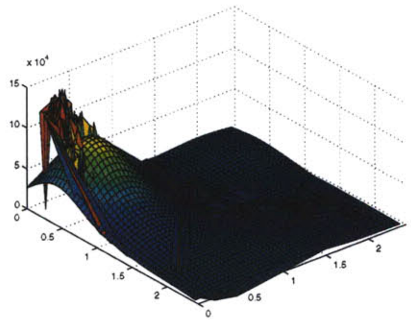


(b) Consensus Error

Figure 5-4: The distance to the estimated centroid, averaged over all the robots in the network is shown in 5-4(a). The plot in 5-4(b) shows a quantity representing the disagreement of parameters among robots.



(a) SwarmBot



(b) Function Approximation

Figure 5-5: The iRobot SwarmBot platform is shown in 5-5(a). The basis function approximation of the light intensity (smooth surface) over the area for one robot is shown in 5-5(b) superimposed over a triangular interpolation of the light intensity measurements of all the robots (jagged surface).

# Chapter 6

# Coverage with Quad-Rotors

## 6.1 Introduction

In this chapter we apply the theory from Chapter 3 to control flying robots with downward facing cameras. This work demonstrates how to incorporate a realistic sensor model of the camera to obtain an appropriate cost function $\mathcal{H}$ to derive the gradient based controller from Equation (2.7). The computation of the gradient controller in this case is more difficult, and the application of the gradient convergence and stability theorems (Theorems 2.3 and 2.4) require the verification of some more intricate technical details.

Multiple collaborating robots with cameras are useful in a broad range of applications, from surveying disaster sites, to observing the health of coral reefs. However, an immediate and difficult question arises in such applications: how should one position the robots so as to maintain the best view of an environment? In this chapter we offer an approach motivated by an information content principle: minimum information per pixel. Using information per pixel as a metric allows for the incorporation of physical, geometric, and optical parameters to give a cost function that represents how well a group of cameras covers an environment. We develop the approach in detail for the particular case of multiple downward facing cameras mounted to robots. The cost function leads to a gradient-based distributed controller for the robots to position themselves in three dimensions so as to best observe a planar environment

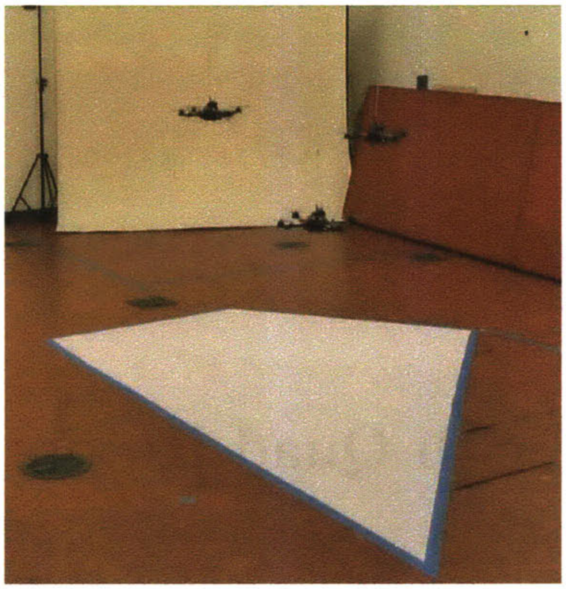


Figure 6-1: This snapshot of an experiment shows three flying quad-rotor robots moving so that their cameras cover the environment represented by the white polygon.

over which they hover. We present simulation results in a Matlab environment. We also present experimental results with three AscTec Hummingbird quad-rotor robots.

Our algorithm can be used in support of a higher-level computer vision task, such as object recognition or tracking. We address the problem of how to best position the robots given that the data from their cameras will be used by some computer vision algorithm. Our design principle can be readily adapted to a number of applications. For example, it could be used to control groups of autonomous underwater or air vehicles to do mosaicing [79], or to produce photometric stereo from multiple camera views [41], for inspection of underwater or land-based archaeological sites, biological environments such as coral reefs or forests, disaster sites, or any other large scale environment of interest. Our algorithm could also be used by autonomous flying robots to do surveillance [21], target tracking [12, 18, 50], or to aid in navigation of agents on the ground [81].

### 6.1.1 Related Work

One recent extension described in [63], Figure 14, proposed an algorithm for the placement of hovering sensors, similar to our scenario. Our method in this chapter is related to this work in that we propose a cost function and obtain a distributed controller by taking its gradient. However, the cost function we propose is different from previous ones in that it does not involve a Voronoi partition. To the contrary, it relies on the fields of view of multiple cameras to overlap with one another. Another distinction from previous works is that the agents we consider move in a space that is different from the one they cover. Previous coverage scenarios have considered agents constrained to move in the environment that they cover, which leads to a constraint that the environment must be convex (to prevent agents from trying to leave the environment). In contrast, we consider agents moving in a space $\mathbb{R}^3$, covering an arbitrary lower dimensional environment $Q \subset \mathbb{R}^2$. This eliminates the need for $Q$ to be convex. Indeed, it need not even be connected. It must only be Lebesgue measurable (since the robots will calculate integrals over it), which is quite a broad specification.

There have also been other algorithms for camera placement, for example a probabilistic approach for general sensor deployment based on the Cramér-Rao bound was proposed in [42], and an application of the idea for cameras was given in [31]. We choose to focus on the problem of positioning downward facing cameras, similarly to [54], as opposed to arbitrarily oriented cameras. Many geometrical aspects of the problem are significantly simplified in this setting, yet there are a number of practical applications that stand to benefit from controlling cameras in this way, as previously described. More generally, several other works have considered cooperative control with flying robots and UAV's. For an excellent review of cooperative UAV control please see [85], or [11] and [83] for two recent examples.

### 6.1.2 Contributions

The main contributions of this chapter are:

1. Applying the ideas of Chapter 3, we propose the minimum information per pixel principle to formulate a cost function for multiple hovering robots with downward facing cameras. We use the cost function to design a gradient descent controller to deploy multiple robots to their optimal positions in a distributed fashion.

2. We implement the proposed controller on three quad-rotor robots and test its performance in experiments.

The proposed robot coordination algorithm is fully decentralized, provably stable, adaptive to a changing number of flying agents and a changing environment, and will work with a broad class of environment geometries, including convex, non-convex, and disconnected spaces.

## 6.2 Optimal Camera Placement

We motivate our approach with an informal justification of a cost function, then develop the problem formally for the single camera case followed by the multi-camera case. We desire to cover a bounded environment, $Q \subset \mathbb{R}^2$, with a number of cameras. We assume $Q$ is planar, without topography, to avoid the complications of changing elevation or occlusions. As in previous chapters, let $p_i \in \mathcal{P}$ represent the state of camera $i$, where the state-space, $\mathcal{P}$, will be characterized later. We want to control $n$ cameras in a distributed fashion such that their placement minimizes the aggregate information per camera pixel over the environment,

$$\min_{(p_1,\dots,p_n)\in\mathcal{P}^n} \int_Q \frac{\text{info}}{\text{pixel}}\, dq.$$

This metric makes sense because the pixel is the fundamental information capturing unit of the camera. Consider the patch of image that is exposed to a given pixel. The information in that patch is reduced by the camera to a low-dimensional representation (i.e. mean color and brightness over the patch). Therefore, the less information content the image patch contains, the less information will be lost in its low-dimensional representation by the pixel. Furthermore, we want to minimize the accumulated information loss due to pixelation over the whole environment $Q$, hence the integral. In the next two sections we will formalize the notion of information per pixel.

### 6.2.1 Single Camera

We develop the cost function for a single camera before generalizing to multiple cameras. It is convenient to consider the information per pixel as the product of two functions, $f : \mathcal{P} \times Q \mapsto (0, \infty]$, which gives the area in the environment seen by one pixel (the "area per pixel" function), and $\phi : Q \mapsto (0, \infty)$ which gives the information per area in the environment. The form of $f(p_i, q)$ will be derived from the optics of the camera and geometry of the environment. The function $\phi(q)$ is a positive weighting of importance over $Q$ and should be specified beforehand (it can also be learned from sensor data, as in Chapter 4). For instance, if all points in the environment are equally important, $\phi(q)$ should be constant over $Q$. If some known area in $Q$ requires more resolution, the value of $\phi(q)$ should be larger in that area than elsewhere in $Q$. This gives the cost function

$$\min_p \int_Q f(p, q)\phi(q)\, dq, \tag{6.1}$$

which is of a general form similar to the one seen in Chapter 3. We will introduce significant changes to this basic form with the addition of multiple cameras.

The state of the camera, $p$, consists of all parameters associated with the camera that effect the area per pixel function, $f(p, q)$. In a general setting one might consider the camera's position in $\mathbb{R}^3$, its orientation in $so(3)$ (the three rotational angles), and perhaps a lens zooming parameter in the interval $(0, \infty)$, thus leading to an optimization in a rather complicated state-space ($\mathcal{P} = \mathbb{R}^3 \times so(3) \times (0, \infty)$) for only one camera. For this reason, we consider the special case in which the camera is downward facing (hovering over $Q$). Indeed, this case is of particular interest in many applications, as described in Section 6.1. We define the field of view, $\mathcal{B}$, to be

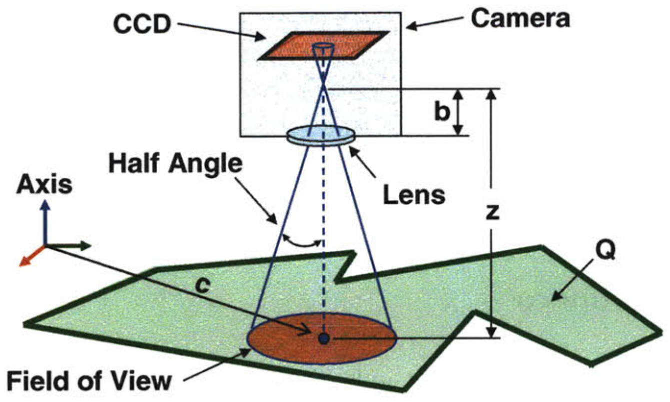


Figure 6-2: The camera optics and the geometry of the environment are shown in this figure.

the intersection of the cone whose vertex is the focal point of the camera lens with the subspace that contains the environment, as shown in Figure 6-2. In Section 6.3.1 we will consider a camera with a rectangular field of view, but initially consider a circular field of view, so the rotational orientation of the downward facing camera is irrelevant. In this case $\mathcal{P} = \mathbb{R}^3$, and the state-space in which we do optimization is considerably simplified from that of the unconstrained camera. Decompose the camera position as $p = [c^T, z]^T$, with $c \in \mathbb{R}^2$ the center point of the field of view, and $z \in \mathbb{R}$ the height of the camera over $Q$. We have

$$\mathcal{B} = \left\{ q \mid \frac{\|q - c\|}{z} \leq \tan\theta \right\} \tag{6.2}$$

where $\theta$ is the half-angle of view of the camera.

To find the area per pixel function, $f(p, q)$, consider the geometry in Figure 6-2. Let $b$ be the focal length of the lens. Inside $\mathcal{B}$, the area/pixel is equal to the inverse of the area magnification factor (which is defined from classical optics to be $b^2/(b-z)^2$) times the area of one pixel [40]. Define $a$ to be the area of one pixel divided by the square of the focal length of the lens. We have,

$$f(p, q) = \begin{cases} a(b-z)^2 & \text{for} \quad q \in \mathcal{B} \\ \infty & \text{otherwise,} \end{cases} \tag{6.3}$$

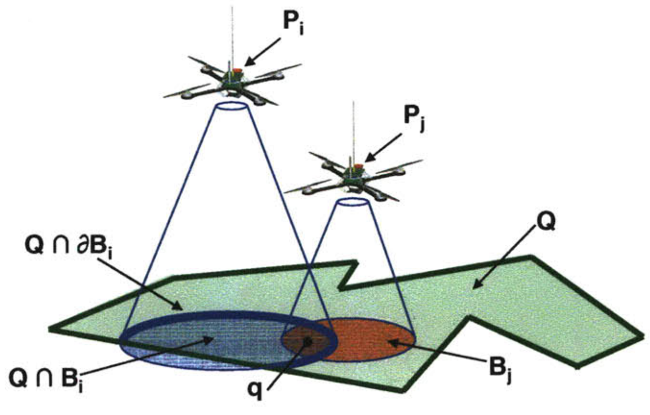


Figure 6-3: This figure shows the relevant quantities involved in characterizing the intersecting fields of view of two cameras.

Outside of the field of view, there are no pixels, therefore the area per pixel is infinite (we will avoid dealing with infinite quantities in the multi-camera case). The cost function in (6.1) takes on an infinite value if any area (of non-zero measure) of Q is outside of the field of view. Indeed, we know there exists a $p \in \mathcal{P}$ such that the cost is finite, since $Q$ is bounded (given $c$ and $\theta$, there exist $z \in \mathbb{R}$ such that $Q \subset \mathcal{B}$). Therefore, we can write the equivalent constrained optimization problem

$$\begin{aligned} &\min_p \int_Q a(b+z)^2 \phi(q)\, dq, \\ &\text{subject to} \quad Q \subset \mathcal{B}. \end{aligned} \tag{6.4}$$

One can see in this simple scenario that the optimal solution is for $p$ to be such that the field of view is the smallest ball that contains $Q$. However, with multiple cameras, the problem becomes more challenging.

## 6.2.2 Multiple Cameras

To find optimal positions for multiple cameras, we have to determine how to account for the area of overlap of the images of the cameras, as shown in Figure 6-3. Intuitively, an area of $Q$ that is being observed by two different cameras is better covered than if it were being observed by only one camera, but it is not *twice* as well covered. Consider a point $q$ that appears in the image of $n$ different cameras. The number of pixels per area at that point is the sum of the pixels per area for each camera. Therefore (assuming the cameras are identical, so they use the same function $f(p_i, q)$) the area per pixel at that point is given by the *inverse* of the sum of the *inverse* of the area per pixel for each camera, or

$$\frac{\text{area}}{\text{pixel}} = \Big(\sum_{i=1}^{n} f(p_i, q)^{-1}\Big)^{-1},$$

where $p_i$ is the position of the $i$th camera. We emphasize that it is the *pixels per area* that sum because of the multiple cameras, not the *area per pixel* because, in the overlap region, multiple pixels are observing the same area. Therefore the inverse of the sum of inverses is unavoidable. Incidentally, this is the same form one would use to combine the variances of multiple noisy measurements when doing sensor fusion.

Finally, we introduce a prior area per pixel, $w \in (0, \infty)$. The interpretation of the prior is that there is some pre-existing photograph of the environment (e.g. an initial reconnaissance photograph), from which we can get a base-line area per pixel measurement. This is compatible with the rest of our scenario, since we will assume that the robots have knowledge of the geometry of the environment $Q$, and some notion of information content over it, $\phi(q)$, which could also be derived from a pre-existing photograph. This pre-existing information can be arbitrarily vague ($w$ can be arbitrarily large) but it must exist. The prior also has the benefit of making the cost function finite for all robot positions. It is combined with the camera sensors as if it were another camera to get

$$\frac{\text{area}}{\text{pixel}} = \Big(\sum_{i=1}^{n} f(p_i, q)^{-1} + w^{-1}\Big)^{-1},$$

Let $\mathcal{N}_q$ be the set of indices of cameras for $q$ is in the field of view, $\mathcal{N}_q = \{i \mid q \in \mathcal{B}_i)$.

We can now write the area per pixel function as

$$g_q(f(p_1, q), \dots, f(p_n, q)) = \big( \sum_{i \in \mathcal{N}_q} f(p_i, q)^{-1} + w^{-1} \big)^{-1}. \tag{6.5}$$

which is very similar to the mixing function $g_\alpha$, with $\alpha = -1$, from Chapter 3. The only difference is the prior $w$. In a probabilisitic setting, as in Section 3.3.2, this would represent the variance of a Gaussian prior. Forming the standard cost function as in Chapter 3, $g_q$ is integrated over the environment to give the cost function

$$\mathcal{H}(p_1, \dots, p_n) = \int_Q g_q(f(p_1, q), \dots, f(p_n, q)) \phi(q) \, dq. \tag{6.6}$$

We will often refer to $g$ and $\mathcal{H}$ without their arguments. Now we can pose the multi-camera optimization problem,

$$\min_{(p_1, \dots, p_n) \in \mathcal{P}^n} \mathcal{H}. \tag{6.7}$$

The cost function (6.6) is of a general form valid for any area per pixel function $f(p_i, q)$, and for any camera state space $\mathcal{P}$ (including cameras that can can swivel on gimbels). We proceed with the special case of downward facing cameras, where $\mathcal{P} = \mathbb{R}^3$ and $f(p_i, q)$ is from (6.3) for the remainder of the chapter.

## 6.3 Distributed Control

We will take the gradient of (6.6) and find that it is distributed among agents. This will lead to a gradient-based controller. We will use the notation $g_{q,i}$ to mean

$$g_{q,i}(f(p_1, q), \dots, f(p_n, q)) = \big( \sum_{j \in \mathcal{N}_q \setminus \{i\}} f(p_j, q)^{-1} + w^{-1} \big)^{-1}, \tag{6.8}$$

where $\mathcal{N}_q \setminus \{i\}$ is the set of all indices in $\mathcal{N}_q$, except for $i$.

