1-7-2011

Cluster Space Gradient Contour Tracking for Mobile Multi-robot Systems

Thomas Adamek
Santa Clara University

SANTA CLARA UNIVERSITY

Department of Mechanical Engineering

Date: January 7, 2011

I HEREBY RECOMMEND THAT THE THESIS PREPARED UNDER MY SUPERVISION BY

Thomas Adamek

Cluster Space Gradient Contour Tracking for Mobile Multi-robot Systems

BE ACCEPTED IN PARTIAL FULFILLMENT OF THE REQUIREMENTS FOR THE DEGREE

OF

MASTER OF SCIENCE IN MECHANICAL ENGINEERING

_____

Christopher Kitts, Thesis Advisor

_____

Timothy Hight, Chairman of Department

Cluster Space Gradient Contour Tracking for Mobile Multi-robot Systems

By

Thomas Adamek

GRADUATE MASTER THESIS

Submitted in Partial Fulfillment of the Requirements
For the Degree of Master of Science
in Mechanical Engineering

Santa Clara, California

Cluster Space Gradient Contour Tracking for Mobile Multi-robot Systems

Thomas Adamek

Department of Mechanical Engineering
Santa Clara University
2011

Abstract

Multi-robot systems have the potential to exceed the performance of many existing robotic systems by taking advantage of the cluster's redundancy, coverage and flexibility. These unique characteristics of multi-robot systems allow them to perform tasks such as distributed sensing, gradient climbing, and collaborative work more effectively than any single robot system. The purpose of this research was to augment the existing cluster space control technique in order to demonstrate effective gradient-based functionality, specifically, that of tracking gradient contours of specified concentration levels. To do this, we needed first to estimate the direction of the gradient and/or contour based on the real-time measurements made by sensors on the distributed robots, and second, to steer the cluster in the appropriate direction.

Successful simulation, characterization, and experimental testing with the developed testbed have validated this approach. The controller enabled the cluster to sense and follow a contour-based trajectory in a parameter field using both a kayak cluster formation and also the land based Pioneer robots. The positive results of this research demonstrate the robustness of the cluster space control while using the contour following technique and suggest the possibility of further expansion with field applications.

Keywords: Cluster Space Control, Autonomous Surface Vessel

Acknowledgements

First and foremost, I would like to thank my advisor, Dr. Chris Kitts, for his guidance, support, encouragement, and patience throughout this research. His dedication, professionalism and willingness to help and educate is unsurpassed.

I would also like to thank everyone in the SCU Robotic Systems Lab, both past and present. They are the ones who spent countless hours trouble-shooting, coding, and researching which laid the foundation for what we have today. Much of this thesis has been built directly on their dedicated work.

A special thanks to my friend and my colleague Vincent Howard. He has worked alongside me on this project and has made many invaluable contributions. None of this would have been possible without his assistance. A mention needs to go out to the people who created and operated the testbeds: Paul Mahacek and Steve Li. Thanks.

Just as importantly, I'd like to extend my appreciation of the children across the street, who made me laugh with those crazy birthday egg fights and reminded me when it was time to go have some fun. A big thank you also to all the other lab mates and family members who have helped out and supported me along the way.

This work has been sponsored, in part, by the funds thru National Science Foundation Grant No. CNS-0619940, NOAA Grant NA03OAR4300104 (via subaward UAF 05-0148 from the University of Alaska Fairbanks), and funding from the Santa Clara University Technology Steering Committee. Any opinions, findings, conclusions or recommendations expressed in this material are those of the authors and do not necessarily reflect the views of the National Science Foundation.

Table of Contents

List of Figures

1.0 Introduction

As technology rapidly develops, robots are improving, and they can offer many advantages to accomplishing tasks given their strength, speed, precision, repeatability, and ability to withstand extreme environments. While most robots perform tasks in an isolated manner, there is a growing interest in the use of collaborative multi-robot systems to enhance performance in current applications and to develop new capabilities [1]. This concept has applications ranging from remote and *in situ* sensing to the physical manipulation of objects, and the domains for such applications include land, sea, air, and space [2].

For some applications, multi-robot systems may be more adept to specific tasks due to superior workload distribution, redundancy, increased coverage, system robustness and cost effectiveness. The ability of multi-robot systems to distribute tasks among several robots allows them to perform tasks much more quickly than any single robot. By dividing the workload, a group of robots surveying a workspace can map a larger area in order to collectively create a comprehensive map in a fraction of the time of any single robot. This speed is due to the ability of multi-robot systems to perform multiple tasks in numerous locations simultaneously.

Another advantage of spatially distributed robots is that data can be collected simultaneously on a large scale which permits the characterization of the target area in the form of a real time synoptic map. This snapshot of parameters over an area is a true representation of the instantaneous value of the parameter field, something that is not possible to do using a single robot that maps the area over time as it moves through the work space. Using the appropriate navigation algorithms, the cluster of robots can complete the distributed sensing tasks efficiently. Through this data gathering technique and interpolation, maps can be formed, targets located and areas searched.

One application of a large multi-robot system is the Centibots Project [3] done at SRI International. The Centibots Project employed 100 autonomous robots to cooperatively map an area and subsequently track objects within the mapped area. An initial group of

mapping robots was used to collectively build and share a distributed map. Following the mapping stage, a smaller group of tracking robots was deployed within the known area to gather data on specific objects of interest. The system successfully demonstrated the ability of a multi-robot system to effectively collaborate through the sharing of communication and data. Figure 1.1 illustrates the Centibots system during a mapping and tracking demonstration.

Figure 1.1-*Centibot Project area mapping [3]*

Another example of real world multi-robot systems is a fleet of autonomous underwater

gliders that was demonstrated during sea trials in Monterey Bay in August 2003. Each vehicle was equipped with sensors for observing the marine environment. The group served as a mobile sensor network capable of monitoring and collecting data in large areas. In the case when the mobile sensor network was to be used to sample the physical and/or biological variables in the water, the range of relevant spatial and temporal scales can be dramatic [4] as Fiorelli et al. demonstrated with the cooperative control of marine robots.

There are still many challenges that need to be addressed in order to field cost-effective multi-robot systems. These challenges include inter-robot communication, relative position sensing and actuation, control paradigms appropriate to real-time multisystem control, interfaces allowing efficient human direction/supervision of these systems, and design approaches supporting the economical production of such systems [1].

1.1 Multirobot Formation Control

The objective of a formation control strategy is to maintain the relative positions of a group of robots. There are different approaches of achieving the specified goal. A wide variety of techniques have been used and continue to be explored, such as centralized vs decentralized approaches, bio-inspired strategies and potential-field based concepts.

The "Follow the Leader" technique is widely used in controlling robot groups in a decentralized manner, and has been adopted by many researchers [5],[6],[7]. In this strategy, one robot is assigned the lead and the other robots are required to maintain set distances and/or bearings from the leader. This method uses feedback linearization to exponentially stabilize the relative distance and orientation of the follower. Follow-the-Leader-techniques require that the leader robot always remains functioning, and if anything happens to that leader, the group may encounter difficulties. In spite of this deficiency, the Follow-the-Leader-technique approach is particularly valued because of its simplicity and scalability

Biologists who have studied the behavior of animal aggregations as seen in schools of

fish and flocks of birds have observed that complex group behavior can emerge from simple animal rules. Such behaviors include techniques for more effective food foraging, and increased energy efficiency for locomotion. In minnows and goldfish, the time to find a patch of food was greatly decreased by working in groups[8]. These are seen as decentralized formation techniques. These techniques have been used in the "Nearest Neighbor Approach' introduced by J. G. Skellam. It is an example where the ratio of expected and observed mean value of the nearest neighbor distances is used to determine if a data set is clustered [9].

Another method of robot formation control has been used in the Autonomous Ocean Sampling Network (AOSN) that is under development at Monterey Bay Aquarium Research Institute (MBARI) [5]. This is an example of a multi-robot system used for adaptive ocean sampling. An adaptive formation control technique has been developed and tested on multiple autonomous underwater gliders [10]. The system uses three-dimensional multi-robot formation control applied to non-holonomic vehicles that are subject to significant environmental disturbance forces. The system is controlled by using a virtual-body and artificial potential multiple vehicle control technique [10].

Over the past several years, faculty and students in SCU's RSL have focused on a new formation control approach for applications requiring full control of formation parameters. This method, termed Cluster Space Control, involves viewing the robot formation as a cluster that is directed by variables such as position, orientation and geometry [1]. These cluster variables are related to robot specific variables through kinematic transforms. This allows a human in the loop to command the cluster at an abstracted, formation-level, with these commands being automatically converted to individual robot commands, thus allowing a simpler interface for controlling the group.

The vision for this cluster space technique includes improving its functionality and applying it to real world problems. Functional improvements are focused on enhancing motion control performance, increasing the number of vehicles, and adding more degrees of freedom by branching out into both aerial and submerged vehicles. Potential

applications and solutions to real world problems could range from Search and Rescue, where robot formations move across inhospitable terrain in search of people, to a cluster of robots following trace amounts of pollutants and detecting the source.

1.2 Gradient Based Navigation

Parameters in the environment (such as temperature or the concentration of certain elements) often vary spatially, and we are often interested in creating maps of these parameters. In some cases, we are only interested in knowing the location of the extreme values of these parameters (e.g., the minimum or maximum). In other cases, we may want to locate specific positions where the parameter has a specific value. In these types of cases, it may be inefficient to map an entire region in order to locate the small fraction of locations of interest. An alternate approach is to use the sensed data during the mapping operation to estimate how the parameter varies, and then to use this information to navigate to areas where the parameter may have a higher or lower value. In effect, the system computes the spatial derivative or gradient of the local parameter field in order to determine which direction will most efficiently lead the robot to areas of higher or lower concentration [11].

The gradient of a scalar field is a vector field which points in the direction of the greatest rate of increase of the scalar field, and whose magnitude is the greatest rate of change. An example of this gradient would be an increase in water temperature over a spatial interval, perhaps due to the effluent from a shore based power plant. In order to find peaks efficiently, the gradient following technique would allow the group of robots to sense the directions of increasing temperature and to head in that direction until the peak is located

In a multi-robot formation, a two vehicle configuration utilizes two vehicles that observe one another to climb up a gradient. As one vehicle follows the gradient, the second vehicle will continue in that direction using that shared knowledge and converge on the point.

One way to achieve gradient following with a single vehicle was demonstrated using the

Autonomous Benthic Explorer (ABE) to find a specified spot in a lake [10]. This approach was based on a chemotaxis behavior model of E.coli, which allowed the vehicle to move towards the desired location. Chemotaxis is defined as an action when bacteria moves either towards or away from certain chemicals. The mechanism used to achieve this chemotaxis [12] is known as the "Run-and-Tumble" [13] method. When the gradient is rich enough, the bacteria continues for longer stretches to move in a forward motion before it tumbles and moves again in a random direction. If the gradient becomes weaker, the bacteria decreases the distance between the direction changes. This approach has some limitations as the convergence rate can be slow and the mobile robot may become incapacitated in the local maxima or minima source concentration [14].

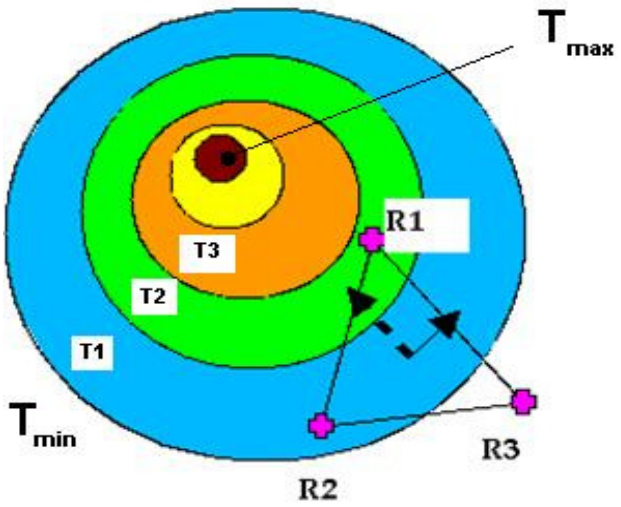


Figure 1.2-*A 3-robot cluster gradient-based example a contour temperate map*

Another study of single-robot source location used four anemometric sensors and four gas sensors in order to determine the direction of an odor source. The anemometric sensors were used for measuring the direction of the air flow carrying the odor molecules. This system used a strategy based on how moths track pheromone plumes [15]. The key feature to the work that has been done in the past and implemented in the robot was to track the chemical plume by driving into the oncoming wind, similar to the way a moth shows upwind surges when it perceives a pheromone [16],[17],[41],[42]. When the robot

by chance leaves the plume, it tries to relocate the lost plume by moving back and forth across the wind. One drawback to a single probe or sensor is that it can only cover a small defined area, which makes it good only for smaller applications. As the target area grows, the number of sensors would need to increase. Scaling becomes an issue, and a strategy using multiple robots could be effective.

In previous work, Hayes is probably the first researcher who used multiple robots to implement odor source localization with real robot hardware. The process for mobile robot based odor source localization was divided into different phases. During the initial phase, contact was made with a plume. Once the plume was detected, the robot group tracked the chemical toward its source. In the final phase the robots located the source. Hayes described the three phases as plume finding, plume tracking, and odor/gas source declaration [18].

More recently, a large scale underwater operation was completed in Monterey Bay in 2003, using multiple SLOCUM gliders [19]. This demonstration used a gradient climbing navigation strategy to locate and track features such as temperature fronts and eddies. The gliders faced strong currents and experienced significant communication delays. Nevertheless, using the glider data, the successful mapping of gradients in the temperature field was completed.

Gradient-based technologies typically face two challenges. The first is when the gradient is so small that it can't be sensed. In this case, a good strategy is to perform a local search (known as "casting" in biological literature) until it is re-acquired. The location of the previous packet encounter provides the best immediate estimate of where the next will occur. This type of surge-cast behavior has been observed in moths [20] and its performance has been studied in simulations [21]. Another issue that may affect performance is the existence of a local minima and maxima. Simple gradient-based search approaches terminate at such locations. If identifying global extremes is important, augmented techniques are required to identify other peaks and valleys in the parameter field.

1.3 Project Statement

The purpose of this research was to augment the existing cluster space control technique in order to demonstrate effective gradient-based functionality, specifically that of tracking gradient contours of specified concentration levels. This work was targeted for applications involving clusters of autonomous surface vessels in order to characterize the marine environment. To achieve this goal, significant effort was invested in the following tasks.

- Derivation of the gradient estimation function based on three simultaneous samples made by a distributed 3-Robot cluster,
- Incorporation of the gradient estimate into the cluster space control architecture in order to support contour tracking,
- Simulation of the control architecture to iteratively develop the method and to characterize performance as a function of the cluster's spatial geometry, and
- Verification of the technique with two hardware-in-the-loop test-beds: a set of three land rovers and a set of three robotic kayaks.

This work resulted in the successful demonstration of gradient contour tracking for robots operating in the field and using simulated spatial gradient fields. It is worth noting that development of this research program was performed in cooperation with fellow graduate student Vincent Howard, who implemented a gradient-based technique in order to determine the locations of gradient minima and/or maxima [22].

1.4 Reader's Guide

This thesis is divided into five chapters and is structured as follows: The first chapter provides an introduction to multi-robot systems and gradient based sensing. The chapter also discusses the motivation for this project and lists the objective of the thesis. The second chapter begins with a review of the cluster control concept and discusses the various gradient climbing and following methods. A literature review of several projects in these fields is reviewed. This provides a basic summary of how the experiments were

conducted. It also includes a discussion of the relevant kinematic transforms and control framework for a 3-robot cluster. Lastly it presents the dynamics of the vessels. The third chapter reviews the computer simulations and how the tests were conducted with a brief explanation of the test setup and procedures. The fourth chapter evaluates the three-ASV cluster space controls determined through hardware experimentation and their test-beds. The fifth chapter reviews and discusses the results of the thesis project.

2.0 Introduction Gradient-Based Cluster Space

Cluster space control treats the formation of robots as a group that is directed by variables such as position, orientation and geometry. This allows a human in the loop to command the cluster at an abstracted, formation-level. These commands are automatically converted to individual robot commands, allowing a simple and versatile interface for controlling the cluster.

This chapter starts with a review of the cluster space control technique to include a discussion of its general kinematic transforms, closed loop control architecture and its singularities. In the second part of the chapter, an introduction to gradient-based techniques is presented, and the integration of gradient based control within the cluster control framework discussed in detail.

2.1 Review of Cluster Space Control

The cluster space control technique promotes simplified specification and monitoring of the motion of mobile multi-robot systems. We wish to specify multi-robot system motion and compute required control actions in the cluster space using cluster state variables. Given that these control actions will be implemented by each individual robot (ultimately by the end effectors or actuators on the robot), we develop formal kinematic relationships relating the cluster space variables and robot space variables described below.

The selection of cluster state variables may be a function of the design, the application, and the operator's preferences and/or ease of the calculation. The cluster space description establishes a cluster reference frame; references to individual robots in the cluster are made with respect to this cluster frame as shown below in Figure 2.1. A conventional description of a cluster of robots provides individual robot frame descriptions with respect to a global frame, typically in the form of a homogeneous transform [1].

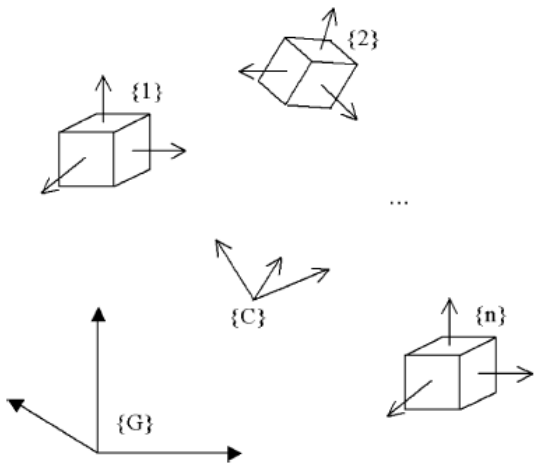


Figure 2.1-*Robot pose using conventional vs cluster space representation[1].*

The cluster space pose variables are defined as:

$$\vec{C} = (c_1, c_2, \ldots, c_n)^T \tag{1}$$

where $(c_1, c_2, \ldots)^T$ includes variables that represent the position, orientation and geometry of the cluster. The robot space state variable is defined as:

$$^G\vec{R} = (x_1, y_1, \theta_1 \ldots x_n, y_n, \theta_n)^T \tag{2}$$

where n is the number of robots and $(x_1, y_1, \theta_i)^T$ defines the position and orientation of robot *i*. Cluster space variables are related to robot space variables through a formal set of kinematic transforms.

$$\vec{C} = \begin{pmatrix} c_1 \\ c_2 \\ . \\ . \\ . \\ c_{mn} \end{pmatrix} = \text{KIN}(^G\vec{R}) = \begin{pmatrix} g_1(r_1, r_2, .., r_{mn}) \\ g_1(r_1, r_2, .., r_{mn}) \\ . \\ . \\ . \\ g_{mn}(r_1, r_2, .., r_{mn}) \end{pmatrix} \tag{3}$$

$$^G\vec{R} = \begin{pmatrix} r_1 \\ r_2 \\ . \\ . \\ . \\ r_{mn} \end{pmatrix} = \text{INVKIN}(\vec{C}) = \begin{pmatrix} h_1(c_1, c_2, .., c_{mn}) \\ h_1(c_1, c_2, .., c_{mn}) \\ . \\ . \\ . \\ h_{mn}(c_1, c_2, .., c_{mn}) \end{pmatrix} \tag{4}$$

We also consider the formal relationship between the robot and cluster space velocities, $\dot{\vec{R}}$ and $\dot{\vec{C}}$. Taking the derivative of equation (3), we may compute the cluster state

velocities; this can be expressed in terms of the Jacobian matrix J, which maps the robot velocities to the cluster velocities in a form of a linear time varying function.

$$\dot{\vec{C}} = \begin{pmatrix} \dot{c}_1 \\ \dot{c}_2 \\ \cdot \\ \cdot \\ \cdot \\ \dot{c}_{mn} \end{pmatrix} = {}^{G}J(\vec{R})\,{}^{G}\dot{\vec{R}} \tag{5}$$

In a similar manner, we can develop the inverse Jacobian ${}^{G}J(\vec{R})\,{}^{G}\dot{\vec{R}}$, which maps cluster velocities to robot velocities. Computing the partial derivatives of the robot space pose variables from (4) yields

$${}^{G}\dot{\vec{R}} = \begin{pmatrix} \dot{r}_1 \\ \dot{r}_2 \\ \cdot \\ \cdot \\ \cdot \\ \dot{r}_{mn} \end{pmatrix} = {}^{G}J^{-1}({}^{G}\vec{R})\dot{\vec{C}} \tag{6}$$

With the formal kinematics defined, the controller is composed such that desired motions are specified and control compensations are computed in the cluster space. These compensation commands are transformed to robot space through the inverse Jacobian relationship. The resulting robot-level velocity commands are transformed to actuator commands through a vehicle-level inverse Jacobian relationship. Sensed cluster space parameters are then compared to the operator specified desired values for the cluster space variables, which might include trajectories of how the cluster centroid should move over time, how the cluster should rotate, how the shape should change, etc. [23]. Figure 2.2 below, presents the control architecture for trajectory based cluster space control.

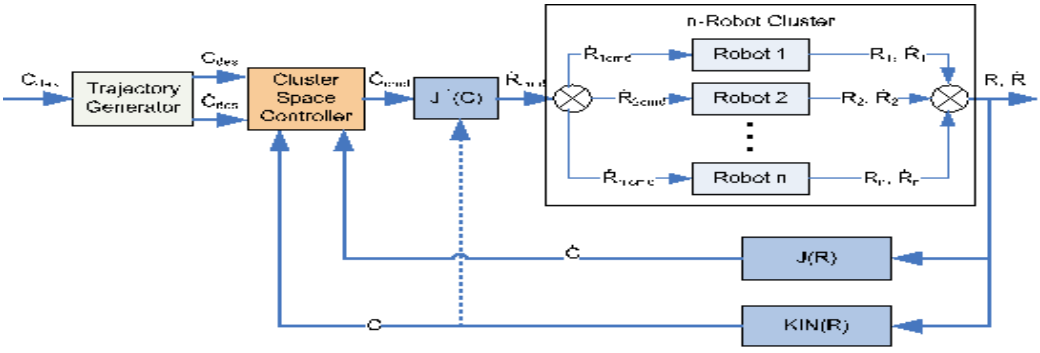


Figure 2.2-*Three robot cluster definition [2]*

In robotic manipulator chains, singularities occur in configurations where the Jacobian and inverse Jacobian matrices become singular. Such singularities are also an issue in the cluster space control of multi-robots. One such singularity often occurs when two or more robots are located at the same location. In practice, we wish to avoid this condition in order to protect the robots, and an avoidance policy could be implemented to maintain an arbitrary separation distance, thereby ensuring that the singularity is avoided [24]. Operation at or near singularities often leads to many challenges, such as possible amplification of sensing errors [25]. In general singularities can be avoided by simply not allowing operations in their vicinity by selecting an alternate set of cluster variables.

We have successfully used the cluster space control approach to demonstrate cluster-space-based versions of regulated motion [26], automated trajectory control [27]-[28], human-in-the-loop piloting [29]-[30]. This work has included experiments with planar land rover clusters [31]-[32], with surface vessel systems [33], for holonomic and non-holonomic robots, for robots negotiating obstacle fields [34] applications such as escorting and patrolling [33]-[35], and aerial robots [36]-[37].

2.2 Three Robot Cluster Definition

To demonstrate, we review the application of the cluster space framework to a simple 3-Robot cluster as presented in [1]. This section reviews the selection of cluster space variables and the resulting kinematic transforms for this example. This 3-Robot formation is used later in this thesis as part of the implementation of a gradient-based contour-following cluster space controller.

