Distributed, adaptive deployment for nonholonomic mobile sensor networks : theory and experiments

Jose Marcio Luna-Castaneda

José Marcio Luna Castañeda

Candidate

Electrical and Computer Engineering

Department

This dissertation is approved, and it is acceptable in quality and form for publication on microfilm:

Approved by the Thesis Committee:

[illegible] , Chairperson

[illegible]

[illegible]

Accepted:

Dean, Graduate School

Date

Distributed, Adaptive Deployment for Nonholonomic Mobile Sensor Networks: Theory and Experiments

by

José Marcio Luna-Castañeda

B.S., Electronics Engineering, Universidad Distrital Francisco José de Caldas, 2004

THESIS

Submitted in Partial Fulfillment of the Requirements for the Degree of

Master of Science
Electrical Engineering

The University of New Mexico

Albuquerque, New Mexico

November, 2009

Dedication

A José, Aura, Claudia, Juan Diego y David, mi única familia.

A Roberta, el amor de mi vida.

A Mauricio, Sebastián, Javier y Marcela, amigos caros a mi corazón.

Acknowledgments

I would like to thank Professor Rafael Fierro, my academic advisor, and Professor John Wood for giving me the opportunity to work on this research, for all their support and help, and for being part of my committee. I would also like to thank Professor Chaouki Abdallah for his important suggestions, his support and time, and for being part of my committee as well. Finally, I would like to thank Andres Cortez and Francisco Rodriguez for their invaluable collaboration during the experimental process of my research.

This work was supported by DOE URPR (University Research Program in Robotics) grant DE-FG52-04NA25590 and by NSF grants ECCS CAREER # 0811347, IIS # 0812338, CNS # 0709329.

Distributed, Adaptive Deployment for Nonholonomic Mobile Sensor Networks: Theory and Experiments

by

José Marcio Luna-Castañeda

ABSTRACT OF THESIS

Submitted in Partial Fulfillment of the
Requirements for the Degree of

Master of Science
Electrical Engineering

The University of New Mexico

Albuquerque, New Mexico

November, 2009

Distributed, Adaptive Deployment for Nonholonomic Mobile Sensor Networks: Theory and Experiments

by

José Marcio Luna-Castañeda

B.S., Electronics Engineering, Universidad Distrital
Francisco José de Caldas, 2004

M.S., Electrical Engineering, University of New Mexico, 2009

Abstract

In this work we show the Lyapunov stability and convergence of an adaptive and decentralized coverage control for a team of mobile sensors. This new approach assumes nonholonomic sensors rather than the usual holonomic sensors found in the literature. The kinematics of the unicycle model and a nonlinear control law in polar coordinates are used in order to prove the stability of the controller applied over a team of mobile sensors.

This controller is adaptive, which means that the mobile sensors are able to estimate and map a density function in the sampling space without a previous knowledge of the environment. The controller is decentralized, which means that each mobile sensor has its own estimate and computes its own control input based on local information. In order to guarantee the estimate convergence, the mobile sensors implement a consensus protocol in continuous time assuming a fixed network topology and zero communication delays.

The convergence and feasibility of the coverage control algorithm are verified through simulations in Matlab and Stage. The Matlab simulations consider only the kinematics of the mobile sensors and the Stage simulations consider the dynamics and the kinematics of the sensors. The Matlab simulations show successful results since the sensor network carries out the coverage task and distributes itself over the estimated density function. The adaptive law which is defined by a differential equation must be approximated by a difference equation to be implementable in Stage. The Stage simulations show positive results, however, the system is not able to achieve an accurate estimation of the density function. In spite of that, the sensors carry out the coverage task distributing themselves over the sampling space.

Furthermore, some experiments are carried out using a team of four Pioneer 3-AT robots sensing a piecewise constant light distribution function. The experimental results are satisfactory since the robots carry out the coverage task. However, the accuracy of the estimation is affected by the approximation of the adaptation law by difference equations, the number of robots and sensor sensitivity.

Based on the results of this research, the decentralized adaptive coverage control for nonholonomic vehicles has been analyzed from a theoretical approach and validated through simulation and experimentation with positive results. As a future work we will investigate: (*i*) new techniques to improve the implementation of the adaptive law in real time,(*ii*) the consideration of the dynamics of the mobile sensors, and (*iii*) the stability and convergence of the adaptive law for continuous-time variant density function.

Contents

List of Figures **xiii**

List of Tables **xvii**

1 Introduction **1**

1.1 Motivation 1

1.2 Coverage Control in Hazardous Environments 2

1.3 Contributions 3

1.4 Organization of the Thesis 3

2 Related Research **5**

2.1 Sensor Networks 6

2.2 Coverage Control 7

2.3 Nonholonomic Multivehicle Control 9

2.4 Consensus Problems 11

3 Mathematical Preliminaries **13**

3.1 Stability Analysis on Nonautonomous Systems 13

3.1.1 Nonautonomous Systems 14

3.1.2 Stability Definitions for Nonautonomous Systems 14

3.1.3 Lyapunov Functions 15

3.1.4 Barbalat's and Lyapunov-like Lemma 16

3.2 Adaptive Control Fundamentals 18

3.2.1 Model-Reference Adaptive Control 18

3.2.2 Self Tuning Controllers 20

3.2.3 The Gradient Estimator 21

3.3 Nonholonomic Mobile Robots 23

3.3.1 The Unicycle Model 25

3.4 Consensus Problem 27

3.4.1 Consensus Problems on Graphs 27

4 Adaptive Algorithm for Holonomic Sensor Networks **30**

4.1 Voronoi Diagrams 31

4.2 Locational Optimization 31

4.3 Adaptive Control for Holonomic Sensors 33

4.3.1 Weighting Functions 36

5 Adaptive Control for Nonholonomic Sensors **37**

5.1 Nonlinear Steering Control 37

5.2 Stability Analysis 39

5.3 Dynamic Density Function 43

6 Simulation Tests **45**

6.1 Simulation in Matlab 46

6.1.1 Simulation Design in Matlab 46

6.1.2 Simulation Results in Matlab 48

6.2 Simulation in Stage 50

6.2.1 Simulation Design in Stage 50

6.2.2 Simulation Results in Stage 51

7 Experimental Verification **58**

7.1 Experimental Design 58

7.2 Experimental Results 59

8 Multivehicle Testbed **63**

8.1 The Pioneer 3 - AT Robots 64

8.1.1 Sensor Suite 66

8.1.2 Power Source 74

8.2 The P3-AT Software Libraries 75

8.2.1 The Player/Stage/Gazebo Project 75

8.2.2 Hardware-Software Interaction 78

9 Conclusions and Discussion 81

A Complement of the Proof of Stability (Theorem 4) 83

References 85

List of Figures

3.1 Model Reference Adaptive Control system. 19

3.2 Self-Tuning Control system. 21

3.3 A unicycle vehicle in the Cartesian plane with the indicated state variables x, y and θ. 26

5.1 Unicycle model and variables in the goal frame $\{G\}$: Notice the vectors and angles which determine our nonholonomic model in polar coordinates. 39

5.2 Illustration of the nonlinear steering control over eight robots with different initial positions. All the robots go to the position $(0, 0, 0)$. . 40

6.1 Plots of the parameter estimation error $\mathcal{K}(q)^T\bar{\tilde{a}}_i(t)$, the error distance $\bar{\rho}(t)$, the angle α_i and the consensus error c_a for the simulation in Matlab. 53

6.2 Mean squared error of the parameter estimation vector $\hat{a}_i$ averaged over the whole population of robots and the real parameter vector a for the simulation in Matlab. The blue line shows the response of the system without changing the parameter estimation vector $\hat{a}_i$ and the red line illustrates the obtained improvement by applying the change at every switching time t_s. 54

6.3 Simulation in Matlab with a population of 20 nonholonomic robots represented by the blue triangles over a dynamic density function indicated by the multicolor ring-shape contour. The yellow color of the contour means maximum density. This results correspond to the simulation changing $\hat{a}_i = \beta$ at every switching time t_s. The Cyan circles represent the estimated centroids $\hat{C}_{V_i}$ of the i-th robot and the green lines surrounding the robots form their Voronoi cells. Notice the centroidal Voronoi tessellations in (b)–(d). 55

6.4 Plots of the error distance ρ_i averaged over the population of robots, and the angle α_i during the simulation in Stage. 56

6.5 The Stage environment with four P3-AT robots used during the simulation. The light distribution goes from A to B after 150 s. 57

6.6 Plots of the estimate density functions obtained by the team of robots in the simulation environment in Stage. 57

7.1 Plots of the error distance $\bar{\rho}(t)$, the angle α_i and the consensus error c_a for the experiments. 61

7.2 The picture in (a) shows a snapshot of an experiment with four P3 - AT robots sensing a light distribution at 107 s. The figure in (b) shows the estimated density function averaged over the four robots and the plot in (c) illustrates a top view done in Matlab with the robot data. The picture in (d) shows the snapshot of the experiment at 259 s. The figure in (e) shows the estimated density function averaged over the four robots, and the plot in (f) illustrates a top view done in Matlab based on the real data 62

8.1 The pictures illustrate the variety of robotic platforms available in the MARHES lab at the University of New Mexico. 64

8.2 Accesories available for the P3-AT robots at the MARHES lab at the University of New Mexico. 66

8.3 One of the four P3-AT robots available in the MARHES lab. Notice the detailed pictures of three different kinds of sensors, namely, Phidgets light precision sensors (top), Phidgets magnetic sensors (middle) and a Hokuyo UHG-08LX laser range finder (bottom), mounted on a custom fixture attached to the robot. 67

8.4 A Phidgets USB interface I/O board with an attached light precision sensor. Notice the USB port at the bottom of the image and the digital input outputs at the right and left edges respectively. The analog inputs are located at the top edge of the board where the light sensor is connected. 68

8.5 The picture in (a) shows The SICK LMS 200 laser at the MARHES lab at The University of New Mexico and the picture in (b) shows the enclosure ProPak-V3 (lower left) and the antenna GPS-702-GG (upper right) for the GPS system. 73

8.6 The picture in (a) shows the front sonar array of a P3-AT robot and the picture in (b) a 12VDC werker battery used in the P3-AT robots. 75

8.7 A Stage model of a P3-AT robot (in red) with a gripper and an LMS 200 laser exploring a virtual world with different kinds of objects, including another P3-AT robots. 78

8.8 The pictures show the appearance of the real P3-AT and the Gazebo model of a P3-AT. 79

8.9 The diagram illustrates the software-hardware interaction and the software architecture of the robot. Notice that the services of Player run locally in the P3-AT robots allowing a local or remote application to access the sensors and actuators of the real or virtual robot. . . . 80

List of Tables

8.1 General specifications of the P3-AT robots. 65

8.2 General specifications of the Phidgets 1127 light precision sensor . . 69

8.3 General specifications of the Phidgets 1108 magnetic sensor 69

8.4 General specifications of the Hokuyo UHG-08LX laser 70

8.5 General specifications of the SICK LMS 200 laser 71

8.6 General specifications of the ProPak-V3 72

8.7 General specifications of GPS-702-GG 73

8.8 General specifications of the motion encoders of the P3-AT robots . 74

8.9 General specifications of the Werker Battery 74

Chapter 1

Introduction

1.1 Motivation

Some of the technological developments which started occurring during the industrial revolution in the 19th century not only had a significant repercussion over lives of people all around the world, but had a great impact over our natural environment. Some accidents in the past, related to the nuclear and petroleum industries have left dangerous waste in the planet. Oil spills and failures in nuclear plants have generated negative consequences on ecosystems, sometimes creating irreparable damages.

Forest fires are a constant threat to our environment as well. Presently, we can see how difficult it is to put out a large scale forest fire, even in developed countries, where the process can take days. This is not only devastating for the animals and vegetation trapped in the fire but it is also a risk for the life of the firefighters involved. Since the accidents we mentioned above have been and are an actual imminent threat, we should design a mechanism able to react in time to prevent critical damage.

The coverage control using mobile sensor networks is an option to overcome these disasters. Such a tool is applicable to some other important cases such as recovery operations, exploration, rescue missions, automatic surveillance, and geological studies. In several of the mentioned cases human lives can be in danger in a given scenario where the sensors should be close to the area of interest (*e.g.,* areas in presence of contaminants or extreme weather). Therefore, if it is possible to substitute human beings by mobile sensors to carry out a mission in a harmful environment, we are reducing the risk of the people who participate in the process.

1.2 Coverage Control in Hazardous Environments

As mentioned in the motivation section, the exploration of hazardous environments by human beings is a very difficult task that sometimes must be carried out by forcing people to risk their lives. Literature related to oil spills [1] and forest fires [2], [3] shows that some systems present motion dynamics that make them more complicated to overcome without putting humans in danger.

With the actual resources for firefighting, the use of manned aerial vehicles requires skillful enough pilots to avoid crashing in the attempt to put out a fire. Moreover, we do not posses efficient mechanisms to follow the evolution of the fire in real time and the firefighters must get a qualitative estimation of the fire dynamics almost by direct observation. In another scenario, Cortez *et al.,* exposed in [4] that building radiation maps involving nuclear material is still done using people for taking measurements close to the radiated area.

Coverage controllers become promising with the latest developments on wireless communications, material science, new sensors and the constant improvement of computational power. The possibilities to send small unmanned aerial, terrestrial or underwater vehicles which coordinate actions to sense and map an area of interest are increasing as the research in decentralized algorithms and hardware progresses.

1.3 Contributions

This work is motivated by the one presented by Schwager *et al.,* in [5] where the authors describe the development of an adaptive coverage control for mobile sensor networks and provide the stability analysis of the controller. The authors assume that the mobile sensors do not have nonholonomic constraints and that the estimated density function is static.

However, several real world vehicles such as aircrafts at cruising attitude, sea vessels and skid-steered mobile robots have nonholonomic constraints and several phenomena cannot be considered static. The performance of the results given for holonomic mobile sensors can be severely affected or even invalidated [6], when they are adapted to nonholonomic mobile sensors.

Our main goal in this thesis is to provide the necessary mathematical background to use nonholonomic mobile sensors along with the adaptive coverage control presented in [5], and guarantee the stability of the system. Furthermore, we will apply our coverage control over dynamic density functions whose parameters are modeled as piecewise constant functions. We will provide simulations to validate our theoretical conclusions. Finally, in order to study the behavior of the controller in a real environment we will carry out some experiments of the controller using a team of skid-steered mobile robots sensing a dynamic light distribution function.

1.4 Organization of the Thesis

The thesis is organized as follows: Chapter 2 provides a short overview with examples and results related to sensor networks, coverage control, nonholonomic multivehicle control and consensus algorithms which are areas directly related with our problem formulation. Chapter 3 provides the necessary mathematical background in nonlinear control related to stability analysis of nonautonomus systems and presents the basics of adaptive control and self-tuning systems. Furthermore we provide the fundamentals of nonholonomic mobile robots and consensus problems. Chapter 4 describes the adaptive coverage control for holonomic sensor networks which inspired this work. The mathematical background related to Voronoi partitions and locational optimization is presented. In Chapter 5 we present our main theoretical result which shows the stability of the adaptive coverage control for nonholonomic sensor networks. Chapter 6 shows simulation results obtained using Matlab and Stage. Chapter 7 illustrates the experimental results obtained by using a team of four Pioneer 3-AT robots sensing a dynamic light distribution. Chapter 8 provides a technical description of the experimental testbed available at the Marhes lab at the University of New Mexico. Lastly, Chapter 9 summarizes the main conclusions and limitations of our approach as well as future work to overcome those limitations.

Chapter 2

Related Research

The problem of controlling networked robots has gained an increasing interest in recent years because of the technological advances in networking and miniaturization of electro-mechanical systems [7] which enabled the implementation of tools and algorithms for sensing. Using a team of robots rather than a single robot allows the exploration of novel solutions for problems like search and recovery operations, manipulation in hazardous environments [7], exploration, rescue missions, automatic surveillance [5], and geological and ecological studies such as the tracking of algae bloom [8] and oil spills [1], among others.

Concepts such as *network sensors*, *coverage control*, *consensus* and *nonholonomic robots* are common to several scenarios involving the control of networked robots such as the work presented in this thesis. For that reason we review some recent applications and results of those concepts in different contexts.

2.1 Sensor Networks

Sensor networks are groups of simple sensors that can be used to monitor, track or survey an area of interest. Every sensor collects data and shares the information with the others in the group. If the sensors have computational capabilities, they can even process the data to carry out a determined mission. The sensor networks we are interested in are dynamic [1], [7], [8], [9], [10]. Some of them can adapt themselves to the environment [5], [11], [12] in such a way that the sensors can react to changes in the environment and take advantage of their mobility to explore the new areas of interest.

We consider convenient to illustrate some applications which are part of the state of the art in the sensor networks field. Mostofi and Sen propose in [13] a compressive sensing approach to build a map of spatial variations of certain parameter of interest in its environment. The compressive sensing approach provides the theoretical frame to guarantee that the nodes on the sensor network can reconstruct the spatial variations with a considerably incomplete sensing of the area.

Based on the compressive sensing theory they present the foundations of a novel non-invasive mapping technique which applies the Fourier slice theorem in order to build a two-dimensional map of an indoor environment (*e.g.,* a building, a store or a house). The authors use mobile sensors organized in transmitter-receiver couples. The receiver gets the data from a beam sent by the transmitter through the indoor environment and samples the Fourier transform of the two-dimensional map. Afterwards, the agents can use the sparse representation of the signal in the partial domain and the robots can solve the map cooperatively.

Hou and Slotine propose in [14] a dynamic region following formation control for a swarm of robots, which is able to adjust a formation with a desired geometric shape by choosing the right function to describe it. The system is scalable allowing a number of robots to leave the formation or fail without affecting the general behavior of the swarm. The communication between the agents is limited to the adjacent neighbors and the robots do not require any identification or order for the formation to work.

The system implements an adaptive control law and a parameter update law which allow a provable convergence analysis of the system using the Lyapunov-like lemma. The general formation is able to scale, rotate and displace. Some simulations using a swarm of 100 robots contracting and expanding the formation in order to pass through a door are shown. Different shapes of the region such as a circle, a square, an ellipse and a ring are illustrated in the simulations as well.

We have presented two specific applications of sensor networks related to two different disciplines namely, compressive sensing and flocking. The presented cases are located in two different scenarios which show their usefulness and feasibility. In what follows this section we present more recent sensor networks results in other contexts.

2.2 Coverage Control

Based on [15], the coverage control problem involving sensor networks provides the notion of quality of service of the sensing task. Having a cost function which determines a problem-dependent metric of the coverage performance we can implement a controller to determine the *optimal placement* of the sensors in an environment. The *locational optimization problem* [7], [16], which will be explained in detail in Section 4.2, is one of the methodologies applied in coverage control. Other approaches for sensing an unknown environment different than locational optimization are the adaptive triangular mesh generation algorithm proposed by Lee *et al.,* [11] and the Bayesian sequential field estimation of Graham *et al.,* [17].

We describe a couple of recent results in coverage control as an illustration not only of the actual applications but of the potential future solutions it can provide. Lee *et al.,* present in [11] an approach of coverage control with environmental sensing. Given a chemical spill the robots should position themselves over an area such that they concentrate themselves in the area with the greatest amount of the chemical or the most *contaminated* area.

The robots are initially distributed in an arbitrary way over the contaminated area. They start calculating an adaptive triangular mesh where each robot p_i chooses the closest neighbor p_1 to be the first triangular neighbor. Afterwards, the second triangular neighbor p_2 is chosen such that the distance $d(p_i, p_2) + d(p_2, p_1)$ is minimal. The control law forces the distribution of robots to form a mesh of equilateral triangles whose sides length are inversely proportional to the contamination level in the area below them. Then the triangular mesh algorithm concentrates more robots in the more contaminated regions. Notice that with this algorithm no explicit communication is needed if the robots are able to detect their neighbors' location using local sensor measurements.

In [18] Schwager *et al.,* propose an optimization criterion to distribute a team of hovering robots with downward facing cameras to obtain the best view of an environment. They propose a metric based on the minimum information per pixel in order to elaborate a cost function. This cost function is minimized as the hovering robots locate themselves in a three-dimensional space in such a way that the downward facing cameras can completely cover a two-dimensional region of interest. Furthermore the authors prove the robots convergence to locally optimal positions using the LaSalle's invariance principle. They carried out some successful simulations and an implementation using hummingbird quadrotors.

We have presented a couple of the most recent advances published in the area of coverage control. Coverage control and sensor networks are found together very often because the coverage control discipline makes more sense if the covered environment is sensed to fulfill a goal. Now, let us discuss a little bit about the core of the research results presented in this thesis namely, nonholonomic multivehicle applications.