**Theorem 6.1 (Gradient Component)** *The gradient of the cost function $\mathcal{H}(p_1, \dots, p_n)$ with respect to a robot's position $p_i$, using the area per pixel function in (6.3) is given* *by*

$$\frac{\partial \mathcal{H}}{\partial c_i} = \int_{Q\cap\partial\mathcal{B}_i} (g_q - g_{q,i}) \frac{(q - c_i)}{\|q - c_i\|} \phi(q)\, dq, \tag{6.9}$$

*and*

$$\begin{aligned}\frac{\partial \mathcal{H}}{\partial z_i} = &\int_{Q\cap\partial\mathcal{B}_i} (g_q - g_{q,i})\phi(q) \tan\theta \, dq \\ &- \int_{Q\cap\mathcal{B}_i} \frac{2g_q^2}{a(b - z_i)^3} \phi(q)\, dq.\end{aligned} \tag{6.10}$$

**Proof 6.1** *We can break up the domain of integration into two parts as*

$$\mathcal{H} = \int_{Q\cap\mathcal{B}_i} g_q \phi(q)\, dq + \int_{Q\backslash\mathcal{B}_i} g_q \phi(q)\, dq.$$

*Only the integrand in the first integral is a function of $p_i$ since the condition $i \in \mathcal{N}_q$ is true if and only if $q \in \mathcal{B}_i$ (from the definition of $\mathcal{N}_q$). However the boundaries of both terms are functions of $p_i$, and will therefore appear in boundary terms in the derivative. Using the standard rule for differentiating an integral, with the symbol $\partial\cdot$ to mean boundary of a set, we have*

$$\begin{aligned}\frac{\partial \mathcal{H}}{\partial p_i} = &\int_{Q\cap\mathcal{B}_i} \frac{\partial g_q}{\partial p_i} \phi(q)\, dq \\ + &\int_{\partial(Q\cap\mathcal{B}_i)} g_q \phi(q) \frac{\partial q_{\partial(Q\cap\mathcal{B}_i)}}{\partial p_i}^T n_{\partial(Q\cap\mathcal{B}_i)}\, dq \\ + &\int_{\partial(Q\backslash\mathcal{B}_i)} g_{q,i} \phi(q) \frac{\partial q_{\partial(Q\backslash\mathcal{B}_i)}}{\partial p_i}^T n_{\partial(Q\backslash\mathcal{B}_i)}\, dq,\end{aligned} \tag{6.11}$$

*where $q_{\partial\cdot}$ is a point on the boundary of a set expressed as a function of $p_i$, and $n_{\partial\cdot}$ is the outward pointing normal vector of the boundary of the set. Decomposing the boundary further, we find that $\partial(Q \cap \mathcal{B}_i) = (\partial Q \cap \mathcal{B}_i) \cup (Q \cap \partial\mathcal{B}_i)$ and $\partial(Q\backslash\mathcal{B}_i) = (\partial Q\backslash\mathcal{B}_i) \cup (Q \cap \partial\mathcal{B}_i)$. But points on $\partial Q$ do not change as a function of $p_i$, therefore* *we have*

$$\frac{\partial q_{(\partial Q \cap \mathcal{B}_i)}}{\partial p_i} = 0 \quad \forall q \in \partial Q \cap \mathcal{B}_i$$
$$\text{and} \quad \frac{\partial q_{(\partial Q \backslash \mathcal{B}_i)}}{\partial p_i} = 0 \quad \forall q \in \partial Q \backslash \mathcal{B}_i.$$

*Furthermore, everywhere in the set $Q \cap \partial\mathcal{B}_i$ the outward facing normal of $\partial(Q\backslash\mathcal{B}_i)$ is the negative of the outward facing normal of $\partial(Q \cap \mathcal{B}_i)$,*

$$n_{\partial(Q\backslash\mathcal{B}_i)} = -n_{(\partial(Q\cap\mathcal{B}_i)} \quad \forall q \in Q \cap \partial\mathcal{B}_i.$$

*Simplifying (6.11) leads to*

$$\frac{\partial \mathcal{H}}{\partial c_i} = \int_{Q\cap\partial\mathcal{B}_i} (g_q - g_{q,i})\phi(q) \cdot \frac{\partial q_{(Q\cap\partial\mathcal{B}_i)}}{\partial c_i}^T n_{(Q\cap\partial\mathcal{B}_i)}\, dq. \tag{6.12}$$

*and*

$$\frac{\partial \mathcal{H}}{\partial z_i} = \int_{Q\cap\partial\mathcal{B}_i} (g_q - g_{q,i})\phi(q) \cdot \frac{\partial q_{(Q\cap\partial\mathcal{B}_i)}}{\partial z_i}^T n_{(Q\cap\partial\mathcal{B}_i)}\, dq - \int_{Q\cap\mathcal{B}_i} \frac{2g_q^2}{a(b-z_i)^3}\phi(q)\, dq, \tag{6.13}$$

*where we used the fact that $\partial g_q/\partial c_i = [0 \quad 0]^T$, and a straightforward calculation yields $\partial g_q/\partial z_i = -2g_q^2/(a(b-z_i)^3)$. Now we solve for the boundary terms,*

$$\frac{\partial q_{(Q\cap\partial\mathcal{B}_i)}}{\partial c_i}^T n_{(Q\cap\partial\mathcal{B}_i)} \text{ and } \frac{\partial q_{(Q\cap\partial\mathcal{B}_i)}}{\partial z_i}^T n_{(Q\cap\partial\mathcal{B}_i)},$$

*which generally can be found by implicitly differentiating the constraint that describes the boundary. Henceforth we will drop the subscript on $q$, but it should be understood that we are referring to points, $q$, constrained to lie on the set $Q \cap \partial\mathcal{B}_i$. A point $q$ on* *the boundary set $Q \cap \partial \mathcal{B}_i$ will satisfy*

$$\|q - c_i\| = z_i \tan \theta, \tag{6.14}$$

*and the outward facing normal on the set $Q \cap \mathcal{B}_i$ is given by*

$$n_{(Q \cap \partial \mathcal{B}_i)} = \frac{(q - c_i)}{\|q - c_i\|}.$$

*Differentiate (6.14) implicitly with respect to $c_i$ to get*

$$\left(\frac{\partial q}{\partial c_i}^T - I_2\right)(q - c_i) = 0,$$

*where $I_2$ is the $2 \times 2$ identity matrix, therefore*

$$\frac{\partial q}{\partial c_i}^T \frac{(q - c_i)}{\|q - c_i\|} = \frac{(q - c_i)}{\|q - c_i\|},$$

*which gives the boundary terms for (6.12). Now differentiate (6.14) implicitly with respect to $z_i$ to get*

$$\frac{\partial q}{\partial z_i}^T \frac{(q - c_i)}{\|q - c_i\|} = \tan \theta,$$

*which gives the boundary term for (6.13). The derivative of the cost function $\mathcal{H}$ with respect to $p_i$ can now be written as in Theorem 6.1*

**Remark 6.1 (Intuition)** *We will consider a controller that moves a robot in the opposite direction of its gradient component. In which case, the single integral for the lateral component (6.9) causes the robot to move to increase the amount of the environment in its field of view, while also moving away from other robots $j$ whose field of view overlaps with its own. The vertical component (6.10) has two integrals with competing tendencies. The first integral causes the robot to move up to bring more of the environment into its field of view, while the second integral causes it to move down to get a better look at the environment already in its field of view.*

**Remark 6.2 (Requirements)** *Both the lateral (6.9) and vertical (6.10) components can be computed by robot $i$ with knowledge of 1) its own position, $p_i$, 2) the extent of the environment $Q$, 3) the information per area function $\phi(q)$, and 4) the positions of all other robots whose fields of view intersect with its own (which can be found by communication or sensing).*

**Remark 6.3 (Network Requirements)** *The requirement that a robot can communicate with all other robots whose fields' of view intersect with its own describes a minimal network graph for our controller to be feasible. In particular, we require the network to be at least a proximity graph in which all agents $i$ are connected to all other agents $j \in \mathcal{N}_i$, where $\mathcal{N}_i = \{j \mid Q \cap \mathcal{B}_i \cap \mathcal{B}_j \neq \emptyset, i \neq j\}$. The controller can be run over a network that is a subgraph of the required proximity graph, in which case performance will degrade gracefully as the network becomes more sparse.*

We form the controller using the gradient according to the standard multi-robot controller in Equation (2.7),

$$\dot{p}_i = -k\frac{\partial \mathcal{H}}{\partial p_i}. \tag{6.15}$$

We can prove the convergence of this controller to locally minimize the aggregate information per area.

**Theorem 6.2 (Convergence)** *For a network of $n$ robots with the closed-loop dynamics in (6.15),*

$$\lim_{t\to\infty} \frac{\partial \mathcal{H}}{\partial p_i} = 0 \quad \forall i \in \{1, \ldots, n\}. \tag{6.16}$$

**Proof 6.2** *The proof is an application of Theorem 2.3 from Chapter 2. The closed-loop dynamics $\dot{p}_i = -\partial\mathcal{H}/\partial p_i$ are a gradient system. We must only show that all evolutions of the system are bounded. To see this, consider a robot at $p_i$ such that $Q \cap \mathcal{B}_i = \emptyset$. Then $\dot{p}_i = 0$ for all time (if the field of view leaves $Q$, the robot stops for all time), so $c_i(t)$ is bounded. Given $Q \cap \mathcal{B}_i \neq \emptyset$, $\mathcal{H}$ is radially unbounded (i.e.*

*coercive) in $z_i$, therefore $\dot{\mathcal{H}} \leq 0$ implies that $z_i$ is bounded for all time. Therefore, all conditions of Theorem 2.3 are satisfied and the trajectories of the system converge to the set of critical points.*

To be more precise, there may exist configurations at which $\frac{\partial \mathcal{H}}{\partial p_i} = 0 \ \forall i$ that are saddle points or local maxima of $\mathcal{H}$. However, since the controller is a gradient controller, only isolated local *minima* of $\mathcal{H}$ are stable equilibria, according to Theorem 2.4.

This controller can be implemented in a discretized setting as Algorithm 2. In general, the integrals in the controller must be computed using a discretized approximation. Let $\widehat{Q \cap \partial \mathcal{B}_i}$ and $\widehat{Q \cap \mathcal{B}_i}$ be the discretized sets of gird points representing the sets $Q \cap \partial \mathcal{B}_i$ and $Q \cap \mathcal{B}_i$, respectively. Let $\Delta q$ be the length of an arc segment for the discretized set $\widehat{Q \cap \partial \mathcal{B}_i}$, and the area of a grid square for the discretized set $\widehat{Q \cap \mathcal{B}_i}$. A simple algorithm that approximates (6.15) is then given in Algorithm 2.

```
**Algorithm 2** Discretized Controller
Require: Robot $i$ knows its position $p_i$, the extent environment $Q$, and the information per area function $\phi(q)$.
Require: Robot $i$ can communicate with all robots $j$ whose field of view intersects with its own.
  loop
    Communicate with neighbors to get $p_j$
    Compute and move to
        $c_i(t+\Delta t) = c_i(t)$
        $-k \sum_{q \in \widehat{Q \cap \partial \mathcal{B}_i}} (g_q - g_{q,i}) \frac{(q-c_i)}{\|q-c_i\|} \phi(q) \Delta q$
    Compute and move to
        $z_i(t+\Delta t) = z_i(t)$
        $-k \sum_{q \in \widehat{Q \cap \partial \mathcal{B}_i}} (g_q - g_{q,i}) \phi(q) \tan\theta \Delta q$
        $+k \sum_{q \in \widehat{Q \cap \mathcal{B}_i}} \frac{2 g_q^2}{a(b-z_i)^3} \phi(q) \Delta q$
  end loop
```

**Remark 6.4 (Time Complexity)** *To determine the computational complexity of this algorithm, let us assume that there are $m$ points in both sets $\widehat{Q \cap \partial \mathcal{B}_i}$ and $\widehat{Q \cap \mathcal{B}_i}$.*

*We can now calculate the time complexity as*

$$T(n,m) \leq \sum_{j=1}^{m}(O(1) + \sum_{k=1}^{n} O(1)) +$$
$$\sum_{j=1}^{m}(O(1) + \sum_{k=1}^{n} O(1) + \sum_{k=1}^{n-1} O(1)) \in O(nm).$$

*When calculating the controller for all robots on a centralized processor (as was done for the simulations in Section 6.5), the time complexity becomes* $T(n,m) \in O(n^2 m)$.

**Remark 6.5 (Adaptivity)** *The controller is adaptive in the sense that it will stably reconfigure if any number of robots fail. It will also work with nonconvex environments,* $Q$*, including disconnected ones. In the case of a disconnected environment, the robots may (or may not, depending on the specific scenario) split into a number of sub-groups that are not in communication with one another. The controller can also track changing environments,* $Q$*, and changing information per area functions,* $\phi(q)$*, provided these quantities change slowly enough. This is not addressed by the proof, but has been shown to be the case in simulation studies.*

**Remark 6.6 (Control Gains and Robustness)** *The proportional control gain,* $k$*, adjusts the aggressiveness of the controller. In a discretized implementation one should set this gain low enough to provide robustness to discretization errors and noise in the system. The prior area per pixel,* $w$*, adjusts how much of the area* $Q$ *will remain uncovered in the final configuration. It should be chosen to be as large as possible, but as with* $k$*, should be small enough to provide robustness to discretization errors and noise in the system.*

### 6.3.1 Rectangular Field of View

Until this point we have assumed that the camera's field of view, $\mathcal{B}_i$ is a circle, which eliminates a rotational degree of freedom. Of course, actual cameras have a rectangular CCD array, and therefore a rectangular field of view. In this section we revisit the gradient component in Theorem 6.1 and calculate it for a rectangular field of view and a robot with a rotational degree of freedom.

Let the state space of $p_i = [c_i^T \quad z_i \quad \psi_i]^T$ be $\mathcal{P} = \mathbb{R}^3 \times \mathbb{S}$, where $\psi_i$ is the rotation angle. Define a rotation matrix

$$R(\psi_i) = \begin{bmatrix} \cos\psi_i & \sin\psi_i \\ -\sin\psi_i & \cos\psi_i \end{bmatrix}, \tag{6.17}$$

where $R(\psi_i)q$ rotates a vector $q$ expressed in the global coordinate frame, to a coordinate frame aligned with the axes of the rectangular field of view. As is true for all rotation matrices, $R(\psi_i)$ is orthogonal, meaning $R(\psi_i)^T = R(\psi_i)^{-1}$. Using this matrix, define the field of view of robot $i$ to be

$$\mathcal{B}_i = \{q \mid |R(\psi_i)(q - c_i)| \leq z_i \tan\theta\}, \tag{6.18}$$

where $\theta = [\theta_1, \theta_2]^T$ is a vector with two angles which are the two half-view angles associated with two perpendicular edges of the rectangle, as shown in Figure 6-4, and the $\leq$ symbol applies element-wise (all elements in the vector must satisfy $\leq$). We have to break up the boundary of the rectangle into each of its four edges. Let $l_k$ be the $k$th edge, and define four outward-facing normal vectors $n_k$, one associated with each edge, where $n_1 = [1 \quad 0]^T$, $n_2 = [0 \quad 1]^T$, $n_3 = [-1 \quad 0]$, and $n_4 = [0 \quad -1]$. The cost function, $\mathcal{H}(p_1, \ldots, p_n)$, is the same as for the circular case, as is the area per pixel function $f(p_i, q)$.

**Theorem 6.3 (Rectangular Gradient)** *The gradient of the cost function $\mathcal{H}(p_1, \ldots, p_n)$ with respect to a robot's position $p_i$ using the area per pixel function in (6.3) and the rectangular field of view in (6.18) is given by*

$$\frac{\partial \mathcal{H}}{\partial c_i} = \sum_{k=1}^{4} \int_{Q \cap l_k} (g_q - g_{q,i}) R(\psi_i)^T n_k \phi(q)\, dq, \tag{6.19}$$

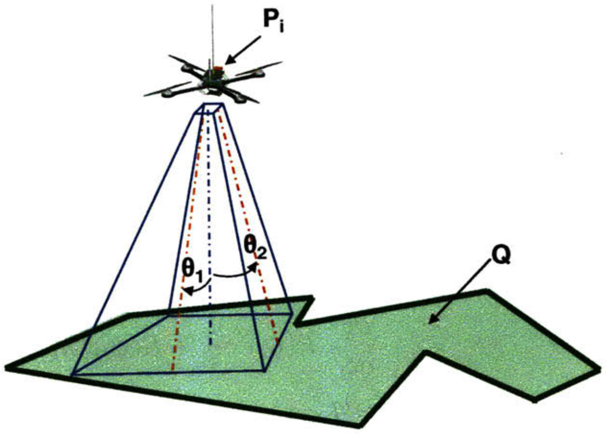


Figure 6-4: The geometry of a camera with a rectangular field of view is shown in this figure.

$$\frac{\partial \mathcal{H}}{\partial z_i} = \sum_{k=1}^{4} \int_{Q \cap l_k} (g_q - g_{q,i}) \tan \theta^T n_k \phi(q)\, dq - \int_{Q \cap \mathcal{B}_i} \frac{2 g_q^2}{a(b - z_i)^3} \phi(q)\, dq, \tag{6.20}$$

*and*

$$\frac{\partial \mathcal{H}}{\partial \psi_i} = \sum_{k=1}^{4} \int_{Q \cap l_k} (g_q - g_{q,i}) \cdot (q - c_i)^T R(\psi_i + \pi/2)^T n_k \phi(q)\, dq. \tag{6.21}$$

**Proof 6.3** *The proof is the same as that of Theorem 6.1 up to the point of evaluating the boundary terms. Equations (6.12) and (6.13) are true. Additionally the angular component is given by*

$$\frac{\partial \mathcal{H}}{\partial c_i} = \int_{Q \cap \partial \mathcal{B}_i} (g_q - g_{q,i}) \phi(q) \cdot \frac{\partial q_{(Q \cap \partial \mathcal{B}_i)}}{\partial \psi_i}^T n_{(Q \cap \partial \mathcal{B}_i)}\, dq.$$

*The constraint for points on the kth leg of the rectangular boundary is*

$$(q - c_i)^T R(\psi_i)^T n_k = z_i \tan\theta^T n_k,$$

*from (6.18). Differentiate this constraint implicitly with respect to $c_i$, $z_i$, and $\psi_i$ and solve for the boundary terms to get*

$$\begin{aligned}
\frac{\partial q}{\partial c_i}^T R(\psi_i)^T n_k &= R(\psi_i)^T n_k, \\
\frac{\partial q}{\partial z_i}^T R(\psi_i)^T n_k &= \tan\theta^T n_k, \\
\text{and} \quad \frac{\partial q}{\partial \psi_i}^T R(\psi_i)^T n_k &= -(q - c_i)^T R(\psi_i + \pi/2)^T n_k,
\end{aligned}$$

*where we have used the fact that*

$$\frac{\partial R(\psi_i)}{\partial \psi_i} = \begin{bmatrix} -\sin\psi_i & \cos\psi_i \\ -\cos\psi_i & -\sin\psi_i \end{bmatrix} = R(\psi_i + \pi/2).$$

*Break the boundary integrals into a sum of four integrals, one integral for each edge of the rectangle. The expression in Theorem 6.3 follows.*

**Remark 6.7 (Intuition)** *The terms in the gradient have interpretations similar to the ones for the circular field of view. The lateral component (6.19) has one integral which tends to make the robots move away from neighbors with intersecting fields of view, while moving to put its entire field of view inside of the environment $Q$. The vertical component (6.20) comprises two integrals. The first causes the robot to go up to take in a larger view, while the second causes it to go down to get a better view of what it already sees. The angular component (6.21) rotate the robot to get more of its field of view into the environment, while also rotating away from other robots whose field of view intersects its own. Computation of the gradient component for the rectangular field of view is of the same complexity as the circular case, and carries the same constraint on the communication topology.*

## 6.4 Experiments

We implemented Algorithm 2 on a group of three AscTec Hummingbird flying quad-rotor robots. Our experiments were performed at CSAIL, MIT in a laboratory equipped with a Vicon motion capture system. The robots' position coordinates $(x, y, z, yaw)$ were broadcast wirelessly at 50Hz via a 2.4 Ghz xBee module. Each robot was equipped with a custom ARM microprocessor module running a PID position control loop at 33Hz. Pitch and roll were fully stabilized by the commercial controller described in [38]. A schematic of the experimental setup is shown in Figure 6-5.

The coverage algorithm was implemented on the same onboard ARM modules, running asynchronously in a fully distributed fashion. The algorithm calculated way points ($c_i(t)$ and $z_i(t)$ from Algorithm 2) at 1Hz. This time-scale separation between the coverage algorithm and the PID controller was required to approximate the integrator dynamics assumed in (2.5). The camera parameters were set to $a = 10^{-6}$ and $b = 10^{-2}$m (which are typical for commercially available cameras), the field of view was $\theta = 35$deg, the information per area was a constant $\phi(q) = 1$, the prior area per pixel was $w = 10^{-6}\text{m}^2$, and the control gain was $k = 10^{-5}$. The environment to be covered was a skewed rectangle, 3.7m across at its widest, shown in white in Figure 6-6.

To test the effectiveness of the algorithm and its robustness to robot failures, we conducted experiments as follows: 1) three robots moved to their optimal positions using the algorithm, 2) one robot was manually removed from the environment, and the remaining two were left to reconfigure automatically, 3) a second robot was removed from the environment and the last one was left to reconfigure automatically. Figure 6-6 shows photographs of a typical experiment at the beginning (Figure 6-6(a)), after the first stage (Figure 6-6(b)), after the second stage (Figure 6-6(c)), and after the third stage (Figure 6-6(d)).