Given the parameters defined by Figure 2.3, the cluster space state variable definition is given by:

$$\vec{C} = (x_c, y_c, \theta, \emptyset_1, \emptyset_2, \emptyset_3, p, q, \beta)^T \quad (7)$$

where $(x_c, y_c, \theta)^T$ are the cluster positions, the $\emptyset_i$'s are the yaw orientation of each rover relative to the cluster, p and q are the distances from rover 1 to rover 2 and 3 respectively, and β is the skew angle with vertex on rover 1 [40]. The robot space state vector is defined as:

$$^{G}\vec{R} = (x_1, y_1, \theta_1, x_2, y_2, \theta_2, x_3, y_3, \theta_3)^T \quad (8)$$

were $(x_I, y_1, \theta i)^T$ defines the position and orientation of robot i.

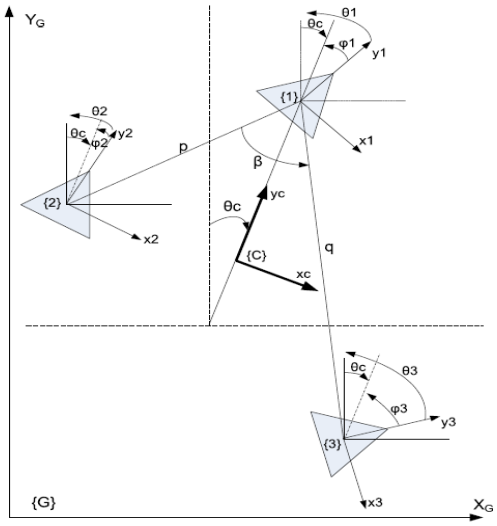


Figure 2.3-*A three robot cluster definition [40]*

Given the aforementioned selection of cluster space state variables, it is possible to express the forward and inverse position kinematics of the 3-Robot system [40].

The forward position kinematics in this experiment are therefore defined by:

$$x_c = \frac{x_1 + x_2 + x_3}{3} \quad (9)$$

$$x_y = \frac{y_1 + y_2 + y_3}{3} \quad (10)$$

$$\theta_C = atan2\frac{2/3(x_1) - 1/3(x_2 + x_3)}{2/3(y_1) - 1/3(y_2 + y_3)} \quad (11)$$

$$\emptyset_1 = \theta_1 + \theta_C \quad (12)$$

$$\emptyset_2 = \theta_2 + \theta_C \quad (13)$$

$$\emptyset_3 = \theta_3 + \theta_C \quad (14)$$

$$p = \sqrt{(x_1 - x_2)^2 + (y_1 - y_2)^2} \quad (15)$$

$$q = \sqrt{(x_3 - x_1)^2 + (y_1 - y_3)^2} \quad (16)$$

$$\beta = acos\frac{(p^2 + q^2) - (x_3 - x_2)^2 - (y_3 - y_2)^2}{2pq} \quad (17)$$

The inverse position kinematics are therefore defined by:

$$x_1 = x_c + \left(\frac{1}{3}\right) r sin\theta_c \tag{18}$$

$$y_1 = y_c + \left(\frac{1}{3}\right) r sin\theta_c \tag{19}$$

$$\theta_1 = \emptyset_1 - \theta_c \tag{20}$$

$$x_2 = x_c + \left(\frac{1}{3}\right) r sin\theta_c - p sin\left(\frac{\beta}{2} + \theta_c\right) \tag{21}$$

$$y_2 = y_c + \left(\frac{1}{3}\right) r sin\theta_c - p sin\left(\frac{\beta}{2} + \theta_c\right) \tag{22}$$

$$\theta_2 = \emptyset_2 - \theta_c \tag{23}$$

$$x_3 = x_c + \left(\frac{1}{3}\right) r sin\theta_c - q sin\left(\frac{\beta}{2} + \theta_c\right) \tag{24}$$

$$y_3 = y_c + \left(\frac{1}{3}\right) r sin\theta_c - q sin\left(\frac{\beta}{2} + \theta_c\right) \tag{25}$$

$$\theta_3 = \emptyset_3 - \theta_c \tag{26}$$

where

$$r = \sqrt{(q + pcos\beta)^{\,2} + (psin\beta)^{\,2}}$$

With the forward and inverse position kinematics determined, the forward and inverse velocity kinematics can be derived by differentiation. The velocity kinematics are placed into matrices known as the Jacobian and inverse Jacobian matrices. These matrices take the form given in equations (27-29). The forward and inverse Jacobians are used to convert velocities from robot space to cluster space and vice versa. Symbolically

$$\dot{\vec{C}} = J\left(\vec{R}\right) * \left(\dot{\vec{R}}\right) \tag{27}$$

where,

$$J\left(\vec{R}\right) = \begin{pmatrix} \frac{\partial c1}{\partial r1} & \cdots & \frac{\partial c1}{\partial r9} \\ \vdots & \ddots & \vdots \\ \frac{\partial c9}{\partial r1} & \cdots & \frac{\partial c9}{\partial r9} \end{pmatrix} \tag{28}$$

And conversely,

$$\dot{\vec{R}} = J^{-1}\left(\vec{C}\right) * \left(\dot{\vec{C}}\right) \tag{29}$$

$$J^{-1}(\vec{C}) = \begin{pmatrix} \frac{\partial r1}{\partial c1} & \cdots & \frac{\partial r1}{\partial c9} \\ \vdots & \ddots & \vdots \\ \frac{\partial r9}{\partial c1} & \cdots & \frac{\partial r9}{\partial c9} \end{pmatrix} \tag{30}$$

Due to limited space, the full algebraic expressions for $J^{-1}(\vec{C})$and $J(\vec{R})$ are presented in a paper done by C. Kitts [40]. Given the transforms, 3-robot architecture is shown below in Figure 2.4.

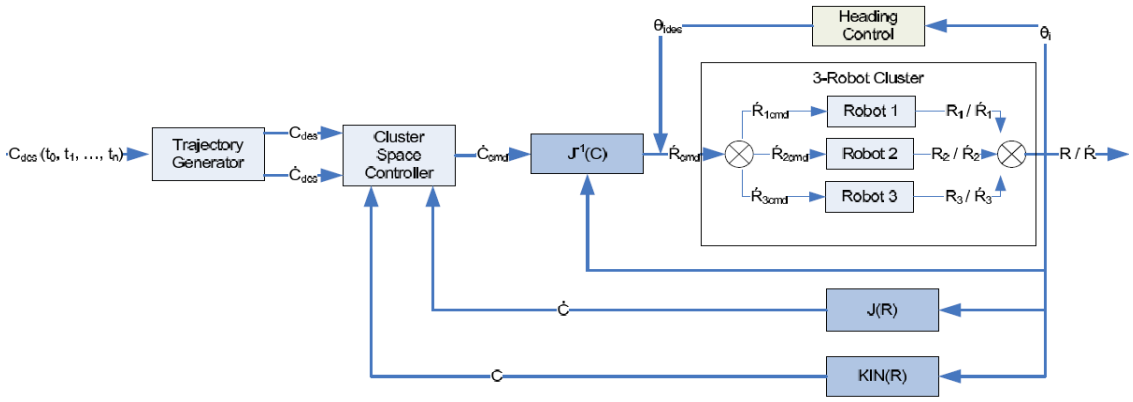


Figure 2.4-*Three robot PID Controller cluster definition*

2.3 Gradient Based Control [44]

We have applications in which navigating a robotic cluster based on the gradient of a parameter field would be useful. This gradient-based functionality can allow a cluster of robots to perform a multitude of tasks such as contour following in bathymetric maps and point source tracking**.** One application could be in a marine environment with a depth sensor used to trace depth contours of an uncharted area and collect scientific data. For search and rescue applications, patrolling by the cluster could be accomplished with the altitude sensors in cases where gridded techniques are not optimal. As an example, on a very steep slope, one might be required to slow down as there is a large surface area below, or the terrain is rugged, making it much more difficult to locate items or sources than on an open smooth surface. Another application of co-operative robot control could be the use of a gas sensor mounted on land robots. This could pinpoint a pollution source

as discussed by V. Howard.

For the research project, we seek to locate and follow specific contours of a gradient field. To do this, we needed first to estimate the direction of the gradient and/or contour based on the real-time measurements made by sensors on the distributed robots, and second, to steer the cluster in the appropriate direction. For this work, we assume that this should be done by orienting the cluster along the contour and having the cluster travel in the direction it is facing.

To understand how the orientation of the gradient field is estimated, consider the diagram in Figure 2.5. In this diagram, robots are represented by the red dots lying in the X-Y plane. The parameter field is represented by dotted red contour lines within the X-Y plane and also as an inclined plane above the X-Y plane; the green dots represent an equivalent position of the robots on the inclined parameter field given that the height of each robot indicates the value of the parameter field that it senses, z_i. The vectors $\bar{R}_{12}$ and $\bar{R}_{13}$ are created as shown in the figure, running from the virtual Robot 1 location to the virtual locations of Robots 2 and 3, respectively.

To compute the direction of the field's gradient, shown laying in the X-Y plane as ∇f, the cross product of $\bar{R}_{12}$ and $\bar{R}_{13}$ is computed and projected into the X-Y plane. The resulting ∇f vector points in the direction of greatest parameter increase, and it is perpendicular to field contour lines. We note that we have assumed that the parameter field is planar at the location of the cluster.

With ∇f computed, the globally referenced bearing to the field contour, shown as λ can be found by: $-arctan2\,(\nabla f)$. This is the contour bearing for what we term the Clockwise (CW) contour direction, which implies a CW rotation around the parameter field if the field was a simple single peak. The bearing of the contour for the Counter Clockwise (CCW) direction is $\pi-\lambda$. The bearing to the gradient direction itself is $90-\lambda$, and the bearing to go "down" the gradient is $-(90+\lambda)$.

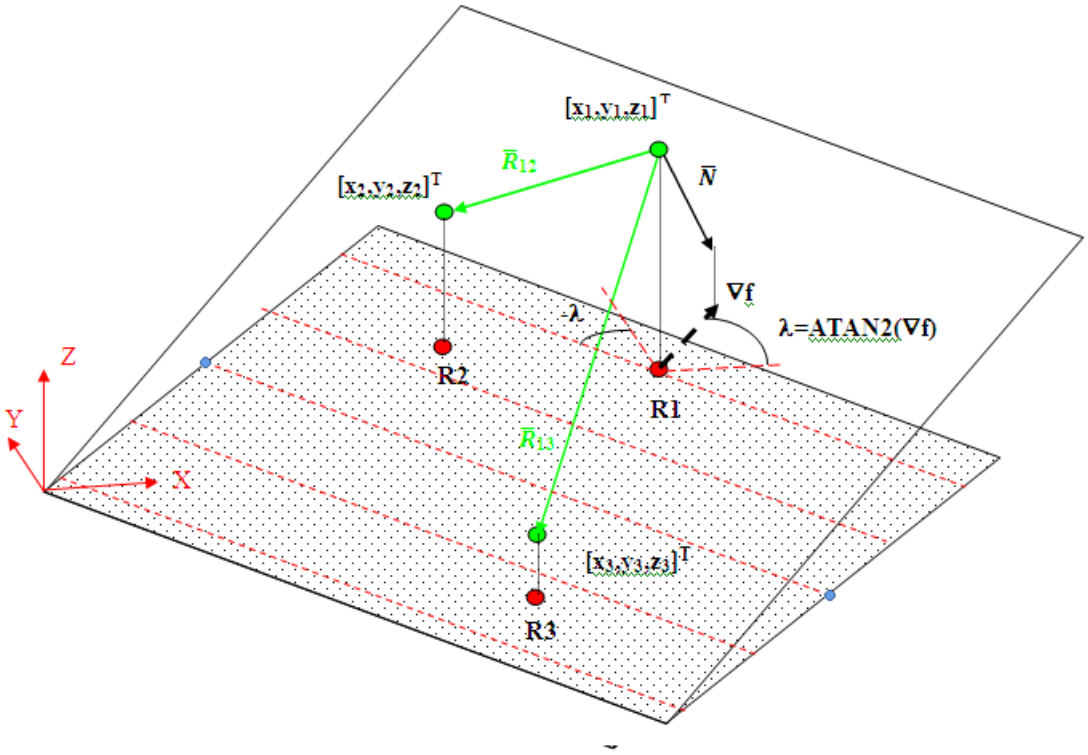


Figure 2.5-*A simple 3-D robot cluster gradient-based description*

To summarize this estimation approach mathematically:

$$\bar{R}_{12} = \begin{pmatrix} x_2 - x_1 \\ y_2 - y_1 \\ z_2 - z_1 \end{pmatrix} \tag{31}$$

$$\bar{R}_{13} = \begin{pmatrix} x_2 - x_1 \\ y_2 - y_1 \\ z_2 - z_1 \end{pmatrix} \tag{32}$$

$$\vec{N} = \bar{R}_{13} \text{ x } \bar{R}_{12} \tag{33}$$

$$\vec{N} = \begin{pmatrix} x_2 - x_1 \\ y_2 - y_1 \\ z_2 - z_1 \end{pmatrix} \text{X} \begin{pmatrix} x_2 - x_1 \\ y_2 - y_1 \\ z_2 - z_1 \end{pmatrix} \tag{34}$$

$$\nabla f = [N_x, N_y]^T \tag{35}$$

Where $(N_x\ , N_y\ )$ are the components of the Normal vector projected onto the xy plane to determine the angle λ.

$$\lambda = arctan2\,(\nabla f) \tag{36}$$

$$\lambda = arctan2\left(\frac{N_y}{\sqrt{N_x^2 + N_y^2} + N_x}\right) \tag{37}$$

Estimation of the parameter field's gradient and contour bearings can be used as the basis for a variety of control strategies. To follow a specific contour, a cross-track contour controller (for CW travel along the contour) is used with the form:

$$(\dot{\theta}_C)_{des} = K[(\theta_C)_{des} - (\theta_C)_{act}] \tag{38}$$

$$(\theta_C)_{des} = [-(\,^{G}(\lambda) + K_{ct}(e)_{ct})] \tag{39}$$

$$e_{ct} = [(z_{des}) - (z_{act})] \tag{40}$$

$$(\dot{\theta}_C)_{des} = K[-(\,^{G}(\lambda) + K_{ct}((z_{des}) - (z_{act}))) - (\theta_C)_{act}] \tag{41}$$

The strategy for this control law is depicted in Figure 2.6. If the cluster (e.g., its centroid) is properly tracking the desired contour, then there is no cross-track error, e_{ct}, and the cluster should be heading in the direction of the contour bearing: $(\theta_C)_{des} = -\lambda$. If the centroid of the cluster is off the desired contour, a non-zero cross track error exists, and the desired cluster heading should be the contour bearing with an angular offset that is proportional to e_{ct}: $(\theta_C)_{des} = [-(\,^{G}(\lambda) + K_{ct}(e)_{ct})]$. By achieving this instantaneous heading set-point, the cluster will head to the desired contour with the level of aggressiveness specified by the value of the proportional control parameter, K_{ct}.
For the cluster controller, a simple proportional law may be used where $(\theta_C)_{des}$ is a simple proportional function of the error in the desired θ_C set-point. Integration of this contour-following control augmentation with the rest of the cluster space controller is depicted in Figure 2.7.

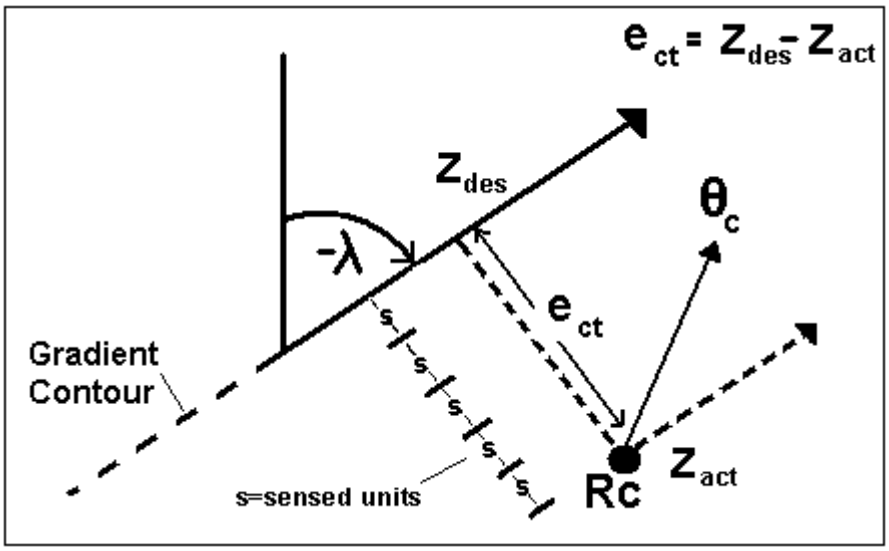


Figure 2.6-*Robot contour tracking error on a gradient*

This steering law, combined with other control objectives, maintains speed along the cluster's heading while also maintaining the cluster's shape, as shown in Eq 40 [44].

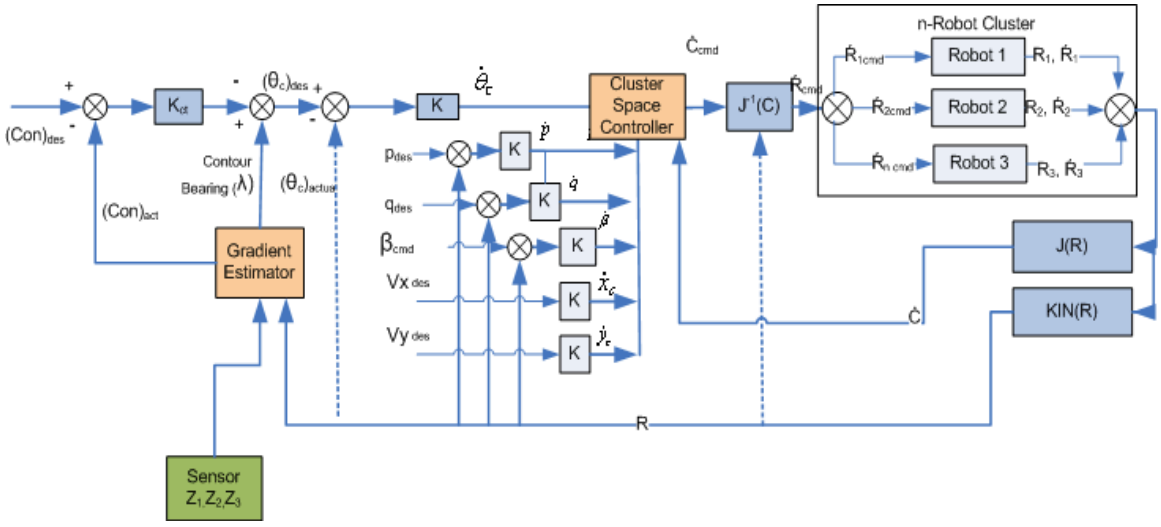


Figure 2.7-*PID Matlab Simulink simulation model with gradient Control*

In the control block diagram, the system is shown as a regulator for cluster speed, size, and shape; cluster orientation is automatically controlled in order to follow the sensed field contour. The desired and actual cluster variables are the inputs to the standard cluster space controller. Through a PID control algorithm, the cluster space controller outputs cluster variable rates. Next, the inverse Jacobian matrix converts the commanded

cluster variable rates into commanded rates of each of the individual robot degrees of freedom. The individual commands for each robot are executed by the onboard actuators. Finally, the resulting positions and orientations of each robot are used as feedback for the gradient estimator and the cluster controller. For the cluster controller feedback, robot positions and velocities are converted to cluster position and velocity variables through the kinematic equations and the Jacobian matrix, respectively.

3.0 Simulation Introduction

A simulation of the proposed control system and the multi-robot experimental testbed has been created using Simulink software. The closed-loop controller used for simulation is identical to that used in the experimental testbed. The following chapter describes the simulation environment and simulation results for a three robot cluster. Results verify the operation of the controller and are used to characterize performance.

3.1 Simulation Environment

To facilitate the development and evaluation of the gradient-based cluster space control concept, a simulator using Matlab/Simulink was adapted from previous RSL student work and used to evaluate the concept prior to committing to hardware experiments. An iterative approach was used to develop the concept and characterize key performance issues. This simulator includes a simple three-dimensional world representation of robot motion using the Virtual Reality Markup Language (VRML) Toolbox as seen in Figure 3.1. The simulator supports the use of robot kinematic and dynamic models of several holonomic and non-holonomic multi-robot systems available for experimentation. It also supports evaluation of automatic controllers.

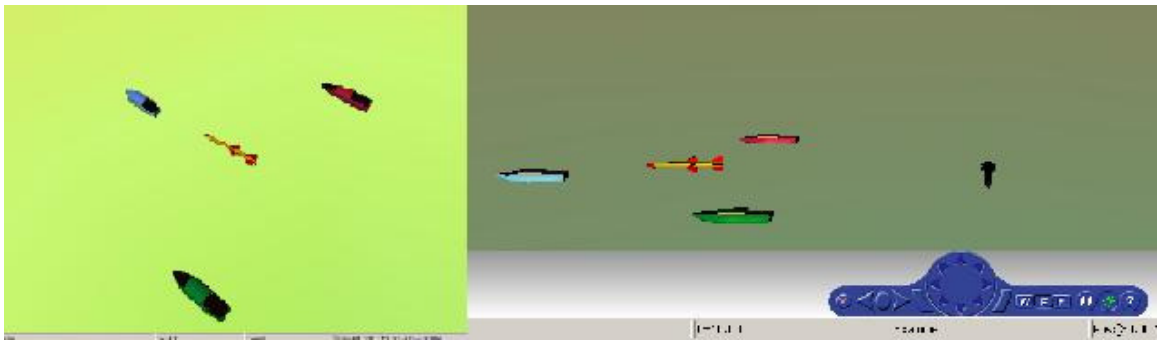


Figure 3.1-*Screen shot of virtual simulation tests*

In previous work [39] model parameters were determined experimentally and were based on the characteristic response of the test-bed vehicle to step inputs. Minor adjustments to the boats, conditions of the motors, and environmental conditions made it necessary to

tune these values during field experiments. Gains were adjusted to optimize the system performance.

Figure 3.2 is the actual Simulink control loop that was used. The desired cluster position and velocity can be specified as a bearing as well as a static location of the group. This is then fed into the controller block as discussed previously in section 2.3.

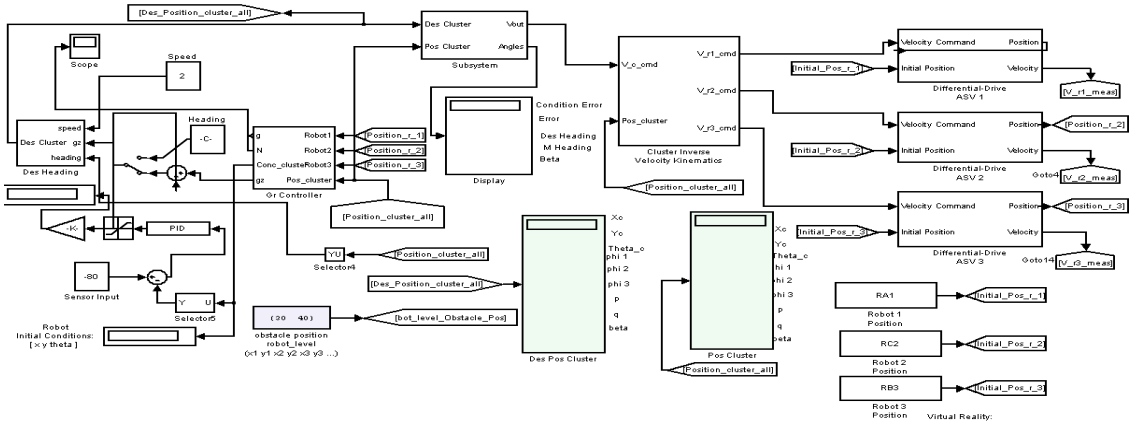


Figure 3.2-*Simulink control block diagram, 3-robot cluster*

3.2 System Base-Line Functionality

In order to show the most basic functionality of the control framework, the simplest possible parameter gradient field that could be conceived, a one-dimensional planar field, was created for test. This was used to verify functionality of the controller and to characterize performance. The tests were run to ensure that the frame transforms and kinematics were correct and that the model behaved as predicted. Two parallel sets of initial tests were conducted; the first round of testing was done assuming an ideal disturbance-free field. In order to produce more realistic field simulations and determine the robustness of the controller, the second series of tests included induced noise for the sensor measurements.