2.3 Nonholonomic Multivehicle Control

Based on [6], Kwok and Martinez state that because of the complexity involved in the analysis of dynamic systems interacting through a network, it is reasonable to consider simple dynamical models such as the popular single integrator [7], [5], [19], [20], or the double and higher order integrators [21], [22], [23]. However, the performance of the existing results can be severely affected or even invalidated under the nontrivial dynamics of nonholonomic systems whose instantaneous movements are restricted.

In this section we present some recent results in the area of nonholonomic multivehicle control to illustrate the complexity involved in considering nonholonomic constraints in some benchmark problems. Lan *et al.,* propose in [24] a hybrid controller to carry out a target tracking with a sensor network. The hybrid controller has two main states: the first one takes the robots to a relatively close position of the target, then the hybrid controller switches to a state where the robots start applying a circular motion around the target in order to capture it. The sensor network is composed by unicycle vehicles. The authors give a formal proof of the trajectory convergence and set invariance of the system.

In [25], Wu and Jiang present a formation control, specifically a leader-follower control for nonholonomic robots with one leader and several followers. The unicycle model is used as well as a switching controller with two states. The first state is activated when the follower is farther from a predefined threshold distance from the leader and applies a non-linear control law to shorten the distance. The follower creates a bidimensional cone which is expanding forwards to contain the leader, whereas it is moving towards the leader. Once the follower reaches the desired distance to the leader, the second state is activated and a finite-feedback controller stabilizes the distance and orientation of the follower. They prove the stability of the switching control and show some experimental results using a team of three Pioneer 3-AT (P3-AT) robots.

Oikonomopoulos *et al.,* present in [26] a multiagent coordination algorithm aimed to work with a special model of aircraft-like mobile agents. The proposed model is an input-constrained hybrid automaton and is known not to be generally safe. They proposed an algorithm based on workspace partitioning, shortest-path graph search and collision detection in order to show the feasibility of the system and correct the non-generally safe feature. Even though the algorithm is verified thorough some simulations it is not fully formalized.

Recently, Kwok and Martinez in [6] have used a hybrid system approach to attack the decentralized control problem described in [7] which is closely related to the problem we are dealing with in this thesis. The authors model the problem as a hybrid automaton with a series of states implementing some forward-left, forward-right, hover-left and hover-right behaviors in a team of unicycle agents with fixed and variable forward velocity. The agents are driven to the centroids of their respective Voronoi partitions assuming a previous knowledge of the sampling space.

The main goal is to optimally position the sensor network over the sampling space in the presence of a density function. The sensors can solve the facility problem known as a locational optimization problem and concentrate more sensors in the areas of interest and less sensors in the remaining areas. The authors assume that the robots have a previous knowledge of the sampling space.

Although we explain the locational optimization problem in Section 4.2, it is worthy to mention that this problem involves a cost function which is minimized by a gradient-descent algorithm, therefore the robots optimize their location over the density function as the algorithm converges. The authors present a convergence analysis of the system based on the invariance principle for hybrid systems [27].

As the reader can notice from the four examples given above, the nonholonomic multivehicle control requires a special treatment even for traditional problems that have been apparently solved in the past, since the consideration of simple dynamics can have destabilizing effects. Now, we introduce some related results on consensus algorithms which as explained in Chapter 4 are used to acquire an agreement within the group of agents involved in the multivehicle control problem.

2.4 Consensus Problems

Based on [9] in a consensus problem a group of nodes in a network topology tries to reach an agreement of a quantity of interest. The nodes exchange information with their neighbors and update their quantities iteratively using a consensus algorithm. We usually find the consensus problem applied to mobile sensor networks in different scenarios such as collective behavior of flocks and swarms, and formation control [28]. In the case of formation control, the consensus algorithm implements a control law which drives the robots to a position that fulfils the agreement requirement.

Now, we present a couple of recent results related to consensus algorithms applied to formation control. Franceschelli *et al.*,, in [21] propose a methodology to solve the consensus problem for multi-agent systems with kinematic constraints. Given a team of agents with nonholonomic constraints and a set of states associated to each agent, we have that the agents should reach a common value of the set of states. The agents should consider network constraints and limitations in the information sharing. The kinematics constraints considered for these problems are related to finite maximum speed and finite maximum acceleration among other possibilities.

The network constraints are related to the graph of the network *e.g.,* a connected graph which implies that all the nodes are always connected to the network. The authors solve the rendezvous problem taking all the agents to the centroid of the network, so the agents should reach the consensus about the network centroid location. The problem is solved in a decentralized fashion through a one-step horizon optimization. They present successful simulations showing all the robots getting to the rendezvous point after reaching the consensus.

Listmann *et al.,* present in [29] a consensus for formation control using nonholonomic constraints. But in contrast to some previous works [21] – [23], which assume fully actuated robotic systems (*e.g.,* the simple and double integrator kinematics equations), the authors assume the kinematic equations of the unicycle model. Furthermore the authors add an obstacle avoidance algorithm in order to improve the performance of the controller.

A rendezvous algorithm is implemented to be solved using the consensus algorithm and an artificial field is implemented in order to keep the robot formation while traveling through the environment. A stability proof of the rendezvous controller is carried out applying the LaSalle-Krasovskii invariance principle and some simulation results are presented.

The former examples illustrate how the four main concepts namely sensor networks, coverage control, nonholonomic multivehicle control and consensus algorithms involved in this research work in different scenarios. As we describe later in Section 5.2, we put all these concepts together in our mobile sensor network deployment problem.

Chapter 3

Mathematical Preliminaries

In this chapter we provide some classic definitions, theorems and lemmas of nonlinear control theory which are necessary tools to be used in the development of our main theoretical ideas behind our decentralized, adaptive algorithm. The concepts of nonlinear control presented in this chapter are mainly taken from [30] and [31] and we do not provide the proofs of the theorems and lemmas since they are available in the indicated references.

3.1 Stability Analysis on Nonautonomous Systems

In this section we provide the definition of nonautonomous systems. Afterwards, we define several types of stability and then we introduce the definition of Lyapunov functions, positive definite functions and decrescent functions. After that, we introduce two important results namely, the Barbalat's lemma and the Lyapunov-like lemma.

3.1.1 Nonautonomous Systems

Given the $n \times 1$ *state vector* $\mathbf{x}$, the $m \times 1$ *input vector* $\mathbf{u}$ and the $n \times 1$ *vector function* $\mathbf{f}$ we can construct a set of n nonlinear equations of the form

$$\dot{\mathbf{x}} = \mathbf{f}(\mathbf{x}, \mathbf{u}, t). \tag{3.1}$$

The variable n is called the *order* of the system. A specific value of the state vector $\mathbf{x}$ at a time t is called a *point* in the state-space. Since the state vector $\mathbf{x}$ is varying with time, it determines some trajectories on the state-space called *state trajectories*.

Another equation to consider is

$$\mathbf{y} = \mathbf{h}(\mathbf{x}, \mathbf{u}, t), \tag{3.2}$$

which is called the *output* equation of the system and determines the state variables that are of particular interest to our model. The equations (3.1) and (3.2) together are usually called the *state-space* model of the system. Now, we provide the definition of nonautonomous systems.

Definition 1. *The nonlinear system given by (3.1) is* <u>*nonautonomous*</u> *if and only if* $\mathbf{f}$ *depends explicitly on time. Otherwise the system in (3.1) is said to be autonomous.*

3.1.2 Stability Definitions for Nonautonomous Systems

Let us start with the definition of equilibrium point. Afterwards, we provide some stability definitions for nonautonomous systems.

Definition 2 (Equilibrium Point)**.** *A state* $\mathbf{x}_{eq}$ *is said to be an* <u>*equilibrium point*</u> *if once the state vector* $\mathbf{x}(t)$ *is equal to* $\mathbf{x}_{eq}$ *it remains equal to* $\mathbf{x}_{eq}$ *for all future time.*

Now, consider the nonautonomous system given by (3.1) where $f : [0,\infty) \times D_x \to \mathbb{R}^n$ is piecewise continuous in t and locally Lipschitz in $\mathbf{x}$ on $[0,\infty) \times D_x$, and $D_x \subset \mathbb{R}^n$ is a domain containing the origin $\mathbf{x} = \mathbf{0}$. Then we have the following definitions of stability.

Definition 3. *The equilibrium point $\mathbf{x} = \mathbf{0}$ is <u>stable</u> at $t = t_0$ if for any $\epsilon > 0$, there exists a positive scalar $\delta(\epsilon, t_0)$ such that*

$$\|\mathbf{x}(t_0)\| < \delta \;\Rightarrow\; \|\mathbf{x}(t)\| < \epsilon \quad \forall t \geq t_0.$$

Definition 4. *The equilibrium point $\mathbf{x} = \mathbf{0}$ is <u>asymptotically stable</u> at $t = t_0$ if it is stable and there exists a positive scalar $\delta(t_0)$ such that*

$$\|\mathbf{x}(t_0)\| < \delta(t_0) \;\Rightarrow\; \mathbf{x}(t) \to 0 \;\; \textit{as} \;\; t \to \infty.$$

Definition 5. *The equilibrium point $\mathbf{x} = \mathbf{0}$ is <u>globally asymptotically stable</u> if*

$$\mathbf{x}(t) \to \mathbf{0} \textit{ as } t \to \infty \textit{ for any } \mathbf{x}(t_0).$$

3.1.3 Lyapunov Functions

Now, we define positive definite functions which allow us to define Lyapunov functions.

Let $\mathbf{x} = \mathbf{0}$ be an equilibrium point for (3.1) and $D_x \subset \mathbb{R}^n$ is a domain containing the origin $\mathbf{x} = \mathbf{0}$. Let $V : D_x \to \mathbb{R}$ be a continuously differentiable function then we have the following definitions.

Definition 6. *The scalar continuous function $V(\mathbf{x})$ is called <u>locally positive definite</u> if $V(\mathbf{0}) = 0$ and inside a ball $\mathcal{B}(\mathbf{0}, r)$ we have that $\mathbf{x} \neq \mathbf{0} \Rightarrow V(\mathbf{x}) > 0$.*

Definition 7. *The scalar continuous function $V(\mathbf{x})$ is called <u>globally positive definite</u> if $V(\mathbf{0}) = 0$ and $\mathbf{x} \neq \mathbf{0} \Rightarrow V(\mathbf{x}) > 0$ in the whole state space.*

Now, we give the definition of Lyapunov functions.

Definition 8. *The function $V(\mathbf{x})$ is said to be a Lyapunov function if it is positive definite inside a ball $\mathcal{B}(\mathbf{0}, r)$, has continuous partial derivatives and its time derivative along any state trajectory of the system in (3.1) is negative semidefinite, i.e.,*

$$\dot{V}(\mathbf{x}) \leq 0.$$

3.1.4 Barbalat's and Lyapunov-like Lemma

Now, we state the definitions of decrescent functions, continuous functions and uniformly continuous functions in order to state the Barbalat's lemma.

Definition 9. *The scalar function $V(\mathbf{x}, t)$ is said to be decrescent if $V(\mathbf{0}, t) = 0$, and if there exists a time-invariant positive definite function $V_1(\mathbf{x})$ such that*

$$V(\mathbf{x}, t) \leq V_1(\mathbf{x})$$

i.e., the function $V(\mathbf{x}, t)$ is decrescent if it is dominated by an invariant positive definite function.

From calculus we have that if a function f is lower bounded and decreasing ($\dot{f} \leq 0$), then $\lim_{t\to\infty} f(t)$ exists and is finite. However if $\lim_{t\to\infty} f(t)$ exists and is finite then it does not imply that $\dot{f}(t) \to 0$. As a counter-example we have that the function $f(t) = \sin(\log t)$ taken from [30] does not converge. However, $\dot{f}(t) = \frac{\cos(\log t)}{t} \to 0$ as $t \to \infty$

Furthermore, if $\dot{f}(t) \to 0$ does not imply that $\lim_{t\to\infty} f(t)$ exists and is finite. As a counter-example we have from [30] that the function $f(t) = e^{-t}\sin(e^{2t}) \to 0$ as $t \to \infty$ but $\dot{f}(t)$ is unbounded.

Then, we can ask what the conditions are to guarantee that $\dot{f}(t) \to 0$ given that $\lim_{t\to\infty} f(t)$ exists and is finite. The answer is given by an important result called

Barbalat's lemma. But before we state the Barbalat's lemma, we need to give the definitions of continuity and uniform continuity.

Definition 10. *A function $f(t)$ is <u>continuous</u> on $[0, \infty)$ if*

$$\forall\ t_1 \geq 0,\ \forall\ \epsilon \geq 0,\ \exists\ \delta(\epsilon, t_1) > 0, \forall\ t \geq 0,\ |t - t_1| < \delta\ \Rightarrow\ |f(t) - f(t_1)| < \epsilon.$$

Definition 11. *A function $f(t)$ is <u>uniformly continuous</u> on $[0, \infty)$ if*

$$\forall\ \epsilon \geq 0,\ \exists\ \delta(\epsilon) > 0, \forall\ t_1 \geq 0,\ \forall t \geq 0,\ |t - t_1| < \delta\ \Rightarrow\ |f(t) - f(t_1)| < \epsilon.$$

Now, we proceed to state the Barbalat's lemma.

Lemma 1 (Barbalat's lemma)**.** *If for the differentiable function $f(t)$, $\lim_{t\to\infty} f(t)$ exists and is finite and $\dot{f}$ is uniformly continuous, then $\dot{f}(t) \to 0$ as $t \to \infty$.*

The Lyapunov-like lemma is a direct consequence of the Barbalat's lemma,

Lemma 2 (Lyapunov-like lemma)**.** *If a scalar function $V(\mathbf{x}, t)$ satisfies the following conditions:*

- *$V(\mathbf{x}, t)$ is lower bounded,*
- *$\dot{V}(\mathbf{x}, t)$ is negative semidefinite,*
- *$\dot{V}(\mathbf{x}, t)$ is uniformly continuous in time,*

then $\dot{V}(\mathbf{x}, t) \to 0$ as $t \to \infty$.

Now, we proceed to introduce another important tool related to this research called adaptive control.

3.2 Adaptive Control Fundamentals

In the nonlinear control scenario the systems are modeled according to specific parameter values. These parameters can be *constant* or be *slowly varying uncertain* parameters. If the parameters start changing with some uncertainty the gains of the controller should be adjusted in order to keep the stability of the system.

As an example of this slowly varying uncertainty we have the robot manipulators [30] which should deal with objects of different sizes, weights and shapes. The gains of the manipulator controller should be adjusted to guarantee the correct behavior of the manipulator for every different object. Other examples are the autopilots for controlling a ship and an aircraft. In both cases the weather conditions and some other phenomena changing the behavior of the fluid (water or air) induce uncertainty in the model parameter values. Those parameters should be estimated by the controller in order to adjust the gain to keep the stability of the system.

Adaptive controllers are able to adjust the controller parameters on line using special mechanisms. It is worthy to mention that the adaptive controller is assumed to be fast enough to react to changes in the parameters of the plant. If the plant is changing too fast to be tracked by the adaptive controller, the stability of the system cannot be guaranteed. Based on [30] there are two main approaches namely, Model-Reference Adaptive Control (MRAC) and Self Tuning Controllers (STC).

3.2.1 Model-Reference Adaptive Control

A general scheme proposed in [30] is given in Fig. 3.1. The *plant* is supposed to have a known structure which means that we know the structure of the dynamic equation of the system but we do not know some parameters in the equation which must be estimated. The *reference model* is a dynamic equation which provides the desired

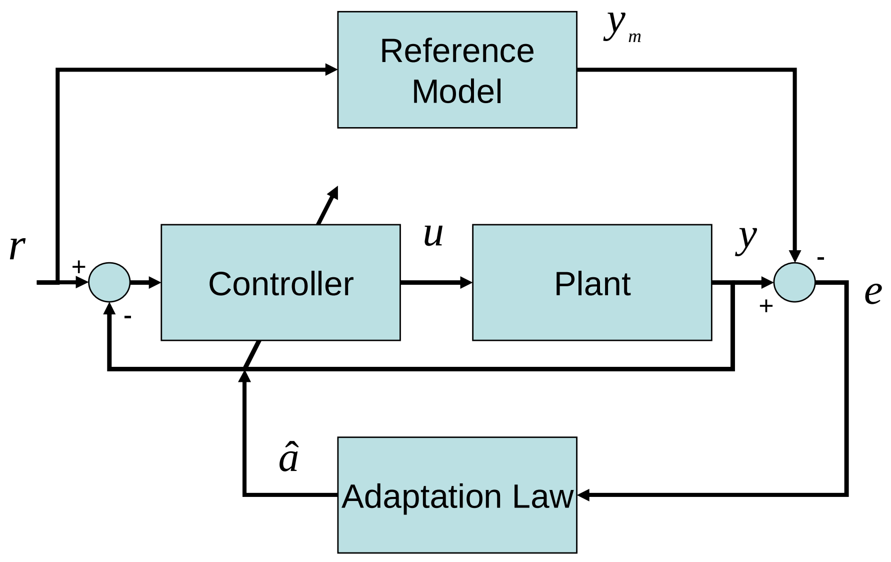


Figure 3.1: Model Reference Adaptive Control system.

response of the plant; it should consider the physical constraints of the plant and the performance specifications such as overshoot, rise time and frequency response among others.

The *controller* makes reference to our control law but in this case, this controller possesses adjustable parameters. Therefore we have a family of controllers rather than a unique controller. This controller is assumed to have *perfect tracking* capacity which means that if the parameters of the plant are known then the response of the system should coincide with the ideal one. If the parameters of the plant are not known then the response of the plant approximates the response of the ideal one asymptotically.

Lastly the *adaptation law* is the mechanism to adjust the parameters of the controller based on the error between the ideal response and the actual response. The main goal is to take the mentioned error to zero asymptotically.

3.2.2 Self Tuning Controllers

In contrast with MRAC, the STC does not use a reference model to approximate the ideal response of the plant. As indicated in Fig. 3.2 the STC has three main blocks namely, the plant, the controller and the adaptation law.

The system starts with some initial plant parameters[1] $\hat{\mathbf{a}}_\mathbf{0}$ which are sent to the controller. The controller calculates an input $\mathbf{u}$ to excite the plant. Simultaneously the estimator takes the input $\mathbf{u}$ and the plant output $\mathbf{y}$ and calculates a new set of estimated parameters $\hat{\mathbf{a}}$, which are sent again to the controller to start a new cycle.

Based on [30] the estimator should be capable of finding the set of parameters that fits the input-output data from the plant. However, the estimation of the real parameters can be guaranteed just under some persistently exciting condition, to be explained later.

The gradient estimator is the simplest on line estimator among the most popular prediction-error-based estimators namely, the *standard least-squares estimator*, and the *least squares with exponential forgetting* described in [30].

[1]The initial parameters are not completely arbitrary since they should fulfill the constraints related to the specific plant.

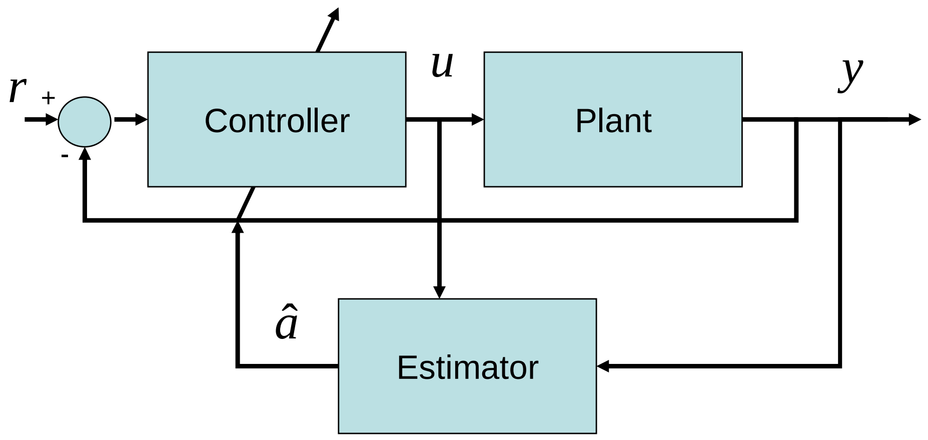


Figure 3.2: Self-Tuning Control system.