The initial positions are shown in Figure 6-6(a), the final positions of the three robots are shown in Figure 6-6(b), the final positions of the two after removing one is

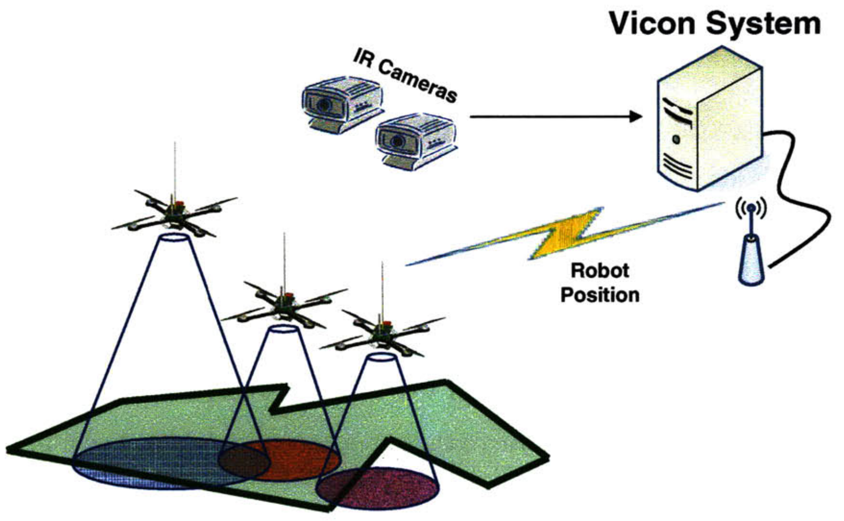
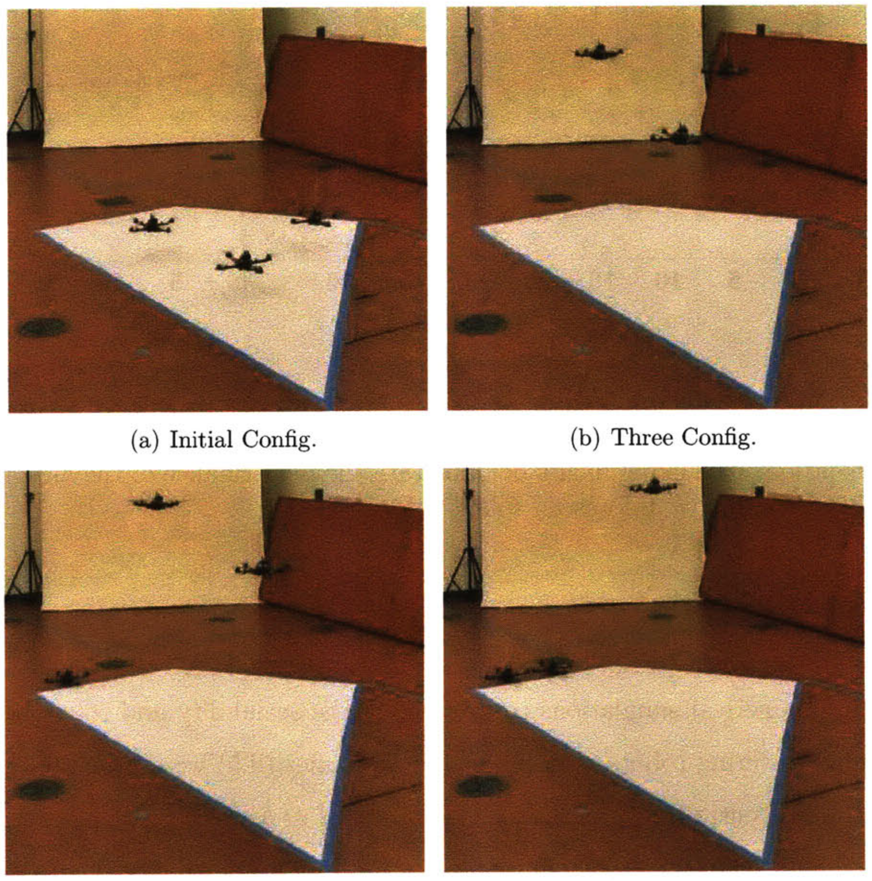


Figure 6-5: This figure shows the experimental setup. The robots positions were captured with an Vicon motion capture system. The robots used their position information to run the coverage algorithm in a distributed fashion.

shown in Figure 6-6(c), and the final position of the last robot after removing the second is shown in Figure 6-6(d). The coverage cost of the robots over the course of the whole experiment, averaged over 19 experiments, is shown in Figure 6-7, where the error bars represent one standard deviation. Notice that when one robot is removed, the cost function momentarily increases, then decrease as the remaining robots find a new optimal configuration. The algorithm proved to be robust to the significant, highly nonlinear unmodeled aerodynamic effects of the robots, and to individual robot failures. This chapter is accompanied by a video showing the experiments and numerical simulations.

We repeated the above experiment a total of 20 times. Of these 19 were successful, while in one experiment two of the robots collided in mid air. The collision was caused by an unreliable gyroscopic sensor, not by a malfunction of the coverage algorithm. With appropriate control gain values, collisions are avoided by the algorithm's natural tendency for neighbors to repel one another.



(a) Initial Config. (b) Three Config.

(c) Two Config. (d) One Config.

Figure 6-6: Frame shots from an experiment with three AscTec Hummingbird quad-rotor robots are shown. After launching from the ground (Figure 6-6(a)), the three robots stabilize in an optimal configuration (Figure 6-6(b)). Then one robot is manually removed to simulate a failure, and the remaining two move to a new optimal position (Figure 6-6(c)). Finally a second robot is removed and the last one stabilizes at an optimal position (Figure 6-6(d)). The robots move so that their fields of view (which cannot be seen in the snapshots) cover the environment, represented by the white polygon.

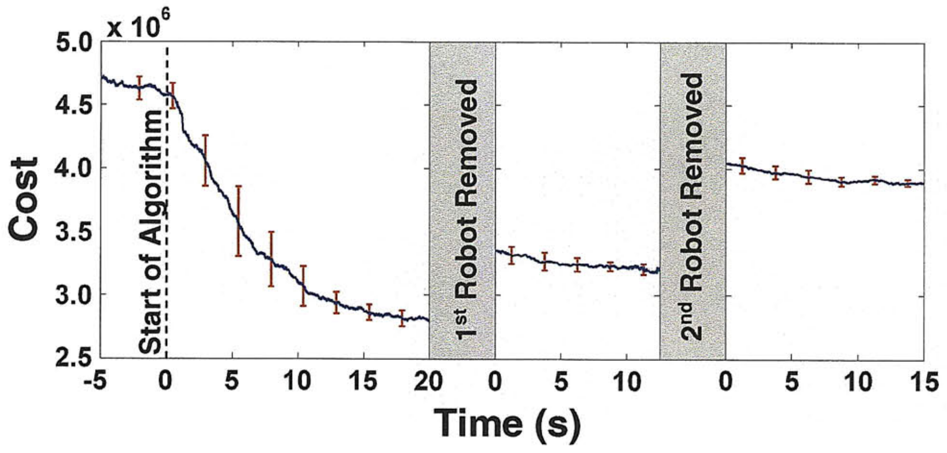
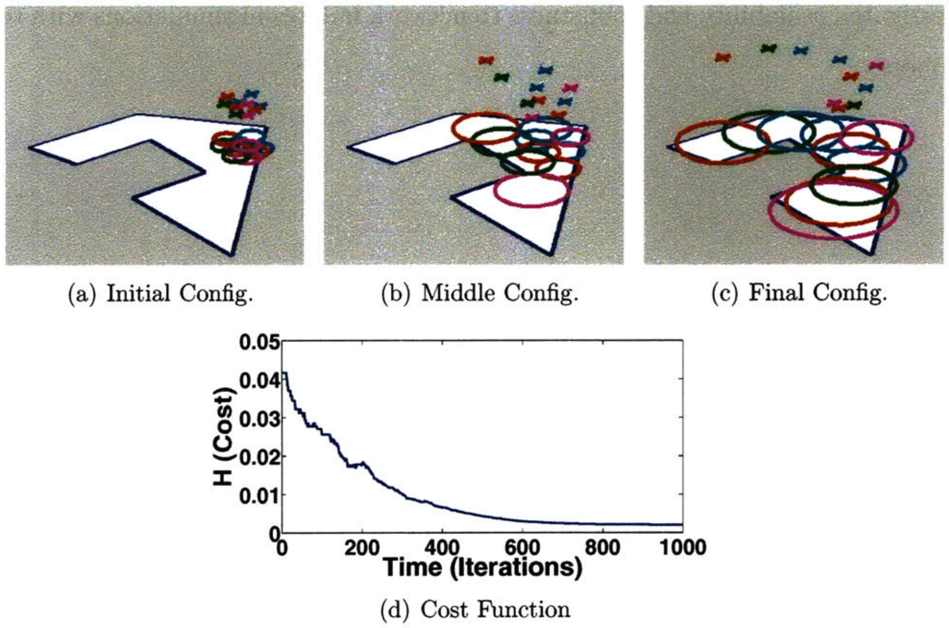


Figure 6-7: The cost function during the three stages of the experiment, averaged over 19 successful experiments, is shown in Figure 6-7. The error bars denote one standard deviation. The experiments demonstrate the performance of the algorithm, and its ability to adapt to unforeseen robot failures.

## 6.5 Simulations

We conducted numerical simulations to investigate the scalability and robustness of the algorithm. Hovering robots with integrator dynamics (2.5) were simulated using Algorithm 2 on a centralized processor. The values of $a$, $b$, $\theta$, $\phi$, $w$, and $k$ were the same as in the experiments. The simulations were over a non-convex environment, as shown in Figure 6-8. Communication constraints were modeled probabilistically. The probability of robot $i$ communicating with robot $j$ was calculated as a linear function of the distance between them decreasing from 1 at a distance of 0, to 0 at a distance of $R = 1.5m$, and the environment width was roughly $3m$. Uncertainty in the robots' velocity was modeled as white Gaussian noise with covariance of $I_3 \times 10^{-4} m^2/s^2$ (where $I_3$ is the $3 \times 3$ identity matrix). Figure 6-8 shows the results of a typical simulation with ten robots. The robots start in an arbitrary configuration and spread out and up so that their fields of view cover the environment. The decreasing value of the cost function $\mathcal{H}$ is shown in Figure 6-8(d). The function does not decrease smoothly because of the simulated communication failures, velocity


(a) Initial Config. (b) Middle Config. (c) Final Config.

(d) Cost Function

Figure 6-8: Results of a simulation with ten robots covering a nonconvex environment are shown. The $\times$'s mark the robot positions and the circles represent the fields of view of their cameras. Communication failures and noise on the robots' velocities are also modeled in the simulation. The initial, middle, and final configurations are shown in 6-8(a), 6-8(b), and 6-8(c), respectively. The decreasing value of the aggregate information per pixel function, $\mathcal{H}$, is shown in 6-8(d). The jaggedness of the curve is due to simulated communication failures, noise, and the discretized integral approximation.

noise, and discretized integral computation in Algorithm 2.

## 6.6 Synopsis

In this chapter we presented an application of the ideas from Chapter 3 to design a distributed control algorithm to allow hovering robots with downward facing cameras to cover an environment. We incorporate a realistic sensor model for the camera and, using this model, formulate the cost function $\mathcal{H}$ representing the aggregate information per pixel of the robots over the environment. The controller is proven to locally minimize the cost function, and can be used in nonconvex and disconnected environments. We implemented the algorithm on a group of three autonomous quad-rotor robots, and experimentally demonstrated robustness to unforeseen robot failures. We also investigated scalability and robustness to network failures in simulations with ten flying robots.

# Chapter 7

# Modeling Animal Herds

## 7.1 Introduction

In this chapter, we demonstrate an application of the potential field controller in Chapter 3 to model the dynamics of groups of animals and robots. The setting is one of system identification: we are presented with position data from a group of agents, and we want to learn a dynamical model with a potential field structure, as in Chapter 3, to represent the data. The method we present is general, however we will demonstrate the technique to model a group of cows. The cows are equipped with GPS sensors that give us position measurements over time. We use system identification techniques to learn the model, that is to tune the parameters of the model to fit the GPS data.

We wish to model groups of interacting dynamic agents, such as flocks, swarms, and herds, using measured data from those agents. For example, we would like to use the trajectories of people in a crowd to develop dynamical models that capture the behaviors of the crowd as a whole. This is a prohibitively complicated problem in general, however, we provide a practical solution by restricting our attention to a special model structure. We embrace a minimalist approach in that we use only position measurements, with a minimum of prior environmental information incorporated into the model. We propose a difference equation model that is decentralized and nonlinear, though it is designed to be linear-in-parameters. The Least Squares method is then used to fit model parameters to position data from a group of agents. Such a model may then be used, for example, to predict future states of the group, to determine individual roles of agents within the group (e.g. leaders vs. followers), or, ultimately, to control the group.

The most immediate application of these ideas is for virtual fencing of livestock [2, 15, 113], in which physical fences are replaced with sensor/actuator devices mounted on the animals. The animals' positions are monitored, and if they stray beyond a virtual fence line, the animals are given cues to return to the desired area. Our modelling techniques will be useful for virtual fencing in several ways. Firstly, our models lead to verified behavioral simulations that can be used to test virtual fencing algorithms in a simulation environment before they are implemented in a costly and time-consuming field test. Secondly, our dynamical models can be used to enhance the animal control algorithm itself, so that it works in conjunction with the animals' natural tendencies. Finally, since our model is inherently distributed and, because of our minimalist approach, requires little computational resources, we envision that the model can run online over the same network of animal-mounted sensor devices that carry out the virtual fencing algorithm. The distributed model can then be used to predict where the group is headed and inform the controller in real time. Simultaneously, the model can be updated to fit the most recent position data collected from the animals. This simultaneous model learning and model-based control is in the spirit of adaptive control.

In addition to livestock management applications, there are many other uses for learned models of distributed dynamical systems. In the case of people, the ability to model group behavior has numerous applications in surveillance, urban planning, and crowd control. Also, the models can be used to drive groups of robots to mimic the behavior of observed groups. This may be useful in reproducing collaborative behaviors exhibited in natural systems, or in producing decoy robots to participate with natural or engineered groups, and even to influence group behavior [39].

### 7.1.1 Related Work

The problem of learning models for groups of interacting dynamic agents lies at the intersection of two fields of research: modeling of distributed dynamical systems, and system identification. A vigorous body of work is emerging from the controls and robotics communities focused on analyzing models of flocks, swarms, and similar distributed dynamical systems. This work, however, has not considered using learning techniques to generate these models from data. Instead, it concentrates on the dynamical properties of models, such as stability of formations [34, 35, 104, 105, 116], asymptotic consensus of agent positions or velocities [27, 46, 71, 111], or designing local controllers from global specifications [7, 106]. These considerations are elemental in describing more complex social phenomena, but they are quite different from the question of learning models from data which we address in this chapter.

Conversely, the rich literature on learning dynamical systems from data, often called system identification, has not yet addressed models of distributed dynamical systems, such as the ones we consider in this thesis. Some related problems have been considered, however. For example, in [22] a system identification technique is used to model global properties of a swarm of robots over time using observed data from the robots. These properties include collision likelihoods of robots and transition probabilities among robot behaviors. There also has been considerable activity in learning behavioral models of individual natural agents. In [69] and [74], system identification is carried out on switching linear systems to learn models of the honey bee waggle dance and human hand motion, respectively, and in [28] a technique is used to find Motion Description Language (MDL) codes from observed ants. These works, however, do not consider group interactions, but investigate the action of individuals isolated from their group roles.

It is our intention in this chapter to bridge the gap between these two research communities by applying system identification techniques to distributed model structures. In addition to this cross-pollination of ideas, we also contribute a new technique for modelling general vector fields (i.e. non-gradient vector fields) in a way that is amenable to system identification. We also pursue our ideas from theory through implementation by testing our method with data from natural agents.

For this purpose, we developed a hardware platform to record position and orientation information of groups of free-ranging cows. The hardware platform is capable of recording GPS position information, head orientation, and is able to provide sound and electrical stimuli, though no stimuli were administered during the data collection for this study. We demonstrate our model learning technique by fitting it to GPS data collected from a group of three and a group of ten free ranging cows, and validate the resulting models by testing the whiteness of the residual error, and by comparing global statistics of simulations verse the actual data. Previous works have considered animal mounted sensor network devices, such as the ZebraNet platform [47], and the sensor/actuator devices described in [2, 15, 84, 113] for automatic livestock management. Our device has several innovations for applying animal control stimuli and for using communication between devices over a network, however we do not describe these innovations in detail in this chapter. In the context of this chapter, the devices were used as a means to collect GPS data for learning and validating dynamical models.

### 7.1.2 Contributions

The main contribution in this chapter is as follows:

1. A model similar to the potential field model of Chapter 3 is proposed for modeling the herding behavior of cows and other groups of agents. Least squares systems identification is then used to tune the parameters of the model (similar to the on-line learning in Chapter 4) to fit GPS data from actual cows. We also analyze the predictive ability of the model and simulate an application to deriving controllers for robots to behave like a herd of cows.

The remainder of this chapter is organized as follows. The model structure is described in Section 7.2. The application of system identification to identify model parameters is described in Section 7.3, along with a review of basic system identification techniques in Section 7.3.1. Our data collection device and experimental method are described in Section 7.4. Results of the system identification technique are presented in Section 7.5 with GPS tracking data from a group of three cows and a group of ten cows, and the quality of the learned models are evaluated in Section 7.5.3. Finally, in Section 7.6 we use a learned model to control a group of mobile robots to behave like the group of three cows. Simulation results of the group of robots are presented. A synopsis and directions for future work are given in Section 7.7.

## 7.2 Model Description

We consider a linear-in-parameters model structure with three naturally distinct parts to describe the motion of coupled physical agents moving over a plane surface. Firstly, each agent is given internal dynamics to enforce the constrains of Newtons laws. Secondly, a force[1] is applied to each agent from its interaction with each of the other agents in the group. Thirdly, a force is applied to each agent as a function of its position in the environment. All remaining effects are modeled as a white noise process.

Throughout this section, we refer to free parameters as $\theta$, and features, or regressors, are denoted by $\phi$. It should be understood that the parameters $\theta$ are left unknown for now. In Section 7.3 we describe how position data is used to tune these parameters to fit the data. A schematic showing the different parts of the model learning process are shown in Figure 7-1.

[1]In this chapter the term "force" is used in a metaphoric sense. When we talk of a "force" we are referring to the intention of the agent to accelerate in a particular way using it's own motive mechanisms.

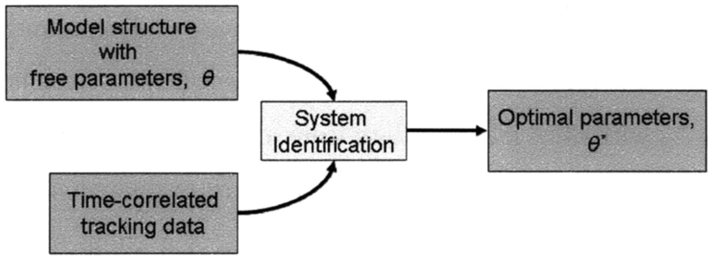

Model structure with free parameters, $\theta$
System Identification
Optimal parameters, $\theta^*$
Time-correlated tracking data

Figure 7-1: A schematic of the method of system identification is shown in this figure. The time correlated (not independent identically distributed) data and the model structure are combined in an optimization procedure to get model parameters tuned to fit the data.