A planar gradient field equation was used to determine functionality of the robot cluster

for preliminary testing and verification. Field equations were defined and inserted into the x-y gradient field simulation function, producing the equivalent of a sloped plane. A robot initial condition function was developed and incorporated which facilitated the initial geometrical configurations and orientations. This function prevented the robots from either starting in a singularity or entering one from the start if the β angle was at 0 or 180°. Based on the knowledge of the testbed, GPS accuracy, kayak speed and space availability, initial values were estimated for the *p&q* values to simulate real life values.

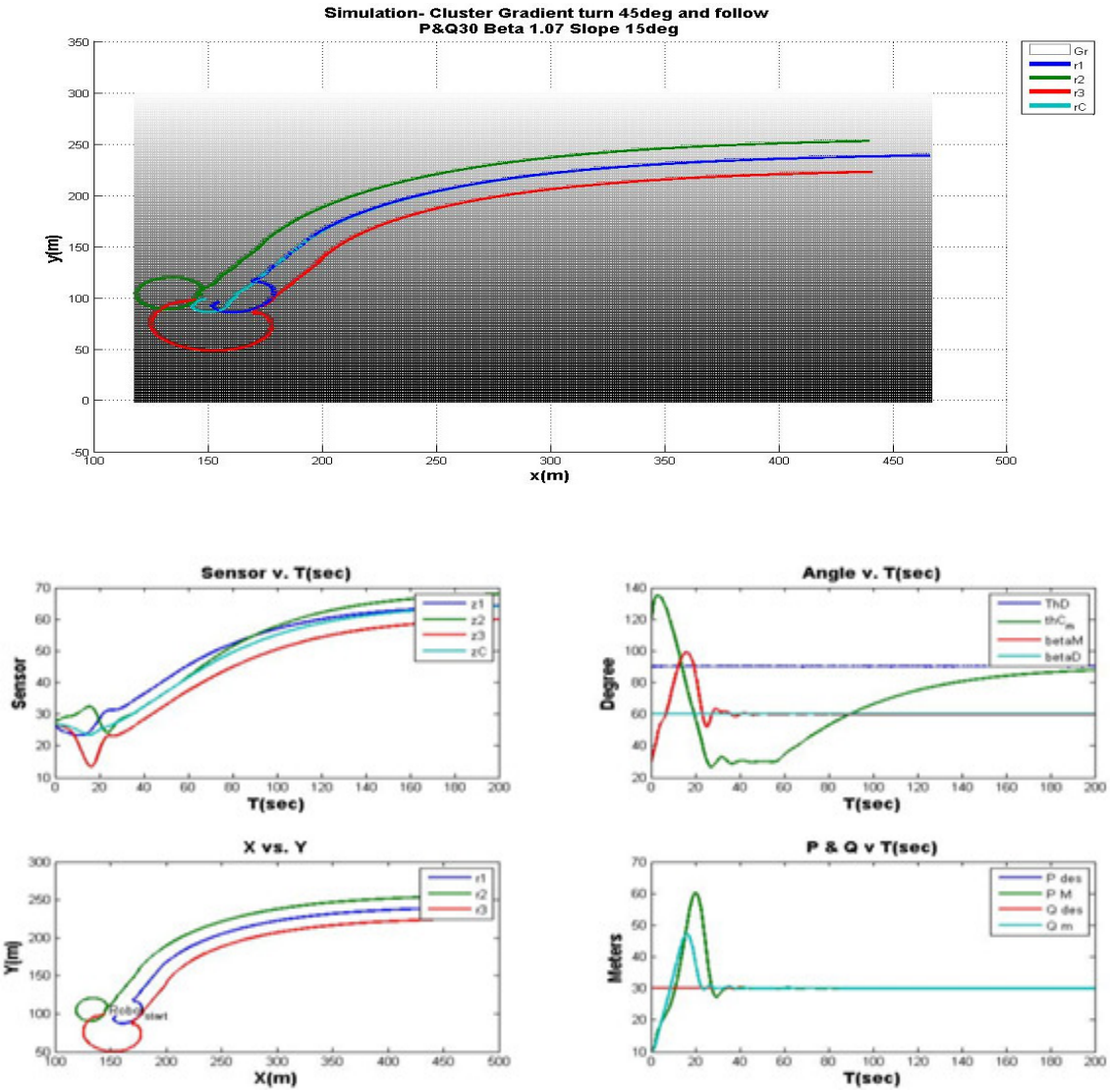


Figure 3.3-*Cluster orientation with constant Beta=75° & p&q=30.*

For the initial simulation, the gradient field was z = y/3.73 units, resulting in a field with

a gradient in the pure y direction and contours along constant values of y. For the first simulation using this field, shown in Figure 3.3 the cluster of robots was commanded to follow a contour of z=65 units while maintaining a shape of [p,q,β]=[30m,30m,60°]. The cluster was initially positioned at approximately 100 units in the parameter field with a heading of approximately 120° and a shape of [p,q,β]=[10m,10m,120°]. As can be seen in the figure, after an initial transient, the cluster maintains its shape while moving up the gradient until it drives along the z = 65 units contour. It is noted that the gradient field estimation function (which operates perfectly given the simulated environment and the planar gradient field) computes a desired heading of 90°, as shown in the Figure.

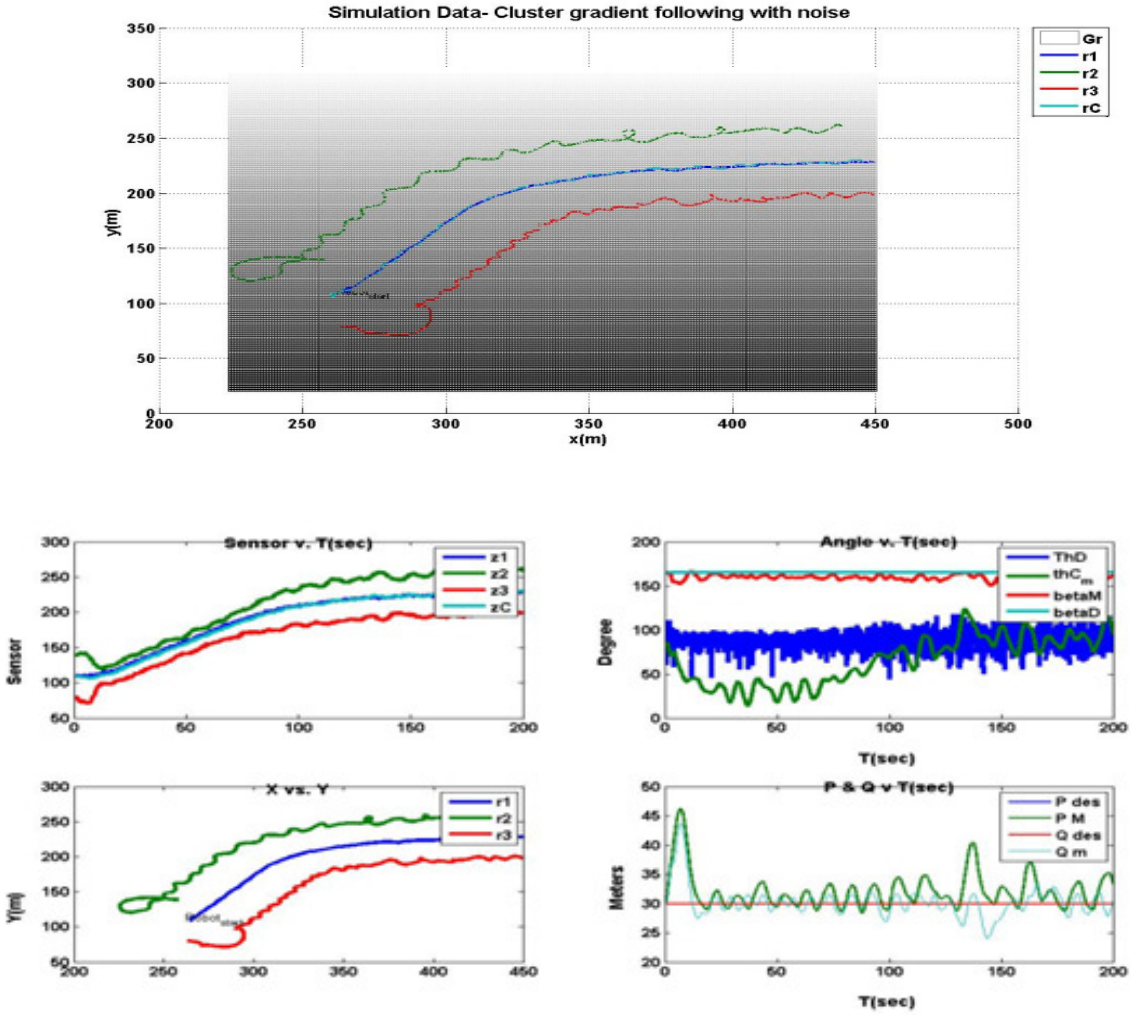


Figure 3.4-*Simulation gradient following data with noise*

In the second part of the basic testing, noise was added to the sensor signal (Figure 3.4) .

This was done in order to determine if sensor noise could have any significant impact on performance. A gradient field of z = y units was used, resulting in a field with a gradient in the pure y direction and contours along constant values of y. The noise was a Simulink normally distributed Gaussian block with a variance of 1 unit. This was incorporated into the virtual sensor readings. (Refer to Appendix D1 for further details). The cluster of robots was commanded to follow a contour of $z = 225$ units while maintaining a shape of [p,q,β]=[30m,30m,160°]. The cluster was initially positioned at approximately $z = 100$ units in the parameter field with a heading of approximately 90°. As can be seen in the figure, after an initial transient, the cluster maintains its shape while moving up the gradient until it drives along the $z = 225$ units contour. Although control is achieved, the robots are unable to precisely settle onto a fixed contour and the system controller is unable to reduce the error. This was due to the addition of noise. The following section characterizes the relationships between noise, the cluster's size and shape, and the gradient field strength.

3.3 Noise Characterization

To characterize non-ideal behavior as a function of cluster shape and gradient field strength, noise functions were inserted in the Simulink model block to create a disturbance. Each sensor in the Simulink model had a normally distributed Gaussian random signal block added to its ideal measure. The settings used had a mean of 0, and a variance of 1 unit, with randomly chosen speed. Increasing noise or decreasing noise has a direct influence on the system stability. The seeds chosen were a random value. A full explanation of variance, mean values and the method of comparison will be discussed further in the chapter. This initial work serves as a preliminary study of this issue; additional exploration of this issue should be the source of future work.

3.3.1 – Size Characterization

The characterization of the effect of the size of the cluster, (*p, q),* on performance was determined by varying cluster size while holding the gradient slope, cluster shape (*Beta*) and noise level constant. As seen from Figure 3.5-8, as size was increased, the cluster

motion became more stable and was able to follow the desired value without losing spatial integrity. This makes sense given that the large cluster size led to larger differences in the sensed field measurements between robots resulting in a smaller variation in these differences as a result of the noise. These sensed differences are a critical element of the computed gradient field estimate, as presented in Chapter 2.

.

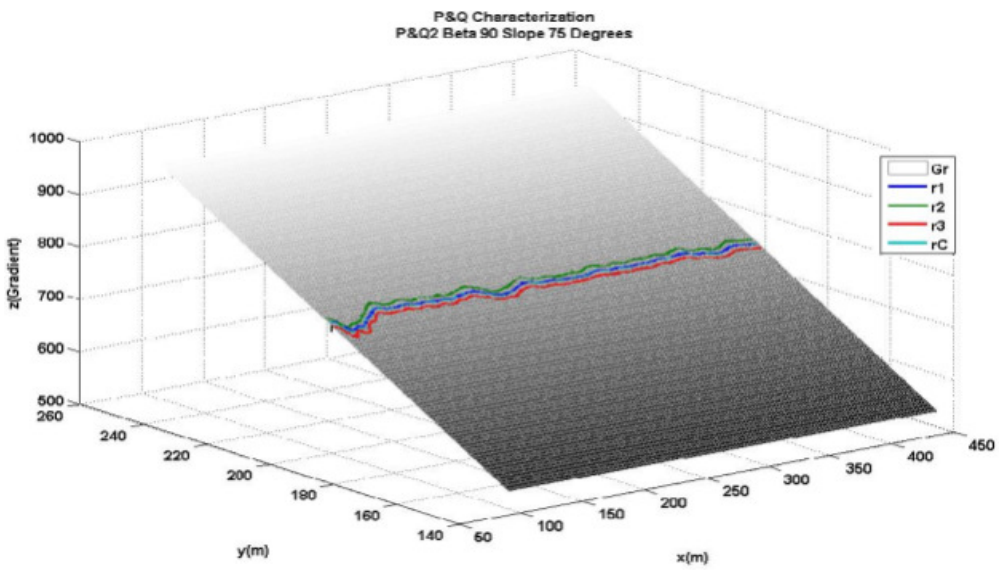


Figure 3.5-*Characterization: p&q=2, constant Beta=90°,slope=75°*

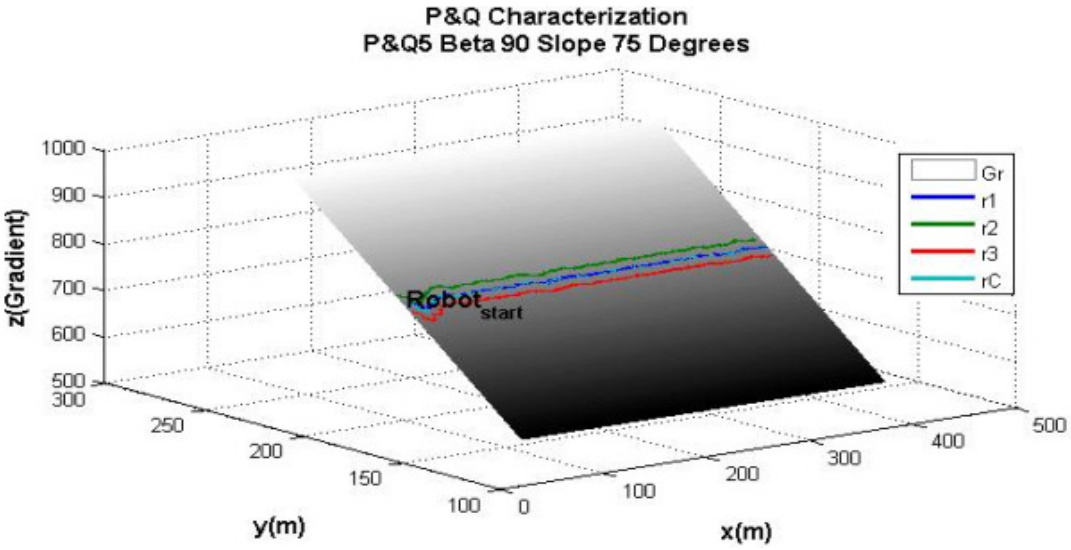


Figure 3.6-*Characterization: p&q=5, constant Beta=90°,slope=75°*

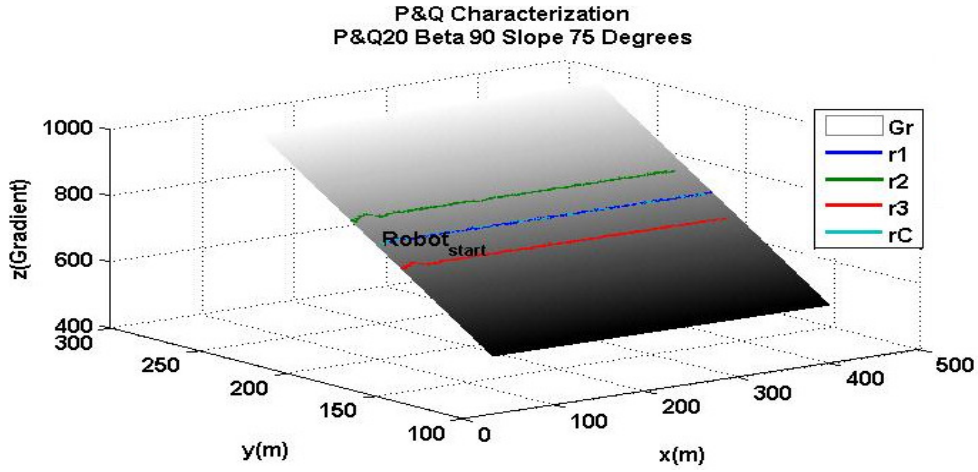


Figure 3.7-*Characterization: p&q=20, constant Beta=90°,slope=75°*

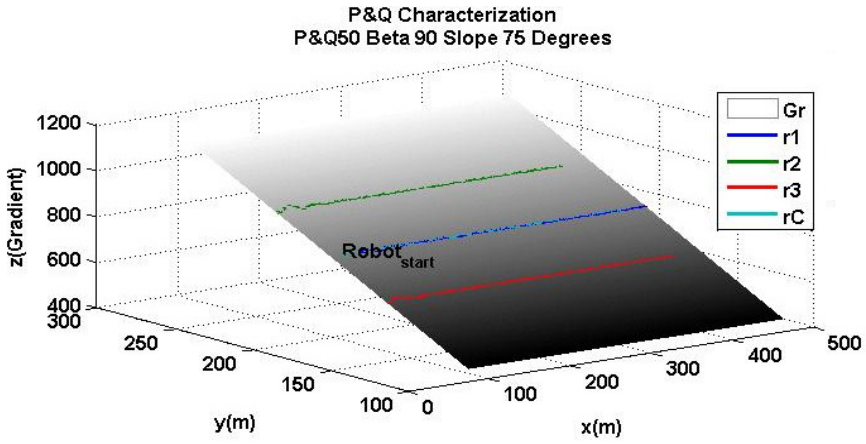


Figure 3.8-*Characterization: p&q=50, constant Beta=90°,slope=75°*

To better understand this relationship, a series of simulations were run in which cluster size was varied from 2-200 meters over three different gradient field slopes: 15°, 45°, 75°. For each simulation, the RMS ThetaC settling error was computed. As shown in Figure 3.5-8, as cluster size increases, settling error decreases as expected. In addition, it can be seen that, for a given size, the settling error also decreases as the strength (or slope) of the parameter gradient field increases. This makes sense since, for a given cluster size, a stronger gradient field leads to the same increase in the difference between robot sensor values within the field (thereby lowering the effect of noise on this difference).

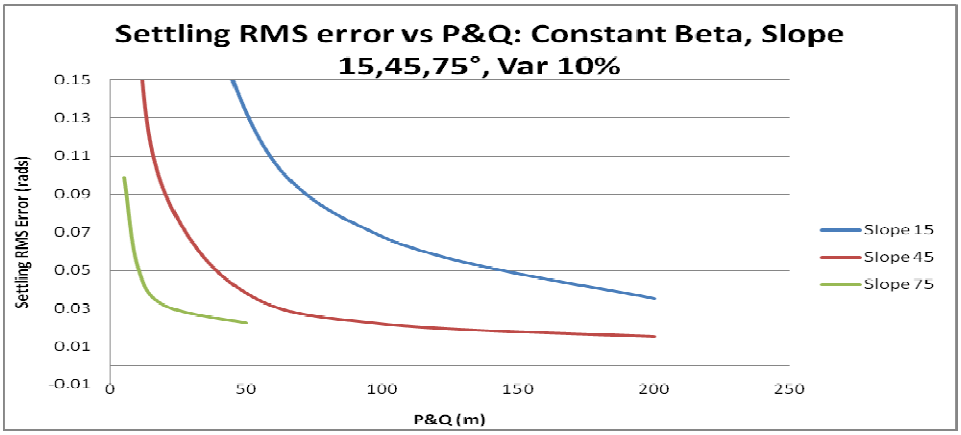


Figure 3.9-*p&q Characterization RMS error vs p&q (three slopes)*

3.3.2 – Gradient Field Strength Characterization

This section characterizes the relationships between noise and the strength of the parameter field. This was determined by increasing the slope and specified increments from 15°,45°,75°, while holding the other cluster shape and size parameters constant (*p&q* fixed at 10, *Beta* held at 90°). The one-dimensional plane was used once again with sensor noise functions inserted in the Simulink model block to create a disturbance in the "*z*" gradient sense field. The settling RMS error is defined as the error that is observed after the cluster has reached its steady state.

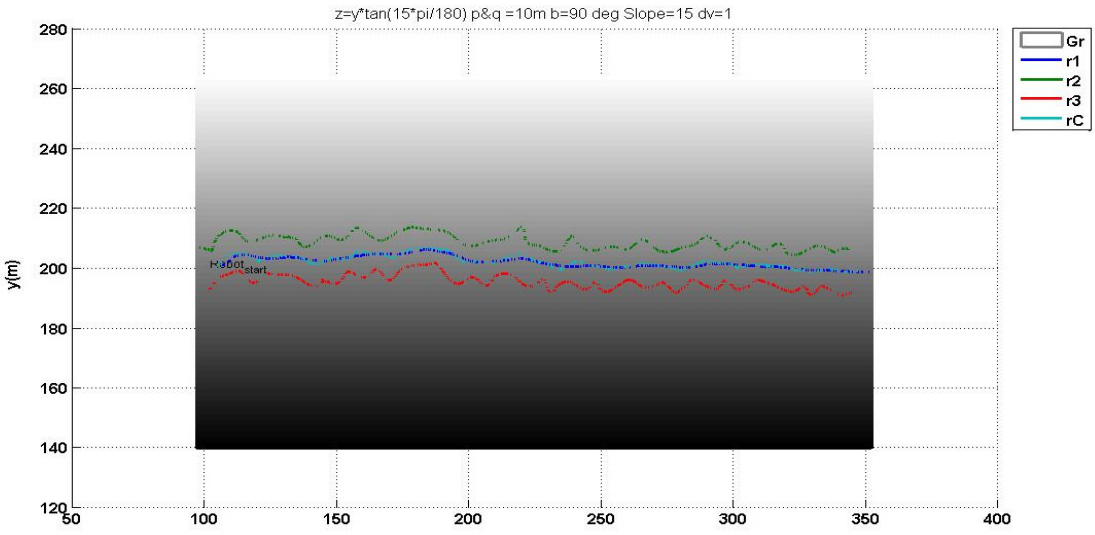


Figure 3.10-*75° slope characterization: p&q is set at 10 & Beta 90°*

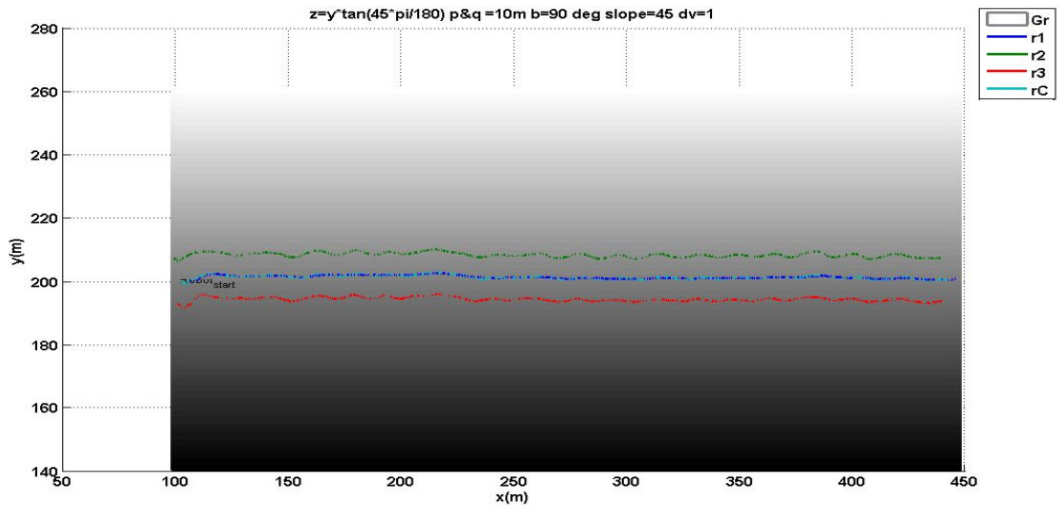


Figure 3.11-*45° slope characterization: p&q is set at 10 & Beta 90°*

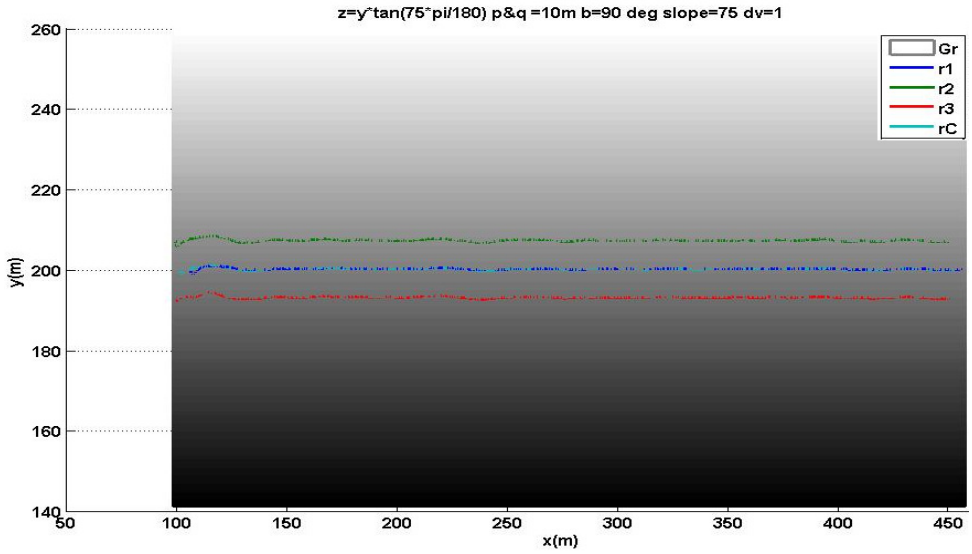


Figure 3.12-*75° slope characterization: p&q is set at 10 & Beta 90°*

We completed another series of simulations in which we changed only one variable, *Beta* and fixed the *slope, p&q*, resulting in tests to generate a characterized relationship. The series of Figures 3.10-12, show that as the slope was increased the cluster became more stable and was able to follow the desired values with less heading error. At a slope of 15°, the RMS error was calculated to be 0.29 rad, and as the slope was increased to 75°,

the RMS error dropped to 0.078 rad. As before, this relationship between an increasing gradient field slope and a reduction in the settling RMS error is due to larger differences in the sensed field measurements such that the effect of noise is reduced. However, it is interesting to note that as the slope increases further, the RMS error eventually rises. This is due to a geometric singularity as can be seen from Eq 41. The low point on the graph indicates a region of stability and can be used as a guideline in order to choose the correct parameters without overstepping into the unstable regions; of course, the location of this point may vary as a function of size and shape.