3.2.3 The Gradient Estimator

The first step during a parameter estimation process is to determine the *estimation model* to be used. The estimation model is basically the assumed mathematical structure used to approximate the behavior of the estimated parameter. A useful estimation model is the *linear parameterization* which is defined as follows

$$\mathbf{y}(t) = \mathbf{W}(t)\mathbf{a}, \tag{3.3}$$

where the $n \times 1$ vector $\mathbf{y}$ represents the "*outputs*" of the system, the $m \times 1$ vector $\mathbf{a}$ represents the plant parameters to be estimated and $\mathbf{W}$ is a signal matrix used to model the system outputs. The vector $\mathbf{y}$ and the signal matrix $\mathbf{W}$ must be known and the parameter vector $\mathbf{a}$ is unknown.

Assume that we have already an estimate $\hat{\mathbf{a}}(t)$ of the parameter vector $\mathbf{a}(t)$ at the time instant t. Then, the estimate of the system output will be

$$\hat{\mathbf{y}}(t) = \mathbf{W}(t)\hat{\mathbf{a}}, \tag{3.4}$$

with $\hat{\mathbf{y}}(t)$ called the predicted output at time t.

The *prediction error* is the difference between the predicted output and the measured output and is given by

$$\mathbf{e} = \mathbf{W}\hat{\mathbf{a}} - \mathbf{W}\mathbf{a} = \mathbf{W}\tilde{\mathbf{a}}, \tag{3.5}$$

where $\tilde{\mathbf{a}} = \tilde{\mathbf{a}} - \mathbf{a}$ is called the *parameter estimation error.*

In the gradient estimator the estimated parameters are updated in the converse direction of the gradient of the squared prediction error, *i.e.,*

$$\dot{\hat{\mathbf{a}}} = -\gamma \frac{\partial[\mathbf{e}^T\mathbf{e}]}{\partial\hat{\mathbf{a}}}, \tag{3.6}$$

where $\gamma > 0$ and is called the *estimator gain.*

From (3.5) we can rewrite (3.6) as

$$\begin{aligned} \dot{\hat{\mathbf{a}}} &= -\gamma\mathbf{W}^T\mathbf{e} \\ &= -\gamma\mathbf{W}^T\mathbf{W}\tilde{\mathbf{a}}, \end{aligned} \tag{3.7}$$

Lastly, proposing the Lyapunov function candidate

$$V = \tilde{\mathbf{a}}^T\tilde{\mathbf{a}}, \tag{3.8}$$

and calculating its derivative we have that

$$\dot{V} = -2\gamma \tilde{\mathbf{a}} \mathbf{W}^T \mathbf{W} \tilde{\mathbf{a}} \leq 0. \tag{3.9}$$

And we found that based on the Lemma 2 (Lyapunov-like lemma) the gradient estimator is always stable, and since the Lyapunov function is the squared parameter error we conclude that the parameter error is always decreasing and we guarantee that the product $\mathbf{W}\tilde{\mathbf{a}} \to 0$ as $t \to \infty$. However, notice that $\tilde{\mathbf{a}}$ not necessarily goes to zero. In fact the convergence of the estimated parameters to the real parameters depends on the excitation of the input signals and is usually called *persistency of excitation* condition and is given by the following theorem.

Theorem 1 (Persistency of Excitation Condition)**.** *If the matrix* $\mathbf{W}(t)$ *fulfills the persistency of excitation condition then there exist* $\alpha > 0$ *and* $T > 0$ *such that*

$$\int_t^{t+T} \mathbf{W}^T \mathbf{W}(r) dr \geq \alpha \mathbf{I} \tag{3.10}$$

then the parameter error vector $\tilde{\mathbf{a}}$ *converges exponentially to zero.*

3.3 Nonholonomic Mobile Robots

Based on [32] we can consider a rigid mobile robot with an associated generalized joint variable vector $\mathbf{q} = [q_1, q_2, \cdots, q_n]^T \in Q \subseteq \mathbb{R}^n$ moving in a workspace Ω. The entries q_i $\forall i = 1, \cdots, k$ of the vector $\mathbf{q}$ represent the state variables of the system which are usually position and orientation variables. Robot motions are *constrained* to a subset of the set of attainable positions, velocities and accelerations because they are usually aimed to carry out tasks by interacting with different objects in the environment. We can model a set of k independent motion constraints using equations of the form

$$\mathbf{a_i}(\mathbf{q}, \dot{\mathbf{q}}, t) = 0; \quad \forall i = 1, 2, \cdots, k, \tag{3.11}$$

so the space of attainable velocities $\dot{\mathbf{q}} \in V_q$ is reduced to a $(n-k)$ dimensional subspace without changing the dimension of the space Q [32].

If we group the independent constraints in a matrix as follows,

$$\mathbf{A}(\mathbf{q},t) = [\mathbf{a_1}(\mathbf{q},\dot{\mathbf{q}},t), \mathbf{a_2}(\mathbf{q},\dot{\mathbf{q}},t), \cdots, \mathbf{a_k}(\mathbf{q},\dot{\mathbf{q}},t)]^T, \tag{3.12}$$

and $\mathbf{A}(\mathbf{q},t) = 0$ then the matrix $\mathbf{A}(\mathbf{q},t)$ represents a set of *holonomic* or *integrable* constraints. Otherwise, the constraints are called *nonholonomic* or *nonintegrable.*

As an example, a kinematically constrained robotic system such as a car-like robot has to deal with nonholonomic constrains given by the impossibility of moving sideways. Therefore, some tasks like the parallel parking of a vehicle in a parking lot would be easier to carry out if car-like vehicles could move sideways, *i.e.,* if they where holonomic.

Different kinds of characterization of nonholonomic constraints besides the one given in (3.12) are possible *e.g.,* nonholonomic constraints related to the position of the robot but not to the velocity $\mathbf{a_i}(\mathbf{q},t) = 0 \;\; \forall i = 1, \cdots, k$, or the representation of k obstacles in a workspace Ω by inequalities of the form,

$$\mathbf{a_i}(\mathbf{q},t) \leq 0 \;\; \forall i = 1, \cdots, k. \tag{3.13}$$

We can associate nonholonomic constraints to robotic manipulators as well; however, for this particular research we limit the discussion to nonholonomic kinematic constraints for mobile robots. Furthermore we assume that the k kinematic constraints are independent of time and can be expressed as,

$$\mathbf{A}(\mathbf{q})\dot{\mathbf{q}} = 0. \tag{3.14}$$

Let us assume a mobile robot with k independent nonholonomic constraints defined by (3.14). If we get a full rank matrix $\mathbf{S}(\mathbf{q})$ of size $(n-k) \times (n-k)$ which spans the null space of $\mathbf{A}(\mathbf{q})$, *i.e.,*

$$\mathbf{S}^T(\mathbf{q})\mathbf{A}^T(\mathbf{q}) = 0, \tag{3.15}$$

then we can find a vector $\nu \in \mathbb{R}^{n-k}$ such that

$$\dot{\mathbf{q}} = \mathbf{S}(\mathbf{q})\nu(t) \quad \forall t, \tag{3.16}$$

where the vector $\nu(t)$ is usually called the velocity vector. For several cases it is defined as $\nu(t) = [v(t), \omega(t)]^T$ where $v(t)$ and $\omega(t)$ are the linear and angular velocity of the mobile robot.

The expression given in (3.16) represents the kinematic equation of the constraints on $\dot{\mathbf{q}}(t)$ in terms of the velocity vector $\nu(t)$. Now, we are ready to analyze the unicycle model which is commonly found in the literature [32]–[33].

3.3.1 The Unicycle Model

In Fig. 3.3 we show the position variables related to the unicycle model. The simple unicycle behavior is based on the fact that neglecting balancing concerns, the vehicle can only move in a direction normal to the axis of the driving wheels. The vehicle should satisfy the *pure rolling* condition given as follows [32],

$$\dot{y}\cos\theta - \dot{x}\sin\theta = 0, \tag{3.17}$$

where $\dot{x} = \frac{dx}{dt}$ and $\dot{y} = \frac{dy}{dt}$ and θ is the heading angle of the robot as shown in Fig. 3.3.

From (3.15) we have that $\mathbf{S}(\mathbf{q})$ is given by,

$$\mathbf{S}(\mathbf{q}) = \begin{bmatrix} \cos(\theta) & 0 \\ \sin(\theta) & 0 \\ 0 & 1 \end{bmatrix} \tag{3.18}$$

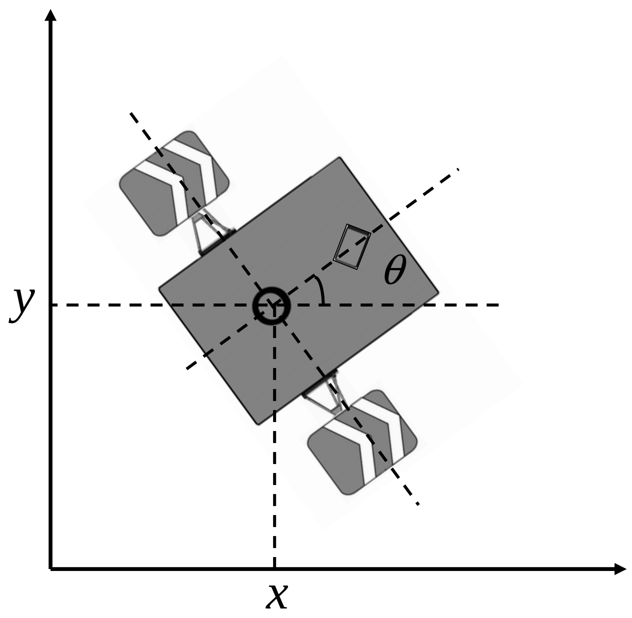


Figure 3.3: A unicycle vehicle in the Cartesian plane with the indicated state variables x, y and θ.

Therefore (3.16) gives the following kinematic equation model of the simple unicycle,

$$\begin{bmatrix} \dot{x} \\ \dot{y} \\ \dot{\theta} \end{bmatrix} = \begin{bmatrix} \cos(\theta) & 0 \\ \sin(\theta) & 0 \\ 0 & 1 \end{bmatrix} \begin{bmatrix} v \\ w \end{bmatrix} \tag{3.19}$$

which are the equations we use to characterize our mobile sensors. Notice that we are neglecting the forces and torques in this model, however, as shown in [32] it is possible to incorporate the dynamics and that is part of our future research.

3.4 Consensus Problem

In several sensor network applications the agents need to reach an agreement of a certain quantity of interest [9]. This quantity can be related to the state variables of every agent or not. The agents should be able to transmit their quantity of interest to their neighbors through the network in order to get the agreement or *consensus*. Furthermore the agents should implement a *consensus protocol* which basically depends on the dynamics of the network.

For this particular case, we assume that the mobile sensor network has a fixed topology and zero communication time delay. Moreover, as described before in Section 3.3.1, the agents motion is modeled using continuous dynamics.

3.4.1 Consensus Problems on Graphs

Let us define the directed graph $\mathbf{G} = (\mathbf{V}, \mathbf{E})$ of order n. The nodes are indicated by the vector $\mathbf{V} = \{v_1, \ldots, v_n\}$, and the edges by $\mathbf{E} = \{e_1, \ldots, e_l\}$, where $e_i = \{v_j, v_k\}$. Furthermore $\mathbf{E} \subseteq \mathbf{V} \times \mathbf{V}$ and the adjacency matrix $\mathbf{A}(i, j)$ is defined as,

$$\mathbf{A}(i,j) = \begin{cases} a_{ij} \geq 0 & \text{for } \{v_i, v_j\} \in E, \\ 0 & \text{otherwise.} \end{cases}$$

where $a_{ij} = 1$ indicates which vertices of the directed graph are adjacent to the jth node. Notice that since the graph is directed $\mathbf{A}(i,j) \neq \mathbf{A}(j,i)$.

The set of neighbors of node v_i is defined as $\mathcal{N}_i = \{v_j \in V | (v_i, v_j) \in E\}$. The out-degree of the node v_i is defined as $\deg_{out}(v_i) = \sum_{i=1}^{n} a_{ij}$. Similarly the in-degree of the node v_i is $\deg_{in}(v_i) = \sum_{i=1}^{n} a_{ji}$.

Definition 12 (Balanced Graphs)**.** *A node v_i of a directed graph $\mathbf{G} = (\mathbf{V}, \mathbf{E})$ is* *balanced if and only if*

$$deg_{out}(v_i) = deg_{in}(v_i). \tag{3.20}$$

A graph $\mathbf{G} = (\mathbf{V}, \mathbf{E})$ *is balanced if and only if all its nodes are balanced i.e.,*

$$\sum_j a_{ij} = \sum_j a_{ji}, \quad \forall i. \tag{3.21}$$

In this work, we assume that our sensor network forms a balanced graph, therefore we can build the degree matrix of the graph G as $\text{diag}_{i=1}^{n}(|\mathcal{N}_i|)$, where $|\mathcal{N}_i| = \text{deg}_{out}(v_i) = \text{deg}_{in}(v_i)$.

The laplacian matrix is defined as $\mathbf{L} = \text{diag}_{i=1}^{n}(|\mathcal{N}_i|) - A$. The row sums of the laplacian matrix is zero, therefore the laplacian matrix has a zero eigenvalue with an associated eigenvector $\mathbf{1} = \{1, \ldots, 1\}^T$.

If any two nodes of a graph can be connected through a path following the direction of the edges the graph is said to be a *connected graph.* We assume that our network topology forms a connected graph therefore $\mathbf{L1} = \mathbf{1}^T\mathbf{L} = 0$. Then we have that $\mathbf{x}^T\mathbf{L}\mathbf{x} \geq 0 \;\; \forall \mathbf{x}$, and $\mathbf{x}^T\mathbf{L}\mathbf{x} = 0$ implies $\mathbf{x} = \mathbf{0}$ or $\mathbf{x} = \mathbf{1}c$ for some $c \in \mathbb{R}$.

In [9] the authors propose the following linear consensus protocol for fixed network topology and zero communication time delay,

$$u_i = \sum_{v_j \in \mathcal{N}_i} a_{ij}(x_j - x_i) \tag{3.22}$$

with the state of the network evolving according to the following linear system,

$$\dot{\mathbf{x}}(t) = -\mathbf{L}\mathbf{x}(t), \tag{3.23}$$

where $\mathbf{L}$ is the graph laplacian.

Finally, Olfati-Saber *et al.,* state and prove the following theorem in [9],

Theorem 2. *Consider the network of integrator agents defined by (3.23) with a fixed topology* $\mathbf{G}(\mathbf{V}, \mathbf{E})$ *which is a strongly connected directed graph. Then the linear consensus protocol given by globally asymptotically solves the average-consensus problem if and only if* $\mathbf{1}^T\mathbf{L} = \mathbf{0}$.

We do not provide the proof of this theorem, however, we exhort the reader to review the Appendix A after reading Chapters 4 and 5 where we provide the proof of the consensus convergence related to our particular case.

Chapter 4

Adaptive Algorithm for Holonomic Sensor Networks

In this section we introduce some additional theoretical background involved in the development of our adaptive and decentralized controller for nonholonomic robots. We do not consider necessary to use the bold notation to differentiate vectors and scalars in the following chapters as we did in Chapter 3.

As mentioned in the introduction, we are interested in employing dynamic sensor networks to achieve an estimation of a density function which represents a concentration of a measurable phenomenon. The sensory function can be considered static (*e.g.,* a stable methane concentration in a garbage dump) or dynamic (*e.g.,* an oil spill [1] or a forest fire [2]).

In some works as the ones presented in [34] and [35], we find applications that involve Voronoi partitions. They are a typical feature of several biological systems [36] and recently have received special attention by mathematicians for their application in disciplines such as cellular biology, image compression, statistics and robotics among others. Before any further discussion, let us start with some necessary definitions.

4.1 Voronoi Diagrams

We based the following definition on the one in [36]

Definition 13. *Given an open set $Q \subseteq \mathbb{R}^N$, the set $\{V_i\}_{i=1}^k$ is called a Voronoi tessellation or diagram of Q if $V_i \cap V_j = \emptyset$ for $i \neq j$ and $\bigcup_{i=1}^k V_i = Q$. Given a set of points $\{p_i\}_{i=1}^k$ belonging to Q, the Voronoi region V_i corresponding to the point p_i is defined by*

$$\begin{aligned} V_i &= \{x \in Q \mid \|x - p_i\| < \|x - p_j\| \\ &\qquad \textit{for } i, j = 1, \ldots, k, j \neq i\}. \end{aligned}$$

Where $\|\cdot\|$ denote the Euclidean norm on $\mathbb{R}^N$. The points $\{p_i\}_{i=1}^k$ are called generator points*, and V_i is the* Voronoi region *associated to the generator point p_i.*

Although Voronoi diagrams can be defined using several distance functions such as the geodesic distance described in [37]. For this particular problem we are using Euclidean distance, and Q is considered a convex polytope in an N-dimensional Euclidean space.

4.2 Locational Optimization

Based on [16], let $Q \subset \mathbb{R}^N$ be a convex polytope including its interior. Assume a mapping $\phi(q) : Q \mapsto \mathbb{R}_+$ with $q \in Q$ called a *distribution density function* (or *sensory* *function*) which represents a measurement of the probability of a specific event on Q. The locational optimization function is then defined as

$$\mathcal{H}(P) = \sum_{i=1}^{n} \int_{V_i} f(\|q - p_i\|)\phi(q)dq, \tag{4.1}$$

where P is the set of all the n generator points $\{p_1, \dots, p_n\} \in Q$ and V_i is the Voronoi partition of the i-th robot.

Now, based on [7] we can adapt some physical concepts namely, the mass M_{V_i}, the first moment L_{V_i}, the polar moment of inertia $J_{V,p}$ and the centroid C_{V_i} of a Voronoi region V_i. Their definitions are given by the following equations,

$$\begin{aligned} M_{V_i} &= \int_{V_i} \phi(q)dq, \\ L_{V_i} &= \int_{V_i} q\phi(q)dq, \\ J_{V,p} &= \int_{V_i} \|q - p_i\|^2 \phi(q)dq, \\ C_{V_i} &= \frac{1}{M_{V_i}} \int_{V_i} q\phi(q)dq. \end{aligned} \tag{4.2}$$

From [7], if we define $f(\|q-p_i\|) = \|q-p_i\|^2$ and replace it in (4.1), after applying a partial derivative with respect to p_i we have that

$$\begin{aligned} \frac{\partial \mathcal{H}_V(P)}{\partial p_i} &= \int_{V_i} \frac{\partial}{\partial p_i} f(\|q - p_i\|)\phi(q)dq \\ &= 2M_{V_i}(p_i - C_{V_i}). \end{aligned} \tag{4.3}$$

Therefore, all the Voronoi tessellation in Q where the generator points are at the same time the centroids of their Voronoi partitions minimize the locational optimization function. These tessellations are usually called *centroidal Voronoi tessellations* [36].

4.3 Adaptive Control for Holonomic Sensors

In [5] the authors propose an approach which guarantees that the network of mobile agents minimizes the cost function $\mathcal{H}_V(P)$ in (4.1). They assume that each agent measures the sensory function without requiring a previous knowledge.

In order to deal with the lack of knowledge of the sampling space they proposed a decentralized adaptive control based on the following assumptions,

Assumption 1 (Matching Conditions)**.** *There exists a parameter vector $a \in \mathbb{R}_+^m$ and a vector function $\mathcal{K} : Q \mapsto \mathbb{R}_+^m$ such that*

$$\phi(q) = \mathcal{K}(q)^T a, \tag{4.4}$$

where $m \in \mathbb{N}$, and $(\cdot)^T$ denotes transpose.

The parameter vector a is unknown by the agents but $\mathcal{K}(q)$ is available to them.

Assumption 2 (Lower Bound)**.** *Given that $a(j)$ is the j-th element of the vector a and $\beta \in \mathbb{R}_+$ then*

$$a(j) \geq \beta \quad \forall j = 1, \ldots, m,$$

The reason for a lower bound for the parameter vector is to avoid that $\mathcal{K}(q)^T a = \phi(q) = 0$ leading to a zero in the denominator of (4.2).

The sensory function estimated by the i-th agent is given by $\hat{\phi}_i = \mathcal{K}(q)^T \hat{a}_i$, where $\hat{a}_i$ is the estimation of the parameter vector a calculated by the agent i. Furthermore the parameter error vector $\tilde{a}_i$ is given by

$$\tilde{a}_i = \hat{a}_i - a_i. \tag{4.5}$$

In [7] the mobile agents are considered holonomic vehicles with first-order continuous dynamics, that is

$$\dot{p}_i = u_i, \tag{4.6}$$

which is called a single integrator.