### 7.2.1 Individual Agent Dynamics

Given a group of $m$ agents, the proposed model structure for an individual agent $i \in \{1, \ldots, m\}$ can be written in state-space, difference equation form as

$$x_i^{\tau+1} = \begin{bmatrix} 1 & 0 & \Delta t & 0 \\ 0 & 1 & 0 & \Delta t \\ 0 & 0 & a_i & 0 \\ 0 & 0 & 0 & a_i \end{bmatrix} x_i^\tau + \begin{bmatrix} 0 & 0 \\ 0 & 0 \\ 1 & 0 \\ 0 & 1 \end{bmatrix} \left( \sum_{j=1, j\neq i}^{m} f_{ij}(p_i^\tau, p_j^\tau) + g_i(p_i^\tau) + w_i^\tau \right). \quad (7.1)$$

Agent $i$'s state $x_i^\tau = [e_i^\tau \; n_i^\tau \; u_i^\tau \; v_i^\tau]^T$ consists of its East position, North position, Eastern component of velocity, and Northern component of velocity after the $\tau$th iteration, and its position is given by, $p_i^\tau = [e_i^\tau \; n_i^\tau]^T$. The time step $\Delta t$ is given by $t^{\tau+1} - t^\tau$, and we assume it is constant for all $\tau$. The term $a_i$ represents damping, $a_i = 1$ for zero damping, and $|a_i| < 1$ for stable systems. The function $f_{ij}(p_i^\tau, p_j^\tau)$ determines the coupling force applied by agent $j$ to agent $i$. The function $g_i(p_i^\tau)$ represents the force applied by the environment to the agent at point $p_i^\tau$. Finally, $w_i^\tau$ is a zero-mean, stationary, Gaussian white noise process uncorrelated with $p_j$ $\forall j$ used to model the unpredictable decision-motive processes of agent $i$. Nonholonomic constraints which are often present in mobile agents, such as people, cattle, or automobiles, are neglected in this treatment, though they could be incorporated with an increase in the complexity of the model structure. Note that the force terms are only applied to

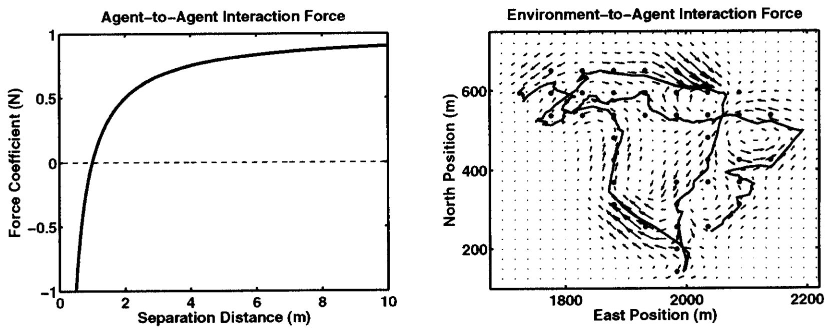


Figure 7-2: The magnitude of the agent-to-agent interaction force is shown on the left for $\theta_1 = \theta_2 = 1$. On the right, the vector field representing the force felt by an agent at each point on the plane is shown for an example agent trajectory. The swirling patterns evident in the field are made possible by a novel parameterization.

affect changes in velocity in accordance with Newton's second law.

### 7.2.2 Agent-to-Agent Interaction Force

Dropping the $\tau$ superscripts for clarity, the form of the agent coupling force $f_{ij}(p_i, p_j)$ is given by

$$f_{ij}(p_i, p_j) = \left(\theta_{1_{ij}} - \frac{\theta_{2ij}}{\|p_j - p_i\|}\right) \frac{n_{ij}}{(m-1)}, \tag{7.2}$$

where $n_{ij} = (p_j - p_i)/\|p_j - p_i\|$ is the unit vector along the line from $p_i$ to $p_j$ (henceforth, $\|\cdot\|$ will denote the $\ell^2$ norm). The factor $(m-1)$ is included to normalize the force exerted by one neighbor by the total number of neighbors.

This is the simplest of a family of force laws commonly used in computational models of physical, multi-body systems. The important feature of this family is that an agent is repulsed from its neighbor at close distances and attracted to its neighbor at far distances. To see this property clearly, examine the magnitude of force exerted by one neighbor $(m - 1 = 1)$ given by $\|f_{ij}\| = \theta_{1_{ij}} - \theta_{2_{ij}}/\|p_j - p_i\|$, and shown in the left of Figure 7-2. Notice that with $\theta_{1_{ij}} > 0$ and $\theta_{2_{ij}} > 0$ the desired characteristic is achieved. Indeed, as $\|p_j - p_i\| \to 0$, $\|f_{ij}\| \to -\infty$, which is repulsive, while $\|p_j - p_i\| \to \infty$, $\|f_{ij}\| \to \theta_{1_{ij}} > 0$, which is attractive. Other, similar force laws can be created to produce unbounded attraction as $\|p_j - p_i\| \to \infty$ and zero attraction as $\|p_j - p_i\| \to \infty$. We chose this law for its simplicity. The function can equivalently be expressed as the gradient of a potential function.

After some manipulation, the sum of $f_{ij}$ over all neighbors $j$ can be expressed as

$$\sum_{j \neq i} f_{ij} = \begin{bmatrix} \phi_{fu_i} \\ \phi_{fv_i} \end{bmatrix} \theta_{f_i}, \tag{7.3}$$

where

$$\begin{aligned}
\phi_{fu_i} &= \begin{bmatrix} \frac{(e_1-e_i)}{\|p_1-p_i\|} & \cdots & \frac{(e_m-e_i)}{\|p_m-p_i\|} & \frac{-(e_1-e_i)}{\|p_1-p_i\|^2} & \cdots & \frac{-(e_m-e_i)}{\|p_m-p_i\|^2} \end{bmatrix} \frac{1}{(m-1)} \\
\phi_{fv_i} &= \begin{bmatrix} \frac{(n_1-n_i)}{\|p_1-p_i\|} \cdots & \frac{(n_m-n_i)}{\|p_m-p_i\|} & \frac{-(n_1-n_i)}{\|p_1-p_i\|^2} & \cdots & \frac{-(n_m-n_i)}{\|p_m-p_i\|^2} \end{bmatrix} \frac{1}{(m-1)}, \text{ and} \\
\theta_{f_i} &= \begin{bmatrix} \theta_{1_{i1}} & \cdots & \theta_{1_{im}} & \theta_{2_{i1}} & \cdots & \theta_{2_{im}} \end{bmatrix}^T,
\end{aligned}$$

and where the indices $j = i$ are excluded from the above vectors (since we do not want an agent to feel a force from itself). This notation will be useful in what follows.

The agent-to-agent force law with the dynamics described above gives a so called potential field based flocking model, the analytical properties of which have been treated extensively in the controls and robotics literature [34, 35, 104, 105, 116]. The environment-to-agent force described below makes our model rather different however, and the inclusion of the noise term $w_i^\tau$ makes the model a random process, which is fundamentally different from the deterministic systems treated in those works.

### 7.2.3 Environment-to-Agent Interaction Force

The agent's preference for certain paths in the environment is modeled as a nonlinear mapping from each point on the plane to a force vector felt by the agent. To this end, two networks of Gaussian basis functions are used, one for each of two perpendicular force components.

In particular, the function $g_i(p_i)$ can be written

$$g_i(p_i) = \begin{bmatrix} \theta_{u_{i1}} & \cdots & \theta_{u_{in}} \\ \theta_{v_{i1}} & \cdots & \theta_{v_{in}} \end{bmatrix} \begin{bmatrix} \phi_{g_1}(p_i) \\ \vdots \\ \phi_{g_n}(p_i) \end{bmatrix}, \tag{7.4}$$

where

$$\phi_{g_{ik}}(p_i) = \frac{1}{2\pi\sigma_{ik}^2} \exp(-\frac{\|p_i - \gamma_{ik}\|^2}{2\sigma_{ik}^2}) \tag{7.5}$$

is the bi-variate Gaussian function, and $k \in \{1, \ldots, n\}$. Each Gaussian is centered at $\gamma_{ik}$, with standard deviation $\sigma_{ik}$, and its strength is represented by the unknown parameters $\theta_{u_{ik}}$ for the Eastern component, and $\theta_{v_{ik}}$ for the Northern component. Gaussian basis functions were chosen for their familiarity; the objective being to demonstrate the modeling approach with a minimum of complications. A number of other basis function types could be used, including wavelets, sigmoidal functions, or splines.

It is important to note that a vector-field parameterized in this way is *not* a potential gradient. A potential gradient field cannot admit circulation around closed paths.[2] We introduce a non-gradient parameterization to enable circulation, as one can imagine agents intending to traverse closed orbits on the plane. For example, a cow may have a routine of passing between a water source, a shaded tree, and a grassy patch in a periodic fashion.

Figure 7-2, on the right, shows a plot of an example force-field parameterized in the above way. The arrows show the forces induced by the field, the heavy dots show the centers of the Gaussian functions, $\gamma_{ik}$, and the curve shows the path of an agent over the vector field. The swirling patterns evident in the vector field would be impossible if it were a gradient field.

The expression in (7.4) can be put into a different form to match that of (7.3). In

[2]proof: Let $\Psi(p)$ be a potential function and $V(p) = -\text{grad}(\Psi)$ its gradient field. Then $\text{curl}(V) = \text{curl}(-\text{grad}(\Psi)) = 0$, thus by Green's Theorem, $\oint_s V ds = \int_{A_s} \text{curl}(V) dA = 0$, where $s$ is any closed curve on the plane, and $A_s$ is the area enclosed by $s$.

particular

$$g_i(p_i) = \begin{bmatrix} \phi_{gu_i} \\ \phi_{gv_i} \end{bmatrix} \theta_{g_i}, \tag{7.6}$$

where

$$\begin{aligned} \phi_{gu_i} &= [\ \phi_{g_1} \ \ \cdots \ \ \phi_{g_n} \ \ \cdots \ \ 0 \ \ \cdots\ ], \\ \phi_{gv_i} &= [\ \cdots \ \ 0 \ \ \cdots \ \ \phi_{g_1} \ \ \cdots \ \ \phi_{g_n}\ ], \text{ and} \\ \theta_{g_i} &= [\ \theta_{u_{i1}} \ \ \cdots \ \ \theta_{u_{in}} \ \ \theta_{v_{i1}} \ \ \cdots \ \ \theta_{v_{in}}\ ]^T. \end{aligned}$$

This form will become useful in what follows.

To consider the computational complexity of this model, consider that the number of agent-to-agent interaction terms grows as the square of the number of agents, $O(m^2)$ and, in the worst case, the number of environment-to-agent interaction terms grows as the product of the time duration and the number of agents $O(Tm)$. Therefore computing successive iterations of the model, not to mention learning the model parameters, will become intractable as the size of the group approaches hundreds or thousands of members. However, we can alleviate these difficulties in a natural way. Firstly, if the area in which the agents move is bounded, the environment-to-agent interaction terms will approach $O(m)$ as the entire area is explored by all the agents. Also, for large groups we could simply add a finite communication radius around each agent, so that neighbor agents outside that radius do not produce a force. This would limit the complexity of agent-to-agent parameters to $O(m)$. Thus we can modify the model to have an overall complexity linear in the number of agents. Also, the model is naturally decentralized, thus it could easily be implemented on a network of, say, $m$ processors, reducing the computation time to a constant independent of the size of the group. In this chapter we do not consider such implementation issues, and the groups of agents we deal with are small enough that computation speed is not a concern.

## 7.3 System Identification with Least-Squares Fitting

We will provide a brief introduction to the field of system identification. Then we will use a Least Squares method to identify optimal parameters for our model. We will also discuss recursive methods for Least Squares fitting that can be used to tune parameters for our model on-line as data is collected.

### 7.3.1 Method Overview

In this section we employ the tools of system identification [59], the basics of which are briefly reviewed here as they may be unfamiliar to the reader. If a stochastic dynamical system is such that its state at the next time step is determined by its state at the current time step and the inputs at the current time step, a state space model of its dynamics can be formed as a difference equation,

$$x^{\tau+1} = F(x^{\tau}, u^{\tau}, w^{\tau}, \tau), \tag{7.7}$$

where $x$ is the state, $u$ is the input, $w$ is a zero mean, stationary, Gaussian white noise process and $\tau$ is the discrete time index. Furthermore, we may formulate a model structure in which several of the parameters, $\theta$, of the model are unknown. If these parameters are time-invariant and occur linearly in the function $F$, and if the noise is additive, we can write the model as

$$x^{\tau+1} = \phi(x^{\tau}, u^{\tau}, \tau)\theta + w^{\tau}, \tag{7.8}$$

where $\phi$ is a row vector of functions of the state, input, and time (these are called statistics, regressors, or features depending upon the research field in which they are used) and $\theta$ is a column vector of the unknown parameters. Suppose we have some arbitrary value of the parameters of the system, $\theta$. Then we can interpret (7.8) as a means of predicting the expected output at the next time step given the state $x^{\tau}$, inputs $u^\tau$, and time $\tau$ measured at the current time,

$$\hat{x}^{\tau+1} = \mathbb{E}[x^{\tau+1} \mid x^\tau, u^\tau, \tau, \theta] = \phi(x^\tau, u^\tau, \tau)\theta, \tag{7.9}$$

where the $\hat{}$ denotes a predicted value ($w^\tau$ drops out in the expectation because it is zero mean). Notice that the predicted output is a function of the parameter values, $\hat{x}^{\tau+1}(\theta)$. If we then compare the predicted value, $\hat{x}^{\tau+1}$, with the actual value, $x^{\tau+1}$, we have an error that gives an indication of how different our model is from the actual system. We form a cost function using this error. One common cost function is constructed from summing over all the squared errors that we have collected from measuring the output of the actual system and comparing it to the predicted output, $J = \sum_\tau \|\hat{x}^\tau(\theta) - x^\tau\|^2$. We can then use an *analytical* optimization method, the Least Square method, to find the parameters $\theta = \theta^*$ that minimize the cost function $J(\theta)$. This can be interpreted as "fitting" the model parameters to the data, and it results in a model that we would expect to give the best prediction of outputs given the inputs. This process is described graphically in Figure 7-1.

System identification shares many similarities with machine learning, however, since it deals with dynamical systems, the training data is time correlated and is presented in a specific order—it is not Independent Identically Distributed (IID). This is a crucial difference between system identification and most other computational learning problems. Because of this fact, much of the machine learning intuition does not apply to system identification, especially with regard to model validation, as described in more detail in Section 7.5.3.

### 7.3.2 Manipulating the Linear Model

The model structure discussed in Section 7.2 has the convenient property that it is linear in its unknown parameters. For this reason, it can be manipulated into a form so that its parameters can be fitted using the system identification technique described above. In keeping with our minimalist approach, we assume that only position measurements, $p_i^\tau$, $\tau = 1, ..., N$, are available to perform the fitting. We can eliminate $u_i$ and $v_i$ from the dynamics in (7.1) to provide a second order equation in the position only. Notice that from (7.1) we can write

$$p_i^{\tau+1} = p_i^\tau + \Delta t \begin{bmatrix} u_i^\tau \\ v_i^\tau \end{bmatrix}, \tag{7.10}$$

and

$$\begin{bmatrix} u_i^{\tau+1} \\ v_i^{\tau+1} \end{bmatrix} = a_i \begin{bmatrix} u_i^\tau \\ v_i^\tau \end{bmatrix} + \sum_{j=1, j\neq i}^{m} f_{ij}^\tau + g_i^\tau + w_i^\tau. \tag{7.11}$$

We can solve (7.10) for $[u_i^\tau \;\; v_i^\tau]^T$ and substitute into the right hand side of (7.11). We then substitute the result back into the right hand side of (7.10), shifting time indices appropriately, to obtain the desired expression

$$p_i^{\tau+2} = p_i^{\tau+1} + (p_i^{\tau+1} - p_i^\tau)a_i + \Delta t \left( \sum_{j=1, j\neq i}^{m} f_{ij}^\tau + g_i^\tau + w_i^\tau \right). \tag{7.12}$$

We can use the above expression to formulate a one-step-ahead predictor in the form of (7.9). First, define the combined regressor vectors $\phi_{u_i}^\tau = [\; (e_i^{\tau+1} - e_i^\tau)/\Delta t \quad \phi_{fu_i}^\tau \quad \phi_{gu_i}^\tau \;]$, and
$\phi_{v_i}^\tau = [\; (n_i^{\tau+1} - n_i^\tau)/\Delta t \quad \phi_{fv_i}^\tau \quad \phi_{gv_i}^\tau \;]$, and a combined parameter vector
$\theta_i = [\; a_i \quad \theta_{f_i}^T \quad \theta_{g_i}^T \;]^T$. By taking the expectation conditioned on the positions, substituting (7.3) and (7.6) for $\sum_{j\neq i} f_{ij}$ and $g_i$, respectively, then making use of the combined regressor and parameter vectors we get

$$\hat{p}_i^{\tau+2} = p_i^{\tau+1} + \Delta t \begin{bmatrix} \phi_{u_i}^\tau \\ \phi_{v_i}^\tau \end{bmatrix} \theta_i, \tag{7.13}$$

where $\hat{p}_i^{\tau+2}$ is the expected value of $p_i$ after $\tau+2$ time steps, given positions up to $\tau+1$, and $w_i^\tau$ drops out in the conditional expectation.

### 7.3.3 Batch Method

The so called Least Squares Batch Method method is now implemented to find the optimal model parameters. Specifically, we wish to find the parameters, $\theta_i$, to minimize the mean squared prediction error over all available time steps. The mean squared prediction error can be written $J_i = 1/(N-2)\sum_{\tau=1}^{N-2}(p_i^{\tau+2} - \hat{p}_i^{\tau+2})^T(p_i^{\tau+2} - \hat{p}_i^{\tau+2})$. Substituting into $J_i$ with (7.13) and (7.10) yields

$$J_i = \frac{\Delta t^2}{N-2}(Y_i - \Phi_i\theta_i)^T(Y_i - \Phi_i\theta_i), \tag{7.14}$$

where

$$Y_i = [\ u_i^2 \ \dots \ u_i^{N-1} \ v_i^2 \ \dots \ v_i^{N-1}\ ]^T, \text{ and } \Phi_i = [\ \phi_{u_i}^{1\ T} \ \dots \ \phi_{u_i}^{N-2T} \ \phi_{v_i}^{1\ T} \ \dots \ \phi_{v_i}^{N-2T}\ ]^T \tag{7.15}$$

and $u_i^\tau$ and $v_i^\tau$ are obtained from (7.10). The Least Squares problem is then formulated as $\theta_i^* = \arg\min_{\theta_i} J_i(\theta_i)$. Following the typical procedure for solving the Least Squares problem we find that

$$\theta_i^* = [\Phi_i^T\Phi_i]^{-1}\Phi_i^T Y_i. \tag{7.16}$$

The right hand side of (7.16) consists entirely of measured data while the left hand side is the vector which represents the optimal parameters of the model. We assume that the data are rich enough that the matrix inversion in (7.16) is possible. The deep implications of this invertibility are discussed in [59]. The myriad merits and deficiencies of Least Squares fitting compared with other learning methods will not be discussed in this chapter.

The white noise signal $w_i^\tau$ can now be estimated using the resulting residual error in the fitting process, so that

$$\hat{w}_i^\tau = \begin{bmatrix} u_i^{\tau+1} \\ v_i^{\tau+1} \end{bmatrix} - \begin{bmatrix} \phi_{u_i}^\tau \theta_i^* \\ \phi_{v_i}^\tau \theta_i^* \end{bmatrix}, \tag{7.17}$$

where $\hat{w}_i^\tau$ is our estimate of $w_i^\tau$. If the "true" system dynamics are represented by the fitted model, we expect to find that $\hat{w}_i^\tau$ is zero-mean, stationary, Gaussian white noise, as this would confirm our initial assumption on the properties of $w_i^\tau$. Specifically, for perfect fitting, $\mathbb{E}[\hat{w}_i(t)\hat{w}_i^T(t+\tau)] = \delta(\tau)Q_i$, where $\delta(\tau)$ is the Kronecker delta function. Therefore, the "whiteness" of $\hat{w}_i$ can be used as an indicator of the goodness of fit that has been achieved. We use this fact in Section 7.5.3 to validate our learned models. For simulation purposes, as in Section 7.6, we would assume $w_i^\tau$ is a white noise process with covariance $Q_i$ equal to the empirical covariance of $\hat{w}_i^\tau$.