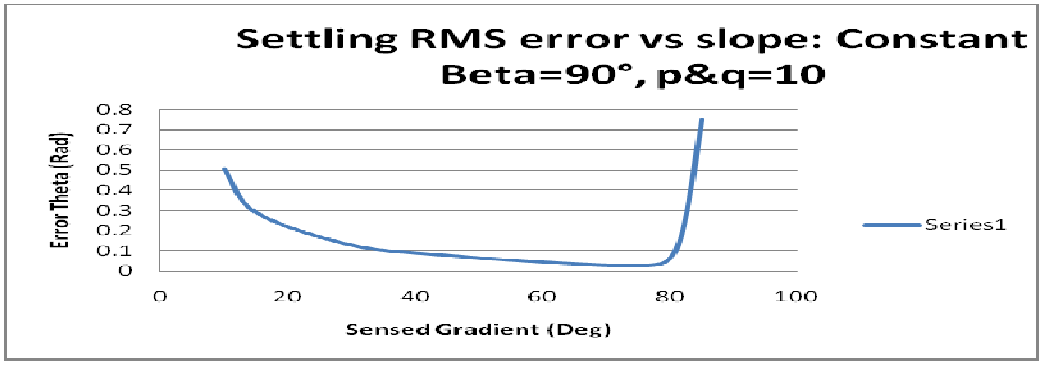


Figure 3.13-*Characterization Settling RMS error vs slope/field sense*

3.3.3 – Beta Characterization

This section looks at the characterization of the relationship between noise and cluster shape (*Beta* angle), given fixed cluster size and gradient field strength. We conducted a series of 16 simulations in which we changed only one variable, *Beta* and fixed the gradient slope and cluster size (*p, q)*. The graph below in Figure 3.14 shows that as the *Beta* angle was increased from 10° to 150° the cluster remained in a more stable region and was able to follow the desired value with less heading error. This relationship between the Settling RMS error and the *Beta*, leads to larger differences on both ends of the *Beta* range in the "z" or sensed gradient measurement field. As the *Beta* approaches either 0° or 180° the system becomes unstable due to previously reported kinematic singularities [2], and the *Theta* error starts to increase. The valley or lower point on the graph indicates a region of stability and can be used as a guideline in order to choose the

correct parameters without overstepping into the unstable regions.

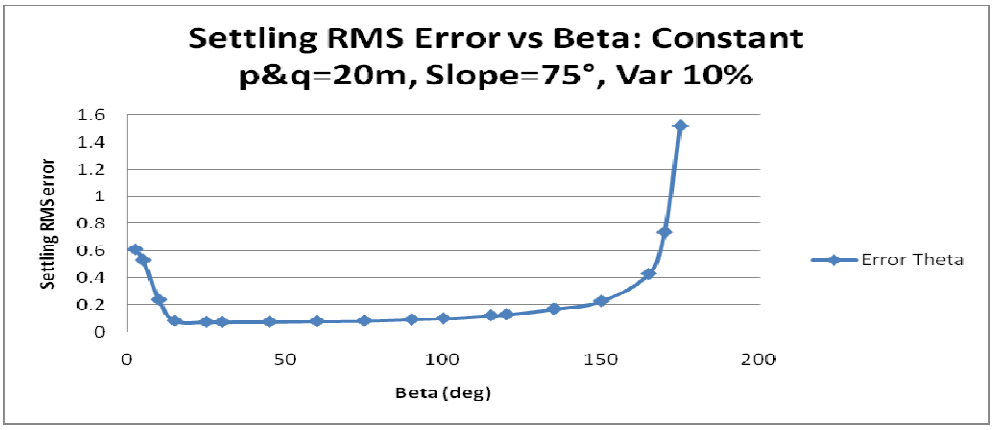


Figure 3.14-*Characterization settling RMS error vs Beta*

3.4 – Examples of Complex Contour Following

Previously shown simulation results have focused on a simple planar gradient field in order to demonstrate functionality and characterize performance in as simple an environment as possible. However, via simulation, it is interesting to explore contour following behavior for more complex gradient fields. To begin, a simple, single-peak, circular cross-section Gaussian field (representing a point source) was created, as expressed in Eq: $z=(x^2+y^2)+100$

Cluster size and shape were set to [p,q,β]=[10m,10m,90°]; these values were chosen according to the size and aspect ratios that where characterized in the previous experiments. The desired contour value for the cluster was set to 90 units, and the cluster was positioned at an initial value of approximately 160 units within the parameter field. As seen in Figures 3.15, the cluster successfully drives down the parameter field and settles at the desired contour level. The desired cluster heading value, which represents the contour bearing estimate, decreases as the cluster moves counter-clockwise about the gradient field; the actual cluster heading converges to this value as the cluster nears the desired contour value. Minor transients in the cluster size can be seen, and they appear to be an artifact of the cluster heading moving between quadrants; this issue was deemed minor and will be resolved in future work.

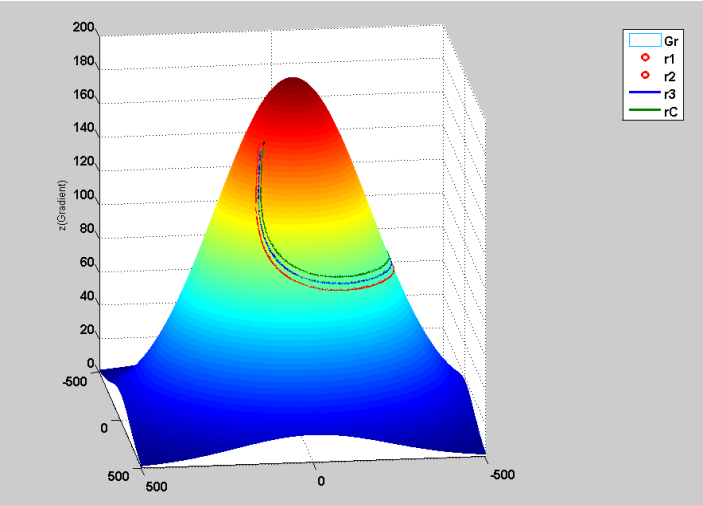


Figure 3.15-*Simulation Gaussian gradient following 3-Bots*

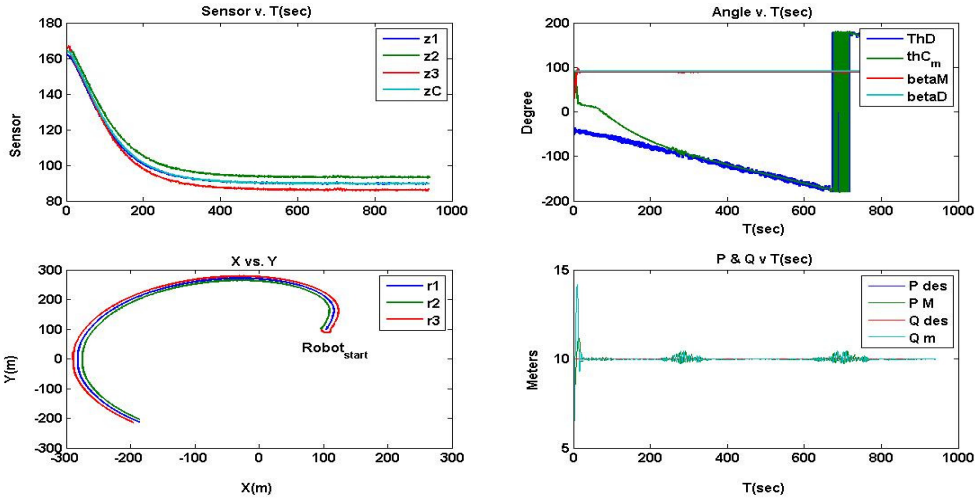


Figure 3.16-*Simulation data,3-bots circling a Gaussian gradient*

A second complex field was explored, this time using a multi-peaked Gaussian parameter field. When the Gaussian field has one or more peaks, saddle points and local maxima will develop. This is an area of research that was explored briefly, but how the robot cluster could deal with these separate multiple maxima and minima locations is beyond the scope of this paper.

For the field shown in Figure 3.17, an initial controller was implemented in order to

maintain the cluster at a specific contour in the field. As can be seen in Figure 3.18, this was successfully achieved, with the cluster maintaining its size and shape as it traversed the field.

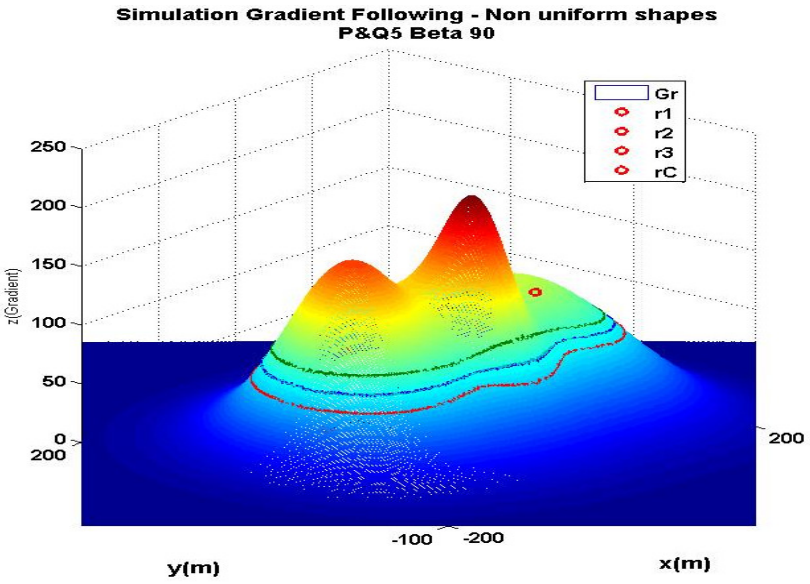


Figure 3.17-*Simulation Gaussian field saddle points*

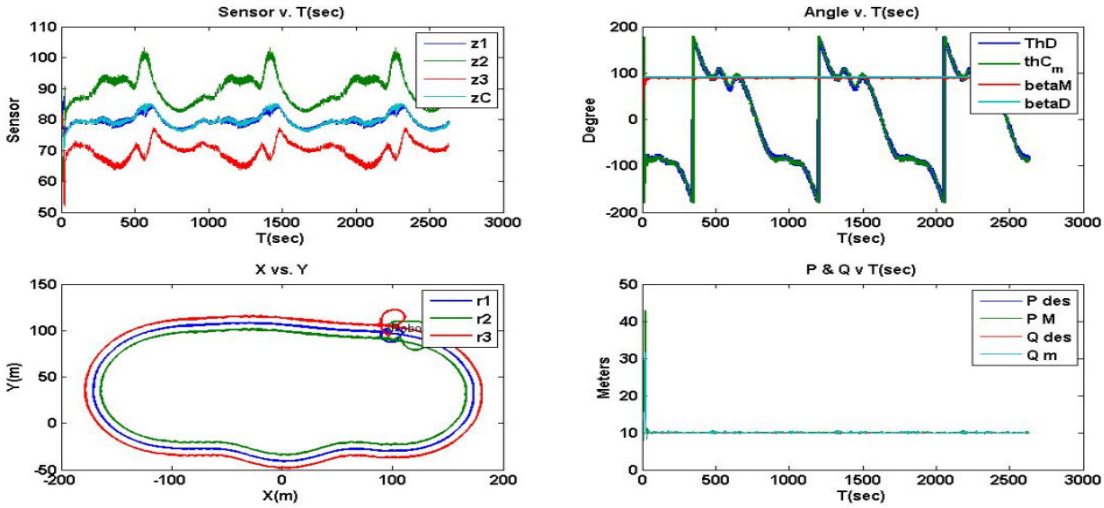


Figure 3.18-*Dimensional plot double Gaussian field without noise*

Using the same multi-peak parameter field, an additional simulation was run in which the desired contour level was slowly incremented by a step size of 10 units; this was done using the same shape and size as before: [p,q,β]=[10m,10m,90°]. Figure 3.18 shows how

the cluster centroid moves over time: first, it slowly circumnavigates the entire field, then it moves between the two peaks (with its inertia carrying it across the saddle point), and then it moves exclusively up one of the peaks (which peak depended on initial conditions; it is noted that there currently is no guarantee that it will explore the maximum peak). Figure 3.19 shows the complete set of time responses, which show the rise in sensed values, the maintenance of cluster shape and size, etc. Figure 3.19 specifically shows the ability of the cluster to follow the estimated field contour.

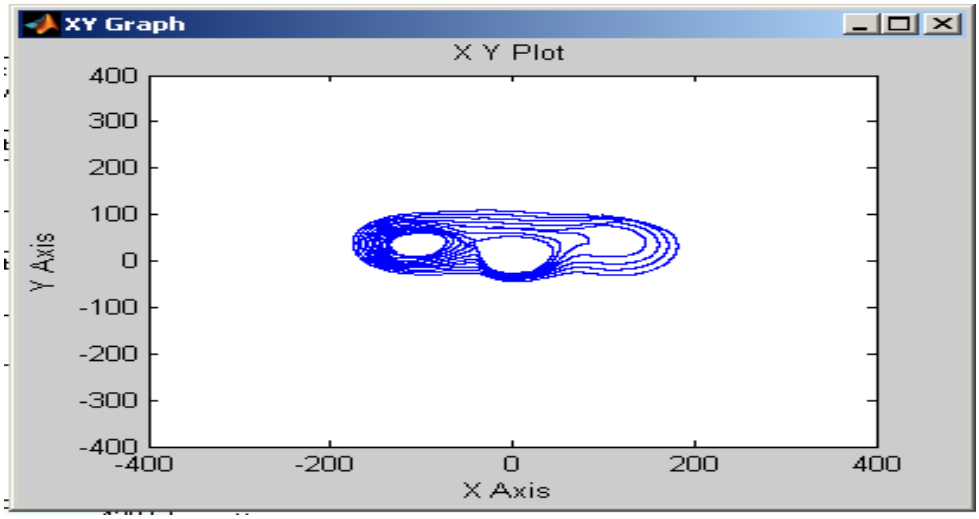


Figure 3.19-*Gradient following in a saddle point Gaussian field*

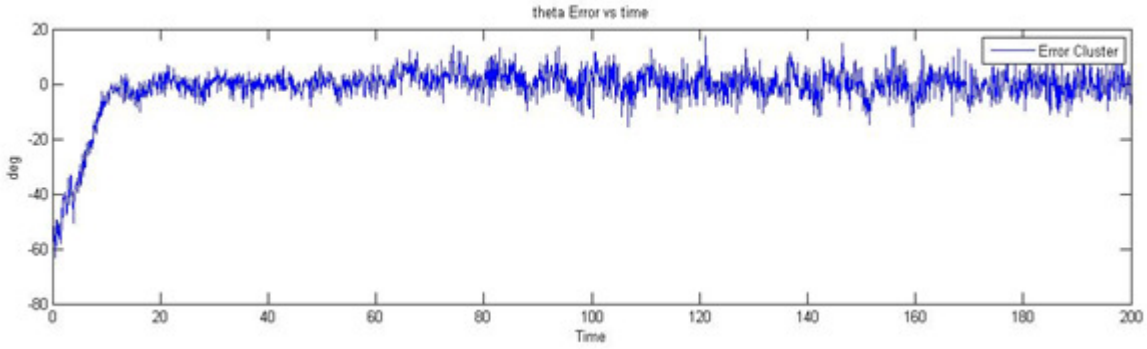


Figure 3.20-*Settling Mean Theta error vs time plot*

4.0 Experimental Testbeds

A cluster control testbed provides experimental capabilities for multi-robot command-and-control and collaboration experiments. Santa Clara University students have developed, and over the years upgraded, the testbed to support a variable number of robots that can be controlled using the internet or a centralized computer. In order to conduct the experiments described in this paper, two different testbeds were used. The Pioneer P3-ATs from Mobilerobot, were operated on a large grass field. In addition, custom powered kayaks were utilized in a local bay. These testbed platforms have been successfully used in the past to demonstrate a variety of 3-Robot cluster experiments.

Figure 4.1-*P3-AT robots*

These testbeds were used to experimentally verify the cluster space control approach and to support further testing and development for future applications. The testbeds share a common electronics architecture that is shown on top of the robots in Figure 4.1. This includes all communication and navigation components for each robot in the cluster. Minor modifications in software due to hardware variations are required when migrating between the marine and land platforms. The use of a common bus architecture across all Robotic Systems Lab (RSL) cluster vehicles enables a rapidly reproducible control system capable of transparently controlling multiple platforms, for land, sea, and air applications.

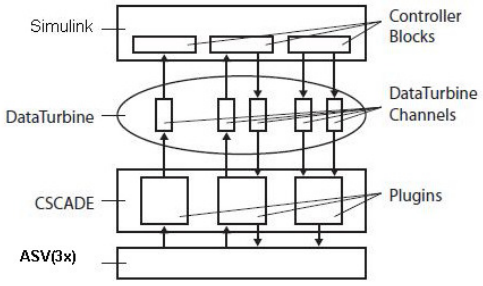


Figure 4.2-*Software flow*

Control software, written in Matlab/Simulink, is integrated with the rest of the system software through the RBNB Data Turbine, an open-source network management program used to route data packet streams over a network and to provide a unified view of both static and streaming data. Shown in Figure 4.2 is a diagram of the software flow. In the test bed system, the Data Turbine is the layer between MatLab/Simulink that handles telemetry data from the robots. In the broad perspective it is used to allow connections to and from anywhere with internet access. For the testing purposes of this thesis, all of this was done on a local computer. CSCADE is middleware used as an open architecture to support data flows from different systems; its purpose is to provide the users with a general framework which has system specific plug-ins to control the flow of telemetry. In order to verify the preliminary work, we have run our experiments with simulated environmental sensors with plans in the future to incorporate a real-time feedback sensor network on the robots. The simulated environmental sensor readings and configuration were incorporated into the specific properties of measurable environmental variables, such as a depth, temperature and other field parameters. These physical measured properties may be used in future work to develop and test the robot navigation control system.

4.1 ASV Testbed and Results

The ASVs used for this testing were chosen to demonstrate a control technique application with robots that have significant physical real world vehicle dynamics. This

coupled along with variable external forces such as wind and current presents cases that the cluster formation and controller may face in the field environment. This testbed will be the first step in the proposed real-world application that eventually will use the ASVs and this technique for monitoring of the marine environment.

4.1.1 Description of the ASV testbed

The kayak design is, by intent, very simple, allowing basic maneuverability and functionality for the purposes of verifying and validating multi-robot cluster control techniques as seen in Figure 4.3. By only including the components essential for full functionality, the ASVs are provided with a practical design that can be expanded to extend functionality as necessary.

Figure 4.3-*Field operation of 3 kayaks*

While traditional kayaks are often designed to be longer and narrower to improve long distance strait-line paddling, the eight-foot sit-on-top style kayaks, which can be seen in Figure 4.4, were selected for their wide ultra-stable flat hull and low cost as well as their maneuverability with dual trolling motors. At only eight feet long and thirty inches wide, the Dragonfly is designed to fit securely inside any standard SUV, station wagon or minivan for security during travel [33].

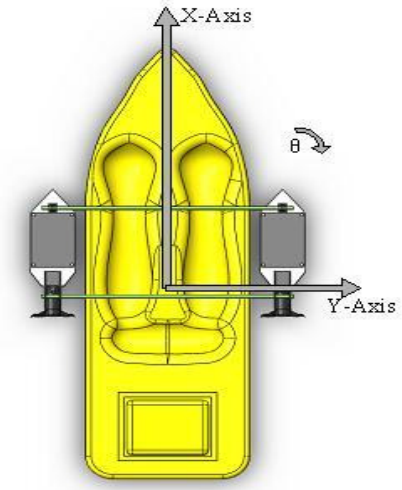


Figure 4.4-*ASV Frame (Z-axis is into the page, RH rule)*

Figure 4.5-*ASV Kayak platform*

The on-board computing stack is made up of two BasicX microcontroller boards. The BasicX-24p a versatile BASIC programmable microcontroller. Its design provides a powerful module capable of fitting in compact applications while still having multi-thread capability and enough onboard memory to carry out non-trivial tasks. The only drawback of the BasicX is the limited number of I/O ports, limiting future expansion of navigation sensors and feedback control. Figures 4.6-4.7, shows the functional and component block diagrams of the configuration of the communication, virtual data and power flow for the ASVs. The power section and common bus architecture have been condensed here and details can be found in previous work [33].