The control law is defined as

$$u_i = k(\hat{C}_{V_i} - p_i), \tag{4.7}$$

where $\hat{C}_{V_i}$ is an estimate of the real centroid C_{V_i} of the i-th Voronoi region defined by

$$\hat{C}_{V_i} = \frac{\hat{L}_{V_i}}{\hat{M}_{V_i}} = \frac{\int_{V_i} q\hat{\phi}(q)dq}{\int_{V_i} \hat{\phi}(q)dq}.$$

Finally, the adaptation law is given by

$$\dot{\hat{a}}_i = \Gamma(\dot{\hat{a}}_{pre_i} - I_{proj_i}\dot{\hat{a}}_{pre_i}), \tag{4.8}$$

with

$$\dot{\hat{a}}_{pre} = -F_i\hat{a}_i - \xi(\Lambda_i\hat{a}_i - \lambda_i) - \zeta\sum_{j\in\mathcal{N}_i}(\hat{a}_i - \hat{a}_j), \tag{4.9}$$

where $\xi, \zeta \in \mathbb{R}_+$ are scalar gains, $\Gamma \in \mathbb{R}^{m\times m}$ is a diagonal positive definite gain matrix. The variables F_i, Λ_i, and λ_i are given by the following equations,

$$F_i = \left[\int_{V_i} \mathcal{K}(q)(q - \hat{C}_{V_i})^T dq\right] \dot{p}_i \tag{4.10}$$

$$\Lambda_i = \int_0^t w(\tau)\mathcal{K}_i(\tau)\mathcal{K}_i(\tau)^T d\tau, \tag{4.11}$$

$$\lambda_i = \int_0^t w(\tau)\mathcal{K}_i(\tau)\phi_i(\tau)d\tau. \tag{4.12}$$

Given a set of indexed vertices $V_e = \{v_1 \ldots, v_n\}$ and a set of edges $E = \{e_1 \ldots e_l\}$, where $e_i = \{v_j, v_k\}$ then $\mathcal{N}_i = \{j|\{v_i, vj\} \in E\}$ *i.e.*, $\mathcal{N}_i$ contains the indexes of the vertices which are neighbors of the vertices associated to the Voronoi partition of the generator point i.

The function $w(t) \in \mathcal{L}^1$ is called a weighting function. Since we are dealing with dynamic density functions, we use a forgetting factor $\delta \in \mathbb{R}$ which encourages the parameter convergence. The weighting function $w(t,\tau) = e^{-\delta(t-\tau)}$ gives more weight to the latest measurements than to the older ones which is suitable to our particular case. Based on [5] there can be other types of weighting functions as we discuss later in Section 4.3.1.

The matrix $I_{proj_i}(j)$ is defined as follows

$$I_{proj_i}(j) = \begin{cases} 0 & \text{for } \hat{a}_i(j) > \beta, \\ 0 & \text{for } \hat{a}_i(j) = \beta \text{ and } \dot{\hat{a}}_{pre_i} \geq 0, \\ 1 & \text{otherwise.} \end{cases} \tag{4.13}$$

The index j denotes the j-th diagonal element of the matrix I_{proj_i} and the j-th element of the vector $\hat{a}_i$. This matrix implements a projection law which prevents the parameter vector $\hat{a}_i$ from taking values less than or equal to the lower bound β.

Lastly, in [5] the authors state and prove the following convergence theorem

Theorem 3 (Convergence Theorem)**.** *Under Assumption 1, for the system of n agents with the dynamics given by (4.6) and the control law in (4.7),*

$$\begin{aligned} \lim_{t\to\infty} \|\hat{C}_{V_i} - p_i\| &= 0, \ \forall i \in I_n, \\ \lim_{t\to\infty} \mathcal{K}(p_i(\tau))^T \tilde{a}_i &= 0, \ \forall \tau \mid w(\tau) > 0 \textit{ and } \forall i \in I_n, \\ \lim_{t\to\infty} \|\hat{a}_i - \hat{a}_j\| &= 0, \ \forall i,j \in I_n, \end{aligned}$$

with $I_n = \{1, \dots, n\}$.

4.3.1 Weighting Functions

There are several options to build the weighting function $w(\cdot)$ in (4.11) and (4.12) as long as it encourages the parameter convergence of the adaptation law. Based on [5], if we choose $w(\tau)$ as a square wave, the integral given in (4.11) does not incorporate any other term in the summation after some fixed time determined by the decay time of the square wave. We can soften the elimination of old terms in the integral using an exponential decay $w(\tau) = e^{-\tau}$ or a decaying sigmoid $w(\tau) = 1/2(\text{erf}(c-\tau)+1)$. If we specifically use the function $w(t,\tau) = e^{-\alpha(t-\tau)}$ the integrals (4.11) and (4.12) become first-order systems, introducing a forgetting factor α which allows the tracking of slow varying density functions.

Chapter 5

Adaptive Control for Nonholonomic Sensors

The stability analyzes of the controllers in [7] and [5] have been conducted assuming holonomic kinematics, but now we propose to formally extend the previous results to nonholonomic vehicles.

5.1 Nonlinear Steering Control

Since several real world vehicles such as aircraft at cruising attitude, sea vessels and skid-steered mobile robots can be modeled as nonholonomic vehicles, we propose to use the unicycle model kinematics equations for a differential steering as a suitable approach. The equations of motion for the i-th agent in the team of robots are given as follows

$$\begin{pmatrix} \dot{x}_i \\ \dot{y}_i \\ \dot{\phi}_i \end{pmatrix} = \begin{pmatrix} u_i \cos\phi_i \\ u_i \sin\phi_i \\ \omega_i \end{pmatrix}, \tag{5.1}$$

where u_i and ω_i are the linear and angular speeds of the i-th robot respectively.

We need to choose an appropriate steering control for nonholonomic vehicles to drive every robot to the centroid of its Voronoi region. For that purpose we use the following kinematics equations given in [38]

$$\begin{pmatrix} \dot{\rho}_i \\ \dot{\alpha}_i \\ \dot{\theta}_i \end{pmatrix} = \begin{pmatrix} -u_i \cos\alpha_i \\ -\omega_i + u_i \frac{\sin\alpha_i}{\rho_i} \\ u_i \frac{\sin\alpha_i}{\rho_i} \end{pmatrix}, \tag{5.2}$$

where

$$\begin{aligned} \alpha_i &= \theta_i - \phi_i, \\ \dot{\phi}_i &= \omega_i. \end{aligned} \tag{5.3}$$

As shown in Fig. 5.1 the position of the agent inside its Voronoi cell is represented in polar coordinates where ϕ_i is the heading angle of the vehicle, ρ_i represents the position error between the agent and the centroid point and α_i is the angle between the principal axis of the robot and the vector error ρ_i.

As in [38] we use the following control law,

$$\begin{pmatrix} u_i \\ \omega_i \end{pmatrix} = \begin{pmatrix} (\gamma \cos\alpha_i)\rho_i \\ k\alpha_i + \gamma \frac{\cos\alpha_i \sin\alpha_i}{\alpha_i}(\alpha_i + h\theta_i) \end{pmatrix}, \tag{5.4}$$

where k, γ and h are positive gains.

The control law in (5.4) allows the agent to reach asymptotically the point $(0,0,0)$. Therefore if we carry out an axis translation to set the centroid at the origin of the plane we can use this control law to drive the robots to their centroidal Voronoi tessellation. We do not include the proof of the stability of the steering control in this section; however, it is developed as part of the stability analysis in Section 5.2. In Fig. 5.2 we show eight robots with different starting positions and initial heading angles $\phi_i = \pi/2$ going to the point $(0,0,0)$.

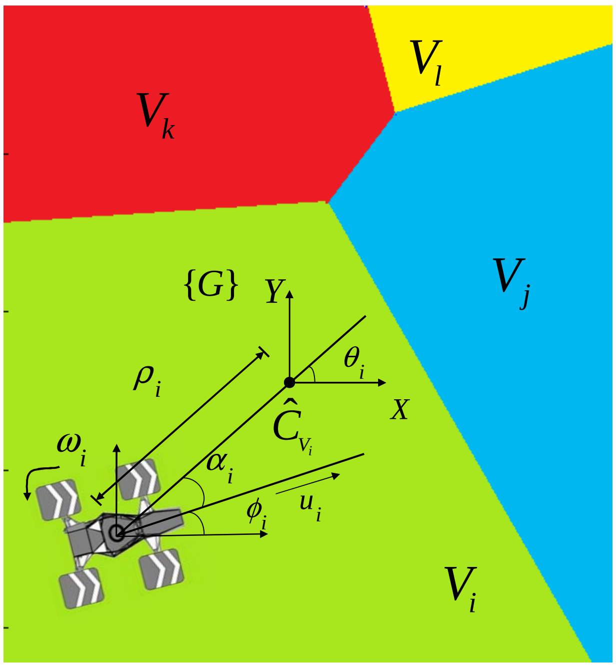


Figure 5.1: Unicycle model and variables in the goal frame $\{G\}$: Notice the vectors and angles which determine our nonholonomic model in polar coordinates.

5.2 Stability Analysis

The following is our extended convergence theorem for the distributed and adaptive control for nonholonomic vehicles.

Theorem 4 (Extended Convergence Theorem). *If Assumptions 1 and 2, are satisfied we have that for the system of n nonholonomic agents with dynamics (5.2) and control law (5.4),*

$$\begin{aligned}
\lim_{t\to\infty} \mathcal{K}(p_i(\tau))^T \tilde{a}_i &= 0, \quad \forall \tau \mid w(\tau) > 0 \textit{ and } \forall i \in I_n, \\
\lim_{t\to\infty} \rho_i, \|\alpha_i\|, \|\theta_i\| &= 0, \quad \forall i \in I_n, \\
\lim_{t\to\infty} \|\hat{a}_i - \hat{a}_j\| &= 0, \quad \forall i, j \in I_n,
\end{aligned}$$

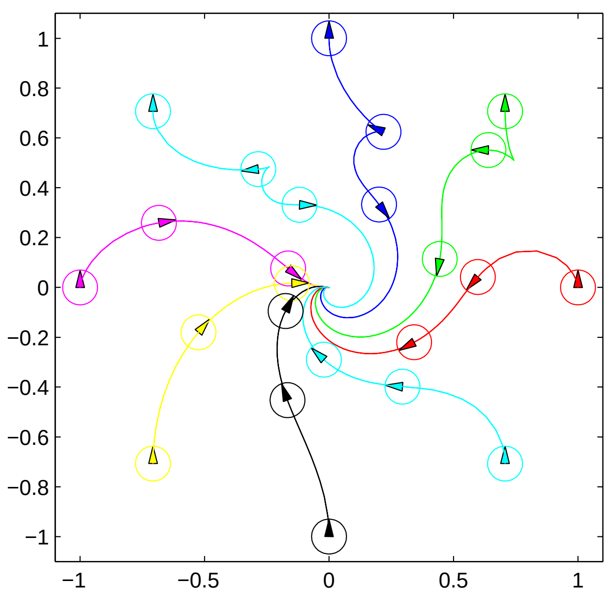


Figure 5.2: Illustration of the nonlinear steering control over eight robots with different initial positions. All the robots go to the position $(0, 0, 0)$.

with $I_n = \{1, \dots, n\}$.

Proof. To carry out the stability analysis we propose the following Lyapunov function candidate

$$V = \mathcal{H} + \sum_{i=1}^{n} \left[\frac{1}{2} \tilde{a}_i^T \Gamma^{-1} \tilde{a}_i + \frac{1}{2} \left(\alpha_i^2 + h\theta_i^2 \right) \right]. \tag{5.5}$$

Notice that compared with the Lyapunov function proposed in [5], the expression in (5.5) has the extra term $\frac{1}{2}\left(\alpha_i^2 + h\theta_i^2\right)$ related to the robot orientation (see Fig. 5.1). The matrix Γ is the same diagonal positive definite matrix in (4.8), $\mathcal{H}$ is described by (4.1), and α_i, θ_i and ρ_i are the state variables in the dynamics in (5.2). Lastly, $\tilde{a}_i$ is the parameter estimation error given by (4.5).

Taking the time derivative of (5.5), we obtain

$$\dot{V} = \sum_{i=1}^{n} \left[\frac{\partial \mathcal{H}}{\partial p_i} \dot{p}_i + \tilde{a}_i^T \Gamma^{-1} \dot{\hat{a}}_i + (\alpha \dot{\alpha}_i + h\theta_i \dot{\theta}_i) \right]. \tag{5.6}$$

Now, replacing (4.3) in (5.6) we get

$$\dot{V} = \sum_{i=1}^{n} \left[M_{V_i}(p_i - C_{V_i})^T \dot{p}_i + \tilde{a}_i^T \Gamma^{-1} \dot{\hat{a}}_i + (\alpha \dot{\alpha}_i + h\theta_i \dot{\theta}_i) \right]. \tag{5.7}$$

Furthermore, it is easy to show that

$$L_{V_i} = M_{V_i} \hat{C}_{V_i} + \tilde{M}_{V_i}(\hat{C}_{V_i} - \tilde{C}_{V_i}) = M_{V_i} C_{V_i}, \tag{5.8}$$

then replacing (5.8) in (5.7), we have

$$\begin{aligned} \dot{V} &= \sum_{i=1}^{n} \Big[-M_{v_i}(\hat{C}_{V_i} - p_i)^T \dot{p}_i + (\tilde{M}_{V_i}\tilde{C}_{V_i} - \tilde{M}_{V_i}\hat{C}_{V_i})\dot{p}_i \\ &\quad + \tilde{a}_i^T \Gamma^{-1} \dot{\hat{a}}_i + (\alpha \dot{\alpha}_i + h\theta_i \dot{\theta}_i) \Big]. \end{aligned} \tag{5.9}$$

Taking into account that

$$\tilde{M}_{V_i}\tilde{C}_{V_i} - \tilde{M}_{V_i}\hat{C}_{V_i} = \tilde{a}_i \int_{V_i} \mathcal{K}(q)^T (q - \hat{C}_{V_i}) dq,$$

and replacing the adaptation law given by (4.8)-(4.12) in (5.9) the final expression for the derivative of the Lyapunov function becomes

$$\begin{aligned} \dot{V} &= -\sum_{i=1}^{n} \Bigg[M_{V_i}(\hat{C}_{V_i} - p_i)^T \dot{p}_i + \xi \int_0^t w(\tau)(\mathcal{K}_i(\tau)^T \tilde{a}_i)^2 \, d\tau \\ &\quad + \tilde{a}_i^T \zeta \sum_{j \in N_i} (\hat{a}_i - \hat{a}_j) + \tilde{a}_i^T I_{proj} \dot{\hat{a}}_{pre_i} - (\alpha_i \dot{\alpha}_i + h\theta_i \dot{\theta}_i) \Bigg]. \end{aligned} \tag{5.10}$$

The second, third and fourth terms in the summation in (5.10) have already been proven to be positive semidefinite [5], considering the negative sign before the summation (see Appendix A). Now, we are interested in proving that the first and fifth terms are positive semidefinite as well.

Calculating $\hat{C}_{V_i} - p_i$ and based on Fig. 5.1, we can assert that

$$\hat{C}_{V_i} - p_i = \begin{pmatrix} x_2 - x_1 \\ y_2 - y_1 \end{pmatrix} = \begin{pmatrix} \rho_i \cos\theta_i \\ \rho_i \sin\theta_i \end{pmatrix} = \begin{pmatrix} \rho_i \cos(\phi_i + \alpha_i) \\ \rho_i \sin(\phi_i + \alpha_i) \end{pmatrix}.$$

Taking the first term $M_{V_i}(\hat{C}_{V_i} - p_i)^T \dot{p}_i$ of (5.10) and replacing $\hat{C}_{V_i} - p_i$ and $\dot{p}_i$ by using the unicycle model in (5.1) with the control law in (5.2) we have

$$\begin{aligned} M_{V_i}(\hat{C}_{V_i} - p_i)^T \dot{p}_i &= M_{V_i} \begin{pmatrix} \rho_i \cos(\phi_i + \alpha_i) \\ \rho_i \sin(\phi_i + \alpha_i) \end{pmatrix}^T \begin{pmatrix} (\gamma \cos\alpha_i)\rho_i \cos\phi_i \\ (\gamma \cos\alpha_i)\rho_i \sin\phi_i \end{pmatrix}, \\ &= M_{V_i} \begin{pmatrix} \rho_i \cos\phi_i \cos\alpha_i - \sin\phi_i \sin\alpha_i \\ \rho_i \sin\phi_i \cos\alpha_i + \cos\phi_i \sin\alpha_i \end{pmatrix}^T \begin{pmatrix} \gamma\rho_i \cos\alpha_i \cos\phi_i \\ \gamma\rho_i \cos\alpha_i \sin\phi_i \end{pmatrix}, \\ &= M_{V_i}\rho_i^2\gamma(\cos^2\phi_i \ \cos^2\alpha_i + \sin^2\phi_i \ \cos^2)\alpha_i, \\ &= M_{V_i}\rho_i^2\gamma\cos^2\alpha_i. \end{aligned} \tag{5.11}$$

Since the mass M_{V_i} of the i-th Voronoi region and the control gain γ are non-negative, the first term in the summation of (5.10) is non-negative.

$$M_{V_i}(\hat{C}_{V_i} - p_i)^T \dot{p}_i = M_{V_i}\rho_i^2\gamma\cos^2\alpha \geq 0.$$

Analyzing the fifth term in (5.10) namely, $-(\alpha_i\dot{\alpha}_i + h\theta_i\dot{\theta}_i)$ we have that replacing the polar kinematics in (5.2) it gives based on [38],

$$\begin{aligned} -(\dot{\alpha}_i + h\theta_i\dot{\theta}_i) &= -\left[\alpha_i\left(-w_i + \frac{u_i \sin\alpha_i}{\rho_i}\right) + h\theta_i\frac{u_i\sin\alpha_i}{\rho_i}\right], \\ &= -\left[\alpha_i\left(-w_i + \frac{(h\theta_i + \alpha_i)}{\alpha_i}\frac{u_i\sin\alpha_i}{\rho_i}\right)\right]. \end{aligned} \tag{5.12}$$

Replacing the control law (5.4) in (5.12) we get

$$-(\alpha_i\dot{\alpha}_i + h\theta_i\dot{\theta}_i) = k\alpha_i^2 \geq 0,$$

and the fifth term $-(\alpha_i\dot{\alpha}_i+h\theta_i\dot{\theta}_i)$ in (5.10) is non-negative. Since V is lower bounded, $\dot{V}$ is negative semidefinite and uniformly continuous in time we conclude that $\dot{V} \to 0$ as $t \to \infty$ by the Lyapunov-like lemma.

From the Lyapunov function derivative in (5.10) it is easy to see that all the limits converge to zero except the third one $\lim_{t\to\infty} \|\theta_i(t)\|$. Differentiating (5.3) and using the equations in (5.2) with the control law in (5.4) we have as in [38] that

$$\begin{aligned}
\dot{\alpha}_i &= \dot{\theta}_i - \dot{\phi}_i,\\
\dot{\alpha}_i &= \gamma \cos\alpha_i \sin\alpha_i - k\alpha_i - \gamma\frac{\cos\alpha_i \sin\alpha_i}{\alpha_i}(\alpha_i + h\theta_i),\\
\dot{\alpha}_i &= -\gamma h\theta_i\frac{\cos\alpha_i \sin\alpha_i}{\alpha_i}.
\end{aligned}$$

Since $\alpha_i \to 0$ and $\theta_i \to c$ as $t \to \infty$, then we have that

$$\lim_{t\to\infty} \dot{\alpha}_i = -\gamma hc.$$

Since $\dot{\alpha}_i$ is a uniformly continuous function, lower bounded and negative semidefinite; therefore, by the Lyapunov-like lemma $\dot{\alpha}_i \to 0$ as $t \to \infty$ and this implies that $\theta_i \to 0$ as well. Therefore the controller guarantees the convergence of the state variables ρ_i, α_i and θ_i to zero under the goal frame $\{G\}$ shown in Fig. 5.1. $\square$

5.3 Dynamic Density Function

We consider the case of estimating the parameters of a time-varying density function $\phi_i(q,t) = \mathcal{K}(q)^T a(t)$ where the j-th entry $a_j(t)$ $(j = 1, 2, \ldots, m)$ of $a(t)$ is a piecewise constant function $a_j(t) : \mathbb{R}_+^m \mapsto \mathbb{R}_+^m$ and is right continuous. It means that every entry of the function vector $a(t)$ has a finite number of discontinuities and takes on constant values between two consecutive discontinuities. This is a reasonable approximation if we consider slow-time varying systems.

Also, we assume that $\lim_{t\to\infty} a(t) = a_c$ where $a_c \in \mathbb{R}_+^m$ is a constant value *i.e.,* the density function reaches a steady state which is reasonable for many real-world phenomena such as oil spills [1] and forest fires [2].