In such a way we learn a cow model for each cow in a herd using measured tracking data. The optimal parameters are found and the characteristics of the random vector $\hat{w}_i^\tau$ are determined for each cow $i = 1, \ldots, m$ to yield parameters for the entire herd. To make the entire process more clear, we have codified it as Algorithm 3.

**Algorithm 3** Batch Identification of Group Dynamics

```
for All agents in the group do
  Apply the measured data to (7.16)
  Use $\theta_i^*$ in (7.13)
  This defines the model for agent $i$
end for
```

### 7.3.4 Recursive Method

**Algorithm 4** Recursive Identification of Group Dynamics

```
for All agents in the group do
  Initialize parameters $\theta_i$ and $P_i$ to an arbitrary value
  Use $P_i$ to calculate $K_i$
end for
loop
  for Each agent in the group do
    Apply one position to (7.18) and (7.20), using $K_i$
    Use resulting $P_i$ to calculate $K_i$ for the next iteration
    Use $\theta_i^*$ in (7.13)
    This defines the model for agent $i$ for one time step
  end for
end loop
```

The Least Squares method can also be formulated recursively, so that each new available measurement becomes integrated into the parameter estimates, tuning them as time progresses. This method would be particularly useful for the parameter identification step in an adaptive control loop.

First, let $\phi_i^\tau = \begin{bmatrix} \phi_{u_i}^{\tau\,T} & \phi_{v_i}^{\tau\,T} \end{bmatrix}^T$, and $y_i^\tau = \begin{bmatrix} u_i^\tau & v_i^\tau \end{bmatrix}^T$. We wish to tune parameters dynamically according to

$$\theta_i^\tau = \theta_i^{\tau-1} + K_i^\tau (y_i^\tau - \phi_i^\tau \theta_i^{\tau-1}), \tag{7.18}$$

where

$$K_i^\tau = P_i^{\tau-1} \phi_i^{\tau T} [\lambda_i I_2 + \phi_i^\tau P_i^{\tau-1} \phi_i^{\tau T}]^{-1}, \tag{7.19}$$

and

$$P_i^\tau = P_i^{\tau-1} - (P_i^{\tau-1} \phi_i^{\tau T} [\lambda_i I_2 + \phi_i^\tau P_i^{\tau-1} \phi_i^{\tau T}]^{-1} \phi_i^\tau P_i^{\tau-1}) / \lambda, \tag{7.20}$$

where $K_i^\tau$ is the parameter gain, $P_i^\tau$ is the parameter covariance matrix, $\lambda_i$ is a forgetting factor ($0 < \lambda_i \leq 1$), and $I_2$ is the $2 \times 2$ identity matrix. This standard algorithm is stated here without derivation. The interested reader can find a thorough discussion in [59]. The algorithm for this method is given in Algorithm 4.

Note that the kinds of systems under consideration are likely to have time varying parameters. For instance cows are likely to change their behavior throughout the day in accordance with sunlight, temperature, their hunger and thirst, etc. For this reason, we would expect the parameter following properties of the recursive algorithm with a forgetting factor to be advantageous. The recursive Least Squares algorithm can be used to learn the model while it is simultaneously being used for prediction in a control algorithm. This would result in an adaptive control algorithm for distributed groups. The results presented in the following sections use the Batch method. We save a detailed study of on-line and distributed learning algorithms for future work.

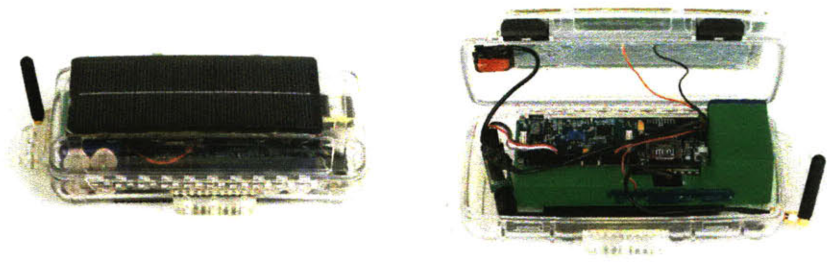


Figure 7-3: The sensor box is shown here with lid closed (left) and lid open (right). The box is roughly 21.5cm×12.0cm×5.5cm and weighs approximately 1kg. It is equipped with a GPS receiver, wireless networking features, and a suite of sensing and actuation capabilities. The Lithium-Ion batteries and solar panel allow for indefinite operation under normal conditions. It can also modularly accommodate expansion boards for various other applications.

## 7.4 Data Collection Experiments

### 7.4.1 Animal Monitoring Hardware

We have developed a small light-weight box (see Figure 7-3) for data collection and animal control for use during our field experiments. The box contains electronics for recording the GPS location of the animal as well as other sensor data which we do not use in this work (a 3-axis accelerometer, a 3-axis magnetometer, and a temperature sensor). The box also contains electronics for networking with other boxes, and for applying sound and electrical stimuli to the animal, though the stimuli were not applied during the data collection experiments described here. Building on the pioneering work of [15, 113] on animal monitoring hardware, we improved the performance of the device by mounting it on top of the animal's head, as shown in Figure 7-4, instead of packaging it as a collar. We found the head mounted device improved several aspects of the device's performance compared to the previous collar mounting: (1) the GPS satellites were more likely to be visible from the top of the head, (2) solar panels on the box were more likely to receive direct sun exposure, (3) networking radio communication was less obstructed by the animal's body, (4) the animal was less able to deliberately rotate the box, and (5) the box was prevented from being dipped in water or mud and was generally better protected.

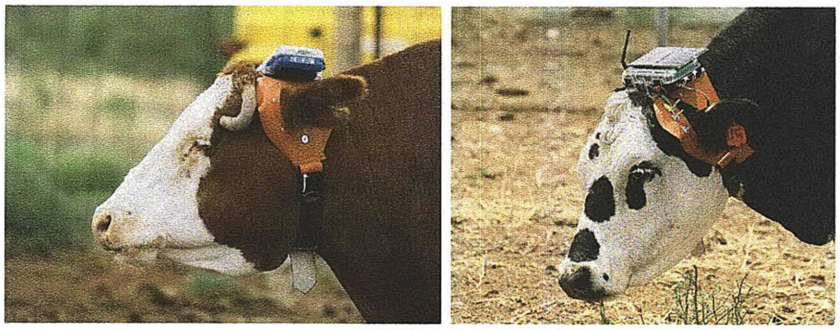


Figure 7-4: The sensor box is mounted to the head of the cow with a custom fitted apparatus made of fabric and plastic. The apparatus is designed to use the cow's ears to keep the box in an upright position, as shown in this figure.

Our sensor box is approximately 21.5cm×12.0cm×5.5cm and weighs approximately 1kg. The processor is a 32bit ARM7TDMI cpu (NXP model LPC2148) with 512kB program memory, 40kB RAM, USB, and a 10 bit A/D converter. The device also has 256kB FRAM (external non-volatile memory with no rewrite limit) and a removable SD card with 2GB storage capacity. Data can be easily and quickly downloaded to a computer by physically transferring the SD card, or by downloading remotely via the radios. There are 2 hardware serials which are multiplexed for a total of 5. The sensors in the box include a GPS engine, 3-axis accelerometer, 3-axis magnetic compass, and an ambient air temperature sensor. There are many general purpose analogue and digital I/O lines, so additional sensors can be included.

The communication system consists of two radios. Firstly, a 900MHz radio (Aerocomm AC4790) with 1 watt transmit power is used for long range, low band width communication. This radio has a claimed 32km range and a claimed 57600b/s transfer rate. However, we observed a maximum of only 2km range and a data transfer rate of only 1000b/s. This is particularly odd as the flat, remote environment in which the radios were tested should have been ideal for radio transmission. The cause for the poor performance of this radio is still unknown. Secondly, the box uses a Bluetooth radio with 100m range and 100kb/s data rate for short range, high band width communication.

Power is provided by a bank of 8 Lithium-Ion batteries with a total capacity of

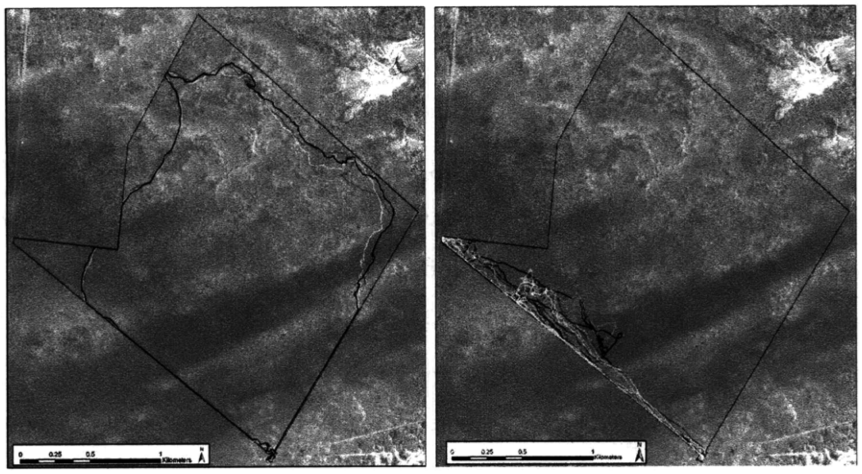


Figure 7-5: The GPS positions of the cows are shown superimposed on satellite images of the paddock in which the data were collected. The left image shows data collected from three cows in the first trial between February 2–5, 2007. The right image shows the data collected from ten cows in the second trial between July 9–11, 2007.

16 watt-hours. The batteries are continuously recharged by a solar panel mounted on the top of the box allowing the box to run indefinitely under normal conditions. The batteries have enough capacity for several days of operation without the solar panels.

Finally, we have a two-tier animal control system consisting of a set of speakers for applying arbitrary, differential sound stimuli and a set of electrodes that enable the application of differential electrical stimuli. The animal control system was not used during the collection of the data described in this chapter.

The box's operating system is a custom designed collaborative multitasking architecture. Processes run as scheduled events which can be scheduled to run at millisecond intervals with no preemption or real-time constraints. The software supports arbitrary network topologies for communication. Users interact with the system via a serial console or a Java user interface. These can be accessed directly through the serial port or remotely over either of the radios. This allows remote reconfiguration of the monitoring devices in the field. The operating system can be completely reprogrammed using an attached serial cable, remotely over the radio, or by placing a file on the SD card.

### 7.4.2 Experimental Methodology

Data were collected during two trials, the first taking place from February 2–5, 2007 and the second from July 9–11, 2007, during which time three head and ten head of cows were monitored, respectively, using the sensor boxes described above. During both trials cows were allowed access to a 466ha, or 4.66 $km^2$, paddock (named 10B) located on the US Department of Agriculture-Agricultural Research Service's (USDA-ARS) Jornada Experimental Range (JER) in Southern New Mexico (32° 37' N, 106° 45'W) which is approximately 37km Northeast of the city of Las Cruces at an elevation of approximately 1260m above sea level. The climate of this arid area has ambient air temperatures that range from a high of 36° C in June to below 13° C in January with 52% of the mean annual precipitation (230mm) falling as rain between July and September [73, 110]. Grasses (39% to 46%) and forbs (36% to 49%) comprise the predominant vegetation while woody shrubs compose 14% to 19% of the remaining standing crop [3,45] that grows in a mosaic pattern across this relatively flat landscape composed of three major landforms [65].

In the first trial, three free-ranging mature beef cattle of Hereford and Hereford $\times$ Brangus genetics, labeled Cow 1–Cow 3, were fitted with the sensor boxes described above. Data were collected over four days from February 2–5, 2007 at a data collection rate of 1Hz. In the second trial, ten free-ranging mature beef cattle of similar genetics, labeled Cow 1–Cow 10 were fitted with the sensor boxes. Data were collected at 1Hz over three days from July 9–11, 2007. The cows 1, 2 and 3 correspond to the same three cows in the first and second trials. The paddock for the experiments was fenced with the geometry shown in Figure 7-5. During these two trials, the animals received no audio or electric cues from the sensor boxes.

When they are introduced to a new paddock, cows commonly trace out the perimeter to familiarized themselves with the extent of their new environment [1]. They then concentrate their activities on certain areas depending upon vegetation and other factors. During the first trial, shown on the left of Figure 7-5, the cows had been recently introduced to paddock 10B from another neighboring paddock (though they had previous experience in paddock 10B), and their perimeter tracing behavior is evident in the plot. In the second trial (on the right of Figure 7-5), the cows had already been in the paddock for some time before data were collected.

## 7.5 Modeling a Group of Cows

The method presented in Section 7.3 was used to model the dynamics of a group of three cows, as well as a group of ten cows. Data collected as described in Section 7.4 was used for fitting the model parameters and for evaluating the resulting model. We will first present modeling results for the three cows as it is less complicated to interpret data for a smaller group, then we will show results for the ten cows. The total number of agent-to-agent interaction forces grows like the square of the number of agents, hence the difficulty in efficiently displaying results for large groups. Finally we discuss the problem of validating the learned models, and propose a statistically justified method for validation. Results of the validation method are shown for both the three and ten cow models.

### 7.5.1 Three Cows

The dynamics of a cow group are known to be modal [90], in the sense that model parameters are approximately constant over contiguous intervals, but can change rapidly when switching between such intervals, for example when the group transitions from resting to foraging. We intentionally selected a 52 minute interval of data (from approximately 18:02hrs to 18:54hrs on February 2, 2007) for learning model parameters that corresponded to a stretch of time when the herd was apparently in a constant foraging mode. For each cow, the data used for the Least Squares fitting consisted of 3100 GPS position entries collected at 1Hz. The data for all animals were artificially synchronized to a common clock using a standard linear interpolation. The characteristic time scale of cow dynamics is considerably longer than 1 second (that is to say, cows move little in the span of 1 second), thus such an interpolation is expected to have a negligible effect on modeling results.

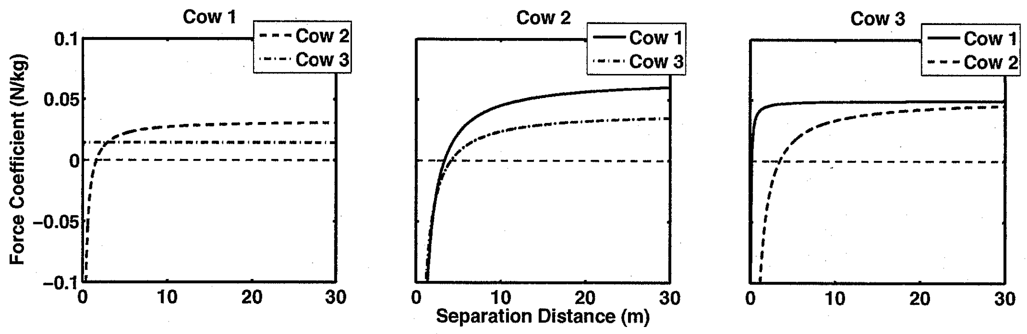


Figure 7-6: The agent-to-agent interaction forces are shown for the three cows. Each curve represents the size of the force imposed by one cow on another as a function of the distance between the cows. A positive value is attractive while a negative value is repulsive.

The data were used to find model parameters as described in Section 7.3. The panels in Figure 7-6 show the agent-to-agent force magnitudes $\|f_{ij}(p_i, p_j)\|$ for the three cows. For each cow, the two curves show the force imposed by each of the two other cows in the group. Note that the forces are not necessarily pair-wise symmetric, that is, $\|f_{ij}\| \neq \|f_{ji}\|$ in general. The force curves are useful for analyzing behavioral traits of the cows. It is well known that groups of cows have complicated social sub-groupings and hierarchies [58]. The plots indicate that Cows 1 and 3 had an affinity for one another, while Cow 2 was comparatively not very attractive to, or attracted by, Cows 1 and 3. We will reexamine the behavior of Cow 2 below in the context of the ten cow group.

The environment-to-agent vector fields are shown in the panels of Figure 7-7 for the three cows. The heavy dots show the centers of the Gaussian basis functions, $\gamma_{ki}$, the arrows show the direction and magnitude of the force felt by a cow at each point, and the curve indicates the position data used for learning. The Gaussian centers were spaced over an even grid containing the trajectory of the cow. If the trajectory did not come within one standard deviation, $\sigma_{ki}$, of a Gaussian function, the Gaussian was dropped from the network. This primitive pruning algorithm was used for simplicity; more complex algorithms could be employed. The Gaussian widths were chosen to be $2/3$ the length of the grid space occupied by the Gaussian. This width was found to

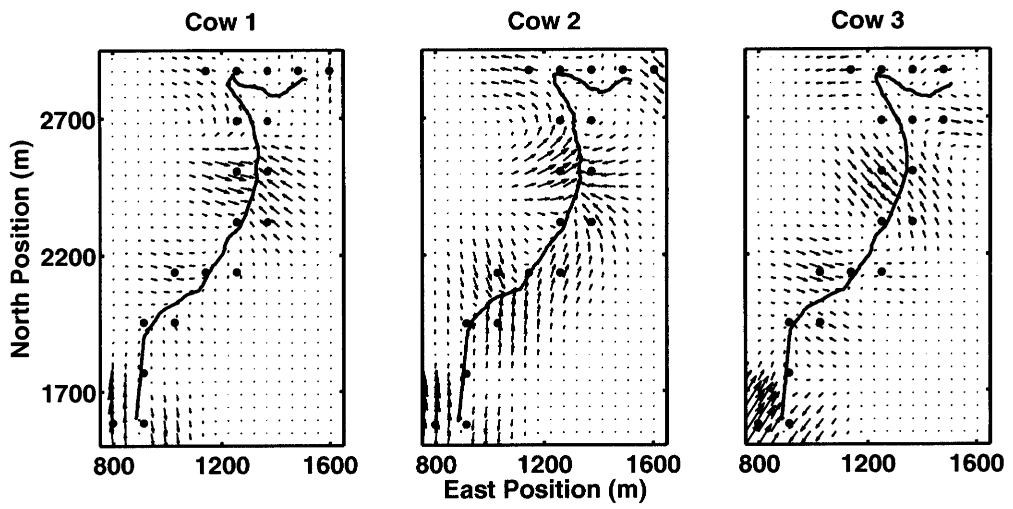


Figure 7-7: The environment-to-agent force fields are shown for the three cows. The heavy dots indicate the centers of the Gaussian functions and the arrows show the forces produced by the learned vector field. The continuous curve marks the actual cow's path over the region.

give good performance with our data. One could imagine including the widths as free parameters in the Least Squares cost function (7.14), but the cost function becomes non-convex in this case and is therefore very difficult to optimize.

### 7.5.2 Ten Cows

For each cow, data consisted of 4000 GPS position entries collected at 1Hz during the second trial described in Section 7.4.2. As before, care was taken to use a contiguous stretch of data (from approximately 11:33hrs to 12:40hrs on July 9, 2007) during which the cow group appeared to be in a foraging mode. Cows 1, 2, and 3 were the same animals as in the first trial. The data for all animals were artificially synchronized to a common clock using a standard linear interpolation as was done for the three cow data.

