MULTIPLE ROBOT BOUNDARY TRACKING WITH PHASE AND WORKLOAD BALANCING

A Thesis

Presented to

the Faculty of California Polytechnic State University

San Luis Obispo

In Partial Fulfillment

of the Requirements for the Degree

Master of Science in Electrical Engineering

by

Michael J. Boardman

June 2010

COMMITTEE MEMBERSHIP

TITLE:	Multiple Robot Boundary Tracking with Phase and Workload Balancing
AUTHOR:	Michael J. Boardman
DATE SUBMITTED:	June 2010
COMMITTEE CHAIR:	Christopher M. Clark, Ph.D.
COMMITTEE MEMBER:	Fred DePiero, Ph.D.
COMMITTEE MEMBER:	Lynne Slivovsky, Ph.D.

Abstract

Multiple Robot Boundary Tracking with Phase and Workload Balancing

Michael J. Boardman

This thesis discusses the use of a cooperative multiple robot system as applied to distributed tracking and sampling of a boundary edge. Within this system the boundary edge is partitioned into subsegments, each allocated to a particular robot such that workload is balanced across the robots. Also, to minimize the time between sampling local areas of the boundary edge, it is desirable to minimize the difference between each robot's progression (i.e. phase) along its allocated sub segment of the edge. The paper introduces a new distributed controller that handles both workload and phase balancing. Simulation results are used to illustrate the effectiveness of the controller in an Autonomous Underwater Vehicle (AUV) under ice edge sampling application. Successful results from experimentation with three iRobot® Creates are also presented.

Contents

List of Tables **vii**

List of Figures **viii**

1 INTRODUCTION **1**

2 BACKGROUND **4**

3 PROBLEM DEFINITION **12**

3.1 Workload 13

3.2 Phase 13

3.3 Implementation 14

4 CONTROLLER DESIGN **15**

5 PARTICLE FILTER LOCALIZATION **19**

6 EXPERIMENT **22**

6.1 Implementation 24

6.1.1 Simulation 24

6.1.2 Hardware 24

6.1.3 Software 26

6.1.4 Environment 28

6.1.5 Particle Filter 29

6.2 Experiment Configuration 31

7 RESULTS **33**

7.1 Simulation 33

7.2 Robot Implementation 35

7.2.1 Odometry 35
7.2.2 Particle Filter Localization 36
8 CONCLUSION **38**
9 FUTURE WORK **40**
Bibliography **42**

List of Tables

7.1 Phase Steady State Error (*radians*) 34
7.2 Workload Steady State Error (m^2) 34
7.3 Experiment Steady State Error 36

List of Figures

1.1 A candidate coordinate system for distributed control of multiple robots on a boundary edge. In this example, multiple AUVs are distributed along an edge of ice rafts. 2

3.1 Distributed Multiple Robot System Performing Boundary Tracking 14

6.1 Experiment setup: Three iRobot® Creates before navigating under a simulated ice edge with global axes displayed. 22

6.2 Desired States of Robot Crossing Patrolling Axis 23

6.3 The iRobot® Create robot is shown from above in (a), and below in (b). The bluetooth module and IR sensor added to the robot can be found in (c). 25

6.4 Hardware Flow Diagram 26

6.5 System Level Algorithm 27

6.6 Workload/Phase Balancing Algorithm 28

6.7 The hardware setup: Three iRobot® Creates navigated underneath an overhanging foam block used to simulate an ice raft. . . 29

6.8 The SICK Laser rangefinder is shown in (a). A scan of the robots at their initial position with particle filter localization is seen in (b). 30

6.9 Particle Filter Test: iRobot® Create moving toward location of known distance (tape). 31

6.10 The Simulator GUI: A top down view of three robots (circles) navigating water (blue) and under ice (white). The algae (green) covers part of the under ice edge in some scenarios. In (a), there is no algae on ice edge. In (b), algae is covering half of ice edge. Scenario simulation (c) has a random coverage of algae. 32

7.1 Simulation results for the case with no algae under the ice and the three robots starting in phase. The phase error is plotted in (a) and the workload error is plotted in (b). 33

7.2 Trajectories of iRobot® Creates in following the simulated ice raft (i.e. the overhanging foam block.) 35

7.3 Experimental results for the case with no algae under the ice and the three iRobot® Create robots starting in phase. The phase error is plotted in (a) and the workload error is plotted in (b). . . 36

7.4 Trajectories of iRobot® Creates using Particle Filter Localization 37

7.5 Experimental results for the case with no algae under the ice and each robot starting 1m apart from one another. The phase error is plotted in (a) and the workload error is plotted in (b). 37

9.1 OceanServer Iver2 AUVs can be deployed in the Arctic 41

Chapter 1

INTRODUCTION

Robots are increasingly being used to perform a large variety of tasks. Commercial applications give robots the ability to assist the disabled, clean homes, and aid in the manufacturing and processing of products. Military applications give robots additional purpose. They can scour fields for mines, search for snipers in urban combat environments, and even maintain full battlefield awareness for soldiers. They also have great potential in scientific exploration. They have the capability to withstand harsh and unforgiving environments thereby giving them the ability to perform tasks humans cannot perform.

In single robot systems, there is a higher likelihood of mission failure. If the single robot fails, the mission fails. Further, a single robot can only cover so much area in a given time frame. Multi-robot systems typically do not suffer from such possibility of a single point failure. Multi-robot systems also allow for accomplishing tasks of larger size and complexity when compared with single robot systems.

This thesis concerns the task of tracking and sampling the (possibly dynamic)

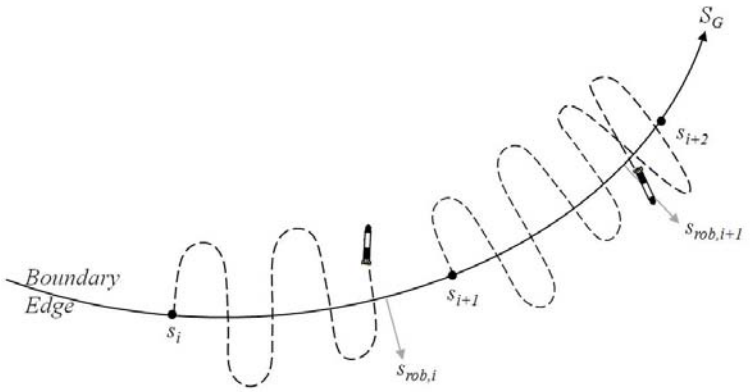


Figure 1.1: A candidate coordinate system for distributed control of multiple robots on a boundary edge. In this example, multiple AUVs are distributed along an edge of ice rafts.

boundary of some entity with multiple robots. This is useful for scientific, military, and even commercial applications. For example, boundary tracking can be used to track a crowd of people, survey an oil spill, or detect the edge of a harmful algae bloom with Autonomous Underwater Vehicles (AUVs).

A goal of this research is to design a distributed controller in which multiple robots track and follow a continuous boundary edge, while balancing both the phase and workload between vehicles. Controlling the workload will allow the coverage along the boundary to be partitioned equally between the multiple robots. Meanwhile, the phase controller will ensure the robots are at the same location within their partition of the boundary.

In chapter 2, a background of multiple robot systems and boundary tracking is discussed. Chapter 3 defines specifics of the problem at hand. Chapter 4 is an overview of the controller method used for the cooperative multiple robot system. Particle filter localization is described in chapter 5. In chapter 6, simulation and real world experimentation implementation are discussed. Results are illustrated in chapter 7. Finally, chapter 8 gives a conclusion of the information discussed

in this thesis, and future work on the topic is revealed in chapter 9.

Chapter 2

BACKGROUND

There are several functional areas needed to create a distributed multiple robot system that can perform phase and workload balancing for the purpose of boundary tracking. The vehicle localization is necessary for robot patrol applications. Boundary tracking and more importantly boundary following is required for this distributed multiple robot application. Area coverage allows a given space to be surveyed in its entirety. Phase balancing and spatio-temporal sampling methods give robots the capability to patrol a given area at a consistent rate.

Related to this research is work performed on the localization and navigation of autonomous robots. In [10], a simple autonomous underwater vehicle utilizes a vision system to identify a laser on a target. This determines the distance from the robot to the target using the image location of where the laser as it hits the target. The navigation system uses a compass module to gain trajectory information, while a laser rangefinder [19] is used for distance information. The calculation controller unit determines the bearing of the robot with respect to the target area. The motor controller linearly reduces the error between the bearing

of the robot and the target area to zero in order to arrive at the desired location.