From now on, we will call *switching time* t_s, the time when each discontinuity happens, where $s = 1, \cdots, k$, and k is the total number of switching times before the density function reaches its final value. This terminology was taken from [39] given the partial similarity with the switching systems.

Moreover let us assume that the adaptation law rate and the angular and linear speeds of the agents are fast enough to follow the dynamics of the density function $\phi(q, t)$.

From (4.8) we know that every robot looks for the centroid of its Voronoi cell while taking measurements of the distribution function on its trajectory. During this time, the tracking error decreases but notice from Theorem 4 that the network of robots converges to a near optimal coverage configuration. Based on Theorem 5.2 this behavior does not necessarily imply that the parameter estimation vector $\tilde{a}(t) \to 0$ as $t \to \infty$.

Furthermore, since we are dealing with a piecewise constant system, the time interval between two switching times $\Delta t_s = t_s - t_{s-1}$ is finite, in contrast with the infinite time necessary to guarantee full parameter convergence.

Chapter 6

Simulation Tests

In this chapter we show simulation results in order to verify our convergence theorem given by Theorem 4. The simulations were carried out using Matlab and Stage (see Section 8.2.1). The reason why we implement the simulations in those two different scenarios is that in Matlab we are able to use just the kinematic model of the unicycle vehicle neglecting the dynamics equations. Therefore, using Matlab we can get simulation results closer to our theoretical approach since we do not consider speed constraints and forces that the real vehicle has to deal with. Furthermore, using the Matlab ode45 solver we can implement the adaptation law given by (4.9) whose numerical calculation in real time makes it unfeasible for a regular computer, however it allows us to confirm that the adaptation law is estimating the right parameters and that the variables indicated in Theorem 4 are convergent.

On the other hand, Stage has the kinematic and dynamic model of the four wheel skid steer system of the P3-AT robot. Furthermore, Stage presents a relation $\frac{\text{Simulation time}}{\text{Real time}} \approx 0.9$ for our particular case, so the simulation is closer to the real application. This implies that the implementation of the adaptation law based on (4.9), which is a continuous differential equation, should be approximated by a difference equation in order to carry out the calculation in real time. Otherwise an ODE solver requires a high computational cost. In this case the simulations in Matlab and Stage are not redundant at all, and together allow us to make conclusions about how the theoretical and practical implementations differ even though, they are strongly related.

6.1 Simulation in Matlab

In this section we give a detailed description of the simulations carried out using Matlab.

6.1.1 Simulation Design in Matlab

For this simulation we used a population of 20 unicycle models randomly distributed over a sample space Q defined as a unit square. We implement the control law given in (5.4) with $\gamma = 3$ and $k = h = 1$.

The parameter values we used in the adaptation law given by (4.8) and (4.9) are $\Gamma = I_{64}$, $\xi = 1000$, $\zeta = 1$ and $\delta = 1$. For the matrix I_{proj_i} defined by (4.13), we have $\beta = 0.1$.

We divided the sampling space Q in a 8×8 grid where the geometric center of every square cell corresponds to the mean μ_i of a bidimensional Gaussian function. Using a function similar to the one in [5] we have that the i-th entry $\mathcal{K}_i$ of the vector function $\mathcal{K}(q)_{64\times 1}$ is calculated as,

$$\mathcal{K}_i = e^{\frac{-(q-\mu_i)^2}{2\sigma_i^2}}, \tag{6.1}$$

with $\sigma_i = 0.05$.

For this simulation we decided to use the unicycle robots to detect a density function which behaves as an expanding circle. The expanding circle recreates a simplified behavior of a forest fire where the higher temperatures are localized at the circumference of the circle.

The dynamics of the expanding circle are modeled by the following parametric equations,

$$\begin{pmatrix} x(t) \\ y(t) \end{pmatrix} = \frac{1}{3} r(t) \begin{pmatrix} \cos\phi \\ \sin\phi \end{pmatrix} + \begin{pmatrix} 0.5 \\ 0.5 \end{pmatrix},$$

with the radius $r(t)$ defined by the following differential equation,

$$\dot{r}(t) = -\frac{5.7}{100} r(t) + 1,$$

with $r(0) = 1 - e^{-0.3}$.

We sample the circumference of the circle using the 64 squared cells which divide the sample space Q as explained before. We assign to each of the k-th cells containing a segment of the circumference a bidimensional Gaussian function defined by (6.1) with $\sigma_k = 0.05$ and mean μ_k equal to the geometric center of the k-th cell. In this way we assign a height to the expanding circle which is maximum at the points located at the circumference of the circle.

Since our approach covers just piecewise constant dynamics we assume that the robots are taking measurements of the density function at the discrete-time instants 0, 20, 40 and 100 s. This means that assuming a slow varying distribution function the robots can reach their respective centroids and rest until some problem dependent condition is fulfilled to start taking measurements again.

6.1.2 Simulation Results in Matlab

In Fig. 6.1 (a)–(d) we show the averaged behavior of the parameter estimation error given by

$$\mathcal{K}(p_i(\tau))^T \tilde{a}_i(t) \; \forall \tau \; | w(\tau) > 0,$$

as well as the error distance $\rho_i(t)$, the angle $\alpha_i(t)$ and the consensus error given by

$$\|\hat{a}_i(t) - \hat{a}_j(t)\| \; \forall i, j \in I_n.$$

Notice that the switching times of the simulation are indicated by the dashed vertical lines in green.

Let us define

$$\bar{\tilde{a}}_i(t) = \frac{1}{n}\left(\sum_{i=1}^{n} \tilde{a}_i(t)\right) \quad \forall t > 0, \tag{6.2}$$

which is the averaged parameter error vector over all robots. In Fig. 6.1 (a) we show the parameter estimation error $\mathcal{K}(q)^T \bar{\tilde{a}}_i(t)$ averaged over the whole population of robots. In a similar way let us define

$$\bar{\rho}(t) = \frac{1}{n}\left(\sum_{i=1}^{n} \rho_i(t)\right) \quad \forall t > 0, \tag{6.3}$$

which is the averaged position error of all the robots in Q, which is plotted in Fig. 6.1 (b).

Notice that the angle $\alpha_i(t)$ plotted in Fig. 6.1 (c) looks noisy. In order to make it clear we just illustrate the angle corresponding to one robot selected randomly.

Finally, for the consensus error let us define the quantity c_a as

$$c_a = \sum_{i=1}^{n} \left\| \sum_{j=1}^{n} (\hat{a}_i - \hat{a}_j) \right\|^2, \tag{6.4}$$

which shows the summation of the squared norm of the vector $\sum_{j=1}^{n}(\hat{a}_i - \hat{a}_j)$ over the whole population of robots and is plotted in Fig. 6.1 (d).

In the plots in Fig. 6.1 (a), (b) and (d) it is easy to note the asymptotic convergence to zero after every switching time t_s, but in the case of α_i this is difficult to see because the approximation of the numerical integrals of the centroids in (4.2) induces some noise in the trajectory of the robots. Furthermore, notice that the transitions of α_i from $-\pi$ to π look like spikes in the plot, however, the robots spend the majority of the time oscillating around the angle $\alpha_i = 0$ as the filtered red signal illustrates in Fig. 6.1 (c).

We calculate the mean square error between the vector a and the averaged parameter estimation error $\bar{\hat{a}} = \frac{1}{20}\sum_{i=1}^{20}\hat{a}_i$ at every time instant t, and plot it using the blue line in Fig. 6.2. Notice that in this case we are just relying on the robustness of the adaptive control to follow changes in the parameters. Furthermore notice that the parameter estimation error in Fig. 6.1 (a) goes asymptotically to zero as the Theorem 4 states whereas the mean squared parameter error in Fig. 6.2 does not since from the Lyapunov function it cannot be guaranteed by Theorem 1. A proof of the additional richness condition to guarantee parameter convergence is given in [5].

Based on the second term on the right side of (4.9), the gradient estimator is excited by the difference between what the robot measures and what the robot estimates. In order to increase that difference to excite the adaptation law, we can change the parameter estimation vector $\hat{a}_i$ at every switching time t_s to guarantee the maximum excitation of the control law in (5.4). This is an interesting problem to be formalized but its solution is part of our future research.

Since we are assuming that the robots do not have information of the density function $\phi_i(q, t)$ we set all the entries of the estimate parameter vector $\hat{a}_i$ to $\beta =$

0.1 (any value $\beta > 0$ from Assumption 2) for every robot at every switching time t_s. Then, the estimated density function is approximately uniform, exciting the adaptation law (if there is not an uniform density function in Q) and improving the parameter estimation error as shown by the red line in Fig. 6.2.

In Fig. 6.3 we present some snapshots of the robot distribution over the piecewise dynamic density function at different times just before the switching time t_s happens. Notice the centroidal Voronoi tessellations shown in Fig. 6.3 (b) – (d).

6.2 Simulation in Stage

Now, we proceed with a detailed description of the simulation carried out using Stage (see Section 8.2.1).

6.2.1 Simulation Design in Stage

The simulation was carried out using Stage 3.0.0 setting a population of four P3-AT robots in the initial configuration shown in Fig. 6.5 (a). The white light distribution function is modeled by (4.4) where the i-th entry $\mathcal{K}_i$ of the vector function $\mathcal{K}(q)$ is given by (6.1) with $\sigma_i = 0.7$ m and μ_i being the geometric centroid of the i-th squared cell of Q.

The adaptation law is a differential equation, although this continuous differential equations provide solid theoretical results, they are difficult to implement in real-world situations, where the arithmetic precision and computational power are limited.

The following approximations of (4.9), (4.11) and (4.12) based on [40]

$$\begin{aligned}
\lambda_i(t+1) &= \lambda_i(t) + \mathcal{K}(p_i(t))\phi_i(t), &(6.5)\\
\Lambda_i(t+1) &= \Lambda_i(t) + \mathcal{K}(p_i(t))\mathcal{K}(p_i(t))^T, &(6.6)\\
\hat{a}_{i_{pre}} &= \hat{a}_i + \xi(\Lambda_i \hat{a}_i - \lambda_i) - \zeta \sum_{j \in \mathcal{N}_i} (\hat{a}_i - \hat{a}_j), &(6.7)\\
\hat{a}_i &= \max(\hat{a}_{i_{pre}}, \beta). &(6.8)
\end{aligned}$$

were used in order to carry out the adaptation law calculation in real time.

The parameter values we used in the approximation of the adaptation law given by (6.5) – (6.8) are $\xi = 1000$, $\zeta = 1$ and $\delta = 1$. The matrix I_{proj_i} defined by (4.13) now is replaced by the max operation in (6.8) $\beta = 0.1$.

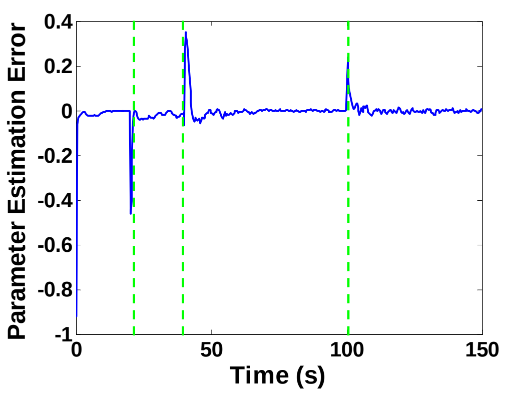
The piece-wise constant property of the density function is simulated by manipulating the parameter vector a such that we have a Gaussian function in the yellow area A indicated in Fig. 6.5 (a). After 150 s of simulation time we change the distribution function such that we have a Gaussian function in the green area B.

6.2.2 Simulation Results in Stage

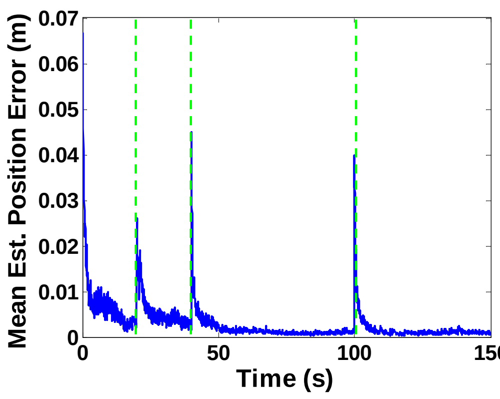
In Fig. 6.4 (a)–(c) we show the behavior of the error distance $\bar{\rho}(t)$ defined in (6.3), the angle α_i, and the consensus error c_a given by (6.4). Notice that all these variables present a different behavior compared with the Matlab simulation due to the consideration of the dynamical model of the P3-AT robots. Another factor affecting the behavior of the plots is the approximation by difference equations of the adaptation law given by (6.5)–(6.8). However, the filtered versions indicated by the red lines in the plots in Fig. 6.4 (a)–(c) show that the average behavior of $\bar{\rho}(t)$, α_i and c_a remains close to zero after the switching time t_s happens.

Notice how in Fig. 6.4 (a) the mean estimated position error goes to zero and after the switching time $t_s = 140$ s the error increases because of the change of the light distribution and start going to zero again according to Theorem 4. However this approximation to zero does not seem to be asymptotic as in the Matlab simulation case since as we mentioned before there is a mechanical model with additional forces perturbing the behavior of the variables. The angle α_i in Fig. 6.4 (b) is oscillating around zero as the filtered signal in red indicates. Lastly, the consensus error c_a in Fig. 6.4 (c) goes quickly to values close to zero, at the beginning of the simulation and during the switching time indicated by the green dashed vertical line.

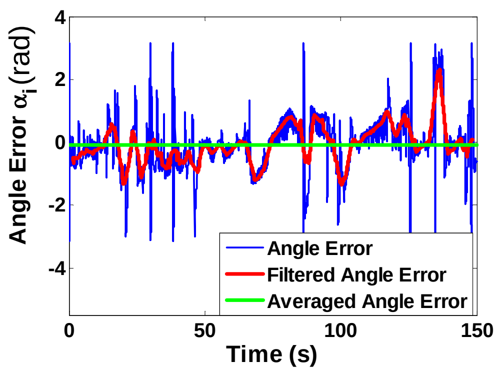
In Fig. 6.5 (b) and (c) we illustrate some snapshots of the Stage simulation. The robots reach their estimated centroidal Voronoi tessellation based on the first light distribution in the position A. Afterwards, they distribute themselves in a second centroidal Voronoi tessellation after the light has been displaced to the position B at $t = 108$ s. in Fig. 6.6 (a) and (b) we can see the average estimated light distribution obtained by the whole population of robots. Notice that given the approximation of the adaptation law with the difference equations (6.5)-(6.8) the values of the distribution function shown in Fig. 6.6 are not the true values and we do not include units in the z axis. However, the peaks in the shown surfaces are well located in correspondence to the simulated light distribution function.



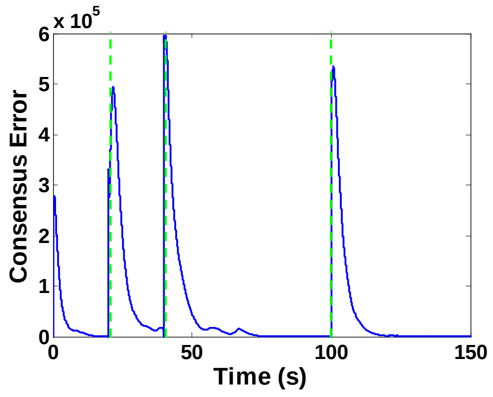
(a)



(b)



(c)



(d)

Figure 6.1: Plots of the parameter estimation error $\mathcal{K}(q)^T\tilde{\bar{a}}_i(t)$, the error distance $\bar{\rho}(t)$, the angle α_i and the consensus error c_a for the simulation in Matlab.

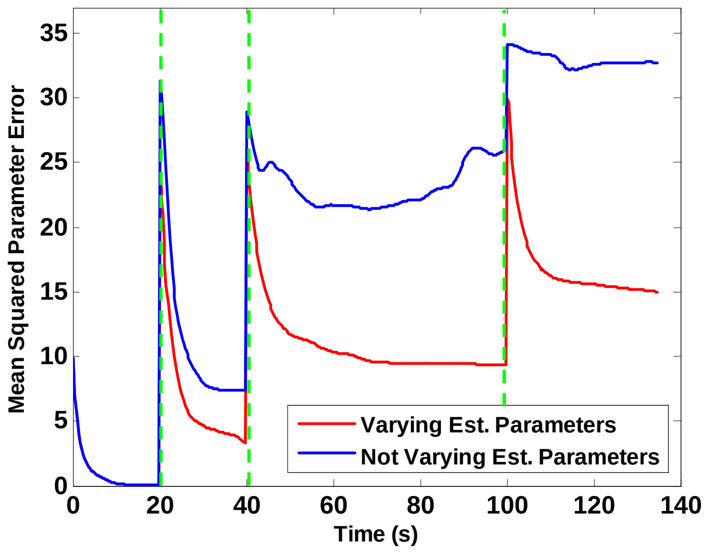
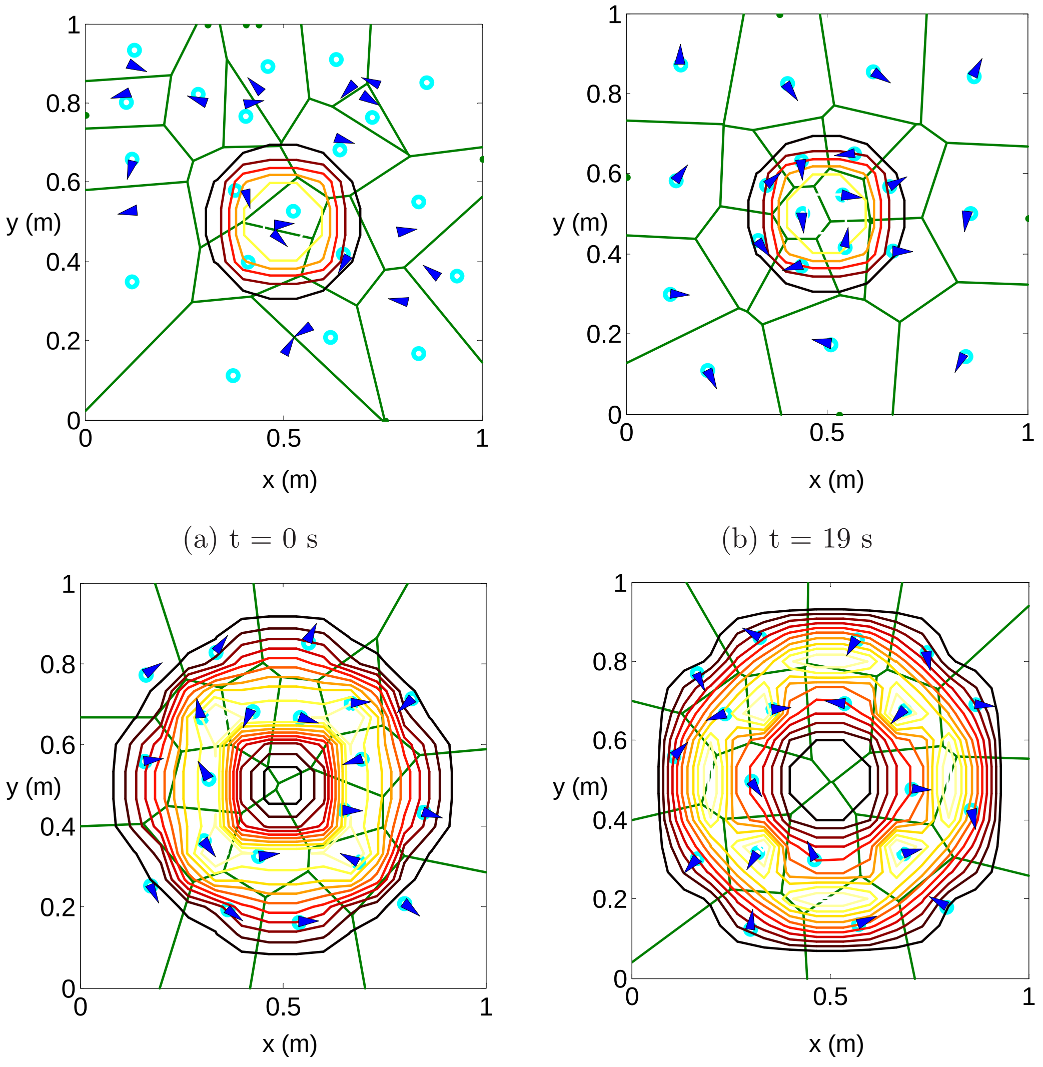


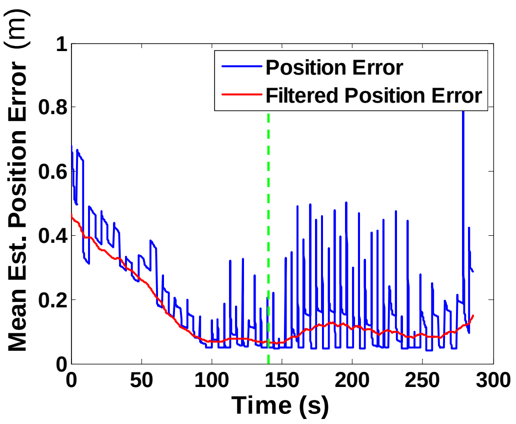
Figure 6.2: Mean squared error of the parameter estimation vector $\hat{a}_i$ averaged over the whole population of robots and the real parameter vector a for the simulation in Matlab. The blue line shows the response of the system without changing the parameter estimation vector $\hat{a}_i$ and the red line illustrates the obtained improvement by applying the change at every switching time t_s.