The data were used to find model parameters as described in Section 7.3. The panels in Figure 7-8 show the magnitude of the agent-to-agent force for the ten cows. The number of agent-to-agent interaction forces is much higher than for three cows

($10 \times 9$ as opposed to $3 \times 2$), so the plots are correspondingly more complicated. In particular, the force plot for each animal shows ten curves. Each of the nine thin curves represents the magnitude of force caused by each of the nine other animals as a function of separation distance. The thick curve shows the mean over all nine force curves. Despite considerable variation over animals (including some inverted force curves) the mean force felt by any one animal as a result of its proximity to all of the others is relatively similar, as indicated by the mean force curve.

The environment-to-agent vector fields are shown in the panels of Figure 7-9 for the ten cows. The heavy dots show the centers of the Gaussian basis functions, $\gamma_{ki}$, the arrows show the direction and magnitude of the force felt by a cow at each point, and the curve shows the position data used for regression. The Gaussian centers were spaced and pruned as described for the three cow trial.

To demonstrate the potential usefulness of the learned model to study animal behavior, consider again the behavior of Cow 2 in the context of the ten cow group. By comparing the mean force curves in Figure 7-8 with the curves in Figure 7-6, we see that Cow 2 does not tend to stay as far from the other cows in the larger group as in the smaller group. It seems, for example, that Cow 3 stays farther from the other cows than does Cow 2 in the larger group. The apparent dependence of animal behavior on group size is a property of interest to the animal behavioral sciences. Of course, there are a number of other factors that could be responsible for this behavior, including time of year, the animals' physiological state, weather conditions, and the quality and quantity of standing crop. However, by analyzing the learned model we have generated a interesting hypothesis about cow behavior, which can be used to guide the design of further experiments.

### 7.5.3 Model Validation

In terms of signal processing, our learning algorithm can be seen as taking a time-correlated velocity signal and producing model parameters and a residual error signal. If our velocity data are rich in temporal correlation it is good for modeling. Also, if our learned model is successful in capturing the relevant correlation of the velocity

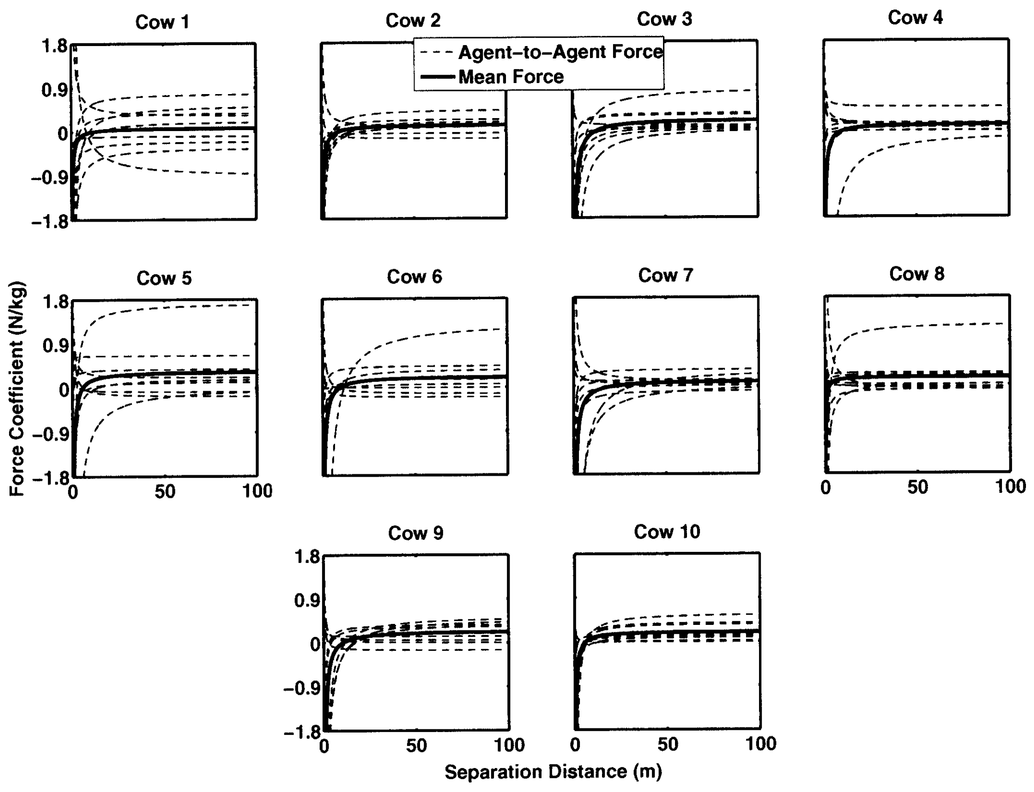


Figure 7-8: The agent-to-agent interaction forces are shown for the group of 10 cows. Each thin, dashed curve represents the size of the force imposed by one cow on another as a function of the distance between the cows. The thick, solid curve shows a mean over all of the individual force curves. A positive value is attractive while a negative value is repulsive.

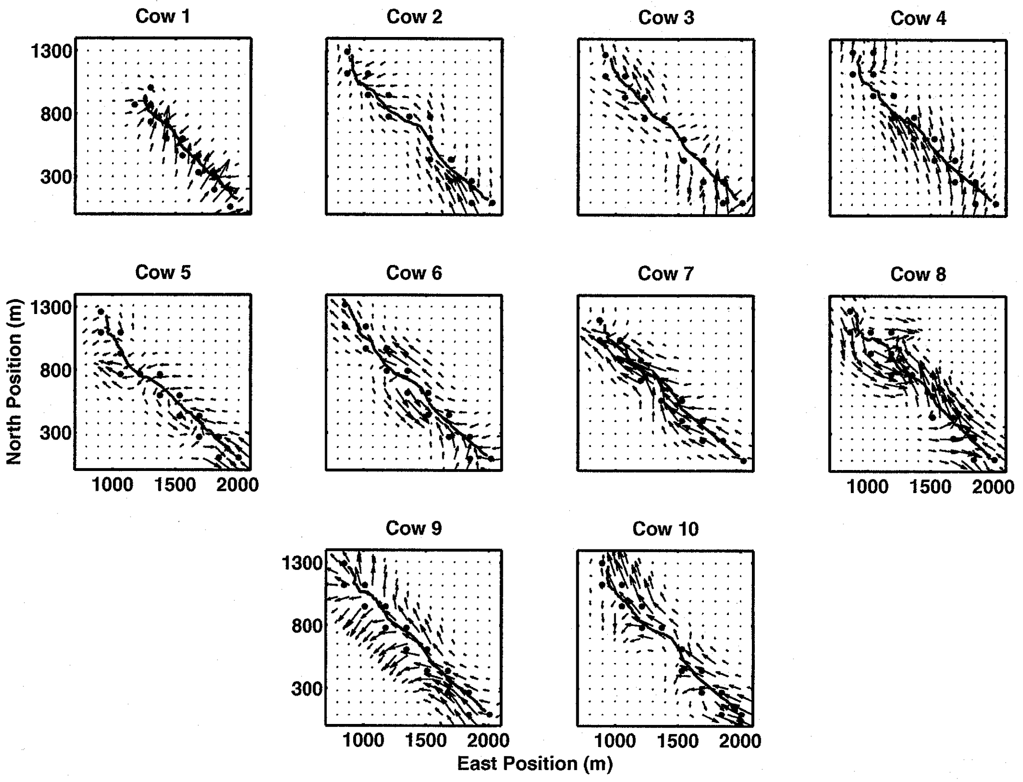


Figure 7-9: The environment-to-agent force fields are shown for the group of 10 cows. Heavy dots indicate the centers of the Gaussian functions and the arrows show the force produced by the learned vector field. The continuous curve marks the cow's actual path over the region.

signal, the residual error signal will have little temporal correlation. More plainly, we want our velocity signal not to be white, and our residual error signal to be white. Therefore, we are interested in testing for "whiteness" in each of these signals by comparing them against a 90% whiteness confidence interval.

To be specific, consider some random signal $x(t)$ generated by a stationary Gaussian white noise process $X(t)$. Each point on the empirical auto-covariance function,

$$K_x(\tau) = \frac{1}{T-\tau}\sum_{t=\tau}^{T} x(t-\tau)x(t), \tag{7.21}$$

is asymptotically normally distributed, with zero mean and variance equal to $\frac{K_x(0)^2}{(T-\tau)}$ (see [59] Lemma 9.A1, or [72]). The 90% confidence interval is then found from the inverse cumulative normal distribution to have boundaries defined by the curves

$$C_5(\tau) = \sqrt{\frac{2}{T-\tau}}K_x(0)\mathrm{erf}^{-1}(2\times .05-1) \quad \text{and} \quad C_{95}(\tau) = \sqrt{\frac{2}{T-\tau}}K_x(0)\mathrm{erf}^{-1}(2\times .95-1),$$

meaning the process $X(t)$ would produce a value $K_x(\tau)$ below $C_5(\tau)$ with probability .05 and below $C_{95}(\tau)$ with probability .95 for each point $\tau$.

Applying this reasoning to our velocity $y_i(t) = [v_i(t) \quad u_i(t)]$ and residual error $\hat{w}_i(t)$ signals, we validate the learned model by examining the empirical auto-covariance functions,

$$K_{y_i}(\tau) = \frac{1}{T-\tau}\sum_{t=\tau}^{T} y_i(t-\tau)y_i(t)^T \quad \text{and} \quad K_{\hat{w}_i}(\tau) = \frac{1}{T-\tau}\sum_{t=\tau}^{T} \hat{w}_i(t-\tau)\hat{w}_i(t)^T,$$

respectively, where the time of the sample is now explicitly written as an argument, for example $\hat{w}_i(t) = \hat{w}_i^t$. If the velocity $y_i(t)$ and the residual error $\hat{w}_i(t)$ were generated by a white noise process, we would expect $K_{y_i}(\tau)$ and $K_{\hat{w}_i}(\tau)$ to fall within their respective whiteness confidence intervals with probability .9 at each $\tau$. Again, we want the velocity signal to fail this test and the residual error signal to pass it.

There are other tests for whiteness, but this is the simplest one with a rigorous statistical interpretation [59]. This whiteness test takes the place of leave-one-out

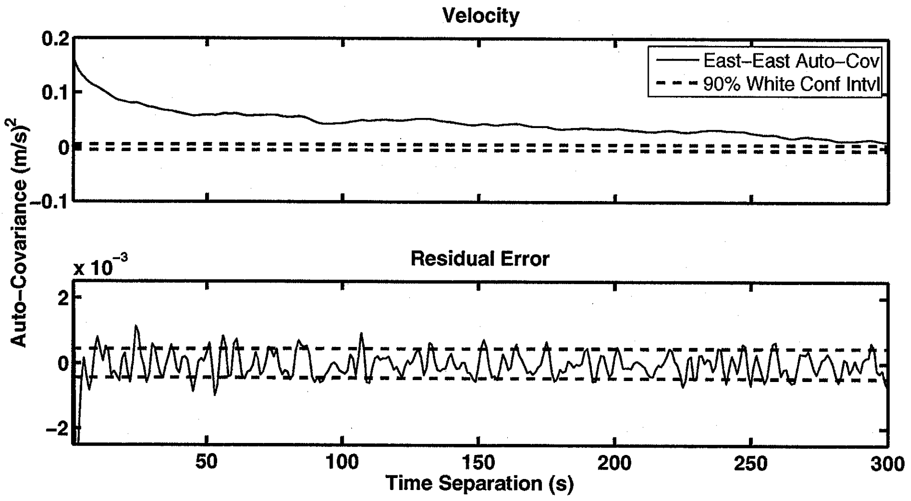


Figure 7-10: The empirical auto-covariance function for the Eastern component of the velocity is shown in the top plot, and for the error residual in the bottom plot. The dotted lines indicate a 90% whiteness confidence interval, meaning that a stationary, Gaussian white, noise process would have generated an empirical auto-covariance inside the interval with probability .9 at each point. By this metric, the velocity signal is not white and the residual error signal is "nearly white," indicating a good model has been learned for the data.

validation, or other similar validation methods common in machine learning applications. We cannot use such methods because our data is not IID, a key assumption in most machine learning algorithms. Indeed, our model is specifically trying to capture correlation between data points, so to leave one data point out would obscure precisely the relationship we want to learn.

The top of Figure 7-10 shows the auto-covariance from the three cow trial of the Eastern component of the velocity for Cow 1, and the bottom figure shows the auto-covariance of the corresponding residual error. Notice there is strong temporal correlation in the velocity, and all points in the plot lie outside the confidence interval, therefore it fails the whiteness test, as desired. For the residual error auto-covariance, there is apparently little temporal correlation and a large majority of the points lie inside the whiteness confidence interval, therefore it passes the whiteness test. Thus, by this measure, the algorithm has done a good job of producing a model to describe the cow's dynamics. The plots for the other components of the auto-

|  | Velocity |  |  | Residual |  |  |
|---|---|---|---|---|---|---|
| Cow Number | 1 | 2 | 3 | 1 | 2 | 3 |
| East-East | 0 | 0 | 0 | 76 | 81 | 87 |
| East-North | 0 | 0 | 5 | 73 | 78 | 83 |
| North-East | 21 | 4 | 17 | 81 | 84 | 91 |
| North-North | 0 | 0 | 0 | 66 | 68 | 87 |

Table 7.1: The table shows what percentage of points lie within the 90% whiteness confidence interval for each of the 3 cows in the first trial, and for each of the four components of the auto-covariance function. According to this test, the velocity signal is not white, and the residual error is approximately white, so the model fits the data well.

covariance functions and for the other cows in the three cow trial are excluded in the interests of space. Instead, we summarize the results in Table 7.1, which shows for each cow, and for each of the four components of the auto-covariance functions $K_{\hat{w}_i}(\tau)$ and $K_{y_i}(\tau)$, the percentage of points within the 90% whiteness interval. The results show that the velocity signals for all cows fail the whiteness test (as desired), while the residual error signals can all be considered nearly white in that nearly 90% of their values were within the confidence interval.

The whiteness test was also carried out for ten cows with similar results as summarize in Table 7.2. The results in the table show that all of the residual errors for the ten cow model are nearly white. As for $K_{y_i}$ in this case, all of the points for all of the components and all of the cows lie outside of the whiteness confidence interval, therefore the velocity is very likely not white for any cow.

## 7.6 Synthetic Control

Simulation experiments were carried out with the model fitted in Section 7.5.1. We simulated a group of three simple mobile robots controlled to have the dynamics in (7.1) with the parameters found in Section 7.5.1. These equations were iterated forward in time in a Matlab environment with the robots started from the same initial positions as the cows. The simulation procedure is summarized in Algorithm 5.

The trajectories of the robots from a typical simulation are shown in the left side

|  | Residual |  |  |  |  |  |  |  |  |  |
|---|---|---|---|---|---|---|---|---|---|---|
| Cow Number | 1 | 2 | 3 | 4 | 5 | 6 | 7 | 8 | 9 | 10 |
| East-East | 75 | 85 | 87 | 81 | 69 | 85 | 87 | 79 | 71 | 84 |
| East-North | 69 | 82 | 76 | 83 | 60 | 75 | 78 | 77 | 74 | 83 |
| North-East | 79 | 80 | 82 | 84 | 61 | 72 | 80 | 80 | 75 | 80 |
| North-North | 68 | 75 | 75 | 84 | 62 | 61 | 74 | 73 | 69 | 83 |

Table 7.2: The table shows what percentage of points lie within the 90% whiteness confidence interval for each of the 10 cows, and for each of the four components of the residual error auto-covariance function. By this metric, the residual errors for all cows are approximately white. For the velocity auto-covariance function (not shown in the table), no point is within the interval for any cow, thus the velocity is very likely not white. By this test, the ten-cow model successfully fits the data.

**Algorithm 5** Synthetic Control Algorithm

```
Execute Algorithm 3 to obtain a set of optimal parameters for each agent
Set initial conditions for simulated group of robots
loop
  for Each robot in the group do
    Use the current state of all the robots and $\theta_i$ obtained from Algorithm 3.
    Apply these to the dynamical equations for agent $i$ (7.1) to produce the next
    robot state
  end for
end loop
```

of Figure 7-11 laid over a schematic showing the fences of the paddock where the actual cow data were recorded. The trajectories of the simulation are similar to those of the real cows. Most importantly, the simulated robots track the fence lines, as did the real cows. This tendency is captured solely through the agent-to-environment force field (described in Section 7.2.3), as the model has no direct knowledge of where fence lines may lie. Furthermore, statistics were gathered for the simulated robots and compared with those from the cow data. Figure 7-12 shows a comparison of the two sets of statistics. Specifically, the distance between cows over time and the speed of the cows over time have similar mean and standard deviation for the real and simulated data. Thus the model preserves global properties of the group, as measured by these statistics.

One should expect the trajectories of the simulation to be *qualitatively* similar to the actual training data, but the question of *how* similar is not a simple one. The model we have constructed is a random process, and two different sets of data generated by the same random process will almost certainly be different. It is also not informative to look at, for example, the mean distance between points of the actual and simulated data, since, again, two signals from the same random process can generate trajectories arbitrarily far from one another. The appropriate test for model validation is the whiteness test described in Section 7.5.3. We show Figures 7-11 and 7-12 only to indicate that the properties verified with the whiteness test lead, in practice, to a *qualitative* match in performance.

It is also important to point out that comparing these simulation results to the cow data is not the same as testing a learned model on training data, a common pitfall in machine learning applications. Indeed, the only training data given to the simulation are the initial positions of the robots. The model recursively generates its own data points which then become inputs for successive time steps. This is a manifestation of the fact that system identification takes place in a non-IID setting, so much of the intuition that applies in typical machine learning problems is not applicable.

This simulation study suggests that our model equations can be used to control a group of robots to exhibit the behavior of the modeled group. In this way con-

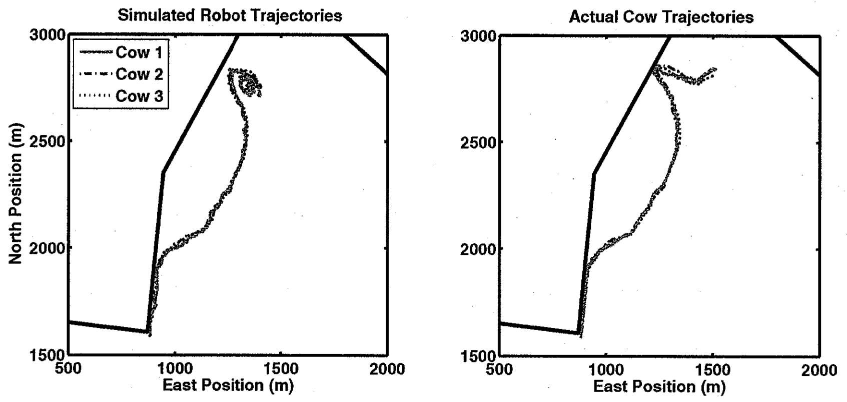


Figure 7-11: The left plot shows trajectories of a team of simulated robots controlled to behave like a group of cows. The robots use dynamical laws generated from the procedure described in this chapter. Their trajectories are superimposed over the fence lines of the paddock where the original cow data were collected, though they have no direct knowledge of fence positions. The right picture shows the actual cow data over the same time window.

trollers can be automatically synthesized for robots to mimic groups that have some desirable collective behavior, such as flocking or herding. One can also imagine introducing artificial members of a group without changing the group dynamics (i.e. without "being noticed") or for the purpose of modifying the group dynamics in a non-disruptive way, for example to influence collective decision making in natural groups, as was done in [39].