Sensor measurements can not be assumed to be exact in real world scenarios. A.W. Stroupe, et. al. [2] performs object tracking by fusing Gaussian distributions of sensor data from multiple robots. The approach used in [2] utilizes Bayes rule with a Kalman filter [17], [19], [23]. Multiple Gaussian distributions of the data are thus combined using simple matrix operations. Fusing the data in this manner provides the ability to more accurately predict the location of an object with multiple robots and permits these robots to be localized within their environment. Kalman filtering is typically used to localize robots within a mapped environment and object tracking. Monte Carlo localization is normally applied to multiple robot localization in unknown environments.

Monte Carlo (or particle filter) localization [13],[23] is utilized to estimate the state of the centroid of a triangle with each robot representing a corner. The particle filter allows for less state information to be required (the position of the centroid and its orientation) and thus less computational complexity. Further, this localization method allows for robots with limited sensor capability to accurately localize themselves within their environment. Though each robot performs the particle filter independently, the robots share their highest belief states with every other robot before particle resampling. Given the distance between each robot, the robots are able to utilize the centroid information to localize themselves within their environment. Though [13] discusses this form of localization in a three robot environment, it is also inferred that this method can be scaled to several robots. This method also requires the robots having the capability to sense one another. Otherwise particle filters must be performed on each individual robot.

Particle SLAM algorithms can use prior low resolution maps of an area to im-

prove the accuracy in a map and trajectory of an autonomous underwater vehicle [20]. The BPSLAM algorithm utilizes a particle's ancestry in the particle filter to create the map of the environment. The prior maps of an area are initialized as the map for the root particle, the particle at the base of the ancestry tree. Particles are weighted based on the quality of their observation with respect to their stored estimates. This method improves localization and mapping accuracy only when prior maps of an area are available.

Several traditional robot navigation strategies have been applied to boundary tracking. To follow boundaries while avoiding obstacles, work in [21] used Artificial Potential Fields. Potential field systems can be used to guide a robot toward a target while avoiding obstacles. Though there is a risk of becoming caught in a local minima, measures can be made to reduce the risk.

Work in [3] describes a method of implementing a global path planner with local sensor data. Instant goals are used to set the path that the robot will follow along with a boundary following algorithm to maintain global boundary following and prevent local minima due to obstacles. This method may detract a robot from observing important elements during observation as an attempt to follow around obstacles.

In [8], non-linear planning is used as an optimization-based approach for path-planning. Given a location that is a significant distance away in an unknown environment, it is not feasible to plan the entire path off line. Real-time path planning occurs as multiple plans are created to plans the robot trajectory to a point on the horizon in the direction toward its goal state.

Using an auction system [24], robots are able to travel to all task points faster than just alternating the assignment of points. Though this motion planner can

be utilized to patrol a boundary area, localized dynamic events may be missed when performing scientific missions requiring such observation.

Q-learning allows the robot to create a path based on a reward system [15]. The Q-learning algorithm gives the robot the ability to not only move based on previous rewards, but also to explore new areas to determine other methods of traveling to the target location. Self-organizing control uses a measurement unit and a modifying unit to control the actual movement of the robot. The measurement unit is a fuzzy incremental controller, while the modifying unit modifies the F-table when controller performance is not met. The purpose of such a controller can be utilized in applications not calling for specific path following.

In [11], a collaborative path planning algorithm assigns one robot as a coordinator and all of the robots comprise a team. The coordinator guides the other robots toward a target. The follower robots are designed to follow the coordinator robot in order to give additional sensing of the environment to the follower robot. Making a single robot the coordinator can lend itself to issues caused by a single robot's failure.

The distance of the robot from a discovered boundary becomes minimized as it moves back and forth across a boundary in [1]. Multiple robots are used to collect and analyze information to follow the boundary closely in a convoy. The research in [1] did result in false positives due to noisy sensor readings. The focus of this thesis was to make all robots traverse the entire area getting as close to the boundary as possible. Another method for coordinating the robots would be to divide the boundary area to be covered.

F. Zhang and S. Haq [9] propose a shape theory approach to boundary following in a multiple mobile robot system. The center of mass of a group of robots,

formed in any shape, (similar to [13]) is used to determine boundary following characteristics. In [9], robots are to keep a distance from the boundary to be followed so as to avoid collision. Collision avoidance is performed by keeping the minimum distance between the robot and center of mass less than the minimum distance between the center of mass and the nearest boundary point. A linear proportional controller is used for the center of mass. Other feedback controls are designed such that the center of mass maintains a desired distance from the boundary and the robots maintain a desired shape. GPS is proven unnecessary in [9] given the robots have sensors to observe the behavior of themselves and of the other robots in the system.

The UUV-gas algorithm [4] can be used to perform boundary tracking comprised of circular motion. The focus of this multiple vehicle cooperative tracking is to prevent vehicle collision while having each vehicle following the same boundary. The circular motion in this algorithm is designed to allow the robot to travel only a set distance within the boundary region and outside the boundary region. Using circular motion only could prevent necessary coverage within the boundary region dependent on the application.

Recent work in area coverage allows single and multiple robots to offer complete coverage of an unknown area. To fully cover an unknown area, *simplices* can partition a 2D region to be covered by multiple robots [22]. The simplices give a path that robots can take to cover the entire unknown area. The path created does not take into account spatio-temporal sampling needs. Further, partitioning the area into separate paths for multiple robots to follow could improve efficiency.

In [6], Boustrophedon decomposition performs complete coverage of an area in single and multiple robot implementations. Adjacency graphs are used to keep track of cells. Cells are created when new areas are explored and obstacles are

reached. Multiple robot implementations create new cells once they no longer have cells to explore to prevent idling. During a single robot failure, this method allows other robots within a multiple robot system to continue coverage of the area; however, overlapping coverage does occur.

Instead of using an adjacency graph, coverage can be achieved using a spanning tree coverage algorithm [14]. On-line multiple robot spanning tree coverage breaks down the work area using cell decomposition. Groups of cells are made from larger cells until each robot is given 4 cells as the area to immediately cover. Robots explore the cells and broadcast their connection to each robot. When multiple robots attempt to assign themselves to the same group of cells, the robot with the least cell coverage is assigned. Spanning tree coverage allows for non-redundancy in most cases. If a robot fails, the area that the robot had covered along with the areas assigned to be done are cleared to be reassigned to other robots. This application involves covering the work area exactly once as opposed to surveillance applications where continuous area coverage is desired.

Coverage of a dynamic environment with obstacles and other robots can also be accomplished using a neural network [5]. Each neuron in the network is defined as an unclean area or an obstacle. The neural network is set to globally attract unclean areas while locally avoiding obstacles. Multiple robots each have their own shunting equation to be used for the neural network. Collision avoidance is also offered as each robot considers other robots to be obstacles.

Y. Guo and M. Balakrishnan [25] offers complete coverage of a bounded area with a car-like mobile robot. Complete coverage is performed using a minimum number of discs, representing sensor coverage of the vehicle, to partition the bounded area. The discs are covered by the robot using a neural network to determine the path to cover all discs. Each neuron is defined as being either

unclean, already cleaned, or an obstacle, where unclean areas are to be explored by the robot. Multiple robot systems are designed whereby the bounded area is divided into sub-regions for each robot.

Utilizing robots for a mobile sensor network [12] also deals with the problem of area coverage. Relationships between robots are defined by a Delaunay tessellation and a Voronoi diagram. The Delaunay tessellation defines properties between robots within one link of one another, and an adjacency matrix is used to specify the connectivity of the Delaunay tessellation. The Voronoi diagram is used to describe the coverage area of each robot along with the coverage area of the entire system. To maximize area coverage of the sensor network, robots must move using a continuous control law (Lloyd's algorithm) to reach the centroid of each Voronoi region. Fault tolerance is considered with the merging and splitting of Voronoi regions. This method maintains maximum area coverage for a group of robots as a sensor network. The robots move in formation to cover different areas in either an asynchronous or synchronous pattern, moving all Voronoi regions with it.

A reactive policy is used on a single unmanned aerial vehicle to cover a given area with all points (referred to as cells) being observed at the same time duration [16]. The single UAV heads toward the cell having the longest time since previously observed (age). When multiple cells have the same age, the robot moves toward the closest cell. A search pattern emerges displaying a spiral pattern along the area boundary heading inward, and then returning to the start position. A multi-agent reactive policy is applied for multiple robot scenarios. This policy acts in a similar pattern, whereby the robot heads toward the cell closest with the maximum age. Another multiple robot approach explored uses space decomposition (SD) to partition the area into regions of cells for cover-

age by each UAV. Genetic algorithms are used to determine AUV assignment to region. Results concluded both multiple UAV approaches valid, with SD being closer to an optimal solution in smaller robot applications.