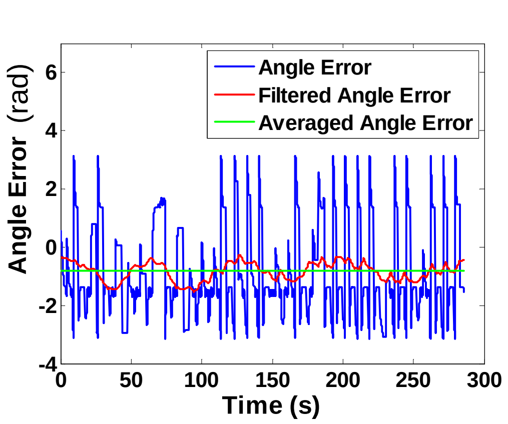
(a) t = 0 s (b) t = 19 s

(c) t = 39 s (d) t = 99 s

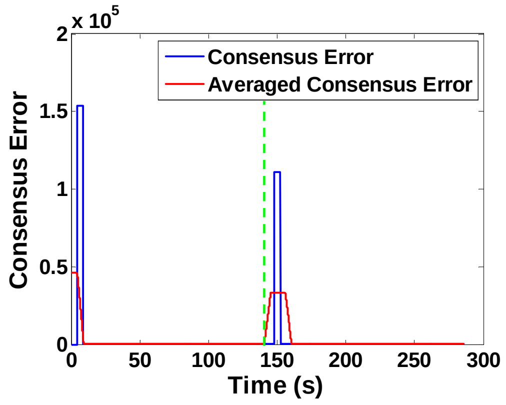
Figure 6.3: Simulation in Matlab with a population of 20 nonholonomic robots represented by the blue triangles over a dynamic density function indicated by the multicolor ring-shape contour. The yellow color of the contour means maximum density. This results correspond to the simulation changing $\hat{a}_i = \beta$ at every switching time t_s. The Cyan circles represent the estimated centroids $\hat{C}_{V_i}$ of the i-th robot and the green lines surrounding the robots form their Voronoi cells. Notice the centroidal Voronoi tessellations in (b)–(d).



(a) Estimated position error

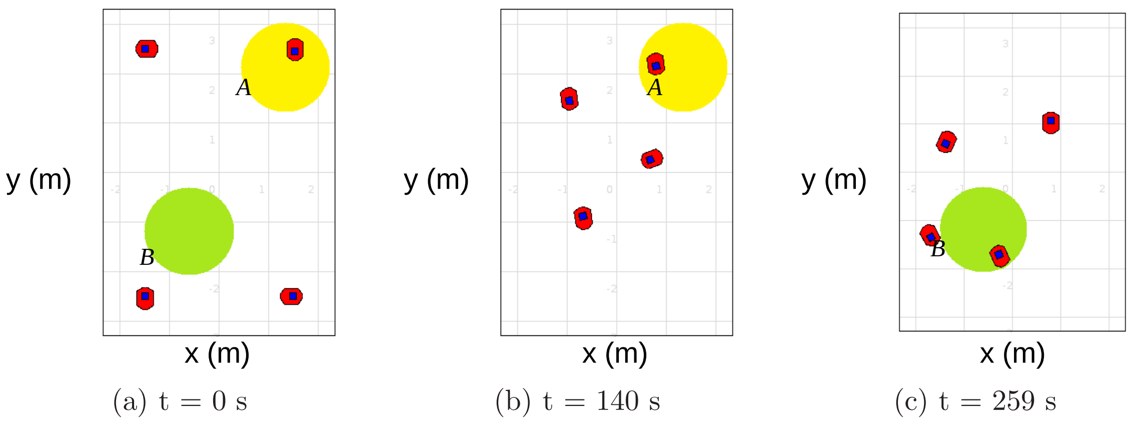


(b) Angle error α_i



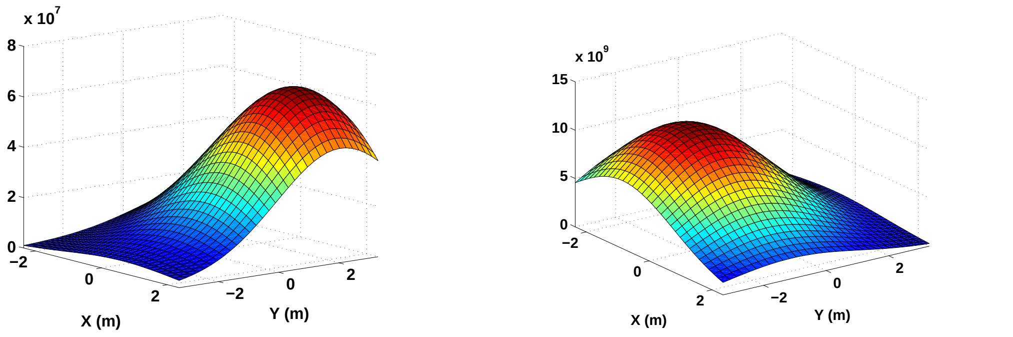
(c) Consensus error

Figure 6.4: Plots of the error distance ρ_i averaged over the population of robots, and the angle α_i during the simulation in Stage.



(a) t = 0 s (b) t = 140 s (c) t = 259 s

Figure 6.5: The Stage environment with four P3-AT robots used during the simulation. The light distribution goes from A to B after 150 s.



(a) Estimated density function t = 140 s (b) Estimated density function t = 250 s

Figure 6.6: Plots of the estimate density functions obtained by the team of robots in the simulation environment in Stage.

Chapter 7

Experimental Verification

This section describes the details of the experiments carried out using the multivehicle testbed described in Chapter 8.

7.1 Experimental Design

The experiments were carried out using a population of four P3-AT robots like the one shown in Fig. 8.3 (see Section 8.1 for a technical description), sensing a white light concentration in a rectangular sampling space of 4.7×6.6 m. Similar to the simulations, the sampling space was divided into a 8×8 grid. The geometric center of each rectangular division corresponds to the mean of a bidimensional Gaussian function given by (6.1) with $\sigma_i = 0.7$ m. As in the Stage simulation, the parameter values we used in the approximation of the adaptation law given by (6.5) – (6.8) are $\xi = 1000$ and $\zeta = 1$. The matrix I_{proj_i} defined by (4.13) now is replaced by the max operation in (6.8) with $\beta = 0.1$.

The light concentration is dynamic under the assumptions presented in Section 5.3. There is one switching time t_s to switch between two different light sources at

108 s of the experiment. The gyroscope and the wheel encoders embedded in the robots are used for robot positioning. A set of four Phidgets light precision sensors (see Section 8.1.1) are set up at the top of the robots and the network communication with the robots is carried out using Player 3.0.0 (see Section 8.2.1) through a Linksys wireless router.

The Voronoi partitions vertices are calculated using the library Voro++ version 0.3 [41] and the centroid integrals were approximated by Riemann summations discretizing the inside of the polygons in an 8×8 grid and adding the volumes of the hexahedra corresponding to every division. The adaptation law was approximated by the difference equations given in (6.5) – (6.8).

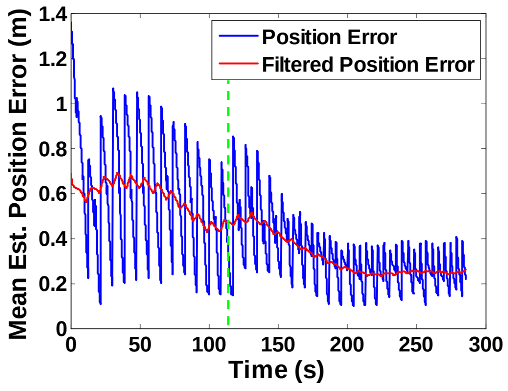
7.2 Experimental Results

In Fig. 7.1 (a)–(c) we show the behavior of the error distance $\bar{\rho}(t)$ defined in (6.3), the angle α_i, and the consensus error c_a given by (6.4). Since we do not know the real parameters to model the light distribution we cannot calculate $\bar{\tilde{a}}_i(t)$.

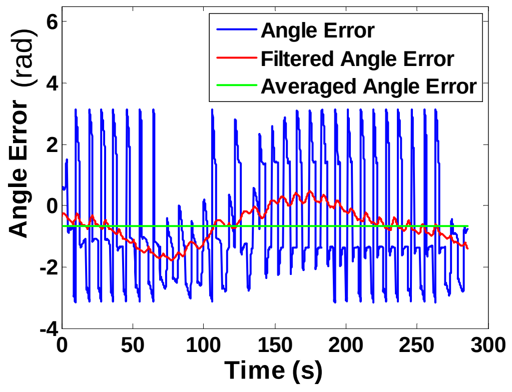
Notice the convergence of the signal in Fig. 7.1 (a) and (c) which are visibly affected by the noise of the real measurements and the numeric approximation of the centroid integrals. In order to make them clear we plot the filtered signal in red to show convergence. Furthermore, the effect of the adaptive law approximation given by (6.8) is evident in Fig. 7.1 (c), where the constant intervals correspond to the moment when the robots are going to their Voronoi cell centroid. The discontinuous edges of the signal represent the moment when the robots take a light measurement and calculate the centroids again.

Also notice that compared with the simulation results in Fig. 6.1 (c), the behavior of the angle α_i does not look as clear in Fig 7.1 (b) because of the presence of the odometry errors and the mentioned noise. However, it is possible to see that again the majority of the time the robots are oscillating around $\alpha_i = 0$.

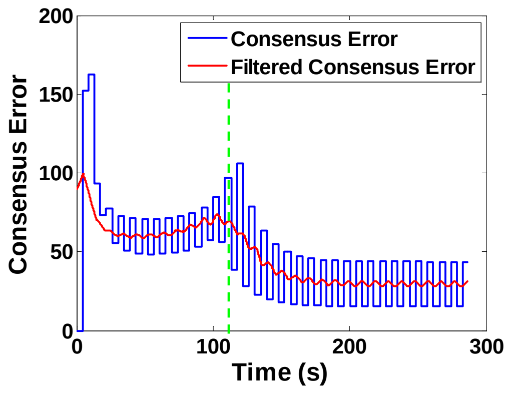
In Fig. 7.2 (a) and (d) we illustrate some snapshots of the experiment. In Fig. 7.2 (b) and (e) we can see the averaged light distribution estimated by the robots. Furthermore, in Fig. 7.2 (c) the top view based on the real data of the experiments show how the robots reach their estimated centroidal Voronoi tessellation related to the first light distribution. Afterwards they calculate a second centroidal Voronoi tessellation shown in 7.2 (f) once the light has been abruptly displaced in the sampling space at $t = 108$ s. Again, as in the Stage simulation the light distribution does not have units in the z axis because the approximation of the adaptation law by difference equations does not give the true distribution function values. However, the peaks of the surface are well located according to the real light distribution.



(a)



(b)



(c)

Figure 7.1: Plots of the error distance $\bar{\rho}(t)$, the angle α_i and the consensus error c_a for the experiments.

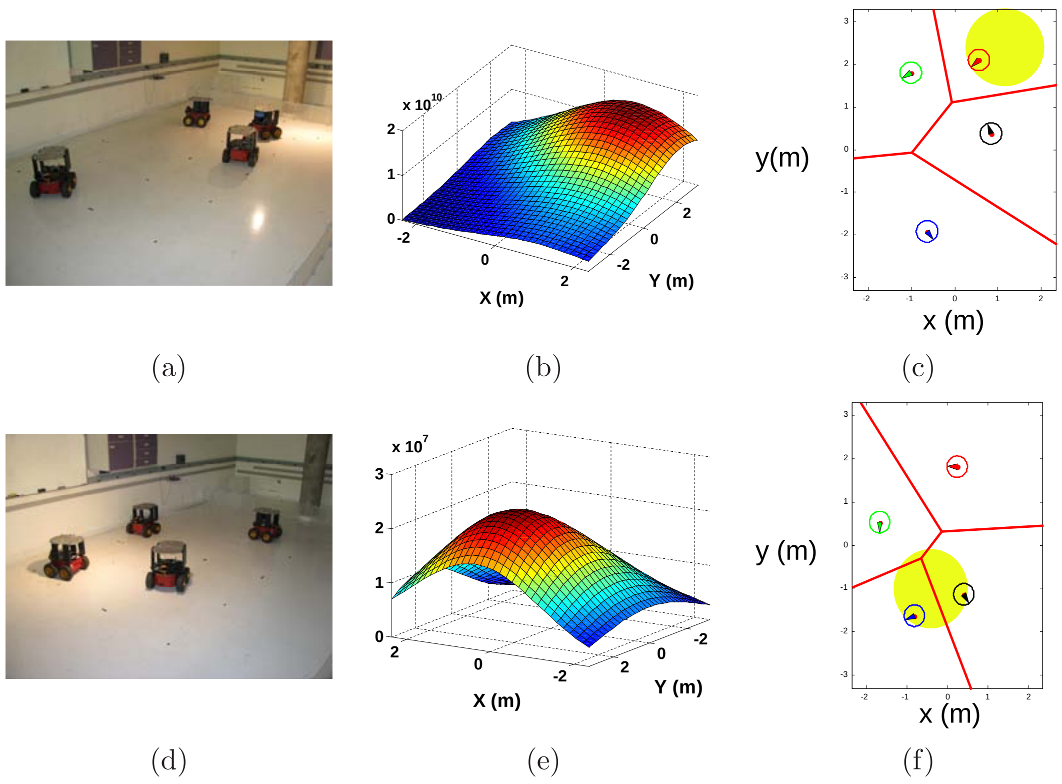


Figure 7.2: The picture in (a) shows a snapshot of an experiment with four P3 - AT robots sensing a light distribution at 107 s. The figure in (b) shows the estimated density function averaged over the four robots and the plot in (c) illustrates a top view done in Matlab with the robot data. The picture in (d) shows the snapshot of the experiment at 259 s. The figure in (e) shows the estimated density function averaged over the four robots, and the plot in (f) illustrates a top view done in Matlab based on the real data

Chapter 8

Multivehicle Testbed

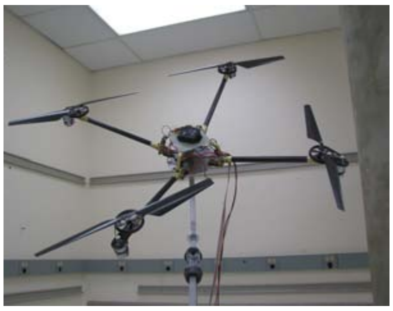
Our new multivehicle testbed in the Marhes lab expands on our original testbed COMET [42] to include mechanisms that allow for environmental sensing. In this way we enable the validation and verification of cooperative control algorithms that depend on measurements of the environment.

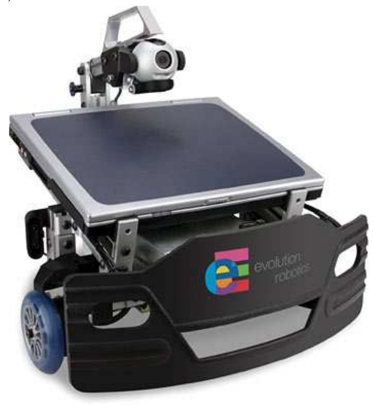
The testbed can accommodate laser-based navigation and Global Positioning System (GPS) navigation, as well as gripper manipulation tasks. Although this work concentrates on experiments involving the P3-AT robots, it is worthy to mention that our testbed contains ten all terrain vehicles which are based on the Tamiya TXT-1 chassis, a Dragonflyer X-Pro quadrotor, and two AscTec Hummingbird quadrotors. Furthermore, we have two Scorpion robots from Evolution Robotics. All the mentioned platforms are shown in Fig. 8.1.

The experimental testbed has its own dedicated IEEE 802.11 WLAN, which provides a low-latency network that is used by the robotic platforms for communication through third party server/client applications such as Player/Stage [43] from the University of Southern California or the Advanced Robotics Interface for Applications (ARIA) [44] from MobileRobots.

(a) A team of 10 TXT-1 robots

(b) Two hummingbird quadrotor


(a) One Dragonflyer quadrotor



(b) Two Scorpion robots

Figure 8.1: The pictures illustrate the variety of robotic platforms available in the MARHES lab at the University of New Mexico.

8.1 The Pioneer 3 - AT Robots

The Pioneer 3 - AT robots (P3-AT) [45] are programmable intelligent platforms equipped with the basic devices for navigation and sensing in the real world. They are part of a large family of robots released in 1995 with the Pioneer 1 which continued with the Pioneer AT, Pioneer 2-DX, the first of the family Pioneer 2, until the most recent family the Pioneer 3 with the Pioneer 3-DX and P3-AT model.

The basic P3-AT robot is provided with high resolution motion encoders, re-

Table 8.1: General specifications of the P3-AT robots.

Length	50.1 cm
Width	49.3 cm
Height	27.7 cm
Weight	14 kg
Payload	40 kg
Translate speed max	0.7 m/s
Rotate Speed max	140°/s

versible DC motors and the motor controllers, as well as the four-wheel skid steer which carries out the balanced drive system of the robot. The power source consists of up to three 12 VDC lead-acid batteries and the control of all sensors and devices in the robot is carried out using a server/client application. The robots are equipped with a radio ethernet board which allows the wireless communication with the sensors and devices attached to the robots.

The P3-AT robot reaches speeds up to 0.7 m/s. Furthermore, it can carry a payload of up to 40 kg and can climb a traversable slope of up to 40%. In Table 8.1 we show some mechanical features of the robot [43].

The main accessories for the P3-AT available in the Marhes lab are shown in Fig. 8.2. As you can notice, some sensors such as the SICK laser and the GPS are aimed to carry out outdoors experiments whereas some others such as the hokuyo laser are oriented to indoors experiments.

A special aluminum plate and mounting system was created to interface the environmental sensor suite. The plate and mounting system allows for multiple sensor configurations as well as the ability to mount multiple accessories on each robotic platform. Fig. 8.3 shows the custom built aluminum plate with four precision light sensors and three magnetic sensors. Also shown is a Hokuyo UHG-08LX laser range finder. The addition of the custom plate and mounting brackets allows for a quick

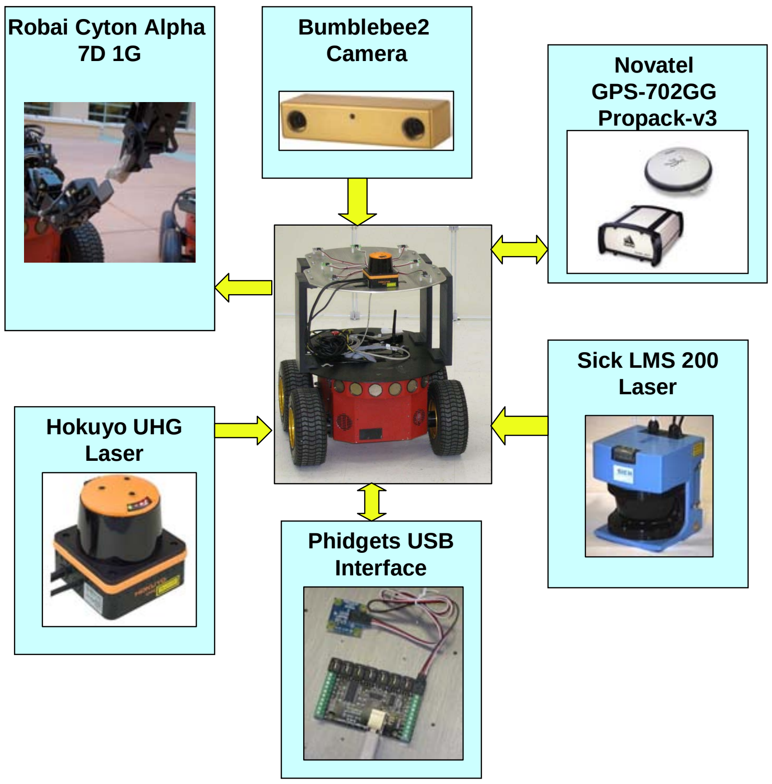


Figure 8.2: Accesories available for the P3-AT robots at the Marhes lab at the University of New Mexico.

swapping of sensors and accessories to address a variety of experimental tests.