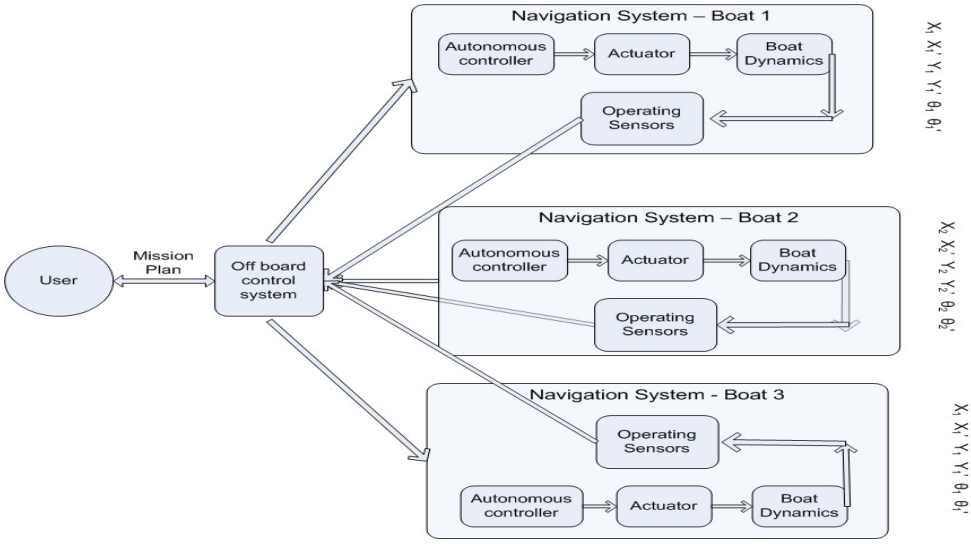


Figure 4.6-*Functional block diagram 3 robot cluster*

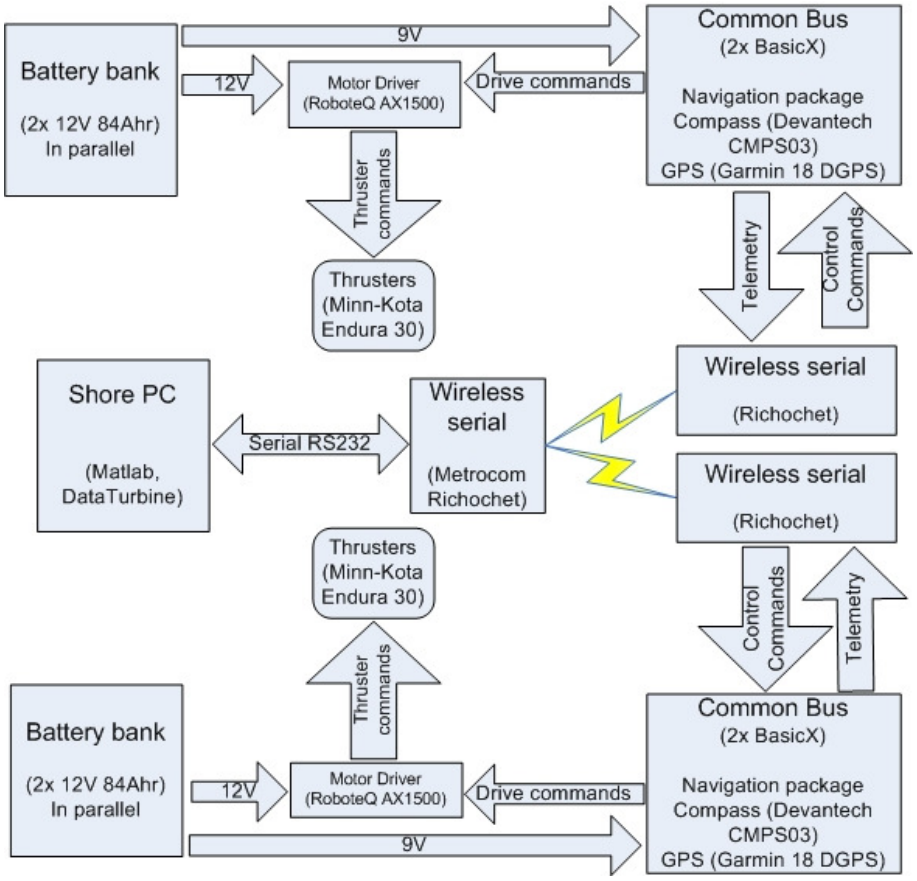


Figure 4.7-*Common ASV component level diagram*

4.1.2 Experimental Results

As mentioned previously, the simulated sensor readings and sensor configuration incorporated the specific properties of measurable environmental variables such as a depth, temperature and other field strengths.

Figure 4.8-*Kayak testbed- Stevens Creek*

In this experiment, the simulated sensors have been overlaid with a virtual gradient field in the specified operational region and have been assigned a depth (m) reading to aid in the understanding and interpretation of the results. Figure 4.8 above shows the ASVs in a real test bed environment.

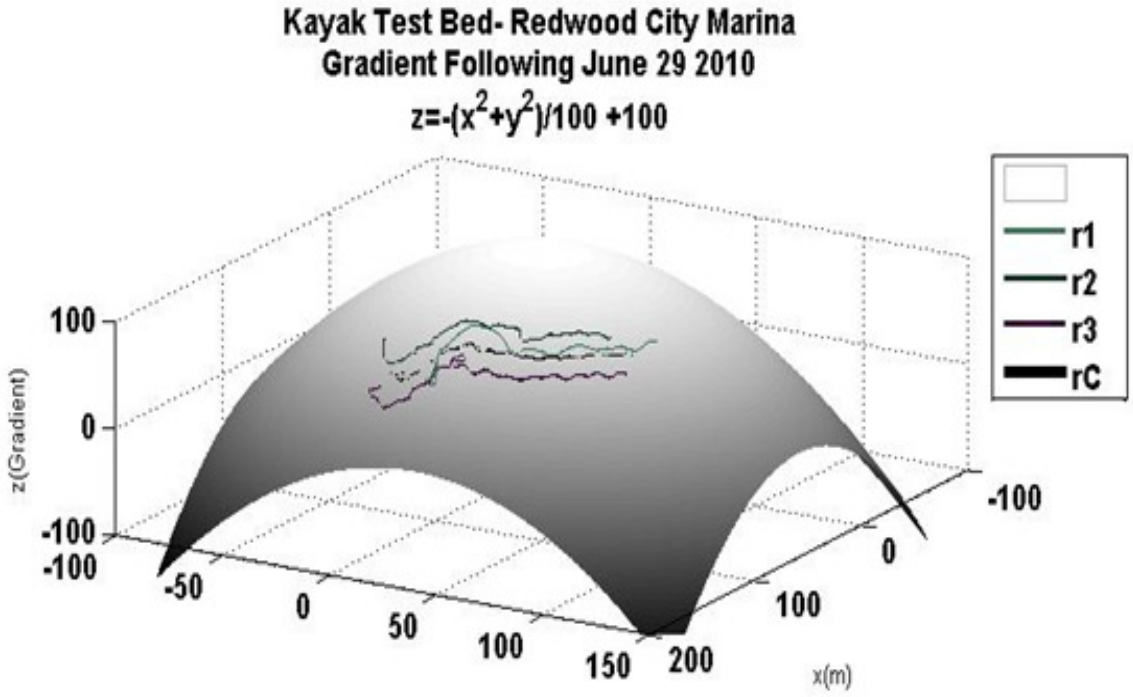


Figure 4.9-*Kayak gradient following Redwood city Gaussian Field 3-D*

Further field work progressed towards verifying the controller algorithm and functionality of a more complex system. For the first test, a three-dimensional virtual Gaussian field was created using Matlab algorithm and was inserted into the controller as seen in Figure 4.9 above. This was used to duplicate a point source that could potentially be found in a natural environment. The robot cluster was started on the virtual gradient field in an arbitrarily chosen heading. The cluster then moved towards the specified values while the distance between the robots and trajectory was controlled. The cluster would continue tracking the gradient thereby encircling the point source.

The gradient field was $z = (x^2+y^2)/100+100$, resulting Gaussian field and contours about

the center. For the first simulation using this field, shown in Figure 4.8, the cluster of robots was commanded to follow a contour of z=50 units while maintaining a shape of [p,q,β]=[20m,20m,60°]. The cluster was initially positioned at approximately 20 units in the parameter field with a heading of approximately 40° and a shape of [p,q,β]=[35m,25m,49°]. As can be seen in the figure, after an initial transient, the cluster maintains its shape while moving up the gradient until it drives along the contour at z = 50 units.

In a simple case, this method works and the gradient can be followed if the values of *p&q* and the slope of the gradient is approximately known. In comparison to the simulation noise, the RMS error becomes much less prominent and the robot cluster trajectory is a smooth line.

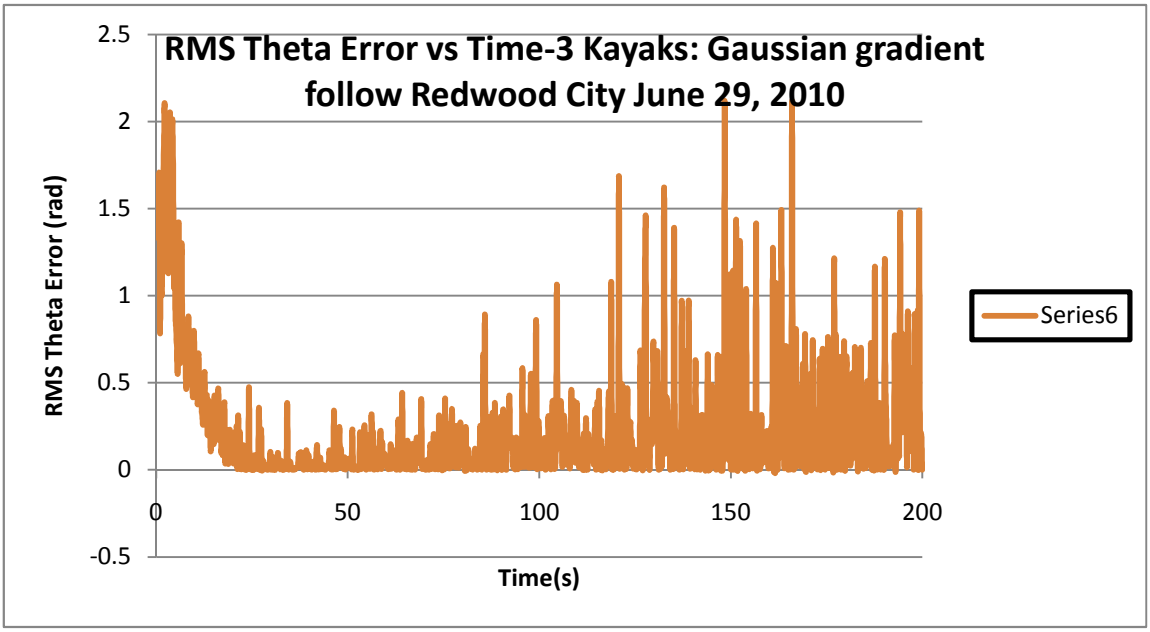


Figure 4.10-*RMS Error of the Kayaks in a Redwood City field test (p&q=20)*

The accuracy of measurement in this system is determined by the degree of closeness of the desired value/angle versus the actual value measured in radians. As shown above in Figure 4.10, the RMS error is calculated to be 0.47 radians.

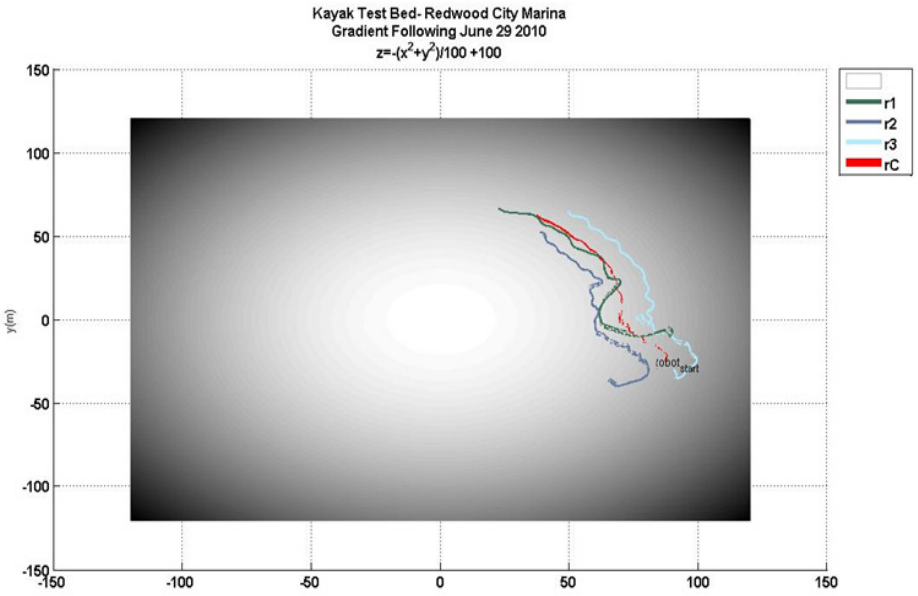


Figure 4.11-*Kayak gradient following, Redwood city Gaussian field 2-D data plots*

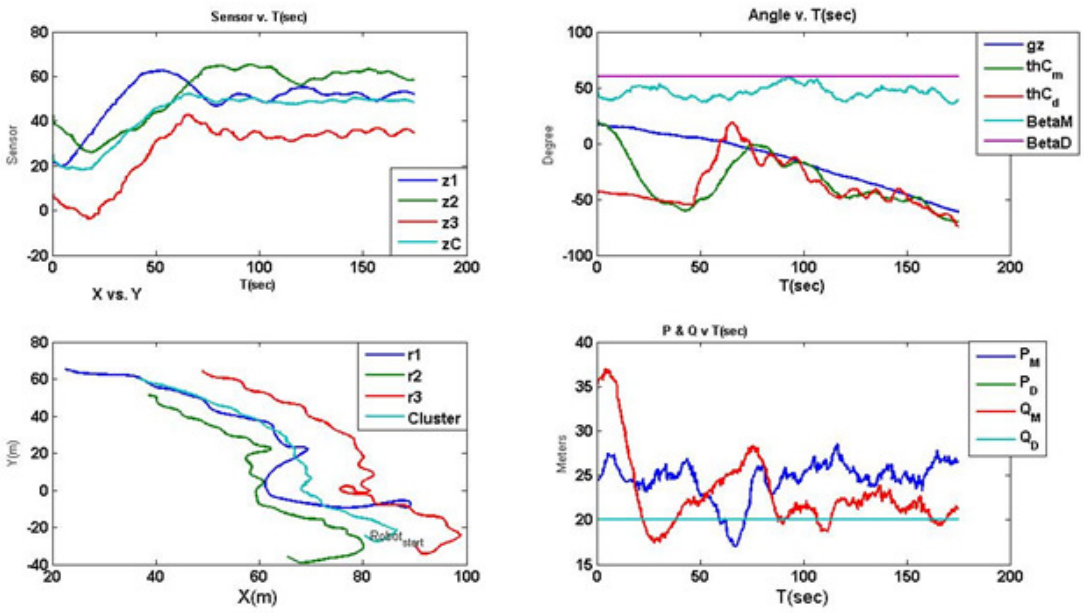


Figure 4.12-*Kayak gradient following Redwood city Gaussian field 2-D data plots*

The results correspond to the simulations with the addition of noise on the outer bounds of the noise characterization. This showed that the model was a good approximation of

environmental variables and that the controller was able to handle the field conditions as expected. Further work is required to properly tune the system and improve performance.

This test shown above in Figures 4.8 thru 4.12, combined the total relationships between the field strength of the parameter field, (ie the sensed values that the robots are tracking), with Beta (β) and cluster size ($p\&q$). Testing showed that the robot cluster oriented in the correct direction, and followed along the specified contour. As it became more stable, it was able to follow the desired value around the Gaussian field without losing spatial integrity.

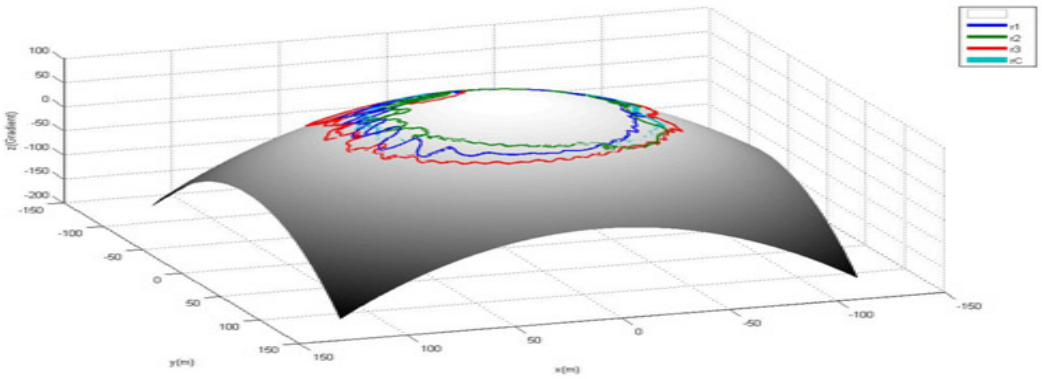


Figure 4.13-*Kayak gradient following virtual Gaussian circle, Redwood City*

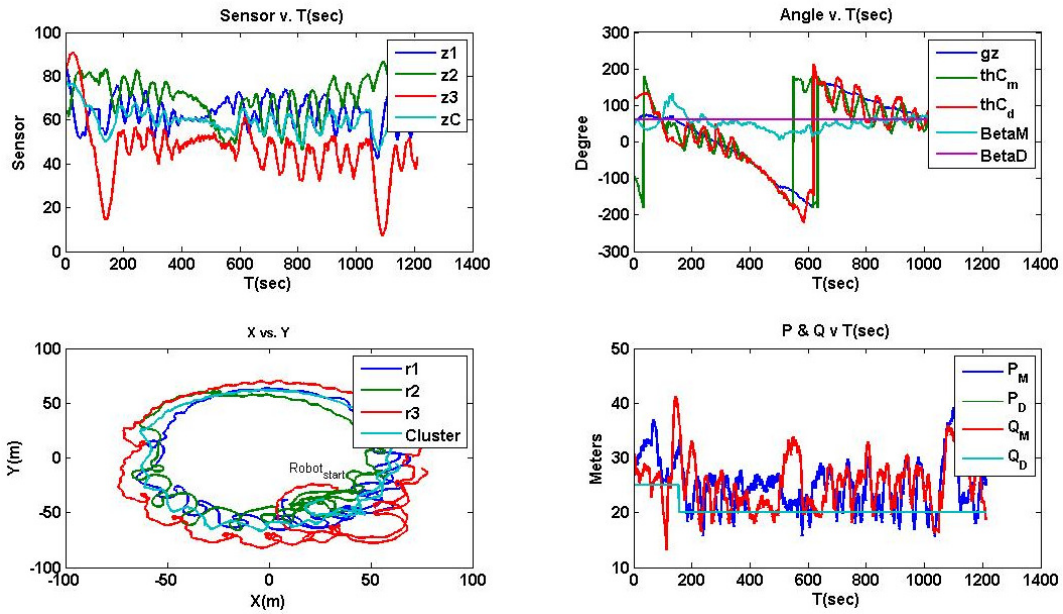


Figure 4.14-*Kayak Gradient following virtual Gaussian Map Redwood City*

The second field test with kayak robots was completed by using a Gaussian field configuration to verify that a cluster can change orientation and maintain control in a slightly more challenging gradient field. The gradient field was $z = (x^2+y^2)/200$, resulting in a field point source field with a contour in a circular pattern about the center. For the test, as shown in Figures 4.13 and 4.14 the cluster of robots was commanded to follow a contour of z= 60 units while maintaining a shape of [p,q,β]=[30m,30m,75°]. The cluster was initially positioned at approximately 75 units in the parameter field with a heading of approximately 45° and a shape of [p,q,β]=[20m,35m,60°]. As seen in the figure, after an initial transient, the cluster maintains its shape while estimated to be moving on the gradient until it drives along the z = 60 units contour with visible disturbances present.

The test shown above in Figure 4.14 was a field example of a virtual gradient that the cluster was commanded to follow. The robot cluster is able to track the gradient in a complete circle and arrive at the same location, while dealing with the winds and currents at the test site. The use of a dynamic controller to handle the outside variables could help improve the results and is currently being investigated [43].

4.2 Pioneer Testbed and Results

In order to have a more complete understanding of the control system, the Pioneer rovers were chosen to demonstrate the technique using a different style of robots. These robots can be modeled as a first order system and have little or no interference from the outside environment, making them easier to work with when applying the technique.

4.2.1 Pioneer Testbed

The Pioneer 3-At robots make up a versatile four-wheel differential drive robotic platform, with the robots capable of linear translation speeds up to 0.8m/s and rotational speeds 300◦/s. They can receive commands and send out telemetry over a 900MHz radio link. The communication link preserves data integrity, but it does not guarantee packet delivery.

Figure 4.15-*Pioneer 3AT robot testbed*

Students at SCU have designed custom sensors and communication subsystems consisting of a Garmin 18-5hz differential GPS unit, a digital Devantech compass and a Ricochet 128Kbits/s radio modem. These subsystems are controlled by BasicX microcontrollers linked through RS-232 interfaces. The package is capable of outputting telemetry at a 5Hz rate with a range of approximately 2 miles in clear/ideal conditions.

The system architecture is almost identical to the kayaks. The motor controllers and software in the stacks have been modified slightly to adopt different type of configurations. The remaining bus hardware, data handling and communications systems are identical to the kayak setup, as seen in the previous Figures 4.6-7.

4.2.2 Pioneer Results

In terms of performance and stability, the land robots proved to be slightly easier to manage due to the fact that they would remain in a set position when not commanded and had little to no dynamic external influences. Testing results showed improved performance in terms of control and stability as compared directly to the ASV testing that was completed on the water. The starting and stopping of the cluster in the correct formation was important in order to duplicate the simulation results.

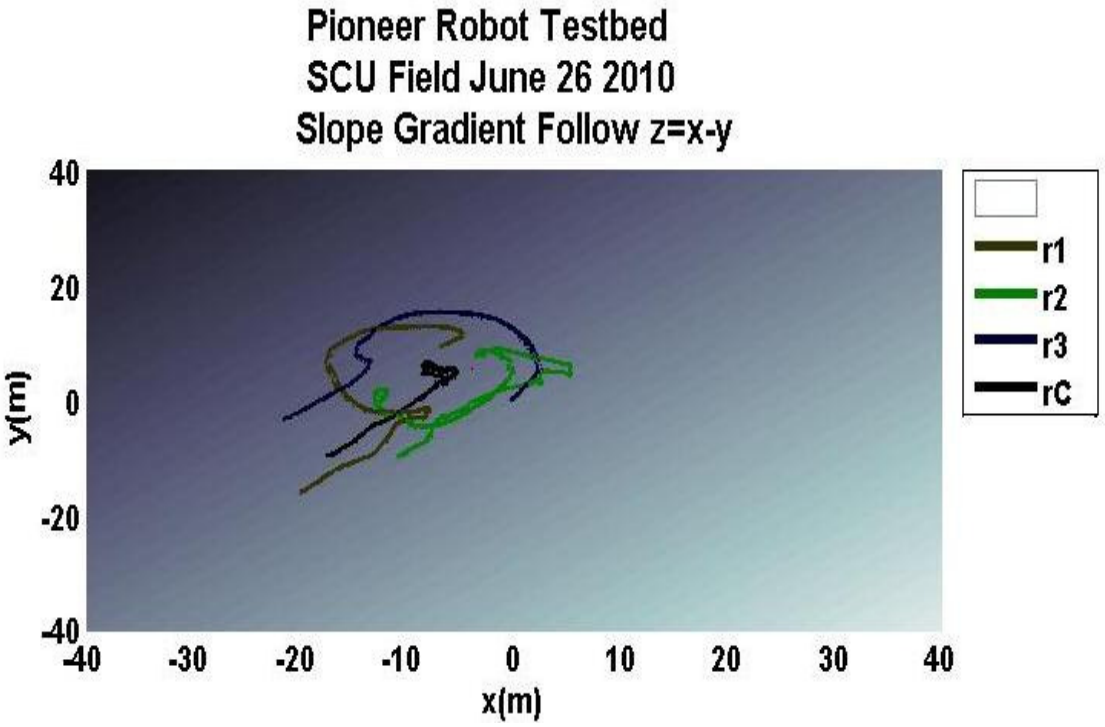


Figure 4.16-*3 cluster rotation & gradient following field test (Beta 90,p&q=12m)*

The first field test with pioneer robots was to set up in an offset triangular configuration to verify that a cluster can change orientation and follow the basic control commands. The gradient field was z =x-y, resulting in a field with an angle of 45° and contours in the same direction. For the first simulation using this field, shown in Figure 4.16 the cluster of robots was commanded to follow a contour of z=(-11) units while maintaining a shape of [p,q,β]=[12m,12m,90°]. The cluster was initially positioned at approximately (-9) units in the parameter field with a heading of approximately 10° and a shape of [p,q,β]=[9m,12m,120°]. As can be seen in the figure, after an initial transient, the cluster maintains its shape while moving up the gradient until it drives along the z = (-11) units contour.