Also, current research on spatio-temporal sampling offers the ability for a robot to maintain a relative location at set time intervals. In [7], phase balancing is used to maintain distance between a fleet of AUV gliders along a set path. Similar phase balancing can be used for spatio-temporal sampling. This system has all robots covering the entire boundary edge, causing overlap of the same location multiple times. If this is combined with workload balancing, the multi-robot system can efficiently partition the coverage of a boundary area along with improving spatio-temporal sampling of the boundary edge.

Though many of the previously discussed methods for multiple robot cooperative control are outlined in [18], the task of coverage with respect to spatio-temporal sampling is also discussed. To perform coverage spatiotemporal sampling, vehicles are allocated to regions to patrol. A region can be divided into polytopes that are assigned to a robot to each region. Coverage control can then be performed by minimizing a function that determines the sensing performance of a robot to a location and the importance of that location. Voronoi decomposition is used to determine a graph of the vehicles. The centroids of the Voronoi decomposition is then utilized in a linear control law to determine critical points for the cost function and give locally optimal coverage. Using Voronoi decomposition allows for robots to only need to communicate with their nearest neighbor for information.

The topics discussed above describe several methods for performing localization, boundary tracking, area coverage, and phase balancing. It is the combined efforts of these principles that can allow for balancing the area covered by mul-

tiple robots while following a boundary edge, keeping each robot at the same relative location along the boundary.

Chapter 3

PROBLEM DEFINITION

Consider a continuous edge segment E defined by two end points s_0 and s_n defined within a coordinate frame where the S_G axis follows the edge. The problem is to partition E into n sub-segments, each of which is allocated to one of n robots that must track the sub-segment. Hence, the i^{th} robot is designated to sample an interval $\Delta s_i = s_{i+1} - s_i$ along the boundary edge between s_i and s_{i+1}. An illustration of the proposed problem can be found in (Fig. 1.1).

To permit repeated sampling measurements over time, the i^{th} robot will travel from s_i to s_{i+1}, and back to s_i. This motion will constitute one *cycle*, where the location of the robot within this cycle is referred to as the *phase* ϕ_i and is measured in radians. The robot's phase ϕ_i relates to the position $s_{rob,i}$ along the S_G axis by:

$$\phi_i = \left\{ \begin{array}{ll} \pi \frac{s_{rob,i} - s_i}{\Delta s_i} & if \ \dot{s_i} > 0 \\ \pi \frac{s_{i+1} - s_{rob,i}}{\Delta s_i} + \pi & else \end{array} \right\} \tag{3.1}$$

In tracking E, it is desirable to balance both $Workload$ and $Phase$.

3.1 Workload

In this thesis, the workload is defined as the area covered by the robot as it traverses the edge. This area will be a function of the distance the robot travels perpendicular to the boundary edge (e.g. the depth under ice an AUV must travel to observe ice algae population density). Further, this perpendicular distance ($d_{a,i}$) can differ along the boundary edge. Hence, to balance the workload, the error to minimize is the difference between each robot's area covered Ψ_i:

$$\begin{aligned} e_{\Psi,i} &= \Psi_{i+1} - \Psi_i \\ &= \int_{s_{i+1}}^{s_{i+2}} d_{des}(s)ds - \int_{s_i}^{s_{i+1}} d_{des}(s)ds \\ &\approx d_{a,i+1}\Delta s_{i+1} - d_{a,i}\Delta s_i \end{aligned} \tag{3.2}$$

In eq. 3.3, the area covered by a robot is approximated as a rectangle and calculated as the product of average depth $d_{a,i}$ and Δs_i. While the d_{des} is a function of the robot position $s_{rob,i}$ along the edge, the boundary values s_i can be controlled by the robot itself. Workload balancing dynamically adjusts the workload of each robot in real time as it traverses the edge.

3.2 Phase

The second goal of this controller is to the improve the spatio-temporal sampling by minimizing the phase difference between AUVs. This will reduce the likelihood of missing a localized dynamic event. A localized dynamic event can occur at any location along the boundary at any given time. Maintaining equal distances within each robot partition permits the robots to be close enough where events are less likely to go unnoticed. Such dynamic events could be important

for observation and surveillance applications. Further, phase balancing allows for consistent sampling of the same location within its respective partition for all robots. The associated error to be minimized is:

$$e_{\phi,i} = \phi_{i+1} - \phi_i \tag{3.3}$$

3.3 Implementation

The controller designed must be able to function on an actual distributed multiple robot system. In (Fig. 3.1), multiple robots are aligned toward a boundary to be followed using sensors. Localization is performed with an external range sensor to place the robots within a global coordinate frame. The robots can communicate with each other and the external sensor via wireless communication method.

It is assumed that the field of view of the rangefinder is at a fixed location during the experiment. Also, the rangefinder must be capable of maintaining all of the robots within its field of view to sufficiently track the robots.

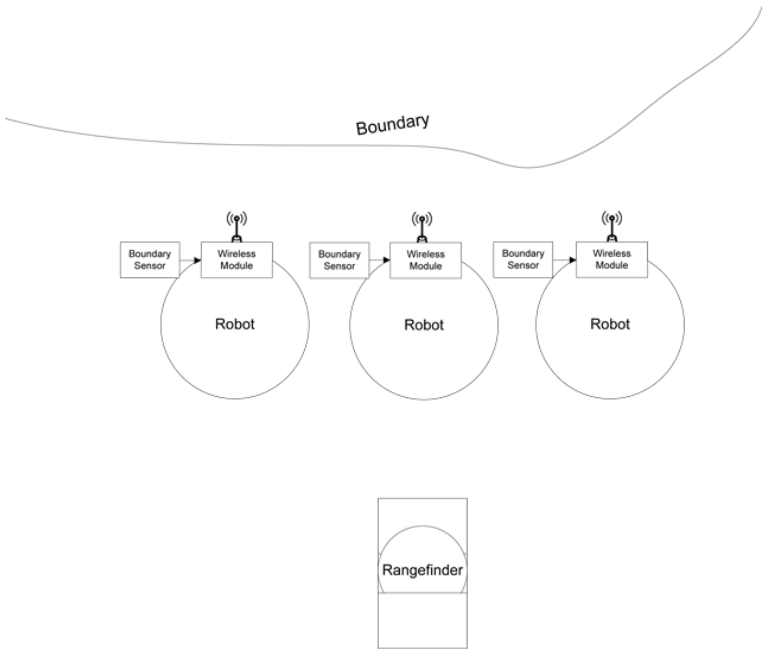


Figure 3.1: Distributed Multiple Robot System Performing Boundary Tracking

Chapter 4

CONTROLLER DESIGN

Since the boundary values s_i can be controlled by the robots themselves, consider the dynamics of the boundary values to be modeled as in eq. 4.1, with a proposed control input $U_{s,t}$ in eq. 4.3.

$$S_{t+1} = S_t + U_{s,t} \tag{4.1}$$

where

$$S_t = [s_0 \ s_1 \ ... \ s_n]_t \tag{4.2}$$

$$U_{s,t} = (0 \ K_s e_{s,0} \ ... \ K_s e_{s,n-2} \ 0)^T \tag{4.3}$$

To understand the error dynamics, consider a three AUV system in which $n = 3$. Considering eigenvalues of the transition matrix in eq. 4.4, the error dynamics can be proven stable for $K_s > 0$.

$$
\begin{aligned}
& E_{s,t+1} \\
&= \begin{pmatrix} e_{s,0} \\ e_{s,1} \end{pmatrix}_{t+1} \\
&= \begin{pmatrix} d_{a,0} & -d_{a,0} - d_{a,1} & d_{a,1} & 0 \\ 0 & d_{a,1} & -d_{a,1} - d_{a,2} & d_{a,2} \end{pmatrix} S_{t+1} \\
&= \begin{pmatrix} 1 - (d_{a,0} + d_{a,1})K_s & d_{a,1}K_s \\ d_{a,1}K_s & 1 - (d_{a,1} + d_{a,2})K_s \end{pmatrix} E_{s,t}
\end{aligned}
\tag{4.4}
$$

While the controller operates using proportional feedback to control boundary coverage (workload balance), a feedback linearization controller is used for robot location (phase balance). The following phase dynamics were used:

$$\Phi_{t+1} = \Phi_t + \delta\Phi_{s,i,t} + U_{\phi,t} \tag{4.5}$$

where

$$\Phi_t = [\phi_0 \ \phi_1 \ ... \ \phi_{n-1}]_t \tag{4.6}$$

As shown in eq 4.7, the proposed control input $U_{\Phi,t}$ for the t^{th} time step consists of several terms, the first of which incorporates the desired phase velocity $\dot{\phi}_{des}$ at which all robot's should maintain, once steady state is reached. Adding the second term $-\delta\phi_{s,i}$ implements feedback linearization to counter the change in phase caused by workload balancing, (i.e. change in s_i). The final term