8.1.1 Sensor Suite

This section describes the different sensors available for use on the experimental testbed at the Marhes lab.

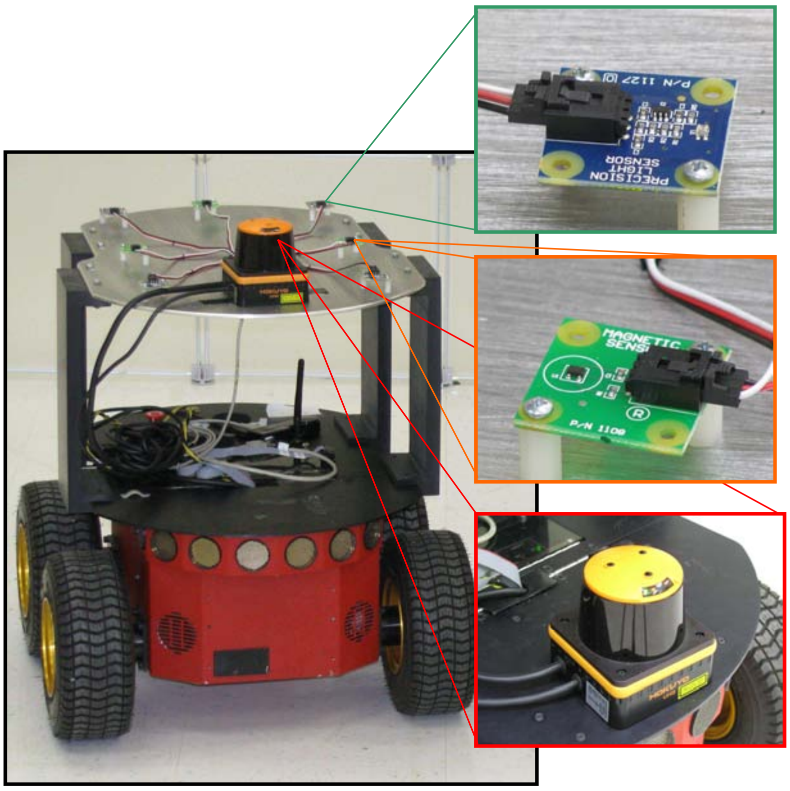


Figure 8.3: One of the four P3-AT robots available in the MARHES lab. Notice the detailed pictures of three different kinds of sensors, namely, Phidgets light precision sensors (top), Phidgets magnetic sensors (middle) and a Hokuyo UHG-08LX laser range finder (bottom), mounted on a custom fixture attached to the robot.

Phidgets Environmental Sensor Suite

The environmental sensor suite consists of a Phidgets USB interface I/O board [46] capable of measuring eight digital and analog inputs, and capable of driving eight digital outputs. The Phidgets I/O board can accommodate pressure, temperature, humidity, light intensity, and magnetic field sensors among others. Furthermore, it is equipped with a digital input hardware filter to eliminate false triggering at the digital inputs. A detailed view of a Phidgets USB interface with an attached light

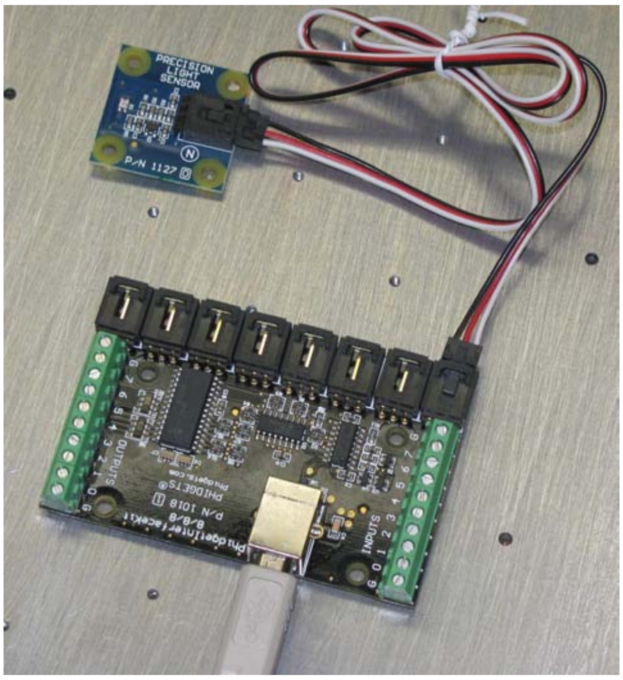


Figure 8.4: A Phidgets USB interface I/O board with an attached light precision sensor. Notice the USB port at the bottom of the image and the digital input outputs at the right and left edges respectively. The analog inputs are located at the top edge of the board where the light sensor is connected.

precision sensor is shown in Fig 8.4.

Light Precision Sensors

The 1127 precision light sensor [47] is a 5 VDC device able to measure from 1 lx to 1000 lx with a typical accuracy of 95%. The formula to convert the sensor data to luminosity units is

$$\text{Luminosity (lx)} = \text{SensorValue}$$

The main features of the device are given in Table 8.2, and a picture of the sensor

Table 8.2: General specifications of the Phidgets 1127 light precision sensor

Current consumption	2 mA
Bandwidth	50 Hz
Minimum Voltage	3.3 VDC
Maximum Voltage	5 VDC
Minimum light level	1 lx
Maximum light level	1000 lx
Error	5%

Table 8.3: General specifications of the Phidgets 1108 magnetic sensor

Current consumption	2 mA
Voltage output range	0.2 - 4.7 VDC
Device supply operating range	4.5 - 5.5 VDC
Typical error (@25°C)	±0.5%
Sensitivity	1 G/SensorValue

is shown in Fig. 8.3 (b).

Magnetic Sensors

The 1108 magnetic sensor [48] is a temperature stable sensor with a sensitivity of 1 G/SensorValue. The formula to convert the sensor value to Gauss units is

$$\Phi\ (\mathrm{G}) = 500 - \mathrm{SensorValue}.$$

We illustrate the main features of this sensor in Table 8.3. The sensor is shown in Fig. 8.3 (c).

Table 8.4: General specifications of the Hokuyo UHG-08LX laser

Power source	12 VDC ±10%
Current consumption	0.3 A
Detection Range	0.03 ∼ 11 m (Accuracy not guaranteed beyond 8 m)
Accuracy	0.1 ∼ 1 m : ±30mm 1 ∼ 8 m : 3% of the distance
Resolution	1 mm
Scan Angle	270°
Angular resolution	0.36°
Scan time	67 ms/scan
Ambient light resistance	Halogen/Mercury: 10000 lx or less Florescent: 6000 lx or less May cause measurement errors under strong light *e.g.* sunlight.
Weight	Approx 500 g
External dimensions	88 × 83 × 83 mm

UHG-08LX Laser

The Hokuyo UHG-08LX Laser [49] is an indoor range sensor which applies phase difference to get its distance measurements, minimizing the effect of the color and reflectance of the detected objects. The laser has a scan angle of 270° and a pitch of 0.36°. The maximum range of the sensor is 8 m with a divergence of 80 mm at that distance. The main features of the laser are illustrated in Table 8.4. The laser device is shown in Fig. 8.3 (d).

SICK LMS 200 Laser

As the UHG-08LX laser the SICK LMS 200 laser [50] is a non-contact measurement system which works in two-dimensions. The laser emits a pulsed laser beam which is reflected back if it reaches an object. The distance is calculated based on the time taken by the beam to go back to the device since this time is proportional to the

Table 8.5: General specifications of the SICK LMS 200 laser

Power source	24 VDC ±15%
Current consumption	current required max. 1.8 A
Operating ambient temperature	$0 - 50°$C
Data interface	RS232
Transfer rate	9.6/19.2/38.4/500 kbaud
Range	max 80 m
Angular resolution	0.25°/0.5°/1° selectable
Response time	53 ms/26 ms/13 ms
System Error	typ ±15 mm (mm mode), range 1 . . . 8 m typ ±4 cm (cm mode), range 1 . . . 20 m
Weight	approx. 4.5 kg
Dimensions	$156 \times 185 \times 136.8$ mm

distance between the laser and the object.

In contrast to the UHG-08LX Laser this device is optimized to be used outdoors and implements an algorithm for fog correction, and for cutting raindrops and snowflakes out.

The main technical features of the SICK LMS 200 laser are shown in Table 8.5. The SICK LMS 200 is shown in Fig. 8.5(a)

Global Positioning System

A Global Positioning system (GPS) is available to be mounted in our testbed for outdoor applications. It is composed of two parts namely, the enclosure and the antenna respectively. The enclosure model we have is a ProPak-V3 [51] and the antenna model is a GPS-702-GG [52].

The ProPak-V3 provides 72 channels, L1 and L2 GPS+GLONASS, a USB interface and SPAN capabilities which links the ProPak-V3 to an IMU. This guarantees

Table 8.6: General specifications of the ProPak-V3

Power source	9 - 18 VDC
Power consumption	2.8 W
Antenna port power output	5 VDC max 100 mA
Communication ports	1 RS-232 (921,600 bps) 1 RS-232 (230,400 bps) 1 USB 1.1 (5 Mbps)
Operating temperature	$-40°C - 75°C$
Horizontal position accuracy (RMS)	Single point L1: 1.8 m Single point L1/L2: 1.5 m
Data Rate	50 Hz
Time accuracy	typ 20 ns
Dimensions	$185 \times 160 \times 71$ mm
Weight	1 kg

stability for position, velocity and attitude measurements even in periods when satellite signals are blocked.

The GPS-702-GG antenna uses L1 and L2 frequencies and the signal reception is combined GPS+GLONASS. Furthermore, the phase center is stable which means that the phase center remains constant as the azimuth and elevation angle of satellites change.

Both devices, the enclosure and the antenna are made of durable, waterproof materials. Furthermore, they are vibration resistant.

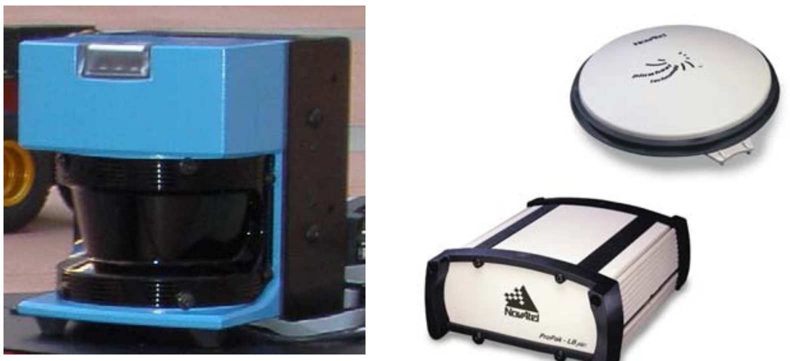
The technical specifications of the ProPak-V3 and the GPS-702-GG are given in Tables 8.6 and 8.7 and the devices are shown in Fig. 8.5(b).

Table 8.7: General specifications of GPS-702-GG

Power source	4.5 - 18 VDC
Power consumption	35 mA
Operating temperature	$-40°C - 85°C$
3dB pass band	L1: 1588.5 ± 2 MHz L2: 1236 ± 18.3 MHz
LNA gain	29 dB
Propagation delay	5 ns (maximum)
Nominal impedance	50 Ω
Dimensions	185 mm diameter × 69 mm
Weight	500 g

Sonar Front Array

This low cost sensor is a classroom oriented tool to give the students experience with obstacle detection sensors [45]. As shown in Fig. 8.6(a), the circular sensors at the front of the robot forms an array of sonars, with 2 sonars on each side and 6 more



(a) (b)

Figure 8.5: The picture in (a) shows The SICK LMS 200 laser at the MARHES lab at The University of New Mexico and the picture in (b) shows the enclosure ProPak-V3 (lower left) and the antenna GPS-702-GG (upper right) for the GPS system.

Table 8.8: General specifications of the motion encoders of the P3-AT robots

Counts/rev	34,000
Counts/mm	49
Counts/rotation	22,500

Table 8.9: General specifications of the Werker Battery

Charge	252 W-hrs
Run Time	2-3 hrs (with PC)
Run Time	4-6 hrs (without PC)
Recharge time	6 hrs/battery (std charger)
Recharge time	2.4 hrs/battery (hight speed charger)

forward covering interval angles of 20°.

Motion Encoders

The P3-AT comes with 4 motion encoders [45] ready-to-use with the features shown in Table 8.8,

8.1.2 Power Source

The P3 - AT robot can accommodate up to three [53] WKA12-7.5F batteries which can store up to 252 W-hrs and provide 12 VDC. If the robot is equipped with good condition batteries the average run time is around 2 to 3 hrs. The werker battery used for our experiments is shown in Fig. 8.6(b).

Some features of the standard batteries are shown in Table 8.9.

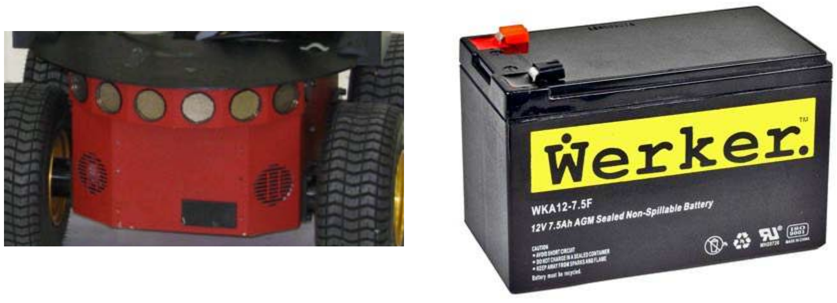


Figure 8.6: The picture in (a) shows the front sonar array of a P3-AT robot and the picture in (b) a 12VDC werker battery used in the P3-AT robots.

8.2 The P3-AT Software Libraries

In this section we describe the software tools available in the MARHES lab, used to access and control the different sensors and devices in the P3-AT robots.

8.2.1 The Player/Stage/Gazebo Project

The Player/Stage/Gazebo project is an open source development oriented to support the research in robotics and sensor systems [43]. Player/Stage/Gazebo is available for download at `http://sourceforge.net/projects/playerstage/files/` under the policies of the GNU General Public License. Any developer has access to the codes in order to modify them, in that way new hardware drivers can be developed and the old drivers can be improved.

The project involves three components namely Player, Stage and Gazebo. The operating systems compatible with Player/Stage/Gazebo are Solaris, Linux and Mac OSX. Recently, just the component Player v-3.0.0 has been ported to run on Windows but any work has been done neither in Stage nor Gazebo yet.

Player

Player [43] is a network server aimed to control the robots and the sensors and actuators attached to it thorough an IP network. A program running in the client communicates to a Player server through a TCP socket. Player provides a network interface for the sensors and actuators on the robot which allows the client program to read data and send commands to the actuators in real time.

Player has a good variety of drivers which support several robot models, sensors and actuators. Some developments in speech and pattern recognition software are supported by Player as well. Furthermore, it is worth mentioning that simulators like Stage and Gazebo are compatible with Player. A complete list of Player supported devices can be found at `http://playerstage.sourceforge.net/doc/Player-2.1.0/player/supported\_hardware.html`.

Using Player, several clients can access the same robot at the same time which makes it a good option for experimenting with multivehicle control and coordination applications. The typical programming languages used for player are C++, Tcl, Java and Python which support TCP sockets. Particularly in the Marhes lab we have been working with C++.

Stage

Stage [54] is a two-dimensional simulator aimed to support research in multiagent autonomous systems. It can simulate populations of sensors, robots, objects and environments to be sensed and manipulated by the robots. It provides simplified models of several devices in order to make the simulations computationally possible, providing a good approach to the real experiment.

Stage is provided with several mechanicals and dynamical models which are realistic enough to try the algorithms out before implementing them in the real robots. Using the Stage plug-in for player *libstageplugin* we can interface Player with Stage in order to control a population of virtual robots in a virtual world thorough a TCP socket.

In order to create a customized virtual world, Stage provides tools to identify blueprints of a building from a common bitmap and allows you to add customized bodies with dynamical and geometric parameters to be manipulated by the virtual robots. Stage has an available model of the P3-AT robot which was used in our simulations. A sampling of a P3-AT robot taking measurements of an obstacle using a Stage model of the SICK LMS 200 laser is shown in Fig. 8.7.

Gazebo

Gazebo [55] is a three-dimensional simulator which provides virtual environments composed of objects and blueprints that can be sensed and manipulated by virtual robots. It is provided with dynamical models which simulate different forces such as friction and gravity that are merely considered in a two-dimensional simulator such as Stage. Furthermore, Gazebo can emulate collisions and even sensor measurements.

Gazebo interfaces with Player in order to implement the client applications in the virtual environment. However, using the Gazebo libraries *libgazebo* the developers are able to interface Gazebo with their own customized server/client applications.

Gazebo provides some tools to design customized models of new sensors, robots or even objects using three-dimensional predefined geometric shapes and different colors and textures. There is a predefined mechanical model of the P3-AT robot for

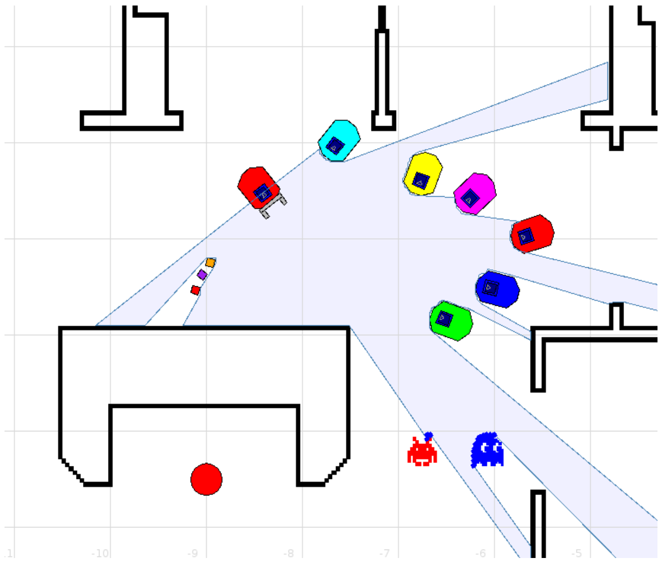


Figure 8.7: A Stage model of a P3-AT robot (in red) with a gripper and an LMS 200 laser exploring a virtual world with different kinds of objects, including another P3-AT robots.

gazebo as the one shown in Fig. 8.8 (b). Notice the resemblance with the picture shown in Fig. 8.8 (a).

8.2.2 Hardware-Software Interaction

After describing the main features of the hardware and software modules involved in our multivehicle testbed we consider convenient to describe the interaction between them. Based on Fig. 8.9 we See that the embedded computer in the P3-AT robot accesses the sensors and actuators in the robot through several interfaces such as the USB, FireWire or RS-232 ports. Using the P3-AT player libraries the computer can communicate with each one of the sensors and actuators using the corresponding

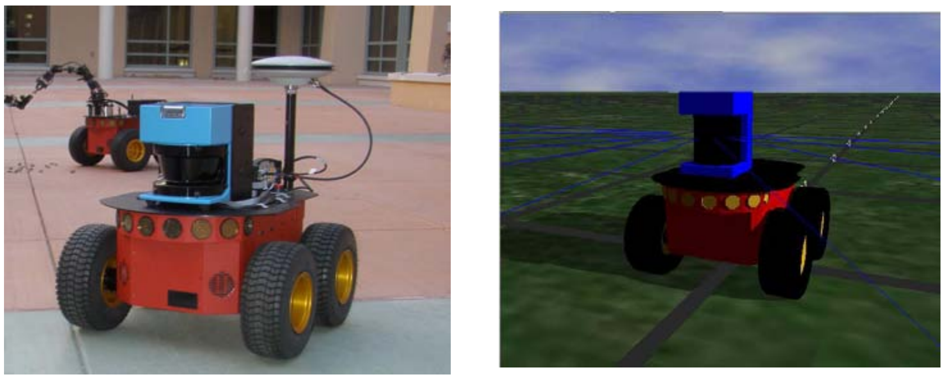

(a) Real P3-AT (b) Virtual P3-AT

Figure 8.8: The pictures show the appearance of the real P3-AT and the Gazebo model of a P3-AT.

drivers.