Figure 4.17 shows that the sensor values converge as the robot cluster rotates and moves along the virtual gradient. The *p&q* values show control by approaching the 12 meter specified input and holding that value for the given test period in the Figure 4.17 below.

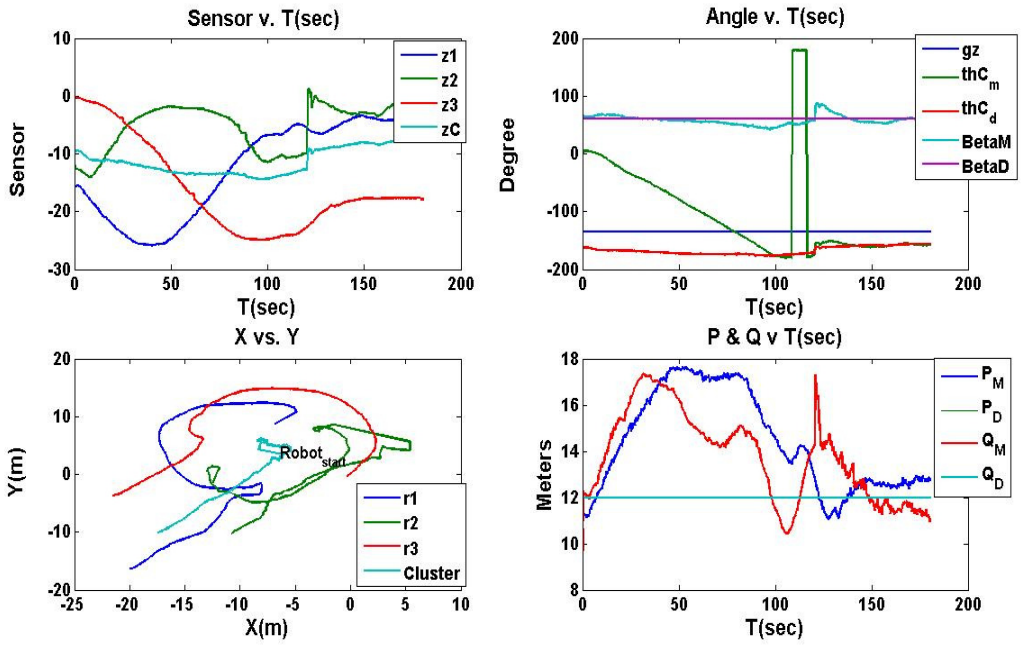


Figure 4.17-*3 cluster rotation & gradient following test error(Beta=90,p&q=12m)*

The accuracy of measurement in this series of tests is calculated in a similar manner to the tests completed using the kayak test bed. The RMS heading error was determined by the degree of closeness of the desired value/angle vs the actual value measured in radians.

The second field test with pioneer robots was to set up Gaussian configuration to verify that a cluster can change orientation and follow the basic control commands in a slightly more challenging gradient field. The gradient field was $z = (x^2+y^2)/100 + 100$, resulting in a field point source field with a contour in a circular pattern around the center. For the test, as shown in Figure 4.18, the cluster of robots was commanded to follow a contour of $z = 85$ units while maintaining a shape of $[p,q,\beta]=[12m,12m,75°]$. The cluster was initially positioned at approximately 100 units in the parameter field with a heading of approximately 10° and a shape of $[p,q,\beta]=[9m,14m,30°]$. As seen in the figure, after an initial transient, the cluster maintains its shape while moving along the gradient until it drives along the $z = 85$ units contour.

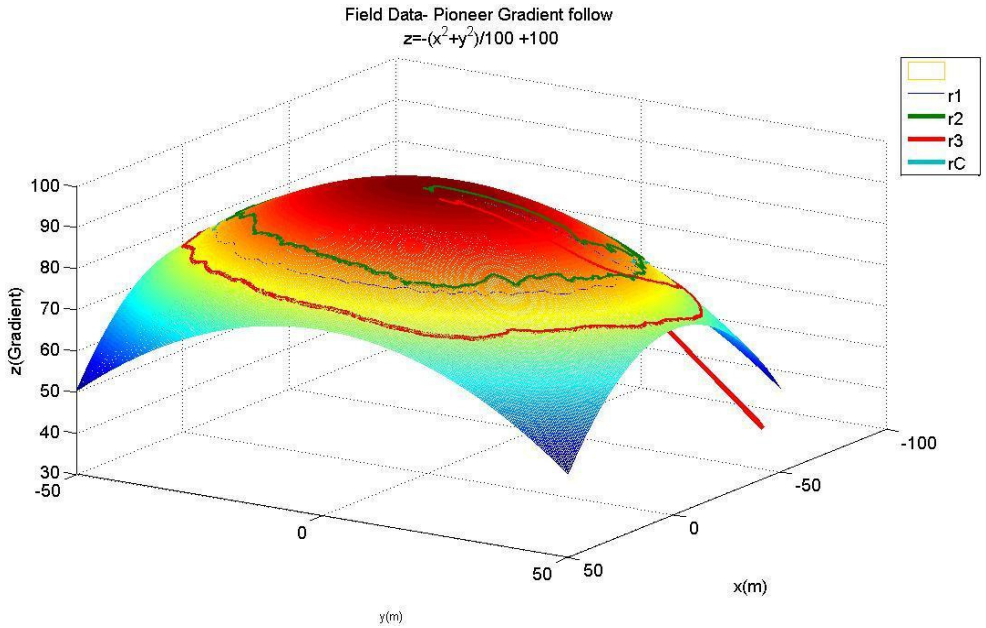


Figure 4.18-*Gradient following of a virtual Gaussian field-3D (β45°,p&q=12m)*

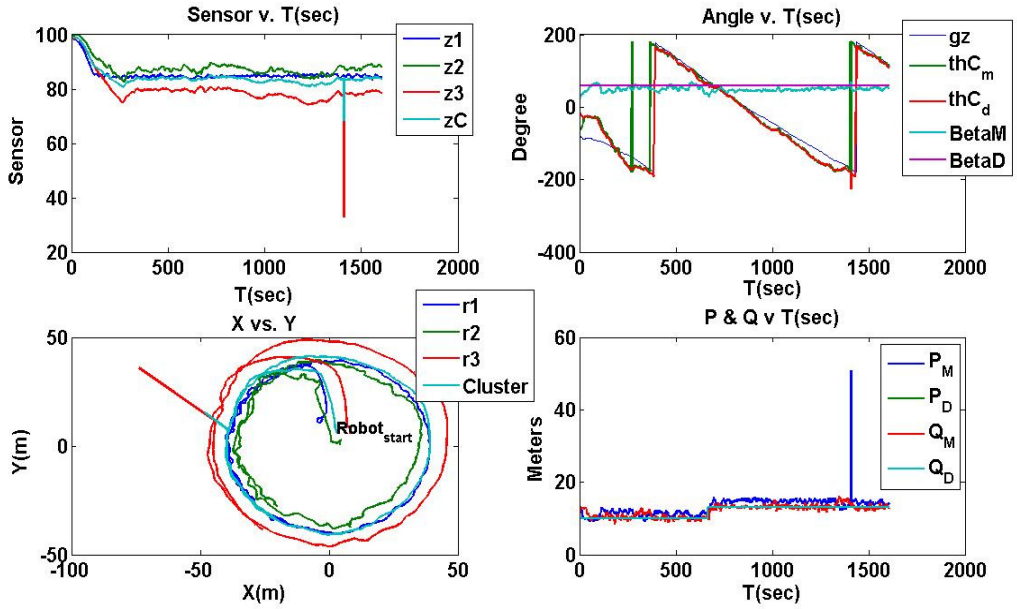


Figure 4.19-*Gradient following of a virtual Gaussian field (β45°,p&q=12m)*

As seen above in Figure 4.18, the robot cluster rotated towards the desired value point and then moved together in formation forming a path seen in the lower left of the Figure 4.19. The *p&q* values dropped as expected, which showed the controller as functioning properly with regards to maintaining the correct distance, velocity and angle of the cluster.

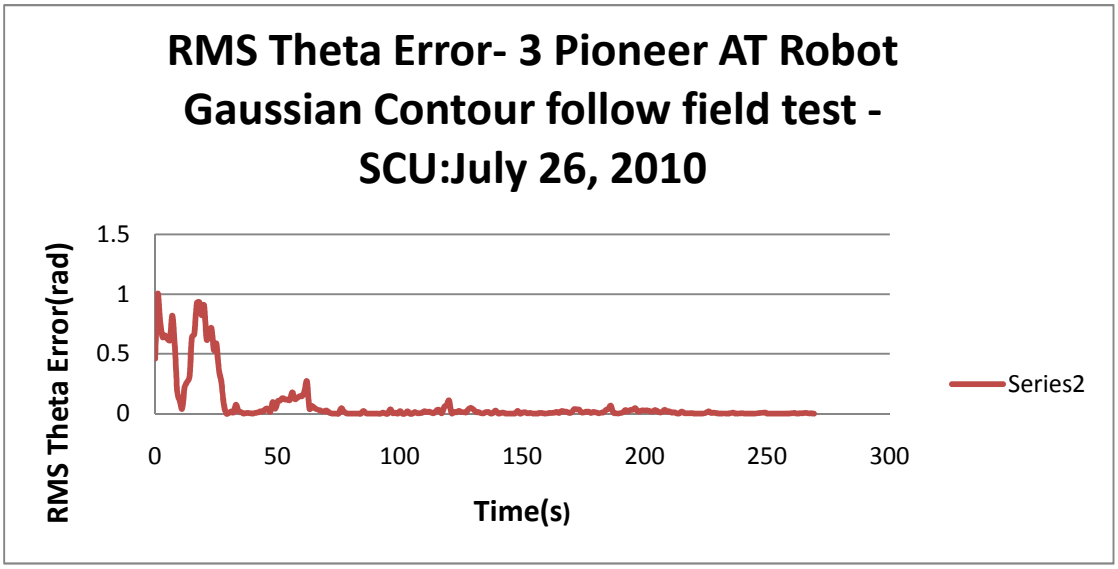


Figure 4.20-*RMS Error of the Pioneer Robots-SCU field test (β45°,p&q=12m)*

As shown above in Figure 4.20, the RMS error is calculated to be 0.27 radians. The results correspond to the simulations with the addition of noise on the outer bounds of the noise characterization. In comparison to the kayak contour following results (RMS error 0.47), the RMS error is less as the robot cluster controller maintains the specified requirements.

4.3 Testing summary

The tests performed in this chapter have shown that the cluster space controller used for gradient contour following functions works on two separate platforms. The controller enabled the cluster to sense and follow a contour-based trajectory in a parameter field using both a kayak cluster formation and also the land based Pioneer robots. Tracking errors were maintained within acceptable accuracy margins.

5.0 Conclusion

This research has successfully demonstrated a cluster space control technique that is capable of controlling a cluster of mobile robots to automatically track a contour in a parameter field. A simplified gradient estimation function appropriate for a cluster of three planar robots was derived and incorporated into the cluster space control architecture to enable contour following. Using this controller configuration, 3-Robot clusters were capable of utilizing virtual sensor data to track contours in a gradient field that were unknown to the robots *a priori*; this was achieved with a cluster of boats as well as with a cluster of land rovers. The test runs with the boats demonstrated the viability of the cluster control approach for vehicles affected by significant dynamics and disturbance forces.

The testing of a virtual gradient test bed with full scale Pioneer and kayaks, operating in their natural environment, showed that the cluster space control technique was successfully implemented in the field. We believe that this control configuration will lead to enhanced capabilities for real-world marine applications. It will also lead to cost-effective improvements in operating such systems through the reduction of the operator/robot ratio required to control such systems. Furthermore, our initial characterization of performance, based on sensor noise and cluster parameters, provides initial guidance to designers and operators to successfully manage this technique in the field.

5.1 Future Work

The cluster space control technique developed in this thesis has a great deal of potential. This technique focuses on the characteristics and applications of cluster motion from the point of view of the operator. Future work might include modifications to the existing cluster definition in order to reduce or eliminate the effects of singularities. Other work may include the extension of the cluster definition to include more robots and create a more robust system to deal with the irregularities of sensor networks. For large clusters, a hierarchical multi-robot specification technique, also known as “cluster of cluster”

control approach, has been in development for applications in concurrence with this research. This approach is currently undergoing experimental verification. Thus far, the results of the 6-Robot cluster have been very promising.

Future work to consider would be the addition of this cluster space control technique to a 3-D application such as blimps, helicopters, airplanes, or underwater robotic vehicles. Each type of vehicle may draw on different capabilities of the cluster; this adaptation may also lead to a better functioning of the cluster control technique in general. This would result in the improvement of sensing mapping.

This paper supports and highlights an alternative direction that multi-robot systems, operating in a three-dimensional workspace, may benefit from. Specific applications could include distributed sensing networks to monitor air or water quality, or distributed antenna systems capable of altering geometry upon command.

The potential benefits of a multi-vehicle configuration are shown in the first part of the paper in the context of gradient tracking. A controller design was developed and implemented using virtual sensors for this application. As the next step, further work might include real time data collection in the ASV environment, in such areas as natural thermal gradients, salinity in estuaries, and bathymetric surveys. We also plan to explore and address how to best manage the effects of the local maxima/minima values within a gradient field and how these affect performance. Another area of development to improve performance would be to increase the noise tolerance levels and to account for environmental factors that can affect the control system stability.

This paper has established a base line for the cluster space control technique performance and has indicated, in its conclusion, some areas of further development for practical applications.

References

[1] C. Kitts, Cluster Space Specification and Control of a 3-Robot Mobile System,2008 *IEEE International Conference on Robotics and Automation Pasadena*, CA, USA, May 19-23, 2008

[2] C. Kitts, I. Mass, Cluster Space Specification and Control of Mobile Multi-robot Systems, C.A.; Robotic Syst. Lab., Santa Clara Univ., Santa Clara, CA , *Mechatronics, IEEE/ASME Transactions* Volume: 14 Issue:2 On page(s): 207 – 218, April 2009

[3] C. Ortiz, K. Konolige, R. Vincent, B. Morisset, A. Agno, M. Eriksen, D. Fox, B. Limketkai, J. Ko, B. Steward, and D.Schulz, Centibots: Very large scale distributed robotic teams, *Proceedings 2004 Sixteenth Innovative Applications of Artificial Intelligence Conference*, pp. 1022-1023, 2004

[4] E. Fiorelli, N.E. Leonard, P. Bhatta, D. Paley, R. Bachmayer, and D.M. Fratantoni, Multi-AUV Control and Adaptive Sampling in Monterey Bay. *Proceedings of IEEE Autonomous Underwater Vehicles 2004: Workshop on Multiple AUV Operations.* Sebasco, ME. June 2004.

[5] J. Fredslund and M. J. Mataric, A general algorithm for robot formations using local sensing and minimal communication, *IEEE Trans. Robot. Automat*., vol. 18, no. 5, pp.837-846, October 2002

[6] A. K. Das, R. Fierro, V. Kumar, J. P. Ostrowski, J. Spletzer and C. J. Taylor, A vision-based formation control framework, *IEEE Trans. Robot. Automat*., vol. 18, no. 5, pp. 813-825, October 2002

[7] Kumar, Controlling formations of multiple mobile robots, *IEEE Intemational Conference on Robotics & Automation Leuven*, Belgium, May 1998

[8] A. E. Magurran and T. J. Pitcher, Foraging, timidity and shoal size in minnows and goldfish, *Behavioral Ecology and Sociobiology* Volume 12, Number 2, pg147-152,

[9] Skellam, J.G. Studies in statistical ecology. I. Spatial pattern, *Biometrica*, 39, 346-362, 1952

[10] E. Burian, D. Yoerger, A. Bradley, and H. Singh.Gradient search with autonomous underwater vehicle using scalar measurements. *Proceedings of the IEEE OES AUV conference*, Monterey, California, June 1996.

[11] C. Kitts, personal communication, September 15, 2010

[12] J. Adler, Chemotaxis in bacteria. *Science*, 153(3737):708716, August 1966.

[13] R. Bachmayer, N. Ehrich Leonard, Experimental Test-Bed for Multi-Vehicle Control, Navigation and Communication, *12th International Symposium on Unmanned Untethered Submersible Technology*, 2001

[14] J. Choi, S.Oh and R. Horowitz, Cooperatively Learning Mobile Agents for Gradient Climbin, *46 IEEE Conference on Decision and Control*, LA, USA, Dec 12, 2007

[15] H. Ishidaa, K. Suetsugua, T. Nakamotoa and T. Moriizumia,Plume-Tracking Robots: A New Application of Chemical Sensors Faculty of Engineering, Tokyo Institute of Technology, 2-12-1 Ookayama, Meguro-ku, Tokyo 152, December 2001. Available online

[16] Willis, M. A., and E. A. Arbas.. Odor-modulated upwind flight of the sphinx moth, Manduca sexta L. J. Comp. *Physiol*. A169:pp. 427–440, 1991

[17] Kaissling, Orientation and Communication in Anthorpods, Switzerland,1997

[18] A. T. Hayes, A. Martinoli and R. M. Goodman. Distributed Odor Source Localization. *IEEE Sensors Journal*, 2 (3): pp. 260-271. 2002,

[19] AOSN Charter (2003). [Online]. Available: http://www.princeton.edu/ ~dcsl/aosn/documents/AOSN_Charter.doc

[20] N. J. Vickers and T. C. Baker, Reiterative responses to single strands of odor promote sustained upwind flight and odor source location by moths, *Proc. Nat. Academy Sci*., vol. 91, pp. 5756–5760, 1994

[21] J. H. Belanger and M. A. Willis, Adaptive control of odor guided locomotion: Behavioral flexibility as an antidote to environmental unpredictability, *Adap. Beh.,* vol. 4, pp. 217–253, 1996

[22] V. Howard, Gradient following Tracking for Mobile Multi-robot Systems . Kitts, Adv. M.S.Thesis, Dept. Mech Eng, Santa Clara University, Santa Clara, CA, 2010, In Draft

[23] S. Agnew, P. Dal Canto, C. Kitts, and S. Li, Cluster Space Control of Aerial Robots. IEEE/*ASME International Conference on Advanced Intelligent Mechatronics*, 2010

[24] K. Stanhouse, Cluster Space Obstacle Avoidance for Mobile Multi-Robot Systems. Advisor: C. Kitts. Santa Clara University Master of Science in Electrical Engineering Thesis, June 2006

[25] I. Mas, O. Petrovic, C. Kitts, Cluster Space Specification and Control of a 3-

Robot Mobile System, Proceedings - *IEEE International Conference on Robotics and Automation*, pp. 3763-3768, 2008

[26] R. Ishizu, The Design, Simulation and Implementation of Multi-Robot Collaborative Control from the Cluster Perspective, C. Kitts, Adv., M.S. Thesis, Dept. Elec Eng, Santa Clara University, Santa Clara, CA, December 2005. Available at http://rsl.engr.scu.edu

[27] P. Connolley, Design and Implementation of a Cluster Space Trajectory Controller for Multiple Holonomic Robots, C. Kitts, Adv., M.S. Thesis, Dept. Mech Eng, Santa Clara University, Santa Clara, CA, June 2006. Available at http://rsl.engr.scu.edu

[28] T. To, Automated Cluster Space Trajectory Control of Two Non-Holonomic Robots, C. Kitts, Adv., M.S. Thesis, Dept. Comp Eng, Santa Clara University, Santa Clara, CA, June 2006. Available at http://rsl.engr.scu.edu

[29] M. Kalkbrenner, Design and Implementation of a Cluster Space Human Interface Controller, including Applications to Multi-Robot Piloting and Multi-Robot Object Manipulation, C. Kitts, Adv., M.S. Thesis, Dept. Mech Eng, Santa Clara University, Santa Clara, CA, June 2006. Available at http://rsl.engr.scu.edu

[30] B. Tully, Cluster Space Piloting of a Nonholonomic Multi-Robot System, C. Kitts, Adv., M.S. Thesis, Dept. Comp Eng, Santa Clara University, Santa Clara, CA, June 2006. Available at http://rsl.engr.scu.edu

[31] I. Mas, O. Petrovic, and C. Kitts, Cluster space specification and control of a 3-robot mobile system, Proceedings of the IEEE International Conference on Robotics and Automation, Pasadena, CA, pp. 3763-3768. 2008

[32] E. Girod, Design and Implementation of Four Robot Cluster Space Control, C. Kitts, Adv., M.S. Thesis, Dept. Mech Eng, Santa Clara University, Santa Clara, CA, June 2008.

[33] P. Mahacek, I. Mas, O. Petrovic, J. Acain, and C. Kitts, Cluster space control of autonomous surface vessels, Marine Technology Society Journal, v 43, n 1, , pp. 13-20, 2009

[34] C. Kitts, K. Stanhouse, and P. Chindaphorn, Cluster Space Collision Avoidance for Mobile Two-Robot Systems, Proceedings IEEE/RSJ International Conference on Intelligent Robots and Systems, St. Louis MO, October 2009.

[35] I. Mas, S. Li, J. Acain, and C. Kitts, Entrapment/Escorting and Patrolling Missions in Multi-Robot Cluster Space Control, *Proceedings IEEE/RSJ International Conference on Intelligent Robots and Systems*, St. Louis MO, October 2009.

[36] M. S. Agnew, Cluster Space Control of a 2-Robot Aerial System, C. Kitts, Adv., M.S. Thesis, Dept. Mech Eng, Santa Clara University, Santa Clara, CA, June 2009. Available at http://rsl.engr.scu.edu.

[37] P. M. Dal Canto, Three Blimp Cluster Space Control: Derivation and Implementation, C. Kitts, Adv., M.S. Thesis, Dept. Mech Eng, Santa Clara University, Santa Clara, CA, June 2009. Available at http://rsl.engr.scu.edu.