$K_\phi e_{\phi,i} - K_\phi e_{\phi,i-1}$ is used to minimize phase error.

$$U_{\phi,t} = \begin{pmatrix} \dot{\phi}_{des}\Delta t - \delta\phi_{s,0} + K_\phi e_{\phi,0} \\ \dot{\phi}_{des}\Delta t - \delta\phi_{s,1} + K_\phi e_{\phi,1} - K_\phi e_{\phi,0} \\ ... \\ \dot{\phi}_{des}\Delta t - \delta\phi_{s,n-2} + K_\phi e_{\phi,n-2} - K_\phi e_{\phi,n-3} \\ \dot{\phi}_{des}\Delta t - \delta\phi_{s,n-1} - K_\phi e_{\phi,n-2} \end{pmatrix}_t \tag{4.7}$$

where

$$\delta\phi_{s,i} = \frac{s_{rob,i,t-1} - s_{i,t}}{\Delta s_{i,t}} - \frac{s_{rob,i,t-1} - s_{i,t-1}}{\Delta s_{i,t-1}} \tag{4.8}$$

The resulting error dynamics are shown below. For clarity, and without losing generalization, only the case with $n = 3$ is shown. This system is guaranteed stable if eigenvalues of the transition matrix are less than 1, requiring the stability condition $K_\phi < 2/3$.

$$\begin{aligned} & E_{\Phi,t+1} \\ &= \begin{pmatrix} e_{\phi,0} \\ e_{\phi,1} \end{pmatrix}_{t+1} \\ &= \begin{pmatrix} -1 & 1 & 0 \\ 0 & -1 & 1 \end{pmatrix} \Phi_{t+1} \\ &= \begin{pmatrix} 1-2K_\phi & K_\phi \\ K_\phi & 1-2K_\phi \end{pmatrix} E_{\Phi,t} \end{aligned} \tag{4.9}$$

While the phase cannot be controlled directly, it can be controlled indirectly through the robot's forward and rotational velocities. For example, inputting the controller into equation 4.5 can yield a desired phase. This phase can be tracked using a linear velocity controller in which the difference between the desired phase

and the actual phase of the vehicle are minimized (4.10).

$$v_i = K_v(\phi_{des,i} - \phi_i) \tag{4.10}$$

As expected, the K_v term is the proportional control gain.

Chapter 5

PARTICLE FILTER LOCALIZATION

A particle filter will be used to localize the robots within the global environment. Odometry measurements from the robots are used for robot particle propagation. Scans from a laser rangefinder are used to measure the robot location in a global frame.

Particle filter localization allows robots to determine their position within a global coordinate frame. Passive localization aids in the navigation of a robot through the environment while the vehicles carry out tasks autonomously. Particle filters work by creating a set of randomly generated entities. Each entity is an individual state estimate with a state ($x_{t,i}$, $y_{t,i}$, $\theta_{t,i}$) in the configuration space and a weight $w_{t,i}$ representing the probability of the belief state being the actual robot. Initially, N particles are randomly drawn from the configuration space and added to a set X_0. These particles are then iterated over the entire time of the experiment. At time t, the particle filter algorithm is performed (5.1).

$$
\begin{aligned}
&For\ i = 1 : N \\
&\quad Pick\ x_{t-1,i}\ from\ X_{t-1,i} \\
&\quad Draw\ x_{t,i}\ with\ probability\ p(x_{t,i}|x_{t-1,i}, o_t) \\
&\quad Calculate\ w_{t,i} = p(z_t|x_{t,i}) \\
&\quad Add\ x_{t,i}\ to\ X_{t,TEMP} \\
&For\ i = 1 : N \\
&\quad Draw\ x_{t,i}\ from\ X_{t,TEMP}\ with\ probability\ w_{t,i} \\
&\quad Add\ x_{t,i}\ to\ X_{t,i}
\end{aligned}
\tag{5.1}
$$

The particle filter is comprised of two main parts, a prediction and a correction step. The prediction step consists of propagating each particle forward and determining the likelihood that particle is the actual state of the robot. The correction step is then utilized to propagate the particles with the highest likelihood. The average state of all particles is thus an estimation of the actual state of the vehicle X_t.

The prediction step of this algorithm is essentially a Bayes Filter. It begins by propagating a given particle $x_{t-1,i}$ forward by adding the odometry measurement plus random gaussian noise to the original state (5.2). The random gaussian noise gives the particle error to distribute the particles around where the robot believes it to be. This is to account for sensor noise that may exist within the system.

$$x_{t,i} = x_{t-1,i} + (o_t + N(\mu = 0, \sigma = 1)) \tag{5.2}$$

The perceptual probability is then determined (5.3). A unimodal (Gaussian) distribution of mean μ, expected range for the particle, and variance, from sensor data, is utilized in order to determine the probability density function for a given

measurement. These weights are normalized using η to transform the weight into appropriate probabilities.

$$w_{t,i} = \eta N(\mu_{t,i}, \sigma_{t,i}) bel(x_{t,i}) \tag{5.3}$$

The correction step selects particles by first randomly selecting a value between 0 and the sum of all weighted values. The particles are iterated through, adding their weights, until the sum is greater than the previously selected value. Particles that satisfy this condition are propagated. This method ultimately gives favor to particles that are closer to the actual robot state having a higher weight.

Chapter 6

EXPERIMENT

The distributed control system was implemented within MATLAB, and tested with a MATLAB simulator as well as with actual robots. In both cases, experiments were designed to represent a system of multiple Autonomous Underwater Vehicles (AUVs) tracking and sampling the underside of an ice raft edge, where ice algae commonly grows. Figure 6.1 shows the testbed design (to be discussed in further detail later) with the global coordinate frame.

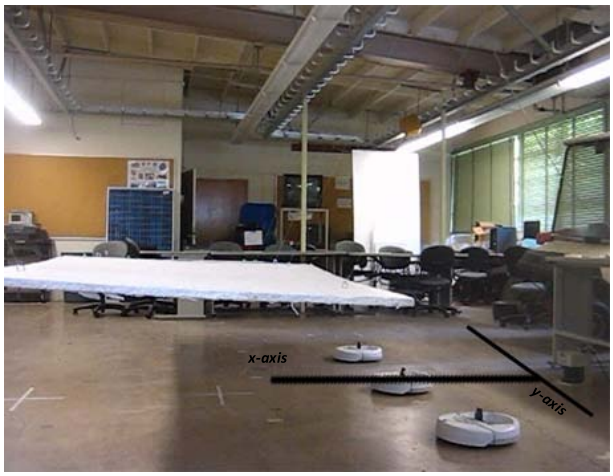


Figure 6.1: Experiment setup: Three iRobot® Creates before navigating under a simulated ice edge with global axes displayed.

To traverse the boundary edge, each robot uses a repeated series of motions that result in a lawnmower pattern that follows the edge. This series of motions includes 1) the robot moving forward until detecting the entering ice edge using upward facing range sensors, 2) driving forward under the ice as long as the presence of algae is still detected, 3) completing a 180 degree turn along a circular arc, 4) driving forward until leaving the ice edge is detected, and 5) completing another 180 degree turn along a circular arc. Throughout these motions, each robot adjusts its forward and rotational velocity to track a desired phase (see equation 4.10). To note, if the robot has reached the limit of the edge segment defined by s_i and s_{i+1}, it will change its boundary edge traversal direction.

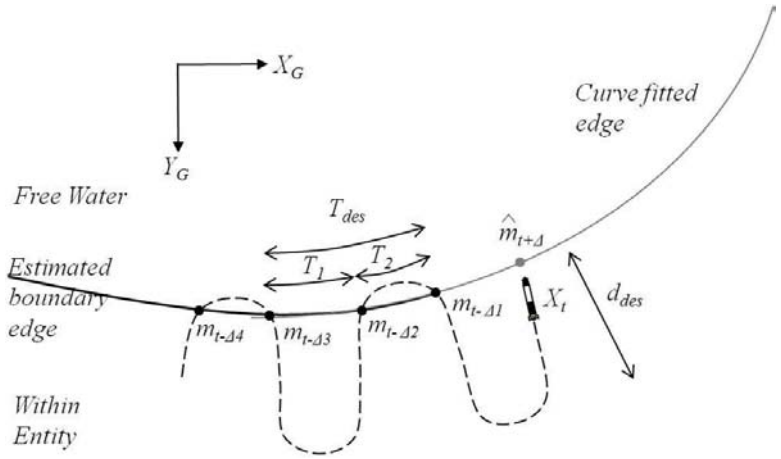


Figure 6.2: Desired States of Robot Crossing Patrolling Axis

Figure 6.2 illustrates the movement of the robot across the patrolling axis, continuing forward until there are no longer algae, turning around, continuing past the patrolling axis, turning around again, and then repeating the pattern. The numbers on the image refer to these different states. Each state is assigned relative to the current position of the robot.