The embedded computer acts as a server and the client program can be running locally or remotely, so that the client uses a TCP socket through a network which most of the time is wireless as indicated in the diagram. A wired connection is possible, however for the mobility of the robots it becomes cumbersome.

Notice that the client program can access the service from the embedded computer in the robot or from one of the simulators we presented previously. Lastly, in the MARHES lab all the P3-AT robots work with a reduced installation of Ubuntu linux version 8.04 which is a stable version suitable for network applications.

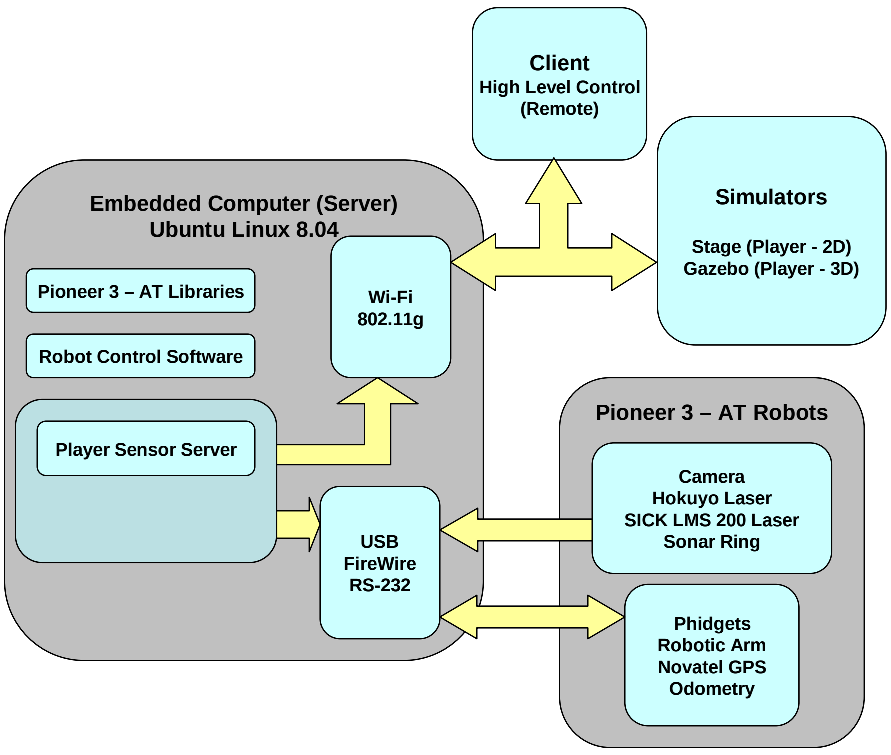


Figure 8.9: The diagram illustrates the software-hardware interaction and the software architecture of the robot. Notice that the services of Player run locally in the P3-AT robots allowing a local or remote application to access the sensors and actuators of the real or virtual robot.

Chapter 9

Conclusions and Discussion

In this thesis we have developed an adaptive controller for deployment of nonholonomic sensor networks to carry out a coverage and estimation of a parameterizable density function in a convex sampling space. We provided a stability theorem which states that the robots distribute themselves in an optimal way over the density function solving the locational optimization problem. The theorem guarantees the convergence of the estimation of the density function parameters based on local measurements of the mobile sensors. The mobile sensors were modeled as unicycle vehicles and a nonlinear steering control law in polar coordinates was used to drive them and guarantee stability.

Previous versions of this controller in the literature [56],[57] have shown that this kind of controller can be useful in search and rescue missions, environmental monitoring and automatic surveillance of buildings or towns. The main difference with similar approaches in the literature is the inclusion of nonholonomic sensor networks as well as the validation of the controller in dynamic density functions with piece-wise constant parameter dynamics.

Through simulations in Stage and Matlab as well as some experimental testing using a team of four P3-AT robots and the Player/Stage/Gazebo project we verified our theoretical results. The experiments showed that the implementation of our coverage controller is feasible and useful in practice.

The assumptions presented in the development of this thesis may or may not be fulfilled in a real scenario. The sampling space is considered convex and the density function is not only assumed to be parameterizable but the parameters should be positive. The communication between the sensors and their neighbors is considered perfect to allow the calculation of the Voronoi partitions and the transmition of their measurements and positions thorough the network. During the experiments we assumed that the odometry was perfect, but usually some error corrections method should be applied.

Our future work is aimed to overcome the mentioned issues. Designing an extension for non-convex sampling spaces similar to the one in [37] and considering obstacles would give us a more realistic approach. Considering possible communication failures in our mathematical model and applying odometry correction mechanisms such as vision and Kalman filtering [58] would help us improve the robustness and accuracy of the system. Furthermore, the development of a theoretical framework for continuous time variant density functions rather than piece-wise constant density functions is encouraging in order to spread the range of applications of our controller. Finally, the study of mechanisms to fulfill the persistency of excitation condition to guarantee the convergence of the parameter estimation vector $\hat{a}_i$ to zero would improve the estimation performance of the system.

Appendix A

Complement of the Proof of Stability (Theorem 4)

Since the weighting function $w(t)$ in the second term $\sum_{i=1}^{n} \gamma \int_0^t w(\tau)(\mathcal{K}_i(\tau)^T \tilde{a}_i)^2 \, d\tau$ in (5.10) is designed to be non-negative (*e.g.,* an exponential function) and is multiplied by a quadratic expression, then the whole term is non-negative.

For the third term in (5.10) $\sum_{i=1}^{n} \tilde{a}_i^T \zeta \sum_{j \in N_i} (\hat{a}_i - \hat{a}_j)$, and following the same procedure as [57], we have that for a graph $G = (V, E)$ with vertices $V = \{v_1, \ldots, v_n\}$ and edges $E = \{e_1, \ldots, e_l\}$, $e_i = \{v_j, v_k\}$, we can associate every agent to one vertex.

Now, we can define the neighborhood set of vertex v_i as $\mathcal{N}_i = \{j | v_i, v_j\} \in E$ with the adjacency matrix A of G defined as

$$A(i,j) = A(j,i) = \begin{cases} 1 & \text{for } \{v_i, v_j\} \in E, \\ 0 & \text{otherwise.} \end{cases}$$

Based on [57] we define the laplacian matrix as $L = \mathrm{diag}_{i=1}^{n}(|\mathcal{N}_i|) - A$. For this specific case the graph is connected, then $L \geq 0$ with a unique eigenvalue in 0 and the associated eigenvector $\mathbf{1} = \{1, \ldots, 1\}$. This means that $L\mathbf{1} = \mathbf{1}^T L = 0$, $x^T L x \geq 0 \forall x$, and $x^T Lx = 0$ implies $x = 0$ or $x = \mathbf{1}c$ for some $c \in \mathbb{R}$.

Now, expressing the third term in (5.10) in terms of the laplacian we have

$$\sum_{i=1}^{n} \tilde{a}_i^T \zeta \sum_{j\in\mathcal{N}_i} (\hat{a}_i - \hat{a}_j) = \zeta \sum_{j=1}^{m} \tilde{b}_j^T L(t)\hat{b}_j, \tag{A.1}$$

where $b_j = a(j)\mathbf{1}$, $\hat{b}_j = [\hat{a}_1(j)\ldots\hat{a}_n(j)]^T$ and $\tilde{b}_j = \hat{b}_j - b_j$.

Then we have that $b_j^T L(t) = a(j)\mathbf{1}^T L = 0$ and (A.1) becomes

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

Since for the connected graph $L(t) \geq 0\ \forall t \geq 0$ then the third term in (5.10) gives

$$\sum_{i=1}^{n} \tilde{a}_i^T \zeta \sum_{j\in N_i} (\hat{a}_i - \hat{a}_j) \geq 0,$$

which is non-negative.

For the fourth term $\sum_{i=1}^{n} \tilde{a}_i^T I_{proj}\dot{\hat{a}}_{pre_i}$ in (5.10) we have that the j-th scalar term in the summation is $\tilde{a}_i^T(j)I_{proj}(j)\dot{\hat{a}}_{pre_i}(j)$.

From (4.13) we have that if $\tilde{a}_i(j) > a_{min}$ or if $\tilde{a}_i(j) = a_{min}$ and $\dot{\hat{a}}_{pre_i} \geq 0$ the scalar term becomes 0. On the other hand if $\hat{a}_i(j) < a_{min}$ and $\dot{\hat{a}}_{pre_i} < 0$ then $\tilde{a}_i(j) = \hat{a}_i(j) - a(j) \leq 0$ and from (4.13) $I_{proj_i}(j) = 1$, then

$$\tilde{a}_i^T(j)I_{proj}(j)\dot{\hat{a}}_{pre_i}(j) \geq 0 \quad \forall j = 1, \ldots, n, \tag{A.2}$$

and we conclude that the fourth term in (5.10) is non-negative.

References

[1] J. Clark and R. Fierro, "Mobile robotic sensors for perimeter detection and tracking," *ISA Trans.*, vol. 45, pp. 3–13, 2007.

[2] D. W. Casbeer, D. B. Kingston, R. W. Beard, T. W. Mclain, S. M. Li, and R. Mehra, "Cooperative forest fire surveillance using a team of small unmanned air vehicles," *International Journal of Systems Sciences*, vol. 37, no. 6, pp. 351–360, 2006.

[3] J. Mandel, J. D. Beezley, J. L. Coen, and M. Kim, "Data assimilation for wildland fires. ensemble kalman filters in coupled atmosphere-surface models," *IEEE Control Systems Magazine*, vol. 29, no. 3, pp. 47–65, 2009.

[4] R. A. Cortez, "Information-driven cooperative control for radiation map building," M.S. Thesis, The University of New Mexico, 2007.

[5] M. Schwager, D. Rus, and J. J. E. Slotine, "Decentralized, adaptive control for coverage with networked robots," *International Journal of Robotics Research*, vol. 28, no. 3, pp. 357–375, March 2009.

[6] A. Kwok and S. Martínez, "Unicycle coverage control via hybrid modeling," *IEEE Transactions on Automatic Control*, submitted, 2008, revised 2009.

[7] J. Cortés, S. Martínez, T. Karatas, and F. Bullo, "Coverage control for mobile sensing networks," *IEEE Transactions on robotics and Automation*, vol. 20, pp. 243–255, 2004.

[8] D. Marthaler and A. L. Bertozzi, "Tracking environmental level sets with autonomous vehicles," in *Recent Developments in Cooperative Control and Optimization*, R. M. S. Butenko and e. P. M. Pardalos, Eds. Kluwer Academic Publishers, 2003, pp. 317–330.

[9] R. Olfati-Saber and R. R. Murray, "Consensus problems in networks of agents with switching topology and time-delays," *IEEE Transactions on Automatic Control*, vol. 49, pp. 1520–1533, 2004.

[10] S. Susca, S. Martínez, and F. Bullo, "Monitoring environmental boundaries with a robotic sensor network," vol. 16, no. 2, pp. 288–296, 2008.

[11] G. Lee, N. Chong, and H. Christensen, "Adaptive triangular mesh generation of self-configuring robot swarms," in *Proc. of the International Conference on Robotics and Automation (ICRA 09)*, Kobe, Japan, May 2009, pp. 2737–2742.

[12] J. M. Luna, R. Fierro, C. T. Abdallah, and J. Wood, "Distributed, adaptive algorithm for deployment of nonholonomic sensor networks," in *IEEE International Conference on Robotics and Automation (ICRA 2010)*, May 2010, submitted.

[13] Y. Mostofi and P. Sen, "Compressive cooperative sensing and mapping in mobile networks," in *Proc. of the American Control Conference (ACC 09)*, St. Louis, Missouri, Jun 10-12 2009, pp. 3397–3404.

[14] S. P. Hou and J. J. E. Slotine, "Dynamic region following formation control for a swarm of robots," in *Proc. of the International Conference on Robotics and Automation (ICRA 09)*, Kobe, Japan, May 12-17 2009, pp. 1929–1934.

[15] A. Deshpande, S. Poduri, D. Rus, and G. S. Sukhatme, "Distributed coverage control for mobile sensors with location-dependent sensing models," in *Proc. of the International Conference on Robotics and Automation (ICRA 09)*, Kobe, Japan, May 2009, pp. 2344–2349.

[16] F. Bullo, J. Cortés, and S. Martínez, *Distributed Control of Robotic Networks*, ser. Applied Mathematics Series, 2009, available at http://www.coordinationbook.info.

[17] R. Graham and J. Cortés, "Distributed sampling of random fields with unknown covariance," in *Proc. of the American Control Conference (ACC 09)*, St. Louis, Missouri, Jun 10-12 2009, pp. 4543–4548.

[18] M. Schwager, B. J. Julian, and D. Rus, "Optimal coverage for multiple hovering robots with downward facing cameras," in *Proc. of the International Conference on Robotics and Automation (ICRA 09)*, Kobe, Japan, May 12-17 2009, pp. 3515–3522.

[19] L. Moreau, "Stability of multiagent systems with time-dependent communication links," *IEEE Transactions on Automatic Control*, vol. 50, no. 2, pp. 169–182, 2005.

[20] W. Ren and E. M. Atkins, "Distributed multi-vehicle coordinated control via local information exchange," *International Journal in Robust and Nonlinear Control*, vol. 17, pp. 1002–1033, 2007.

[21] M. Franceschelli, M. Egerstedt, A. Giua, and C. Mahulea, "Constrained invariant motions for networked multi-agent systems," in *Proc. of the International Conference on Robotics and Automation (ICRA 09)*, Kobe, Japan, May 12-17 2009, pp. 5749–5754.

[22] D. V. Dimarogonas and K. J. Kyriakopolous, "On the rendezvous problem for multiple non-holonomic agents," *IEEE Transactions on Automatic Control*, vol. 52, no. 5, pp. 916–922, 2007.

[23] W. Dong and J. A. Farrel, "Cooperative control of multiple non-holonomic vehicle," *IEEE Transactions on Automatic Control*, vol. 53, no. 6, pp. 1434–1448, 2008.

[24] Y. Lan, G. Yan, and Z. Lin, "A hybrid control approach to cooperative target tracking with multiple mobile robots," in *Proc. of the American Control Conference (ACC 09)*, St. Louis, Missouri, Jun 10-12 2009, pp. 2624–2629.

[25] J. Wu and Z. Jiang, "On the switching control of multiple mobile robots formation," in *Proc. of the International Conference on Robotics and Automation (ICRA 09)*, Kobe, Japan, May 12-17 2009, pp. 2711–2716.

[26] A. Oikonomopoulos, S. G. Loizou, and K. J. Kyriakopoulos, "Coordination of multiple non-holonomic agents with input constraints," in *Proc. of the International Conference on Robotics and Automation (ICRA 09)*, Kobe, Japan, May 12-17 2009, pp. 869–874.

[27] R. G. Sanfelice, R. Goebel, and A. R. Teel, "Results on convergence in hybrid systems via detectability and invariance principle," in *Proc. of the American Control Conference (ACC 09)*, 2005, pp. 551–556.

[28] R. Olfati-Saber and R. M. Murray, "Consensus and cooperation in networked multi-agent systems," *Proceedings of the IEEE*, vol. 95, no. 1, pp. 215–233, 2007.

[29] K. D. Listmann, M. V. Masalawala, and J. Adamy, "Consensus for formation control of nonholonomic mobile robots," in *Proc. of the International Conference on Robotics and Automation (ICRA 09)*, Kobe, Japan, May 12-17 2009, pp. 3886–3891.

[30] J. J. E. Slotine and W. Li, *Applied Nonlinear Control*. Upper Saddle River, NJ 07458: Prentice Hall, 1991.

[31] H. K. Khalil, *Nonlinear Systems*, 3rd ed. Upper Saddle River, NJ 07458: Prentice Hall, 2002.

[32] R. Fierro and F. L. Lewis, *Encyclopedia of Electrical and Electronics Engineering.* John Wiley & Sons Inc., 1999, ch. Robot Kinematics, pp. 559–571.

[33] S. M. LaValle, *Planning Algorithms.* Cambridge, U.K.: Cambridge University Press, 2006, available at http://planning.cs.uiuc.edu/.

[34] M. Pavone, A. Arsie, E. Frazzoli, and F. Bullo, "Equitable partitioning policies for robotic networks," in *Proc. of the International Conference on Robotics and Automation (ICRA 09)*, Kobe, Japan, May 2009, pp. 2356–2361.

[35] J. G. M. Fu, T. Bandyopadhyay, and M. H. A. Jr, "Local voronoi decomposition for multi-agent task allocation," in *Proc. of the International Conference on Robotics and Automation (ICRA 09)*, Kobe, Japan, May 12-17 2009, pp. 1935–1940.

[36] Q. Du and M. Gunzburguer, "Centroidal voronoi tessellations: Aplications and algorithms," *SIAM Review*, vol. 41, pp. 637–676, 1999.

[37] L. C. A. Pimenta, V. Kumar, R. C. Mesquita, and G. A. S. Pereira, "Sensing and coverage for a network of heterogeneous robots." in *CDC.* IEEE, 2008, pp. 3947–3952.

[38] G. Casalino, M. Aicardi, A. Bicchi, and A. Balestrino, "Closed loop steering and path following for unicycle-like vehicles: a simple lyapunov function based approach," *IEEE Robotics and Automation Magazine*, vol. 2, no. 1, pp. 27–35, March 1995.

[39] D. Liberzon, *Switching in Systems and Control*, ser. Systems & Control: Foundations & Applications, Birkhauser, Ed., June 2003.

[40] M. Schwager, J. McLurkin, J. J. E. Slotine, and D. Rus, "From theory to practice: Distributed coverage control experiments with groups of robots," in *Proceedings of the International Symposium on Experimental Robotics*, Athens, Greece, July 2008.

[41] "Voro++ documentation," [online] Available: http://math.lbl.gov/voro++/doc/, 2009.

[42] D. Cruz, J. McClintock, B. Perteet, O. Orqueda, Y. Cao, and R. Fierro, "Decentralized cooperative control: A multivehicle platform for research in networked embedded systems," *IEEE Control Systems Magazine*, vol. 27, no. 3, pp. 58 – 78, June 2007.

[43] "The player robot device interface," [online] Available: http://playerstage.sourceforge.net/doc/Player-2.1.0/player/index.html, 2007.

[44] "Aria," [online] Available: http://robots.mobilerobots.com/wiki/ARIA#Documentation, 2007.

[45] "Pioneer 3 operations manual," [online] Available: http://robots.mobilerobots.com/docs/all_docs/P3OpMan5.pdf, 2007.

[46] "Product manual. 1018 - phidgetinterfacekit 8/8/8," [online] Available: http://www.phidgets.com/documentation/Phidgets/1018.pdf, 2007.

[47] "Product manual. 1127 - precision light sensor," [online] Available: http://www.phidgets.com/documentation/Phidgets/1127.pdf, 2007.

[48] "Product manual. 1108 - magnetic sensor," [online] Available: http://www.phidgets.com/documentation/Phidgets/1108.pdf, 2007.

[49] "Scanning laser range finder uhg-08lx specifications," [online] Available: http://www.acroname.com/robotics/parts/R311-HOKUYO-LASER2s.pdf, 2007.

[50] "Lms200/211/221/291 laser measurement systems," [online] Available: http://www.mysick.com/saqqara/get.aspx?id=IM0012759, 2006.

[51] "Enclosures. propack-v3," [online] Available: http://www.novatel.com/Documents/Papers/ProPakV3.pdf, 2009.

[52] "Antennas. gps-701-gg and gps-702-gg," [online] Available: http://www.novatel.com/Documents/Papers/GPS701_702GG.pdf, 2009.

[53] "Werker. wka12-7.5f. sealed lead acid absorbed glass mat. technical specifications," [online] Available: http://www.security1call.com/pdf_folder/WKA12-7%205F.pdf.

[54] "The stage robot simulator," [online] Available: http://playerstage.sourceforge.net/doc/Stage-3.2.1/, 2007.

[55] "Gazebo," [online] Available: http://playerstage.sourceforge.net/doc/Gazebo-manual-0.8.0-pre1-html/, 2007.

[56] M. Schwager, J. J. Slotine, and D. Rus, "Descentralized adaptive control for coverage with networked robots," in *Proceedings of International Conference on Robotics and Automation*, Rome, April 2007.

[57] ——, "Consensus learning for distributed coverage control," in *Proceedings of International Conference on Robotics and Automation*, Pasadena, CA, May 2008.

[58] A. K. Das, R. Fierro, V. Kumar, J. P. Ostrowski, J. Spletzer, and C. J. Taylor, "A vision-based formation control framework," *IEEE Transactions in Robotics and Automation*, vol. 18, pp. 813–825, 2002.