[38] C. Kitts, personal communication, date**,** Sept. 7 2010

[39] I. Mas, J. Acain, O. Petrovic, and C. Kitts, Error characterization in the vicinity of singularities in multi-robot cluster space control, *Proceedings of the IEEE International Conference on Robotics and Biomimetics*, Bangkok, Thailand, pp. 1911-1917, December 2008,

[40] Dr. Christopher Kitts, Cluster Space Specification and Control of Mobile Multi-Robot Systems:Part 1 – Conceptual Framework Robotic Systems Laboratory, Santa Clara University

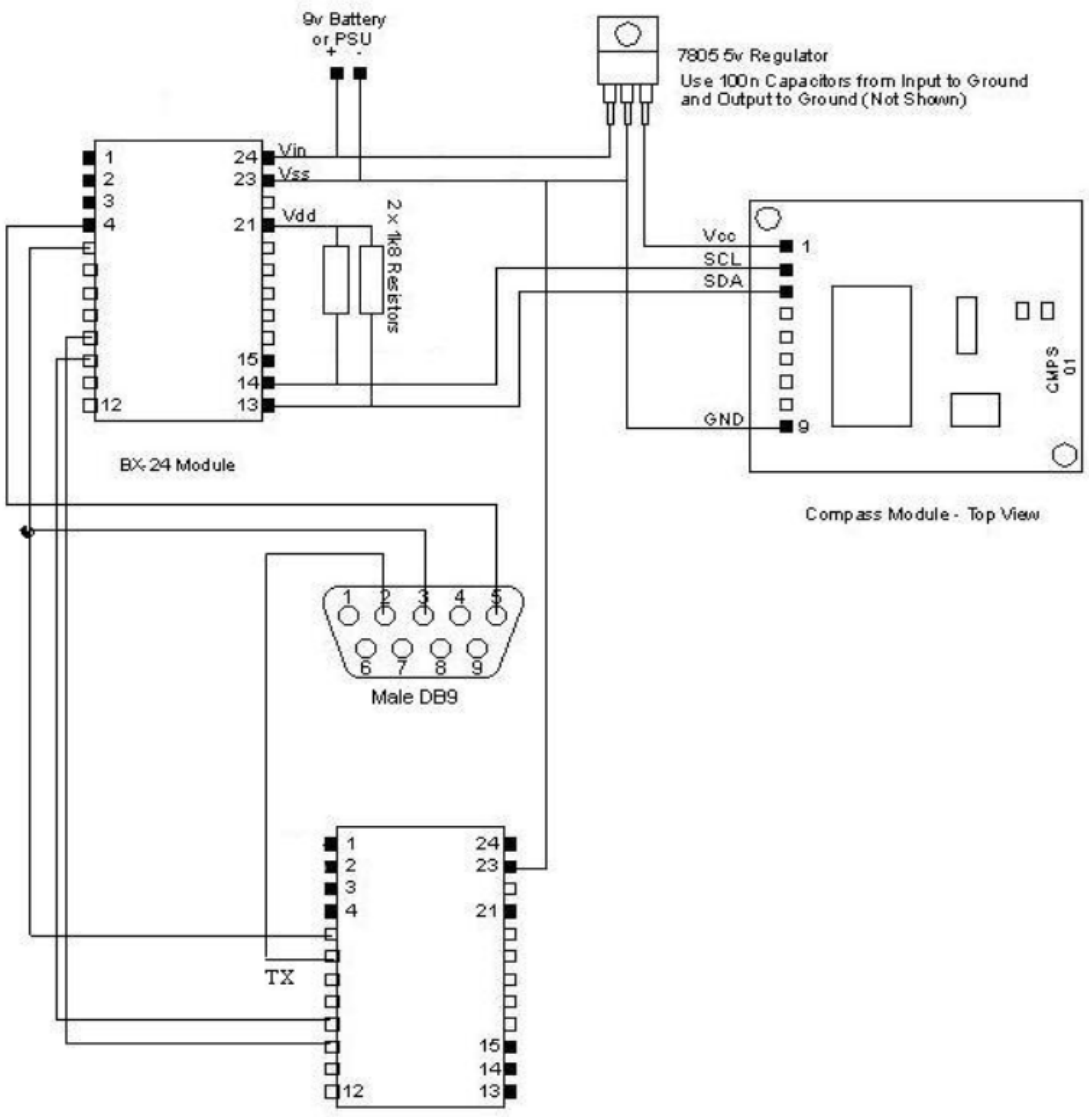
[41] R A Russell, D Thiel, Mackay-Sim Alan. Sensing odour trails for mobile robot navigation. In: *Proc. of IEEE International Conference on Robotics and Automation*, pp. 2672-2677, 1994

[42] Wei Li, Jay A. Farrell, Shuo Pang, et al. Moth-Inspired Chemical Plume Tracing on an Autonomous Underwater Vehicle. IEEE Transactions on Robotics, 22 (2): pp292-307, April 2006

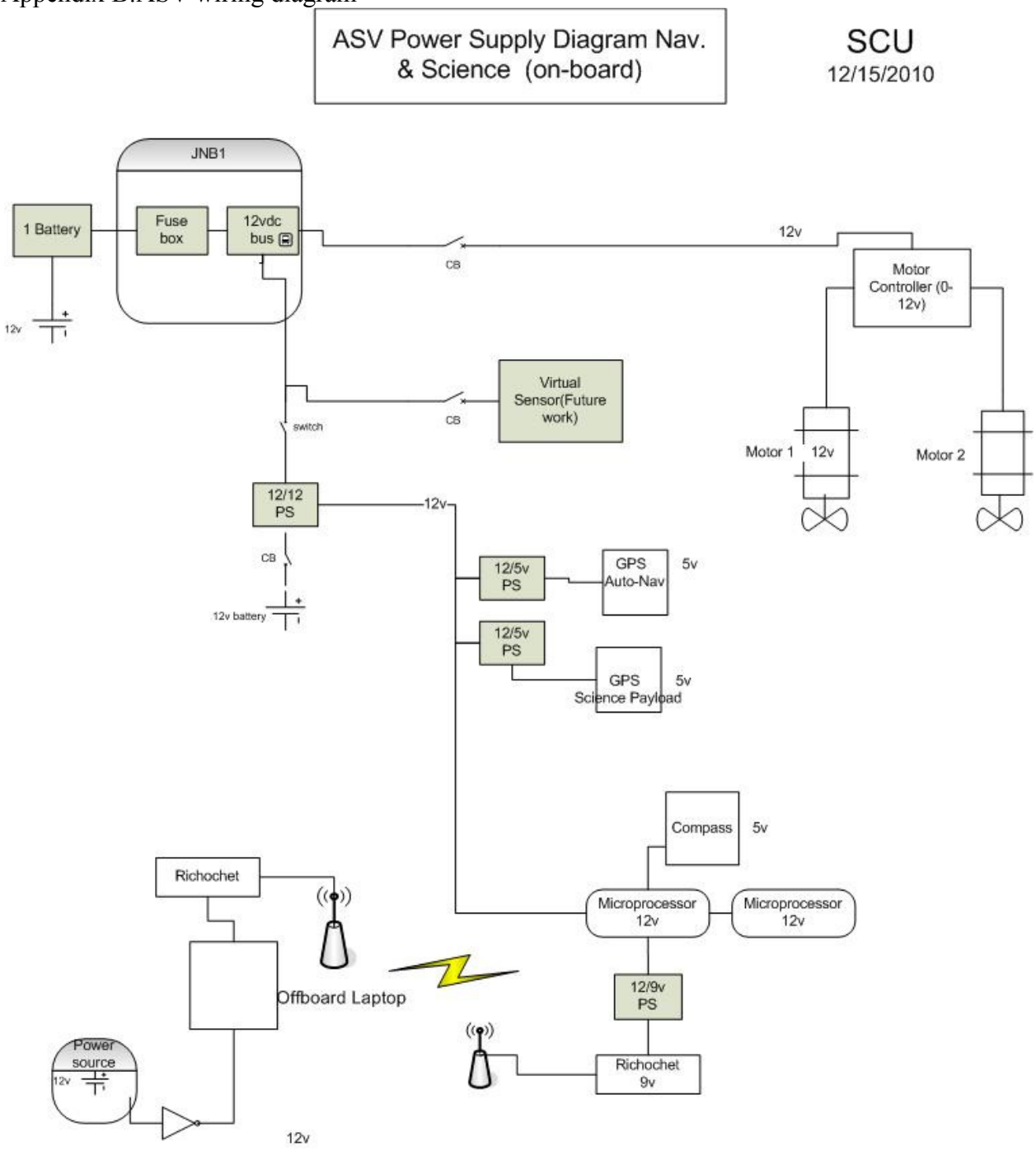
[43] K. Rasel, A study of an ASV Robot and the impact of environmental variables using a robust dynamic control system C. Kitts, Adv., M.S. Thesis, Dept. Mech Eng, Santa Clara University, Santa Clara, CA, Dec 2010. In Draft

[44] C. Kitts, Thomas Adamek, and Vincent Howard, Parameter Field Gradient and Contour Bearing Estimation, Robotic Systems Laboratory Technical Document, Santa Clara University December 22, 2010.

Appendix A: Wiring diagram for common systems



Appendix B:ASV wiring diagram

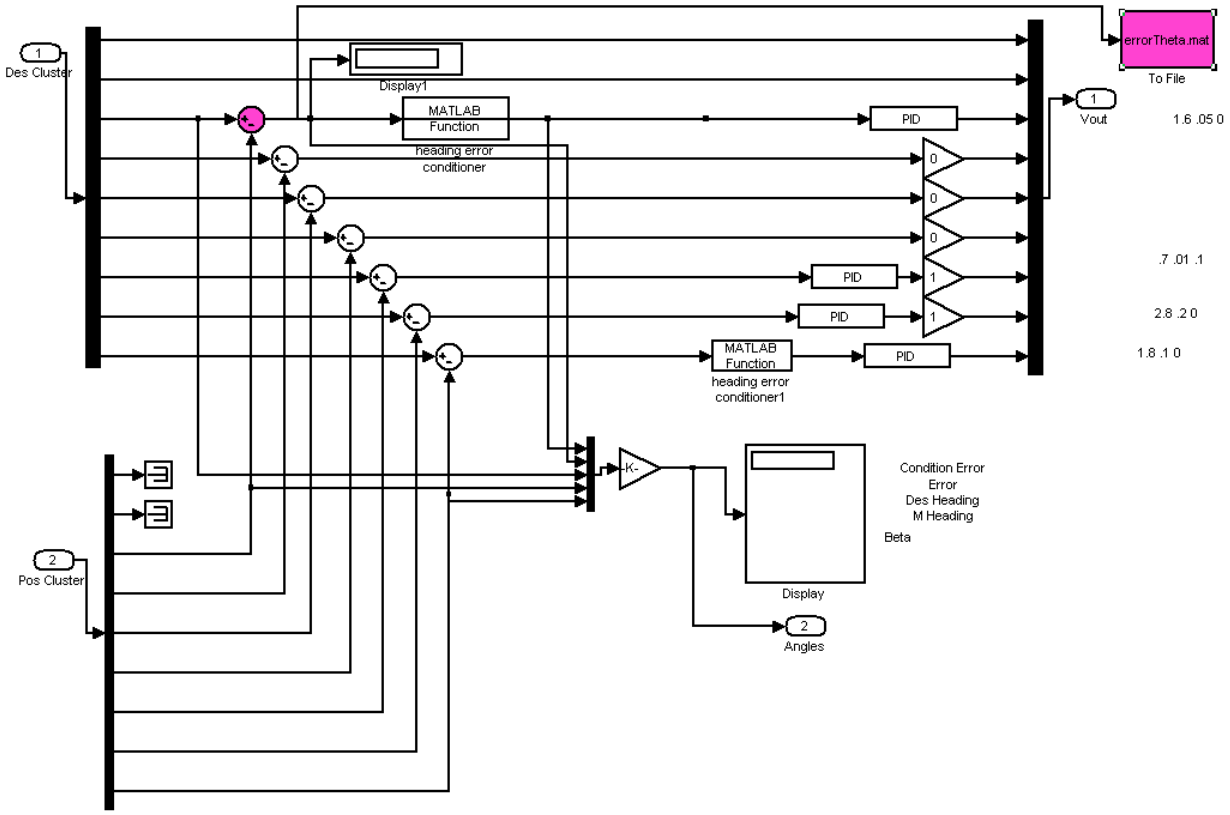


Appendix C:ASV bill of materials

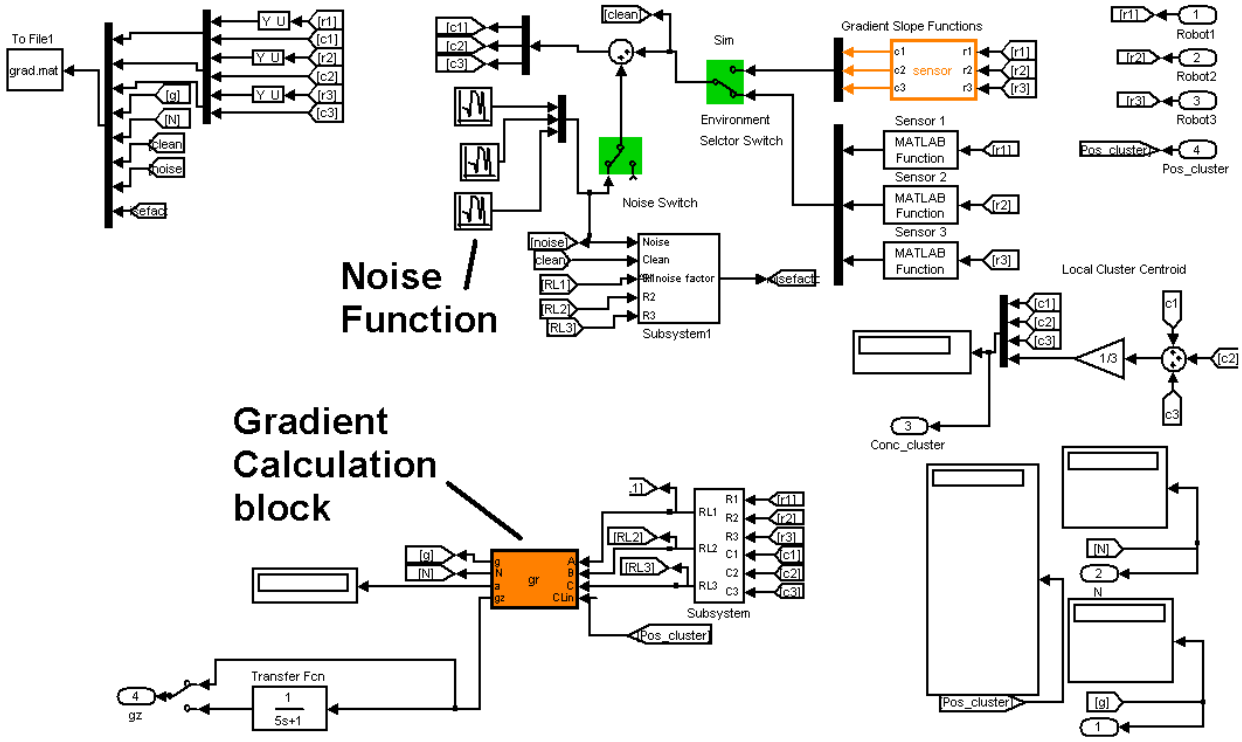
BOM 04 02 2009

Item	Model	Part#	Voltage	Current (spec)	Current (Actual)	Spare	Quantity	Vendor	Quick Ref.Specs	PDF File		
GPS	18 PC	010-00321-34	5v	50m@13.8v		☑	0	Garmin	http://www8.garmin.com	425_TechnicalSpecification.pdf		
12/5V reg.	12 volt	101072306	12/5v			☑	2	Garmin	N/A	N/A		
Processor-(Com)	BasicX-24p	BasicX-24p	13.4v			☑	1	Basic X	http://www.basicx.com	bx-24toPcConnections.GIF		
Processor-(Nav)	BasicX-24p	BasicX-24p	13.4v	20 mA		☑	1	Basic X	http://www.basicx.com	bx-24toPcConnections.GIF		
Stack	bx24devboard2	bx24devboard2	13.4v			☑	2	Basic X	http://www.basicx.com	Boat Navigation Board.JPG		
Richochet- Onboard	MetroCom	Curly	9v			☑	1	SCU				
Richochet- Shore	MetroCom	Snow	9v			☐	1	SCU				
Motor Controller	AX3500	AX3500	12v	60A		☑	1	roboteq	http://www.roboteq.com	ax3500datasheet.pdf		
Fuse	Glass 1/4"	GGC5	12v			☐	4	Digi	http://www.fusecoinc.c	GSA-GSA-V&GDL-GDL-V.pdf		
Battery	Sealed	23289B	12v	5A/h		☑	1	Radio Shack				
Battery	Sealed	PVX-1040T	12v	104 A/h		☑	4		http://store.solar-electri	1		
Regulator	12 volt reg	7805	12/5v			☑	1		http://www.radioshack.	7805.pdf		
Regulator 12/9v		PT78ST109H-ND	12/9v			☑		Digi	http://parts.digikey.com	slts059a_9volt.pdf		
breaker	UL1077	281-2528-ND		5A		☑	2	Digi	http://parts.digikey.de/	1		
breaker	UL1077	281-2531-ND		10A		☑	2	Digi	http://parts.digikey.de/	1		
Devantech Compass	CMPS03	CMPS03	5	20mA		☑	1		http://www.acroname.c	1		
Battery	interstate	SRM-24	12	85A/H		☑	1					
Kayak	Beachcraft	NA	-	-		☑	1		Dimensions- 33"x93"x1	1		
Motor	Endura	1352304				☑	2		http://www.minnkotamo	1		

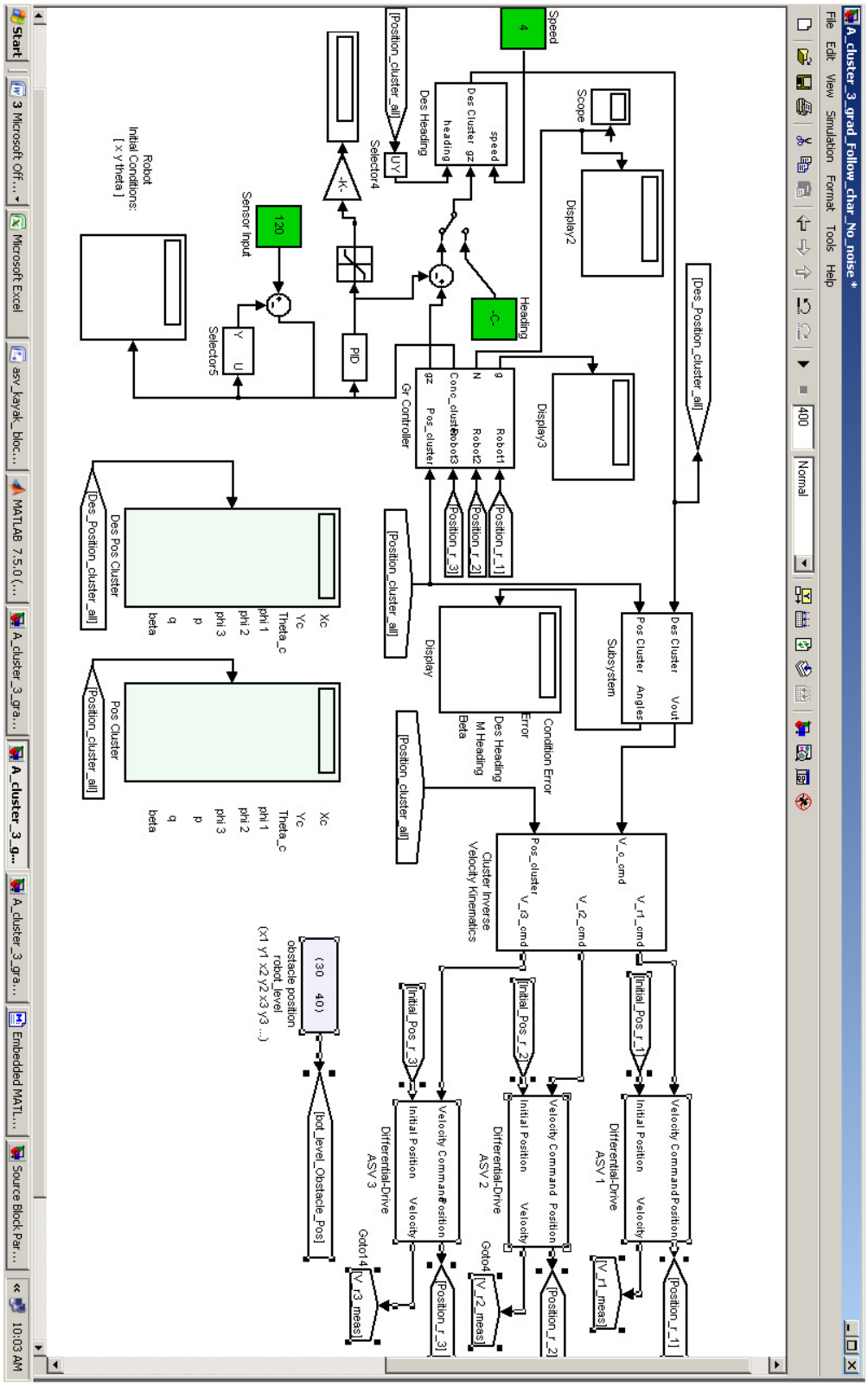
Appendix D: Simulink model - desired cluster heading & actual cluster heading error block



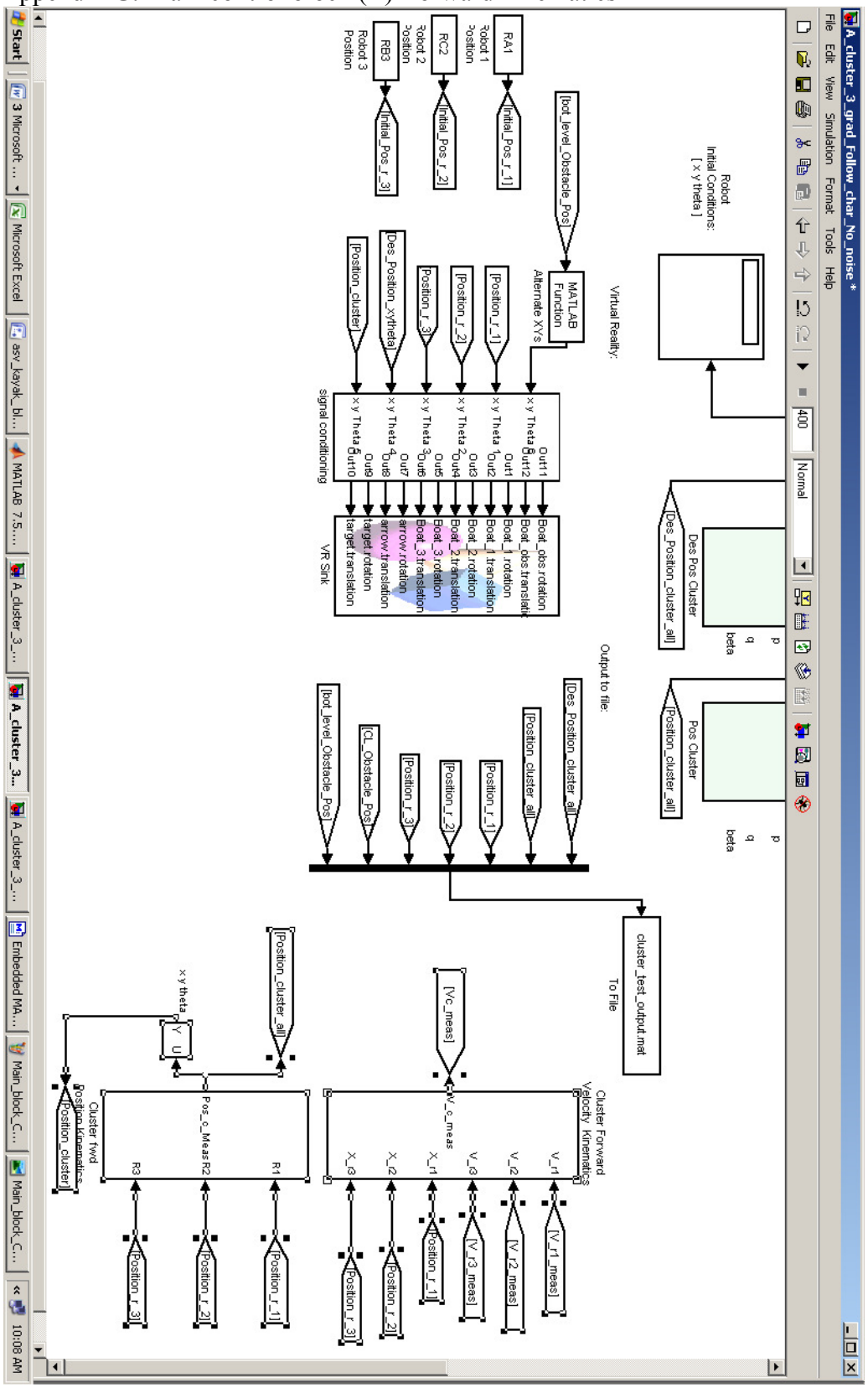
Appendix E: Simulink model- function and gradient calculation blocks



Appendix F: Main Simulink block (A)- Inverse kinematics



Appendix G: Main control block (B)- forward kinematics



Appendix H: Matlab m files

Gradient calculation block

```
function [g,N,a,gz] = gr(A,B,C,CLin)
% Grad
N=[0;0;0];
Cn=(A+B+C)/3;
xy=[1;1;0]/sqrt(2);

N=cross(B,C)*sign(CLin(9)); %Cross product of Robot vectors
a=0;
speed=1;
apart=1;
N=N/sqrt(N(1)^2+N(2)^2+N(3)^2);  %normalization of Normal
gz=atan2(N(2),N(1));
gxy=atan2(N(3),cos(gz)*N(2)+sin(gz)*N(1))-pi/2;
a=gxy*180/pi;
 %N=-N;
gy=atan2(N(3),N(1));
gx=atan2(N(3),N(2));
gz=atan2(N(2),N(1));  %Projection of Normal onto the xy Plane
%gz=pi/2;
g=[gx;gy;gz]*180/pi;  %Converting into Degrees
end
```