6.1 Implementation

6.1.1 Simulation

MATLAB scripts and functions are used to test the functionality of the distributed multiple robot control system. Using the equations in the theory section, the multi-robot system can be used on an assortment of maps to evaluate the system under various conditions. Using a simulator offers ease in troubleshooting and analysis in different scenarios and environments before adapting theory to the real-world application.

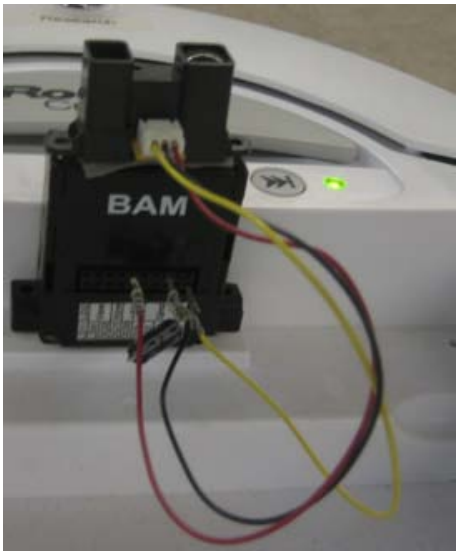
The locations of the simulated algae are represented in maps using booleans to determine if there exists algae in a discretized grid. Rounding functions are used to determine if a robot is located within a grid that contains algae. The other type of map created determines the location of the simulated ice raft. Values of either 0.8 or 1 refer to whether the ice raft is present or not, respectively.

6.1.2 Hardware

Real-world testing is vital to observe the ability of actual robots to carry out the desired task. Three iRobot® Create robots are used for the robot implementation of the control system. These robots have a differential drive system with a stabilizing wheel. Odometry measurements are made via wheel encoders located on the servos for the main wheels of the robot. The robots have been outfitted with an upward facing infrared sensor to determine the presence of the boundary edge. This is attached to a bluetooth® communication module that replaces the standard serial communication cable.

A flow diagram of the hardware system can be seen in (Fig. 6.4).

(a) (b)



(c)

Figure 6.3: The iRobot® Create robot is shown from above in (a), and below in (b). The bluetooth module and IR sensor added to the robot can be found in (c).

The Create robots utilize the same MATLAB controller application as the simulator. Modifications to the MATLAB application include initializing and operating multiple sensors and running serial communication. The robots communicate with a computer running the MATLAB program via bluetooth® wireless communication that is converted to a serial interface. The robots receive wheel velocities from the computer and return odometry and IR measurements. The computer also collects scans of data from a SICK Laser rangefinder con-

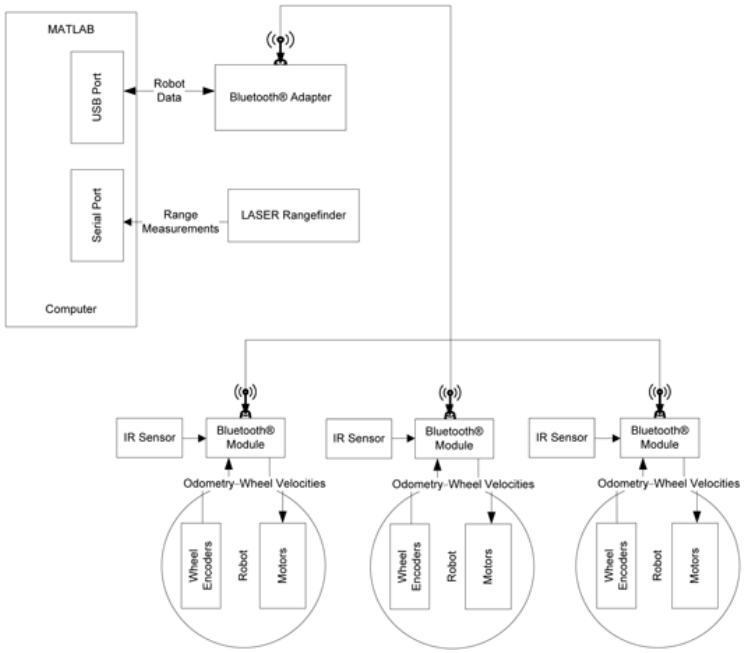


Figure 6.4: Hardware Flow Diagram

nected via serial interface. The main computer may act as a centralized system, however, the program architecture is decentralized.

6.1.3 Software

The main algorithm (Fig. 6.5) implemented in MATLAB operates for each robot. The robots first acquire measurements from their IR sensors. This information determines whether the vehicles are crossing the boundary edge. If the border edge is reached, the robots must alternate their direction to continue following the boundary. If the boundary has been discovered by all robots, phase and workload balancing occurs and the appropriate control laws are utilized to determine the wheel velocities. Particle filter localization determines the global location of each robot in the environment. Figure 6.6 performs the workload and phase balancing for the MATLAB program. The area covered by the robot is

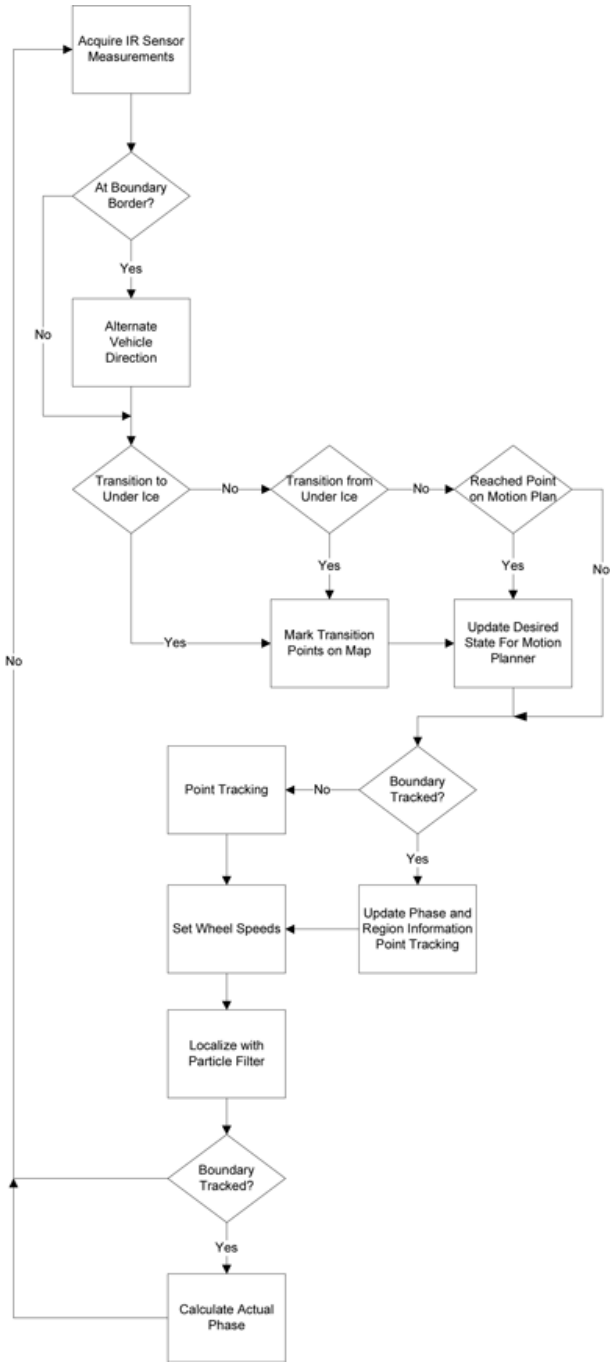


Figure 6.5: System Level Algorithm

determined. The phase of the robot is then found and the error between desired and actual phase is determined. The same occurs for the region the robot is to cover. Feedback linearization occurs to counter the phase change caused by

workload balancing (as discussed earlier). Finally the desired phase and resulting phase error are calculated for the linear proportional control.