## 7.7 Synopsis

In this chapter, we presented a method to generate behavior models of groups of dynamical agents, such as cow herds, using observations of the agents' positions over time. We formulated a physically motivated difference equation model, and used Least Squares system identification to fit the model to data. We demonstrated the method by learning models for a group of three cows and a group of ten cows using GPS position data. The position data were collected with specially designed sensor boxes fitted to the heads of free-ranging cows. An important and surprising contribution of this chapter is the demonstration that a minimalist approach to modeling group

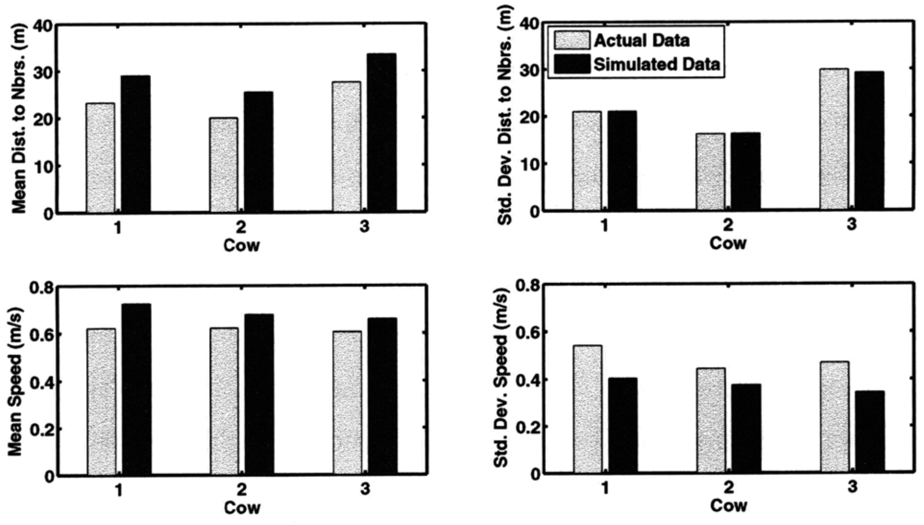


Figure 7-12: The bar charts compare statistics for the actual cow data and the simulated robots. The top charts show the mean and standard deviation of the distance from one cow to the other two cows in the group. The bottom charts show the mean and standard deviation of the speed of each cow.

interactions using only position data leads to meaningful group dynamical models.

Our approach is minimalist in that no information is included in the model about the geometry and configuration of the environment, nor about any attractive (e.g. vegetation) or repulsive (e.g. fences) features in the environment. It was shown in Section 7.6, however, that our method can be used to infer the locations of such features, since the robots avoided a fence obstacle even though they were given no prior indication of the fence's existence. An interesting research direction is to investigate the trade-offs between including additional information about features in the environment and the quality of the resulting model. More specifically, we can explicitly model obstacles in the space as a force field with some free parameters that are learned from the position data. We can also include dependencies upon weather and other ambient environmental conditions for which measurements are available. The question is, does the performance improvement of the learned model justify the extra complexity and prior information required for such a model? Our preliminary studies with explicit fence models show that this additional information leads to models that give similar behavior to those without the explicit obstacle features, but the explicit inclusion of the obstacle gives the ability to enforce hard position constraints on the agents. We generally prefer the minimalist approach described in this chapter in that it is amenable to situations where no detailed environmental information is available.

Our work has provided some insights into developing a minimalist approach to modeling group behavior, however many questions remain to be resolved. Learning models of complex natural and artificial groups is an exercise in balancing tradeoffs between model fidelity and model complexity. The systems we are interested in modeling are too sophisticated to characterize their motion in its entirety, but we have shown in this chapter that a simple model structure with a simple learning algorithm can give enough prediction power to be practically useful for controlling, simulating, and interacting with groups of dynamical agents.

# Chapter 8

# Conclusions, Lessons Learned, and Future Work

This thesis considers a method, based on gradient optimization, for controlling groups of agents to reach a goal configuration. We focus on the problem of deploying robots over an environment to do sensing, a task called coverage. We show that coverage is actually of a general enough nature to represent a number of problems not normally associated with it, for example consensus and herding. We augment the multi-agent gradient controller with learning to allow for robots to adapt to unknown environmental conditions. The learning is incorporated with provable performance and stability guarantees using a Lyapunov proof technique. We implemented the multi-agent learning controller on a group of 16 mobile robots and performed experiments in which they had to learn the intensity of light in the environment. We also implemented a multi-agent coverage controller on a group of flying quad-rotor robots with downward facing cameras. The controller used a realistic model of the camera as a sensor. Experiments were performed with 3 quad-rotor robots and were shown to provide multi-robot sensor coverage as predicted. Finally, we used the multi-robot dynamics to model the motion of cows in a herd. We used system identification techniques to tune the model parameters using GPS positions from a herd of actual cows.

In the course of this research we learned several lessons concerning the design and implementation of multi-robot controllers. Firstly, we learned that the coverage optimization can represent many different multi-robot behaviors. This is one of the main themes of this thesis: the unification of multi-robot control under the umbrella of a single optimization problem. That optimization problem can be specialized to specific multi-robot tasks and specific robot capabilities, and a stable controller can be derived from the gradient of the cost function. We show that this simple formula for designing multi-robot controllers produces robust, practical controllers that are feasible on a range of robot platforms.

A second lesson that we learned is that in a multi-robot setting, consensus algorithms can sometimes substitute for centralized knowledge. For example, consensus was used in our learning algorithm in Chapter 4 to propagate sensor information around the network. Each robot was able to asymptotically learn the sensory function as well as if they had direct access to all the robots' sensor measurements. In deed, it appears that consensus algorithms could be a fundamental and practical tool for enabling distributed learning in general, and have compelling parallels with distributed learning mechanisms in biological systems.

Finally, we learned that analysis is only useful up to a point, after which the true proof of performance is in the implementation of the controller on real robot platforms. This is a fundamental engineering point of view. No tractable mathematical model will be able to capture the intricacies of the dynamics, noise, and computational processes, so the final proof must be an implementation. We have found that the best policy is to model the systems as simply as possible, derive as many properties as possible from the simple model, test in simulation to make sure the model makes sense, then implement on actual robot platforms to verify the analysis. We have endeavored to follow this policy throughout the research in this thesis.

The research in this thesis also points toward several lines of future work. The most important immediate work to be done is in finding a suitable way to implement a controller over a given communication graph. Let the graph induced by a controller as described in Section 2.4.1 be denoted $\mathcal{G}_i$. Let the communication graph be denoted $\mathcal{G}_c$. If the graph induced by the controller is a subgraph of the communication graph, $\mathcal{G}_i \subset \mathcal{G}_c$, then the controller is feasible. Of course it may very well be the case that the controller is not feasible given the communication graph, in which case an approximation must be used. Two methods are suggested in this thesis: 1) each robot computes its controller using only the information from neighbors available to it, and 2) each robot maintains an estimator of the positions of all the robots required to compute its controller, and makes its computations using these estimates. In the future the stability and robustness properties of these methods should be characterized and other methods should be investigated.

Also our recognition that coverage problems stem from nonconvex optimizations suggests some new research directions. Gradient descent controllers, which are the most common type in the multi-robot control literature, in general can only be expected to find local minima of nonconvex cost functions. Therefore it is worth while to look for special cases of multi-robot cost functions that might allow for global minima to be reached with gradient controllers. For example, if the cost function has a single minimum despite being nonconvex, it may be possible to prove convergence to that minimum with a gradient controller. As another example, if all of the minima are global (they all take on the same minimal value of $\mathcal{H}$) then gradient controllers will find global minima. Alternately, we are motivated to consider other nonconvex optimization methods besides gradient descent that can be implemented in a multi-robot setting.

Another direction for future work is to move beyond the class of cost functions considered here. It would be interesting to consider cost function that depend, for example, upon the time history of the robots, as in optimal control, or that incorporate more complicated robot dynamics than the simple integrator dynamics. Cost functions such as the one in this thesis that only depend upon the positions of the robots lead to behaviors in which the robots move to a configuration and remain fixed. An expanded class of cost functions would lead to more complex, dynamic muti-robot behaviors.

In the case of system identification for groups of dynamical agents, an important problem to address in the future is capturing the modal changes in the dynamics of groups of agents over long time scales. We collected data at 1Hz continuously over several days for the cows in Chapter 7, but as discussed previously, we only expected our model to describe the cow group dynamics over an interval of approximately an hour, during which time the group is in a single behavioral mode. In the future, we would like to broaden the model class to include switching state-space models. That is, we would model both the motion of the group while it is in one mode and the transitions among modes. With such a model structure we expect to be able to capture the behavior of the cow group over extended periods of time and to be able to model other natural and artificial groups that exhibit modal properties (e.g. traffic motion, which is congested during rush hour and less so at other times). Unfortunately, exact system identification is known to be intractable for switching state space models [19]. A topic of current research in the system identification and learning communities is to find approximately optimal parameters using, e.g. variational approaches [36], or Markov-Chain Monte Carlo (MCMC) methods [69].

We expect that these open questions will motivate new results and new insights for multi-robot control. We hope that the gradient optimization approach in this thesis will yield new insights and enable new multi-robot control algorithms. We also hope that our emphasis on incorporating learning in multi-robot systems will be a step toward multi-robot technologies that interact flexibly with an uncertain world, gathering information from their environment to proceed toward a common goal. The way forward for multi-robot control is to unify the diverse specialized results which abound in the field, and to work toward a simple policy to create multi-robot controllers that will help us to monitor and manipulate our environment for the better.

# Appendix A

# Proofs of Lemmas

**Lemma A.1 (Uniform Continuity for Basic Controller)** *For the basic controller, $\dot{\mathcal{V}}$ is uniformly continuous.*

**Proof A.1** *We will bound the time derivatives of a number of quantities. A bounded derivative is sufficient for uniform continuity. Firstly, notice that $\hat{C}_{V_i}, p_i \in V_i \subset Q$, so $\hat{C}_{V_i}$ and $p_i$ are bounded, which implies $\dot{p}_i = K(\hat{C}_{V_i} - p_i)$ is bounded. Consider terms of the form*

$$\frac{d}{dt}\left(\int_{V_i} f(q,t)\, dq\right) \tag{A.1}$$

*where $f(q,t)$ is a bounded function with a bounded time derivative $\frac{d}{dt}f(q,t)$. We have*

$$\frac{d}{dt}\left(\int_{V_i} f(q,t)\, dq\right) = \int_{V_i} \frac{df(q,t)}{dt}\, dq + \int_{\partial V_i} f(q,t) n_{\partial V_i}^T \sum_{j=1}^{n} \frac{\partial(\partial V_i)}{\partial p_j} \dot{p}_j\, dq, \tag{A.2}$$

*where $\partial V_i$ is the boundary of $V_i$ and $n_{\partial V_i}^T$ is the outward facing normal of the boundary. Now $\frac{\partial(\partial V_i)}{\partial p_j}$ is bounded for all $j$, $\dot{p}_j$ was already shown to be bounded, and $f(q,t)$ is bounded by assumption, therefore $d/dt(\int_{V_i} f(q,t)\, dq)$ is bounded.*

*Notice that $\dot{\hat{C}}_{V_i}$ is composed of terms of this form, so it is bounded. Therefore $\ddot{p}_i = K(\dot{\hat{C}}_{V_i} - \dot{p}_i)$ is bounded, and $\dot{p}_i$ is uniformly continuous.*

*Now consider*

$$\dot{\mathcal{V}} = \sum_{i=1}^{n} \left[ -\int_{V_i} (q - p_i)^T \phi(q)\, dq\, \dot{p}_i + \tilde{a}_i^T \Gamma^{-1} \dot{\hat{a}}_i \right]. \tag{A.3}$$

*The first term inside the sum is uniformly continuous since it is the product of two quantities which were already shown to have bounded time derivatives, namely $\int_{V_i}(q - p_i)^T \phi(q)\, dq$ (an integral of the form (A.2)) and $\dot{p}_i$. Now consider the second term in the sum. It is continuous in time since $\dot{\hat{a}}_i$ is continuous. Expanding it using (4.14) and (4.13) as*

$$\tilde{a}_i^T \Gamma^{-1} (I_{proj_i} - I)(F_i \hat{a}_i + \gamma(\Lambda_i \hat{a}_i - \lambda_i)) \tag{A.4}$$

*shows that it is not differentiable where the matrix $I_{proj_i}$ switches. However, the switching condition (4.15) is such that $\dot{\hat{a}}_i(t)$ is not differentiable only at isolated points on the domain $[0, \infty)$. Also, at all points where it is differentiable, its time derivative is uniformly bounded (since $\dot{\hat{a}}_i$ and the integrands of $\Lambda_i$ and $\lambda_i$ are bounded, and $F_i$ is composed of the kind of integral terms of the form (A.2)). This implies that $\tilde{a}_i^T \Gamma^{-1} \dot{\hat{a}}$ is uniformly continuous. We conclude that $\dot{\mathcal{V}}$ is uniformly continuous.*

**Lemma A.2 (Uniform Continuity for Consensus Controller)** *For the consensus controller, $\dot{\mathcal{V}}$ is uniformly continuous.*

**Proof A.2** *We have*

$$\dot{\mathcal{V}} = \sum_{i=1}^{n} \left[ -\int_{V_i} (q - p_i)^T \phi(q)\, dq\, \dot{p}_i + \tilde{a}_i^T \Gamma^{-1} \dot{\hat{a}}_i \right], \tag{A.5}$$

*therefore the reasoning of the proof of Lemma A.1 applies as long as $\dot{\hat{a}}_i$ can be shown to be uniformly continuous. But $\dot{\hat{a}}_i$ only differs from the basic controller in the presence of the term*

$$\zeta \sum_{j=1}^{n} l_{ij} (\hat{a}_i - \hat{a}_j). \tag{A.6}$$

*The Voronoi edge length, $l_{ij}$, is a continuous function of $p_k$, $k \in \{1, \dots, n\}$. Furthermore, where it is differentiable, it has uniformly bounded derivatives. It was shown in the proof of Lemma A.1 that $\dot{p}_k$ is bounded, so similarly to $\dot{\hat{a}}_i$, the points at which $l_{ij}(p_1(t), \dots, p_n(t))$ is not differentiable are isolated points on $[0, \infty)$. Therefore $l_{ij}$ is uniformly continuous in time. All other terms in $\dot{\hat{a}}_{pre_i}$ were previously shown to be uniformly continuous, so $\dot{\hat{a}}_{pre_i}$ is uniformly continuous. As shown in the proof of Lemma A.1, the projection operation preserves uniform continuity, therefore $\dot{\hat{a}}_i$ is uniformly continuous.*

# Bibliography

[1] D. M. Anderson and S. Urquhart. Using digital pedometers to monitor travel of cows grazing arid rangeland. *Applied Animal Behaviour Science*, 16:11–23, 1986.

[2] Dean M. Anderson. Virtual fencing - past, present, and future. *The Rangelands Journal*, 29:65–78, 2007.

[3] Dean M. Anderson, J. N. Smith, and C. V. Hulet. Livestock behavior - The neglected link in understanding the plant/animal interface. In F. Baker and D. Childs, editors, *Proceedings of the Conference on Multispecies Grazing*, pages 116–148, International Institute for Agricultural Development, Morrilton, 1985.

[4] A. Arsie and E. Frazzoli. Efficient routing of multiple vehicles with no explicit communications. *International Journal of Robust and Nonlinear Control*, 18(2):154–164, January 2007.

[5] I. Barbălat. Systèmes d'équations différentielles d'oscillations non linéaires. *Revue de Mathématiques Pures et Appliquées*, 4:267–270, 1959.

[6] M. A. Batalin and G. S. Sukhatme. Sensor coverage using mobile robots and stationary nodes. In *Proceedings of the SPIE Conference on Scalability and Traffic Control in IP Networks II (Disaster Recovery Networks)*, pages 269–276, Boston, MA, August 2002.

[7] C. Belta and V. Kumar. Abstraction and control for groups of robots. *IEEE Transactions on Robotics and Automation*, 20(5):865–875, October 2004.

[8] D. Bertsekas, A. Nedić, and A. E. Ozdaglar. *Convex Analysis and Optimization.* Athena Scientific, Nashua, NH, 2003.

[9] D. P. Bertsekas and J. N. Tsitsiklis. *Parallel and Distributed Computation: Numerical Methods.* Prentice Hall, 1989.

[10] D. P. Bertsekas and J. N. Tsitsiklis. Comments on "coordination of groups of mobile autonomous agents using nearest neighbor rules". *IEEE Transactions on Automatic Control*, 52(5):968–969, 2007.

[11] B. Bethke, M. Valenti, and J. How. Cooperative vision based estimation and tracking using multiple uav's. In *Advances in Cooperative Control and Optimization*, volume 369 of *Lecture Notes in Control and Information Sciences*, pages 179–189. Springer, Berlin, 2007.

[12] J. Black and T. Ellis. Multi camera image tracking. *Image and Vision Computing*, (11):1256–1267, 2006.

[13] V. D. Blondel, J. M. Hendrickx, A. Olshevsky, and J. N. Tsitsiklis. Convergence in multiagent coordination, consensus, and flocking. In *Proceedings of the Joint IEEE Conference on Decision and Control and European Control Conference*, pages 2996–3000, Seville, Spain, December 2005.

[14] F. Bullo, J. Cortés, and S. Martínez. *Distributed Control of Robotic Networks.* June 2008. Manuscript preprint. Electronically available at http://coordinationbook.info.

[15] Z. Butler, P. Corke, R. Peterson, and D. Rus. From robots to animals: Virtual fences for controlling cows. *International Journal of Robotics Research*, 25:485–508, 2006.

[16] Z. J. Butler, A. A. Rizzi, and R. L. Hollis. Complete distributed coverage of rectilinear environments. In *Proceedings of the Workshop on the Algorithmic Foundations of Robotics*, Hanover, NH, March 2000.

[17] Z. J. Butler and D. Rus. Controlling mobile sensors for monitoring events with coverage constraints. In *Proceedings of the IEEE International Conference of Robotics and Automation*, pages 1563–1573, New Orleans, LA, April 2004.

[18] Q. Cai and J. K. Aggarwal. Tracking human motion in structured environments using a distributed-camera system. *IEEE Transactions on Pattern Analysis and Machine Intelligence*, 21(12):1241–1247, 1999.

[19] C. B. Chang and M. Athens. State estimation for discrete systems with switching parameters. *IEEE Transactions on Aerospace and Electronic Systems*, 14(3):418–424, 1978.

[20] H. Choset. Coverage for robotics—A survey of recent results. *Annals of Mathematics and Artificial Intelligence*, 31:113–126, 2001.

[21] R. Collins, A. J. Lipton, H. Fujiyoshi, and T. Kanade. Algorithms for cooperative multisensor surveillance. *Proceedings of the IEEE*, 89(10):1456–1477, 2001.

[22] N. Correll and A. Martinoli. System identification of self-organizing robotic swarms. In *Distributed Autonomous Robotic Systems 7*, pages 31–40. Springer Japan, 2006.

[23] N. Correll and A. Martinoli. Towards multi-robot inspection of industrial machinery - from distributed coverage algorithms to experiments with miniature robotic swarms. *IEEE Robotics and Automation Magazine*, 16(1):103–112, March 2009.

[24] J. Cortés. Discontinuous dynamical systems - a tutorial on solutions, nonsmooth analysis, and stability. *IEEE Control Systems Magazine*, 28(3):36–73, April 2008.

[25] J. Cortés, S. Martínez, and F. Bullo. Spatially-distributed coverage optimization and control with limited-range interactions. *ESIAM: Control, Optimisation and Calculus of Variations*, 11:691–719, 2005.