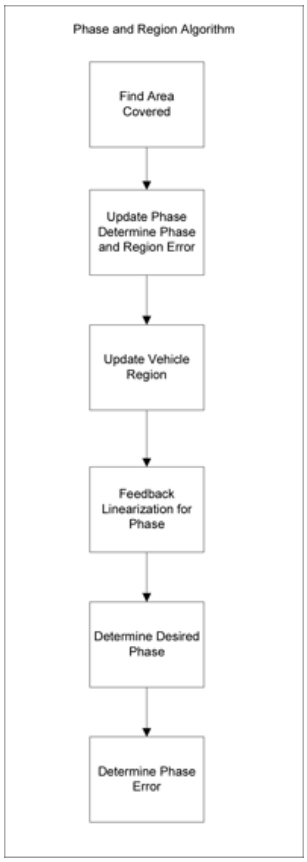


Figure 6.6: Workload/Phase Balancing Algorithm

6.1.4 Environment

The testbed environment for the robots consists of an empty room with a foam board hanging from the ceiling. The foam board is approximately 4m in length and hangs approximately 0.8m above the ground.

The robots are initially placed 1m from the foam board's edge, and are aligned with each other at varying distances parallel to the S_G axis. IR sensors on the robots detect the boundary edge based on the difference in IR height measure-

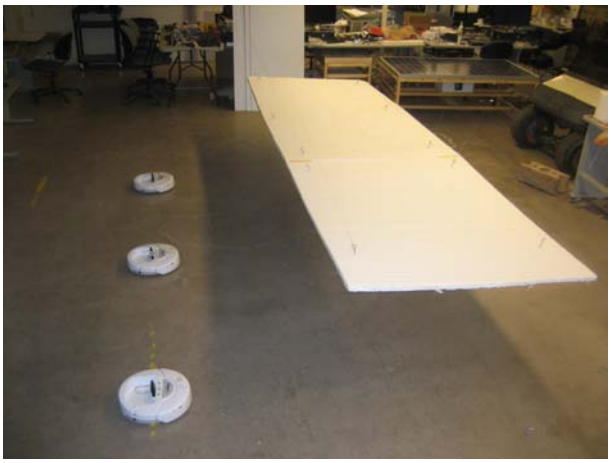


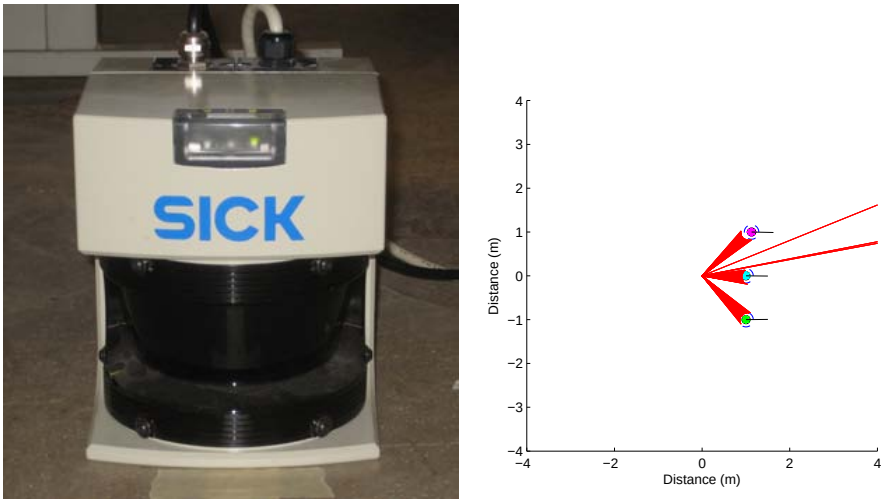
Figure 6.7: The hardware setup: Three iRobot® Creates navigated underneath an overhanging foam block used to simulate an ice raft.

ments between the ceiling and the foam board. Four different experiments were performed as in simulations. The first experiment begins with robots first reaching the foam's edge in phase with each other. The next experiment is initialized with the first robot 72 degrees out of phase with the other two robots. Another experiment begins with the middle robot 72 degrees out of phase with the other two robots. The final experiment sets the third robot 72 degrees out of phase with the other two robots. The results from this data should illustrate that the robots can recover from being out of phase quickly while sampling the underside of the overhanging foam block and evenly disperse the workload between them.

6.1.5 Particle Filter

The particle filter utilizes a SICK Laser rangefinder (6.8a) placed 1m behind the robots, and centered along the boundary edge. This sensor has a scanning angle of 180 degrees with a resolution of 0.5 degrees and maximum distance set to 8m. The SICK Laser rangefinder combined with the odometry measurements from the robots themselves, create the basis for the particle filter localization

used (6.8b).



(a) (b)

Figure 6.8: The SICK Laser rangefinder is shown in (a). A scan of the robots at their initial position with particle filter localization is seen in (b).

Since multiple samples of the robot are taken from a single scan, these measurements need to be translated into a single decisive point that would designate the robots' center, of which all other localization data is based. To simplify this calculation, the shortest sample in a scan where samples curved along the robot was taken. The known radius of the robot was added to the range of the sample and the bearing was maintained. This estimated the location of the robot center.

Further, the accuracy of the particle filter was tested prior to implementation with phase and workload balancing (Fig. 6.9). To test the accuracy in the x-direction (direction perpendicular to the boundary), tape was placed exactly 1m from the starting position of the robot. The robot is then programmed to drive past the tape. Video is recorded of the vehicle driving over the tape. The timestamp from the video is then used to determine the time at which the robot believes it passed over the tape. Five runs are performed, and the average error

Figure 6.9: Particle Filter Test: iRobot® Create moving toward location of known distance (tape).

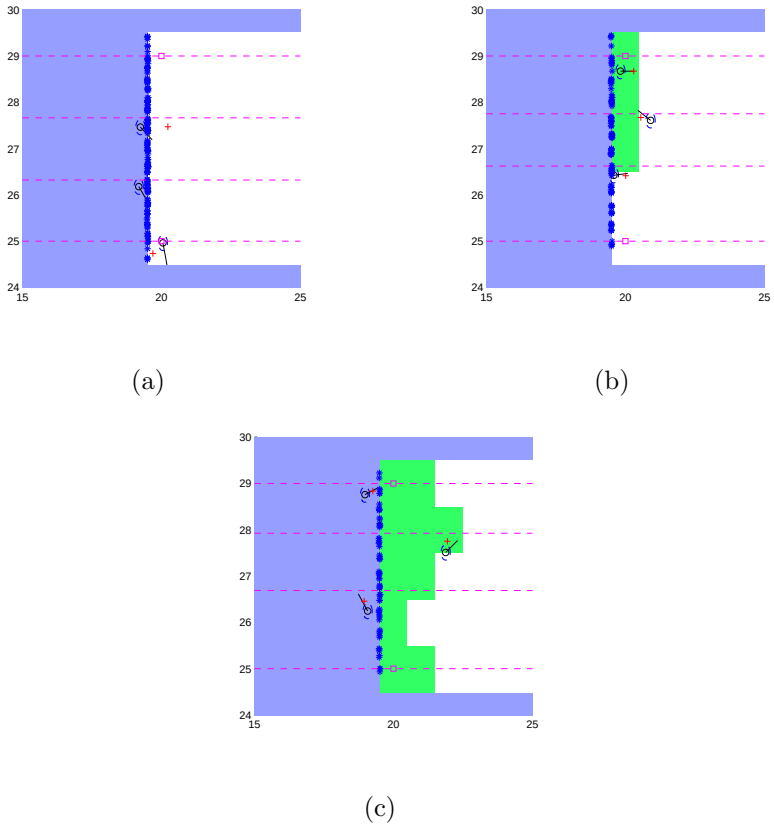
was found. The same test is then performed in the y-direction (parallel to the boundary). The average error in the x-direction was 6.73cm and 6.88cm in the y-direction. This error is believed to be negligible for the sake of this application.

6.2 Experiment Configuration

Three different scenarios were simulated, each with a different algae population. The first scenario used has no algae. In this scenario, each robot will travel an equal distance underneath the ice and return out. Here the boundaries should remain equal and all robots should remain in phase. Figure 6.10a shows the simulated environment. The white area represents the ice and the blue area is the water surrounding the ice raft.

The next scenario has a large amount of algae across half of the ice raft, and no algae across the other half. For this simulation, one robot will be completely submerged in the algae-side as well as half of another robot's boundary. The last robot will only have the axis to patrol. In this case, the boundaries along the

patrolling axis should be significantly shorter for one robot, longer for the second robot, and longest for the robot without algae to observe. The algae in (Fig. 6.10b) is illustrated by the green areas.



(a) (b)

(c)

Figure 6.10: The Simulator GUI: A top down view of three robots (circles) navigating water (blue) and under ice (white). The algae (green) covers part of the under ice edge in some scenarios. In (a), there is no algae on ice edge. In (b), algae is covering half of ice edge. Scenario simulation (c) has a random coverage of algae.

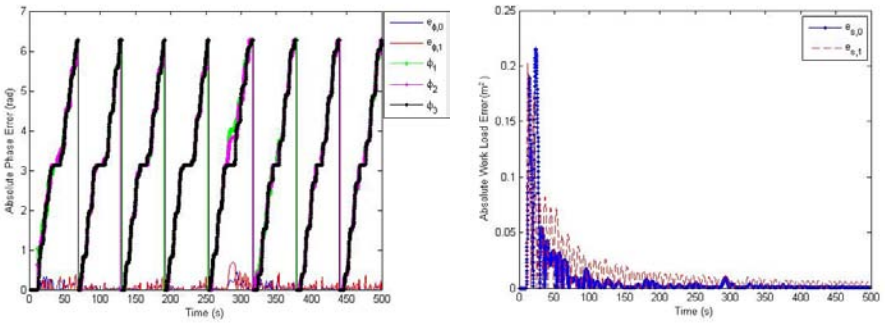
Finally, the third scenario involves algae growing to random lengths from the ice edge, (Fig. 6.10c). To test the phase balancing aspect of this experiment, the robots were first placed in phase with one another. Then, the next three test cases involve initially placing one of the robots 72 degrees out of phase from the other robots. The goal is then for the robots to return to a state of equilibrium with each other while balancing the workload between them.

Chapter 7

RESULTS

7.1 Simulation

In the first simulated test scenario, the three robots were initially in phase with each other and followed the boundary of a simulated ice edge with no algae. In Fig. 7.1, the error in phase and workload is illustrated. In order to compare this nominal error, the actual phase of each robot is displayed in the figure.



(a) (b)

Figure 7.1: Simulation results for the case with no algae under the ice and the three robots starting in phase. The phase error is plotted in (a) and the workload error is plotted in (b).

It can be seen in Fig. 7.1 that the phase remains constantly tracked. Also, the workload does adjust slightly and the error approaches zero.

The other simulations gave similar results. The steady state error for the phase can be found in Table 7.1. The steady state error differs between scenarios, but it always decreases over time. This method lends itself to a worst case average of 4.1% steady state error.

Table 7.1: Phase Steady State Error (*radians*)

Ice Edge	All Robots Initially In Phase	Robot 1 Initially Out of Phase	Robot 2 Initially Out of Phase	Robot 3 Initially Out of Phase
No Algae	0.073	0.093	0.073	0.073
Half Algae	0.169	0.189	0.206	0.207
Random Algae	0.222	0.240	0.257	0.238

The steady state error for the workload is in Table 7.2. The steady state error does not increase significantly between no algae and half algae scenarios. However, the random algae scenario shows a significant increase compared to the other scenarios. Despite this increase, the steady state error remains only a few cm^2.

Table 7.2: Workload Steady State Error (m^2)

Ice Edge	All Robots Initially In Phase	Robot 1 Initially Out of Phase	Robot 2 Initially Out of Phase	Robot 3 Initially Out of Phase
No Algae	0.010	0.013	0.015	0.012
Half Algae	0.015	0.015	0.016	0.019
Random Algae	0.037	0.036	0.043	0.043

7.2 Robot Implementation

7.2.1 Odometry

Three iRobot® Creates were tested to track the boundary of a 4m long foam board without any simulated algae. Figure 7.2 displays the resulting boundary following with robot trajectories.

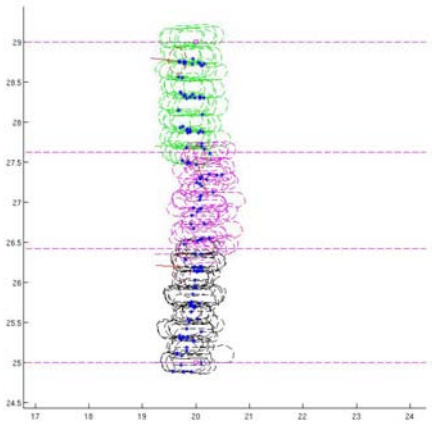


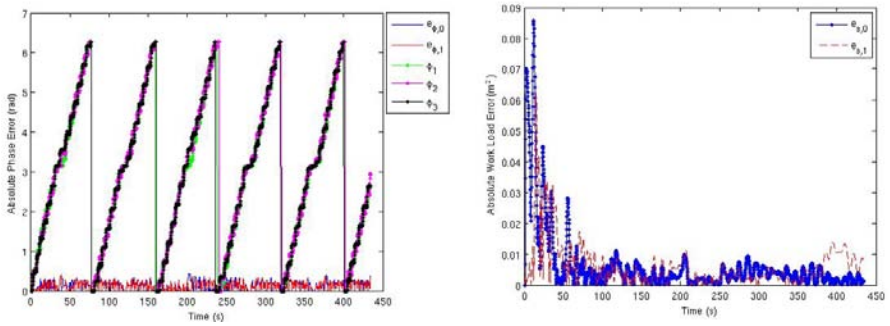
Figure 7.2: Trajectories of iRobot® Creates in following the simulated ice raft (i.e. the overhanging foam block.)

In Fig. 7.3, the error in phase and workload is displayed. In order to compare this nominal error, the actual phase of each robot is displayed in the figure.

It can be seen in Fig. 7.3 that the phase remains constantly tracked. Also, the workload does adjust slightly and the error reduces to zero just after one cycle of the robot between its boundaries.

Other cases produced similar results. Table 7.3 displays the steady state error for the phase in all cases. In the worst case the steady state error remains below 2.5%.

The steady state error for the workload is also seen in Table 7.3. The steady



(a) (b)

Figure 7.3: Experimental results for the case with no algae under the ice and the three iRobot® Create robots starting in phase. The phase error is plotted in (a) and the workload error is plotted in (b).

Table 7.3: Experiment Steady State Error

Steady State Error	All Robots Initially In Phase	Robot 1 Initially Out of Phase	Robot 2 Initially Out of Phase	Robot 3 Initially Out of Phase
Phase ($radians$)	0.129	0.1474	0.1381	0.152
Workload (m^2)	0.014	0.018	0.016	0.015

state error for the workload remains constant. This makes sense since the workload should not have to change across the various cases.

7.2.2 Particle Filter Localization

As with odometry, three iRobot® Creates tracked the boundary of a 4m long foam board without any simulated algae. These robots utilized a particle filter to localize themselves within their global area. Figure 7.4 displays the resulting boundary following with robot trajectories.

In Fig. 7.5, the error in phase and workload is displayed. The actual phase

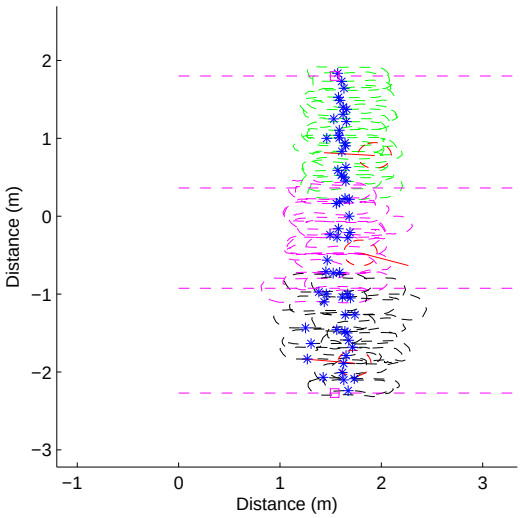
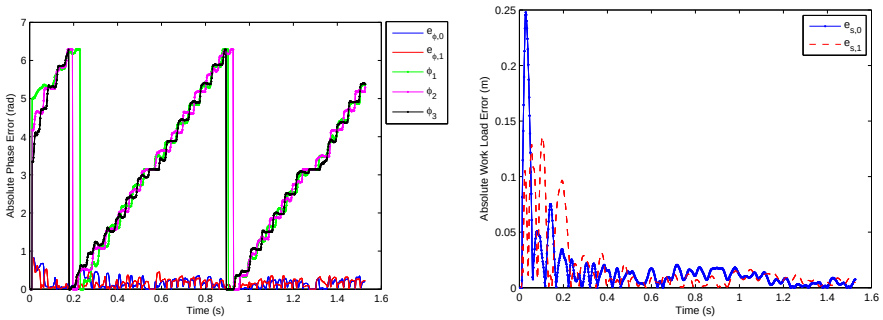


Figure 7.4: Trajectories of iRobot® Creates using Particle Filter Localization

of each robot is displayed in the figure to illustrate the nominal error.



(a) (b)

Figure 7.5: Experimental results for the case with no algae under the ice and each robot starting 1m apart from one another. The phase error is plotted in (a) and the workload error is plotted in (b).

It can be seen in (Fig. 7.5) that the phase and workload are properly balanced as it did when the robots used only odometry for localization. This proves the ability to globally localize the robots in an environment passively while still having the capability to perform accurate phase and workload balancing.