[26] J. Cortés, S. Martínez, T. Karatas, and F. Bullo. Coverage control for mobile sensing networks. *IEEE Transactions on Robotics and Automation*, 20(2):243–255, April 2004.

[27] F. Cucker and S. Smale. Emergent behavior in flocks. *IEEE Transactions on Automatic Control*, 52(5):852–862, May 2007.

[28] F. Delmotte, M. Egerstedt, and A. Austin. Data-driven generation of low-complexity control programs. *International Journal of Hybrid Systems*, 4(1&2):53–72, March & June 2004.

[29] D. V. Dimarogonas and K. J. Kyriakopoulos. Connectedness preserving distributed swarm aggregation for multiple kinematic robots. *IEEE Transactions on Robotics*, 24(5):1213–1223, October 2008.

[30] Z. Drezner. *Facility Location: A Survey of Applications and Methods.* Springer Series in Operations Research. Springer-Verlag, New York, 1995.

[31] F. Farshidi, S. Sirouspour, and T. Kirubarajan. Optimal positioning of multiple cameras for object recognition using Cramér-Rao lower bound. In *Proc. of the IEEE International Conference on Robotics and Automation*, pages 934–939, Orlando, Florida, 2006.

[32] Harley Flanders. Differentiation under the integral sign. *American Mathematical Monthly*, 80(6):615–627, 1973.

[33] P. Frasca, R. Carli, F. Fagnani, and S. Zampieri. Average consensus on networks with quantized communication. *Submitted*, 2008.

[34] V. Gazi and K. M. Passino. Stability analysis of swarms. *IEEE Transaction on Automatic Control*, 48(4):692–697, April 2003.

[35] V. Gazi and K. M. Passino. A class of repulsion/attraction forces for stable swarm aggregations. *International Journal of Control*, 77(18):1567–1579, 2004.

[36] Z. Ghahramani and G. Hinton. Variational learning for switching state-space models. *Neural Computation*, 12:831–864, 2000.

[37] C. Godsil and G. Royle. *Algebraic Graph Theory.* Springer, New York, 2004.

[38] D. Gurdan, J. Stumpf, M. Achtelik, K.-M. Doth, G. Hirzinger, and D. Rus. Energy-efficient autonomous four-rotor flying robot controlled at 1kHz. In *Proc. of the 2007 IEEE International Conference on Robotics and Automation*, pages 361–366, Rome, Italy, April 2007.

[39] J. Halloy, G. Sempo, G. Caprari, C. Rivault, M. Asadpour, F. Tache, I. Said, V. Durier, S. Canonge, J. M. Ame, C. Detrain, N. Correll, A. Martinoli, F. Mondada, R. Siegwart, and J.L. Deneubourg. Social integration of robots into groups of cockroaches to control self-organized choices. *Science*, 318(5853):1155–1158, 2007.

[40] E. Hecht. *Optics.* Addison Wesley, Reading, MA, 3 edition, 1998.

[41] C. Hérnandez, G. Vogiatzis, and R. Cipolla. Multiview photometric stereo. *IEEE Transactions on Pattern Analysis and Machine Intelligence*, 30(3):548–554, 2008.

[42] M. L. Hernandez, T. Kirubarajan, and Y. Bar-Shalom. Multisensor resource deployment using posterior Cramér-Rao bounds. *IEEE Transactions on Aerospace and Electronic Systems*, 40(2):399–416, 2004.

[43] M. W. Hirsch and S. Smale. *Differential Equations, Dynamical Systems, and Linear Algebra.* Academic Press, Inc., Orlando, FL, 1974.

[44] A. Howard, M. J. Matarić, and G. S. Sukhatme. Mobile sensor network deployment using potential fields: A distributed, scalable solution to the area coverage problem. In *Proceedings of the 6th International Symposium on Distributed Autonomous Robotic Systems (DARS02)*, Fukuoka, Japan, June 2002.

[45] C. V. Hulet, D. M. Anderson, V. B. Nakamatsu, L. W. Murray, and R. D. Pieper. Diet selection of cattle and bonded small ruminants grazing arid rangeland. *Sheep Research Journal*, 8:11–18, 1982.

[46] A. Jadbabaie, J. Lin, and A. S. Morse. Coordination of groups of mobile autonomous agents using nearest neighbor rules. *IEEE Transactions on Automatic Control*, 48(6):988–1001, June 2003.

[47] Philo Juang, Hidekazu Oki, Yong Wang, Margaret Martonosi, Li-Shiuan Peh, and Daniel Rubenstein. Energy efficient computing for wildlife tracking: Design and early experiences with ZebraNet. In *Proceedings of Conference on Architectural Support for Programming Languages and Operating Systems (ASPLOS)*, volume 5, pages 96–107, San Jose, CA, October 2002.

[48] A. Kashyap, T. Başar, and R. Srikant. Quantized consensus. *IEEE Transaction on Automatic Control*, 43(7):1192–1203, July 2007.

[49] H. Khalil. *Nonlinear Systems*. Prentice-Hall, Upper Saddle River, NJ, 2002.

[50] S. M. Khan and M. Shah. Consistent labeling of tracked objects in multiple cameras with overlapping fields of view. *IEEE Transactions on Pattern Analysis and Machine Intelligence*, 25(10):1355–1360, 2003.

[51] C. S. Kong, N. A. Peng, and I. Rekleitis. Distributed coverage with multi-robot system. In *Proceedings of the Interenational Conference of Robotics and Automation (ICRA 06)*, pages 2423–2429, Orland, FL, May 2006.

[52] A. Krause and C. Guestrin. Near-optimal observation selection using submodular functions. In *Proceedings of 22nd Conference on Artificial Intelligence (AAAI)*, Vancouver, Canada, July 2007.

[53] A. Krause, C. Guestrin, A. Gupta, and J. Kleinberg. Near-optimal sensor placements: Maximizing information while minimizing communication cost. In *In Proceedings of Information Processing in Sensor Networks (IPSN)*, Nashville, TN, April 19-21 2006.

[54] R. Kumar, H. Sawhney, S. Samarasekera, S. Hsu, H. Tao, Y. Guo, K. Hanna ans A. Pope, R. Wildes, D. Hirvonen, M. Hansen, and P. Burt. Aerial video surveillance and exploration. *Proceedings of the IEEE*, 89(10):1518–1539, 2001.

[55] J. LaSalle. Some extensions of liapunov's second method. *IRE Transactions on Circuit Theory*, 7(4):520–527, 1960.

[56] D. T. Latimer IV, S. Srinivasa, V.L. Shue, S. Sonne adnd H. Choset, and A. Hurst. Towards sensor based coverage with robot teams. In *Proceedings of the IEEE International Conference on Robotics and Automation*, volume 1, pages 961–967, May 2002.

[57] W. Li and C. G. Cassandras. Distributed cooperative coverage control of sensor networks. In *Proceedings of the IEEE Conference on Decision ans Control, and the European Control Conference*, Seville, Spain, December 2005.

[58] A. C. Lindberg. Group life. In L. J. Keeling and H. W. Gonyou, editors, *Social Behaviour in Farm Animals*, pages 37–58. CABI Publishing, New York, 2001.

[59] L. Ljung. *System Identification: Theory for the User*. Prentice-Hall, Upper Saddle River, New Jersey, 1999.

[60] S. P. Lloyd. Least squares quantization in pcm. *IEEE Transactions on Information Theory*, 28(2):129–137, 1982.

[61] H. Logemann and E. P. Ryan. Asymptotic behaviour of nonlinear systems. *The American Mathematical Monthly*, 111(10):864–889, December 2004.

[62] K. M. Lynch, I. B. Schwartz, P. Yang, and R. A. Freeman. Decentralized environmental modeling by mobile sensor networks. *IEEE Transactions on Robotics*, 24(3):710–724, June 2008.

[63] S. Martínez, J. Cortés, and F. Bullo. Motion coordination with distributed information. *IEEE Control Systems Magazine*, 27(4):75–88, 2007.

[64] J. McLurkin. Stupid robot tricks: A behavior-based distributed algorithm library for programming swarms of robots. Master's thesis, MIT, 2004.

[65] H. C. Monger. Soil development in the Jornada basin. In K. M. Havstad, L. F. Huenneke, and W. H. Schlesinger, editors, *Structure and Function of a Chihuahuan Desert Ecosystem*, pages 81–106. Oxford University Press, New York, 2006.

[66] N. Moshtagh and A. Jadbabaie. Distributed geodesic control laws for flocking of nonholonomic agents. *Accepted for publication in IEEE Transactions of Automatic Control*, 2006.

[67] K. S. Narendra and A. M. Annaswamy. *Stable Adaptive Systems*. Prentice-Hall, Englewood Cliffs, NJ, 1989.

[68] P. Ögren, E. Fiorelli, and N. E. Leonard. Cooperative control of mobile sensor networks: Adaptive gradient climbing in a distributed environment. *IEEE Transactions on Automatic Control*, 49(8):1292–1302, August 2004.

[69] S. M. Oh, J. M. Rehg, T. Balch, and F. Dellaert. Data-driven MCMC for learning and inference in switching linear dynamic systems. In *Proceedings of $20^{th}$ National Conference on Artificial Intelligence*, pages 944–949, Pittsburgh, USA, 2005. AAAI press.

[70] A. Okabe, B. Boots, K. Sugihara, and S. N. Chiu. *Spatial Tessellations: Concepts and Applications of Voronoi Diagrams*. Wiley Series in Probability and Statistics. Wiley, Chichester, England, 2 edition, 2000.

[71] R. Olfati-Saber and R. R. Murray. Consensus problems in networks of agents with switching topology and time-delays. *IEEE Transactions on Automatic Control*, 49(9):1520–1533, September 2004.

[72] S. Orey. A central limit theorem for m-independent random variables. *Duke Mathematics Journal*, 25:543–546, 1958.

[73] H. A. Paulsen and F. N. Ares. Grazing values and management of black grama and tobosa grasslands and associated shrub ranges of the Southwest. Technical Report 1270, Forrest Service Technical Bulletin, U.S. Government Printing Office, Washington, 1962.

[74] V. Pavlovic, J. M. Rehg, and J. MacCormick. *Advances in Neural Information Processing Systems 13 (NIPS*2000)*, chapter Learning Switching Linear Models of Human Motion. MIT Press, 2001.

[75] M. Pavone, A. Arsie, E. Frazzoli, and F. Bullo. Equitable partitioning policies for robotic networks. In *Proceedings of the International Conference on Robotics and Automation (ICRA 09)*, pages 2356–2361, Kobe, Japan, May 12–17 2009.

[76] M. Pavone, S. L. Smith, F. Bullo, and E. Frazzoli. Dynamic multi-vehicle routing with multiple classes of demands. In *Proceedings of American Control Conference*, St. Louis, Missouri, June 2009.

[77] L. C. A. Pimenta, V. Kumar, R. C. Mesquita, and G. A. S. Pereira. Sensing and coverage for a network of heterogeneous robots. In *Proceedings of the IEEE Conference on Decision and Control*, Cancun, Mexico, December 2008.

[78] 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 *Proceedings of the Eighth International Workshop on the Algorithmic Foundations of Robotics (WAFR 08)*, Guanajuato, Mexico, December 2008.

[79] O. Pizarro and H. Singh. Toward large area mosaicing for underwater scientific applications. *IEEE Journal of Oceanic Engineering*, 28(4):651–672, 2003.

[80] V. M. Popov. *Hyperstability of Automatic Control Systems.* Springer Verlag, New York, 1973.

[81] R. Rao, V. Kumar, and C. J. Taylor. Planning and control of mobile robots in image space from overhead cameras. In *IEEE International Conference on Robotics and Automation*, Barcelona, Spain, April 18-22 2005.

[82] I. Rekleitis, G. Dudek, and E. Milios. Mulit-robot collaboration for robust exploration. *Annals of Mathematics and Artificial Intelligence*, 31:7–49, 2001.

[83] W. Ren and R. W. Beard. Cooperative surveillance with multiple UAV's. In *Distributed Consensus in Multi-vehicle Cooperative Control*, Communications and Control Engineering, pages 265–277. Springer, London, 2008.

[84] S. M. Rutter, R. A. Champion, and P. D. Penning. An automatic system to record foraging behaviour in free-ranging ruminants. *Applied Animal Behaviour Science*, 54:185–195, 1997.

[85] A. Ryan, M. Zennaro, A. Howell, R. Sengupta, and J. K. Hedrick. An overview of emerging results in cooperative UAV control. In *Proc. of the 43rd IEEE Conference on Decision and Control*, volume 1, pages 602–607, Nassau, 2004.

[86] E. P. Ryan. An integral invariance principle for differential inclusions with applications in adaptive control. *SIAM Journal of Control and Optimization*, 36(3):960–980, May 1998.

[87] S. Salapaka, A. Khalak, and M. A. Dahleh. Constraints on locational optimization problems. In *Proceedings of the Conference on Decision and Control*, volume 2, pages 1430–1435, Maui, Hawaii, USA, December 2003.

[88] R. Sanner and J.J.E. Slotine. Gaussian networks for direct adaptive control. *IEEE Transactions on Neural Networks*, 3(6):837–863, 1992.

[89] S. S. Sastry and M. Bodson. *Adaptive control: stability, convergence, and robustness.* Prentice-Hall, Inc., Upper Saddle River, NJ, 1989.

[90] M. Schwager, D. M. Anderson, Z. Butler, and D. Rus. Robust classification of animal tracking data. *Computers and Electronics in Agriculture*, 56(1):46–59, March 2007.

[91] M. Schwager, D. M. Anderson, and D. Rus. Data-driven identification of group dynamics for motion prediction and control. In *Proceedings of the Conference on Field and Service Robotics*, Chamonix, France, July 2007.

[92] M. Schwager, F. Bullo, D. Skelly, and D. Rus. A ladybug exploration strategy for distributed adaptive coverage control. In *Proceedings of the International Conference on Robotics an Automation (ICRA 08)*, pages 2346–2353, Pasadena, CA, May 19-23 2008.

[93] M. Schwager, C. Detweiler, I. Vasilescu, D. M. Anderson, and D. Rus. Data-driven identification of group dynamics for motion prediction and control. *Journal of Field Robotics*, 25(6-7):305–324, June-July 2008.

[94] M. Schwager, B. Julian, and D. Rus. Optimal coverage for multiple hovering robots with downward-facing cameras. In *Proceedings of the International Conference on Robotics and Automation (ICRA 09)*, pages 3515–3522, Kobe, Japan, May 12–17 2009.

[95] M. Schwager, J. McLurkin, and D. Rus. Distributed coverage control with sensory feedback for networked robots. In *Proceedings of Robotics: Science and Systems II*, pages 49–56, Philadelphia, PA, August 2006.

[96] M. Schwager, J. McLurkin, J. J. E. Slotine, and D. Rus. From theory to practice: Distributed coverage control experiments with groups of robots. In O. Khatib, V. Kumar, and G. Pappas, editors, *Experimental Robotics: The Eleventh International Symposium*, volume 54 of *Springer Tracts in Advanced Robotics (STAR)*, pages 127–136, Berlin, 2008. Springer-Verlag.

[97] M. Schwager, D. Rus, and J. J. Slotine. Decentralized, adaptive coverage control for networked robots. *International Journal of Robotics Research*, 28(3):357–375, March 2009.

[98] M. Schwager, J. J. Slotine, and D. Rus. Unifying geometric, probabilistic, and potential field approaches to multi-robot coverage control. In *Proceedings of the 14th International Symposium of Robotics Research (ISRR 09)*, Lucerne, Switzerland, Aug. 31–Sept. 3 2009. Accepted.

[99] M. Schwager, J. J. E. Slotine, and D. Rus. Decentralized, adaptive control for coverage with networked robots. In *Proceedings of the International Conference on Robotics and Automation (ICRA 07)*, pages 3289–3294, Rome, April 10-14 2007.

[100] M. Schwager, J. J. E. Slotine, and D. Rus. Consensus learning for distributed coverage control. In *Proceedings of the International Conference on Robotics and Automation (ICRA 08)*, pages 1042–1048, Pasadena, CA, May 19-23 2008.

[101] J. J. E. Slotine and J. A. Coetsee. Adaptive sliding controller synthesis for nonlinear systems. *International Journal of Control*, 43(6):1631–1651, 1986.

[102] J. J. E. Slotine and W. Li. Composite adaptive control of robot manipulators. *Automatica*, 25(4):509–519, July 1989.

[103] J. J. E. Slotine and W. Li. *Applied Nonlinear Control.* Prentice-Hall, Upper Saddle River, NJ, 1991.

[104] H. G. Tanner, A. Jadbabaie, and G. J. Pappas. Flocking in fixed and switching networks. *IEEE Transactions on Automatic Control*, 52(5):863–868, May 2007.

[105] H. G. Tanner, G. J. Pappas, and V. Kumar. Leader-to-formation stability. *IEEE Transactions on Robotics and Automation*, 20(3):443–455, June 2004.

[106] G. Ferraru Trecate, A. Buffa, and M. Gati. Analysis of coordination in multi-agent systems through partial difference equations. *IEEE Transactions on Automatic Control*, 51(6):1058–1063, June 2006.

[107] J. N. Tsitsiklis. *Problems in Decentralized Decision Making and Computation.* PhD thesis, Department of EECS, MIT, November 1984.

[108] J. N. Tsitsiklis, D. P. Bertsekas, and M. Athans. Distributed asynchronous deterministic and stochastic gradient optimization algorithms. *IEEE Transactions on Automatic Control*, 31(9):803–812, 1986.

[109] T. Vicsek, A. Czirok, E. Ben-Jacob, I. Cohen, and O. Shochet. Novel type of phase transition in a system of self-driven particles. *Physical Review Letters*, 75(6):1226–1229, August 1995.

[110] J. Wainright. Climate and climatological variations in the Jornada Basin. In K. M. Havstad, L. F. Huenneke, and W. H. Schlesinger, editors, *Structure and Function of a Chihuahuan Desert Ecosystem*, pages 44–80. Oxford University Press, New York, 2006.

[111] W. Wang and J. J. E. Slotine. On partial contraction analysis for coupled nonlinear oscillators. *Biological Cybernetics*, 23(1):38–53, December 2004.

[112] W. Wang and J. J. E. Slotine. A theoretical study of different leader roles in networks. *IEEE Transactions on Automatic Control*, 51(7):1156–1161, July 2006.

[113] Tim Wark, Chris Crossman, Wen Hu, Ying Guo, Philip Valencia, Pavan Sikka, Peter Corke, Caroline Lee, John Henshall, Kishore Prayaga, Julian O'Grady, Matt Reed, and Andrew Fisher. The design and evaluation of a mobile sensor/actuator network for autonomous animal control. In *IPSN '07: Proceedings of the 6th international conference on Information processing in sensor networks*, pages 206–215, New York, NY, USA, 2007. Association for Computing Machinery.

[114] A. Weber. *Theory of the Location of Industries.* The University of Chicago Press, Chicago, IL, 1929. Translated by Carl. J. Friedrich.

[115] M. M. Zavlanos and G. J. Pappas. Controlling connectivity of dynamic graphs. In *Proceedings of the 44th IEEE Conference on Decision and Control*, pages 6388–6393, Seville, Spain, December 2005.

[116] M. M. Zavlanos and G. J. Pappas. Potential fields for maintaining connectivity of mobile networks. *IEEE Transactions on Robotics*, 23(4):812–816, August 2007.

[117] F. Zhang and N. E. Leonard. Cooperative filters and control for cooperative exploration. *IEEE Transactions on Automatic Control*, Submitted, 2008.