Chapter 8

CONCLUSION

This work presents a distributed boundary edge tracking controller for multiple robot systems. The controller balances workload and phase. It balances workload with a proportional controller that adjusts the boundaries that each robot works within in order to appropriately disburse the total coverage area between each robot. It balances the phase using a feedback linearization controller that allows robots to match their edge traversal progression within their individual boundaries.

The controller is provably stable to drive differences between robot workloads and phase differences to zero. This was demonstrated with simulations of a three robot system. The simulations tested conditions where simulated algae coverage covered half of the map at the same depth, all of the map at random depths, and none of the map at all. The simulations also tested these maps against robot locations having various phase differences with respect to one another. Real-world experiments with three iRobot® Create robots displayed similar results. The robot implementation illustrated the ability to perform boundary tracking while performing phase and workload balancing by testing the robot traversal of

varying distances underneath the simulated ice raft as well as varying the initial phases of the robots with respect to one another. Steady state error remained only a few percent in the worst cases among all scenarios.

Particle filter localization is applied to the multiple robot system. Particle filter localization allowed the vehicles to be passively localized within a global coordinate frame. Odometry only methods fail to account for measurement drift and thus display inaccurate tracking information. The particle filter increased the accuracy of the robots' map of the ice edge through localizing the robots and accounting for process noise. Running the particle filter on the multiple robot system permitted the robots to perform boundary tracking while balancing their workload and phase.

Chapter 9

FUTURE WORK

Future work for this thesis will first consist of an elaboration of the current testbed. Due to space and time constraints, the robot implementation was unable to test full workload balancing through adjusting the location of simulated algae (as was performed in MATLAB simulation). In order to test workload balancing in area coverage on the current robot implementation, additional foam board could be added to the bottom of the current hanging foam board. The IR sensor can then be calibrated to decipher the lower areas as coverage areas as opposed to solely being underneath the boundary. Currently, the distributed system is tested using a centralized computer. However, in order to accurately test the scalability of the system, it is necessary to build a proper distributed system. The current computer can barely handle the coordination of all three robots. Giving each robot a microcomputer to perform only the algorithm for that robot, should allow the system to operate smoother and be scaled to as many robots as is necessary. Also, the particle filter on the current system could use further adjustment. One such modification would be to make the particle filter localization into a PF SLAM algorithm. Combining the mapping of the boundary edge with robot

localization can increase accuracy and reduce complexity through integration.

Figure 9.1: OceanServer Iver2 AUVs can be deployed in the Arctic

Other work will ideally involve placing these controllers on Autonomous Underwater Vehicles (AUVs) deployed in Arctic expeditions where ice is abundant. This system can be designed to deploy on multiple OceanServer Iver2 AUVs (Fig. 9.1). These AUVs prove themselves useful for environmental monitoring and ideal for coastal applications. The distributed system can be placed on computers on each robot, using GPS and Sonar, along with acoustic modems for communication.

A key difficulty with AUV localization is underwater positioning error. GPS measurements can be taken only when the vehicle is surfaced, which can allow for measurement error propagation. Utilizing sensors to detect the locations of other robots could prove useful for a multiple robot particle filter to increase position accuracy. Further, beacons may also be able to be utilized with sensor fusion for global localization of multiple underwater vehicles.

Utilizing phase and workload balancing with large numbers of these autonomous underwater vehicles can one day allow for the exploration, and understanding for the growth of harmful algae blooms. An understanding of this knowledge can further research to repair polar ecosystems that are damaged by such algae. Similarly, this same research can aid in exploring and observing the

growth of indigenous algae that is dying out due to climate changes.

Bibliography

[1] A. Joshi, et. al., "Experimental Validation of Cooperative Environmental Boundary Tracking with On-board Sensors", *in American Control Conference*, pp.2630-2635, June 2009.

[2] A.W. Stroupe, et. al., "Distributed Sensor Fusion for Object Position Estimation by Multi-Robot Systems", *Proceedings of the IEEE International Conference on Robotics & Automation* , pp. 1092-1098, 2001.

[3] B. Chow, et. al., "Assigning Closely Spaced Targets to Multiple Autonomous Underwater Vehicles", *Proceedings of the Unmanned Untethered Submersible Technology*, 2009.

[4] C.H. Hsieh, et. al., "Experimental validation of an algorithm for cooperative boundary tracking", *Proceedings of the American Control Conference*, pp. 1078-1083, 2005.

[5] C. Luo and S.X. Yang, "A Real-Time Cooperative Sweeping Strategy for Multiple Cleaning Robots", *Proceedings of the IEEE International Symposium on Intelligent Control*, pp. 660-665, 2002.

[6] C.S. Kong, et. al., "Distributed Coverage with Multi-Robot System", *Pro-*

ceedings of the IEEE International Conference on Robotics & Automation, pp. 2423-2429, 2006.

[7] D.A. Paley, et. al., "Cooperative Control for Ocean Sampling: The Glider Coordinated Control System", *IEEE Transactions on Control Systems Technology*, vol. 16, no.4, pp.735-744, July 2008.

[8] D. Kogan and R.M. Murray, "Optimization-Based Navigation for the DARPA Grand Challenge", *Conference on Decision and Control*, pp. 1-6, 2006.

[9] F. Zhang and S. Haq, "Boundary Following by Robot Formations without GPS", *Proceedings of the IEEE International Conference on Robotics & Automation*, pp. 152-157, 2008.

[10] H. Majeed, et. al., "A Cost Efficient Design for an Autonomous Underwater Vehicle Capable of Localizing and Navigating within a Bounded Body of Water", *International Conference on Computational Intelligence for Modeling Control and Automation*, pp. 987-992, 2008.

[11] J. Chen and L. Li, "Path Planning Protocol for Collaborative Multi-Robot Systems", *in IEEE International Symposium on Computational Intelligence in Robotics and Automation*, pp.721-726, June 2005.

[12] J. Tan, et. al., "Modeling Multiple Robot Systems for Area Coverage and Cooperation", *Proceedings of the IEEE International Conference on Robotics & Automation*, pp. 2568-2573, 2004.

[13] M. Peasgood, C.M. Clark, and J.McPhee, "Localization of Multiple Robots with Simple Sensors", *Proceedings of the IEEE International Conference on Mechatronics & Automation*, pp. 671-676, 2005.

[14] N. Hazon, et. al., "Towards Robust On-line Multi-Robot Coverage", *Proceedings of the IEEE International Conference on Robotics & Automation*, pp. 1710-1715, 2006.

[15] N. Kim, et. al., "Intelligent Navigation and Control of an Autonomous Underwater Vehicle based on Q-Learning and Self-organizing Control", *ICROS-SICE International Joint Conference*, pp. 630-634, 2009.

[16] N. Nigam and I. Kroo, "Persistent Surveillance Using Multiple Unmanned Air Vehicles", *IEEE Aerospace Conference* , pp. 1-14, 2008.

[17] R.G. Brown, P.Y.C. Hwang, *Introduction to Random Signals and Applied Kalman Filtering*, 3rd. ed. Hoboken, NJ: John Wiley & Sons, 1997.

[18] R.M. Murray, "Recent Research in Cooperative Control of Multivehicle Systems", *Transactions of the ASME Journal of Dynamic Systems Measurement and Control*, pp. 571-583, 2007.

[19] R. Siegwart, I.R. Nourbakhsh, *Introduction to Autonomous Mobile Robots*. Cambridge, MA: Bradford, 2004.

[20] S. Barkby, et. al., "Incorporating Prior Maps with Bathymetric Distributed Particle SLAM for Improved AUV Navigation and Mapping", *OCEANS*, pp. 1-7, 2009.

[21] S. Charifa and M. Bikdash, "Adaptive boundary-following algorithm guided by artificial potential field for robot navigation", *IEEE Workshop on Robotic Intelligence in Informationally Structured Space*, pp.38-45, May 2009.

[22] S.S. Ge, et. al., "Boundary Following and Globally Convergent Path Planning Using Instant Goals", *IEEE Transactions on Systems, Man, and Cybernetics*, vol.35, no.2, pp.240-254, April 2005.

[23] S. Thrun, W. Burgard, D. Fox, *Probabilistic Robotics*. Cambridge, MA: MIT Press, 2006.

[24] X. Wang and V. Syrmos, "Coverage Path Planning for Multiple Robotic Agent-Based Inspection of an Unknown 2D Environment", *17th Mediterranean Conference on Control and Automation*, pp. 1295-1300, June 2009.

[25] Y. Guo and M. Balakrishnan, " Complete Coverage Control for Nonholonomic Mobile Robots in Dynamic Environments", *Proceedings of the IEEE International Conference on Robotics & Automation* , pp. 1704-1709, 2006.