Environmental generator - gaussian

```
%GEN_environment.m
% points=[1 3 100 .001; -22 40 100 .0001; 120 30 100 .0001;-120 35 100
.0003;-120 50 30 .0001]
 points=[0 0 100 .00001;0 0 100 .00001;] %center
% <<<<<<< GEN_environment.m
% points=[1 3 1000;-2 -4 100]
% environment_generator(points,[6;.05])
%points=[1 3 100 .0001; -2 4 100 .0001; -20 30 100 .0001]
environment_generator(points,[500;1])
load env_map.mat
global env_h
%environment_evaluator([x y],50,1,env_h)
```

Centroid & start location

```
% Centroid_Start.m
% finding the cluster space robots using
 %define triangle
beta=90*pi/180
pq=15;
r=pq/2*sec(beta/2);
%Rc=[150 100]% mult circle % Speed 12 %Sensor 35-45  P&Q32 (P&Q 8
Sensor 66)
%thCs=-10*pi/180; %mult circle
 Rc=[400 400]% mult circle % Speed 12 %Sensor 35-45  P&Q32 (P&Q 8
Sensor 99 &110

thCs=85*pi/180
%thCs=0*pi/180; %mult circle
%thCs=15*pi/180
```

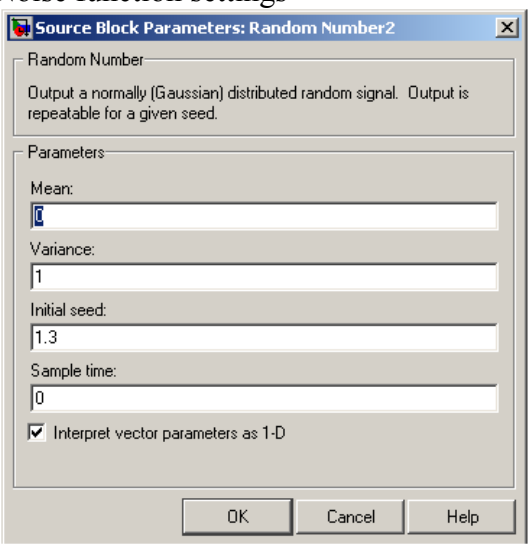
```
%thCs=45*pi/180
%thCs=60*pi/180
%thCs=90*pi/180
%thCs=120*pi/180
%thCs=150*pi/180
%thCs=180*pi/180

%Rc=[150 100] %single circle %speed 12 %sensor input 142-148
%thCs=-60*pi/180 %single circle

R=[cos(thCs) -sin(thCs);sin(thCs) cos(thCs)];
display('Beta')
RC2=[-sin(beta)*r -cos(beta)*r]
RB3=[+sin(beta)*r -cos(beta)*r]
RA1=[0 r]
 display('Thetc')
RC2=RC2*R;
RB3=RB3*R;
RA1=RA1*R;
 display('Global')
RC2=Rc+RC2;
RB3=Rc+RB3;
RA1=Rc+RA1;
 RC2=[RC2 thCs];
RB3=[RB3 thCs];
RA1=[RA1 thCs];
 % Rinv=[cos(thCs) sin(thCs);-sin(thCs) cos(thCs)]
% Rl1=(RA1(1:2)-Rc)*Rinv
% Rl2=(RB3(1:2)-Rc)*Rinv
% Rl3=(RC2(1:2)-Rc)*Rinv
```

Environmental evaluator

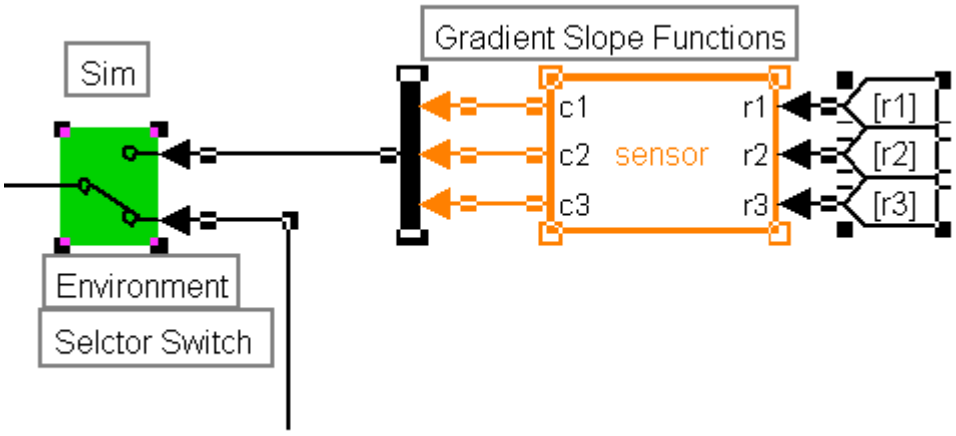
```
function val = environment_evaluator( xy_pt, env_max, step_size, env_h
)
% xy_pt: [x y]
% env_bound: env_max
% step_size (scalar)
% env_h: height field n x 2 matrix
% load env_map.mat
 env_bound(1) = -env_max;
env_bound(2) = env_max;
 %check if [x_pt y_pt] outside of env_map
    if ((xy_pt(1) > env_bound(2)) || (xy_pt(1) < env_bound(1))) ||
((xy_pt(2) > env_bound(2)) || (xy_pt(2) < env_bound(1)))
        val = inf;
    else %if inside, then get row index from env_y; col index from
env_x...
        ind_x = round(1+(xy_pt(2)-env_bound(1))/step_size);
        ind_y = round(1+(xy_pt(1)-env_bound(1))/step_size);
        val = env_h(ind_x,ind_y);
%         [env_x(ind_x,ind_x) ind_x env_y(ind_y,ind_y) ind_y
env_h(ind_x, ind_y)]

    end
```

```
%      env_x(1,ind_x);
%      env_y(ind_y,1);
end
```

Noise function settings



Gradient Slope Function- Simulink and Matlab code (embedded)




```
function [c1,c2,c3] = sensor(r1,r2,r3)
% Grad
x=[r1(1);r2(1);r3(1)];
y=[r1(2);r2(2);r3(2)];
%c=y;
%c=100*(sin((x+y)/1000-3)+cos(y/100-3)); %slope, non linear
%c=100*(sin((x+y)/1000-3)+cos(y/100-3));
%c=-(x.^2+y.^2)+200;
c=tan(45*pi/180)*y;                 % constant slope 45deg, linear
%c=(x-y)
c1=c(1);
c2=c(2);
c3=c(3);
```

Appendix I: Inverse Jacobian m files

```
function Output = three_bots_centroid_inv_jacobian_matrix_exact(u)
%This function computes the robot velocities based on the cluster
%velocities.
%arguments:     u = [theta_c p q beta]
%output:     output = [J_inv]

%Initialize variables
theta=u(1);
p=u(2);
q=u(3);
beta=u(4);


J_inv = [[  1,  0,    1/3*(p^2+q^2+2*p*q*cos(beta))^(1/2)*cos(theta),
0,   0,   0,
1/3*sin(theta)*(q*cos(beta)+p)/(p^2+q^2+2*p*q*cos(beta))^(1/2),
1/3*sin(theta)*(q+p*cos(beta))/(p^2+q^2+2*p*q*cos(beta))^(1/2),     -
1/3/(p^2+q^2+2*p*q*cos(beta))^(1/2)*sin(theta)*p*q*sin(beta)]
        [   0,  1,  -1/3*(p^2+q^2+2*p*q*cos(beta))^(1/2)*sin(theta),
0,   0,   0,
1/3*cos(theta)*(q*cos(beta)+p)/(p^2+q^2+2*p*q*cos(beta))^(1/2),
1/3*cos(theta)*(q+p*cos(beta))/(p^2+q^2+2*p*q*cos(beta))^(1/2),     -
1/3/(p^2+q^2+2*p*q*cos(beta))^(1/2)*cos(theta)*p*q*sin(beta)]
        [   0,  0,  -1, 1,   0,   0,    0,  0,  0]
        [   1,  0,  1/3*(p^2+q^2+2*p*q*cos(beta))^(1/2)*cos(theta)-
p*sin(atan2(q*sin(beta),q*cos(beta)+p)-atan2(-
1/3*(p^2+q^2+2*p*q*cos(beta))^(1/2)*cos(theta),-
1/3*(p^2+q^2+2*p*q*cos(beta))^(1/2)*sin(theta))), 0,   0,   0,
1/3*(sin(theta)*p^3+3*sin(theta)*p^2*q*cos(beta)+2*sin(theta)*p*q^2*cos
(beta)^2+sin(theta)*p*q^2+sin(theta)*q^3*cos(beta)+3*cos(atan2(q*sin(be
ta),q*cos(beta)+p)-atan2(-
1/3*(p^2+q^2+2*p*q*cos(beta))^(1/2)*cos(theta),-
1/3*(p^2+q^2+2*p*q*cos(beta))^(1/2)*sin(theta)))*(p^2+q^2+2*p*q*cos(bet
a))^(1/2)*p^2+6*cos(atan2(q*sin(beta),q*cos(beta)+p)-atan2(-
1/3*(p^2+q^2+2*p*q*cos(beta))^(1/2)*cos(theta),-
1/3*(p^2+q^2+2*p*q*cos(beta))^(1/2)*sin(theta)))*(p^2+q^2+2*p*q*cos(bet
a))^(1/2)*p*q*cos(beta)+3*cos(atan2(q*sin(beta),q*cos(beta)+p)-atan2(-
1/3*(p^2+q^2+2*p*q*cos(beta))^(1/2)*cos(theta),-
1/3*(p^2+q^2+2*p*q*cos(beta))^(1/2)*sin(theta)))*(p^2+q^2+2*p*q*cos(bet
a))^(1/2)*q^2+3*p*sin(atan2(q*sin(beta),q*cos(beta)+p)-atan2(-
1/3*(p^2+q^2+2*p*q*cos(beta))^(1/2)*cos(theta),-
1/3*(p^2+q^2+2*p*q*cos(beta))^(1/2)*sin(theta)))*q*sin(beta)*(p^2+q^2+2
*p*q*cos(beta))^(1/2))/(p^2+q^2+2*p*q*cos(beta))^(3/2),
1/3*(3*sin(theta)*p*q^2*cos(beta)+sin(theta)*q*p^2+sin(theta)*q^3+2*sin
(theta)*p^2*cos(beta)^2*q+sin(theta)*p^3*cos(beta)-
3*p^2*sin(atan2(q*sin(beta),q*cos(beta)+p)-atan2(-
1/3*(p^2+q^2+2*p*q*cos(beta))^(1/2)*cos(theta),-
1/3*(p^2+q^2+2*p*q*cos(beta))^(1/2)*sin(theta)))*sin(beta)*(p^2+q^2+2*p
*q*cos(beta))^(1/2))/(p^2+q^2+2*p*q*cos(beta))^(3/2),    -
1/3*p*q*(2*sin(theta)*sin(beta)*p*q*cos(beta)+sin(theta)*sin(beta)*p^2+
sin(theta)*sin(beta)*q^2+3*sin(atan2(q*sin(beta),q*cos(beta)+p)-atan2(-
1/3*(p^2+q^2+2*p*q*cos(beta))^(1/2)*cos(theta),-
1/3*(p^2+q^2+2*p*q*cos(beta))^(1/2)*sin(theta)))*(p^2+q^2+2*p*q*cos(bet
```

```
a))^(1/2)*p*cos(beta)+3*sin(atan2(q*sin(beta),q*cos(beta)+p)-atan2(-
1/3*(p^2+q^2+2*p*q*cos(beta))^(1/2)*cos(theta),-
1/3*(p^2+q^2+2*p*q*cos(beta))^(1/2)*sin(theta)))*(p^2+q^2+2*p*q*cos(bet
a))^(1/2)*q)/(p^2+q^2+2*p*q*cos(beta))^(3/2)]
        [   0,  1,  -1/3*(p^2+q^2+2*p*q*cos(beta))^(1/2)*sin(theta)-
p*cos(atan2(q*sin(beta),q*cos(beta)+p)-atan2(-
1/3*(p^2+q^2+2*p*q*cos(beta))^(1/2)*cos(theta),-
1/3*(p^2+q^2+2*p*q*cos(beta))^(1/2)*sin(theta))),  0,   0,   0,
1/3*(cos(theta)*p^3+3*cos(theta)*p^2*q*cos(beta)+2*cos(theta)*p*q^2*cos
(beta)^2+cos(theta)*p*q^2+cos(theta)*q^3*cos(beta)-
3*sin(atan2(q*sin(beta),q*cos(beta)+p)-atan2(-
1/3*(p^2+q^2+2*p*q*cos(beta))^(1/2)*cos(theta),-
1/3*(p^2+q^2+2*p*q*cos(beta))^(1/2)*sin(theta)))*(p^2+q^2+2*p*q*cos(bet
a))^(1/2)*p^2-6*sin(atan2(q*sin(beta),q*cos(beta)+p)-atan2(-
1/3*(p^2+q^2+2*p*q*cos(beta))^(1/2)*cos(theta),-
1/3*(p^2+q^2+2*p*q*cos(beta))^(1/2)*sin(theta)))*(p^2+q^2+2*p*q*cos(bet
a))^(1/2)*p*q*cos(beta)-3*sin(atan2(q*sin(beta),q*cos(beta)+p)-atan2(-
1/3*(p^2+q^2+2*p*q*cos(beta))^(1/2)*cos(theta),-
1/3*(p^2+q^2+2*p*q*cos(beta))^(1/2)*sin(theta)))*(p^2+q^2+2*p*q*cos(bet
a))^(1/2)*q^2+3*p*cos(atan2(q*sin(beta),q*cos(beta)+p)-atan2(-
1/3*(p^2+q^2+2*p*q*cos(beta))^(1/2)*cos(theta),-
1/3*(p^2+q^2+2*p*q*cos(beta))^(1/2)*sin(theta)))*q*sin(beta)*(p^2+q^2+2
*p*q*cos(beta))^(1/2))/(p^2+q^2+2*p*q*cos(beta))^(3/2),
1/3*(3*cos(theta)*p*q^2*cos(beta)+cos(theta)*q*p^2+cos(theta)*q^3+2*cos
(theta)*p^2*cos(beta)^2*q+cos(theta)*p^3*cos(beta)-
3*p^2*cos(atan2(q*sin(beta),q*cos(beta)+p)-atan2(-
1/3*(p^2+q^2+2*p*q*cos(beta))^(1/2)*cos(theta),-
1/3*(p^2+q^2+2*p*q*cos(beta))^(1/2)*sin(theta)))*sin(beta)*(p^2+q^2+2*p
*q*cos(beta))^(1/2))/(p^2+q^2+2*p*q*cos(beta))^(3/2),    -
1/3*p*q*(2*cos(theta)*sin(beta)*p*q*cos(beta)+cos(theta)*sin(beta)*p^2+
cos(theta)*sin(beta)*q^2+3*cos(atan2(q*sin(beta),q*cos(beta)+p)-atan2(-
1/3*(p^2+q^2+2*p*q*cos(beta))^(1/2)*cos(theta),-
1/3*(p^2+q^2+2*p*q*cos(beta))^(1/2)*sin(theta)))*(p^2+q^2+2*p*q*cos(bet
a))^(1/2)*p*cos(beta)+3*cos(atan2(q*sin(beta),q*cos(beta)+p)-atan2(-
1/3*(p^2+q^2+2*p*q*cos(beta))^(1/2)*cos(theta),-
1/3*(p^2+q^2+2*p*q*cos(beta))^(1/2)*sin(theta)))*(p^2+q^2+2*p*q*cos(bet
a))^(1/2)*q)/(p^2+q^2+2*p*q*cos(beta))^(3/2)]
        [   0,  0,  -1,  0,   1,   0,  0,   0,  0]
        [   1,  0,  1/3*(p^2+q^2+2*p*q*cos(beta))^(1/2)*cos(theta)-
q*sin(-beta+atan2(q*sin(beta),q*cos(beta)+p)-atan2(-
1/3*(p^2+q^2+2*p*q*cos(beta))^(1/2)*cos(theta),-
1/3*(p^2+q^2+2*p*q*cos(beta))^(1/2)*sin(theta))),  0,   0,   0,
1/3*(sin(theta)*p^3+3*sin(theta)*p^2*q*cos(beta)+2*sin(theta)*p*q^2*cos
(beta)^2+sin(theta)*p*q^2+sin(theta)*q^3*cos(beta)+3*q^2*sin(-
beta+atan2(q*sin(beta),q*cos(beta)+p)-atan2(-
1/3*(p^2+q^2+2*p*q*cos(beta))^(1/2)*cos(theta),-
1/3*(p^2+q^2+2*p*q*cos(beta))^(1/2)*sin(theta)))*sin(beta)*(p^2+q^2+2*p
*q*cos(beta))^(1/2))/(p^2+q^2+2*p*q*cos(beta))^(3/2),
1/3*(3*sin(theta)*p*q^2*cos(beta)+sin(theta)*q*p^2+sin(theta)*q^3+2*sin
(theta)*p^2*cos(beta)^2*q+sin(theta)*p^3*cos(beta)+6*cos(-
beta+atan2(q*sin(beta),q*cos(beta)+p)-atan2(-
1/3*(p^2+q^2+2*p*q*cos(beta))^(1/2)*cos(theta),-
1/3*(p^2+q^2+2*p*q*cos(beta))^(1/2)*sin(theta)))*(p^2+q^2+2*p*q*cos(bet
a))^(1/2)*p*q*cos(beta)+3*cos(-beta+atan2(q*sin(beta),q*cos(beta)+p)-
atan2(-1/3*(p^2+q^2+2*p*q*cos(beta))^(1/2)*cos(theta),-
1/3*(p^2+q^2+2*p*q*cos(beta))^(1/2)*sin(theta)))*(p^2+q^2+2*p*q*cos(bet
a))^(1/2)*p^2+3*cos(-beta+atan2(q*sin(beta),q*cos(beta)+p)-atan2(-
```

```
1/3*(p^2+q^2+2*p*q*cos(beta))^(1/2)*cos(theta),-
1/3*(p^2+q^2+2*p*q*cos(beta))^(1/2)*sin(theta)))*(p^2+q^2+2*p*q*cos(bet
a))^(1/2)*q^2-3*q*sin(-beta+atan2(q*sin(beta),q*cos(beta)+p)-atan2(-
1/3*(p^2+q^2+2*p*q*cos(beta))^(1/2)*cos(theta),-
1/3*(p^2+q^2+2*p*q*cos(beta))^(1/2)*sin(theta)))*sin(beta)*p*(p^2+q^2+2
*p*q*cos(beta))^(1/2))/(p^2+q^2+2*p*q*cos(beta))^(3/2),    -
1/3*p*q*(2*sin(theta)*sin(beta)*p*q*cos(beta)+sin(theta)*sin(beta)*p^2+
sin(theta)*sin(beta)*q^2-3*sin(-beta+atan2(q*sin(beta),q*cos(beta)+p)-
atan2(-1/3*(p^2+q^2+2*p*q*cos(beta))^(1/2)*cos(theta),-
1/3*(p^2+q^2+2*p*q*cos(beta))^(1/2)*sin(theta)))*(p^2+q^2+2*p*q*cos(bet
a))^(1/2)*q*cos(beta)-3*sin(-beta+atan2(q*sin(beta),q*cos(beta)+p)-
atan2(-1/3*(p^2+q^2+2*p*q*cos(beta))^(1/2)*cos(theta),-
1/3*(p^2+q^2+2*p*q*cos(beta))^(1/2)*sin(theta)))*(p^2+q^2+2*p*q*cos(bet
a))^(1/2)*p)/(p^2+q^2+2*p*q*cos(beta))^(3/2)]
        [   0,  1,  -1/3*(p^2+q^2+2*p*q*cos(beta))^(1/2)*sin(theta)-
q*cos(-beta+atan2(q*sin(beta),q*cos(beta)+p)-atan2(-
1/3*(p^2+q^2+2*p*q*cos(beta))^(1/2)*cos(theta),-
1/3*(p^2+q^2+2*p*q*cos(beta))^(1/2)*sin(theta))),  0,   0,   0,
1/3*(cos(theta)*p^3+3*cos(theta)*p^2*q*cos(beta)+2*cos(theta)*p*q^2*cos
(beta)^2+cos(theta)*p*q^2+cos(theta)*q^3*cos(beta)+3*q^2*cos(-
beta+atan2(q*sin(beta),q*cos(beta)+p)-atan2(-
1/3*(p^2+q^2+2*p*q*cos(beta))^(1/2)*cos(theta),-
1/3*(p^2+q^2+2*p*q*cos(beta))^(1/2)*sin(theta)))*sin(beta)*(p^2+q^2+2*p
*q*cos(beta))^(1/2))/(p^2+q^2+2*p*q*cos(beta))^(3/2), -1/3*(-
3*cos(theta)*p*q^2*cos(beta)-cos(theta)*q*p^2-cos(theta)*q^3-
2*cos(theta)*p^2*cos(beta)^2*q-cos(theta)*p^3*cos(beta)+6*sin(-
beta+atan2(q*sin(beta),q*cos(beta)+p)-atan2(-
1/3*(p^2+q^2+2*p*q*cos(beta))^(1/2)*cos(theta),-
1/3*(p^2+q^2+2*p*q*cos(beta))^(1/2)*sin(theta)))*(p^2+q^2+2*p*q*cos(bet
a))^(1/2)*p*q*cos(beta)+3*sin(-beta+atan2(q*sin(beta),q*cos(beta)+p)-
atan2(-1/3*(p^2+q^2+2*p*q*cos(beta))^(1/2)*cos(theta),-
1/3*(p^2+q^2+2*p*q*cos(beta))^(1/2)*sin(theta)))*(p^2+q^2+2*p*q*cos(bet
a))^(1/2)*p^2+3*sin(-beta+atan2(q*sin(beta),q*cos(beta)+p)-atan2(-
1/3*(p^2+q^2+2*p*q*cos(beta))^(1/2)*cos(theta),-
1/3*(p^2+q^2+2*p*q*cos(beta))^(1/2)*sin(theta)))*(p^2+q^2+2*p*q*cos(bet
a))^(1/2)*q^2+3*q*cos(-beta+atan2(q*sin(beta),q*cos(beta)+p)-atan2(-
1/3*(p^2+q^2+2*p*q*cos(beta))^(1/2)*cos(theta),-
1/3*(p^2+q^2+2*p*q*cos(beta))^(1/2)*sin(theta)))*sin(beta)*p*(p^2+q^2+2
*p*q*cos(beta))^(1/2))/(p^2+q^2+2*p*q*cos(beta))^(3/2),    1/3*p*q*(-
2*cos(theta)*sin(beta)*p*q*cos(beta)-cos(theta)*sin(beta)*p^2-
cos(theta)*sin(beta)*q^2+3*cos(-beta+atan2(q*sin(beta),q*cos(beta)+p)-
atan2(-1/3*(p^2+q^2+2*p*q*cos(beta))^(1/2)*cos(theta),-
1/3*(p^2+q^2+2*p*q*cos(beta))^(1/2)*sin(theta)))*(p^2+q^2+2*p*q*cos(bet
a))^(1/2)*q*cos(beta)+3*cos(-beta+atan2(q*sin(beta),q*cos(beta)+p)-
atan2(-1/3*(p^2+q^2+2*p*q*cos(beta))^(1/2)*cos(theta),-
1/3*(p^2+q^2+2*p*q*cos(beta))^(1/2)*sin(theta)))*(p^2+q^2+2*p*q*cos(bet
a))^(1/2)*p)/(p^2+q^2+2*p*q*cos(beta))^(3/2)]
        [   0,  0,  -1, 0,   0,   1, 0, 0,  0]];

Output = J_inv;
```

Appendix J: Jacobian m files

```
function Output = three_bots_centroid_jacobian_matrix_beta_atan(u)
%This function computes the cluster velocities based on robots
%velocities.
%arguments:     u = [x1 y1 x2 y2 x3 y3 ]
%output:     output = [J]

%Initialize variables
x1  = u(1);
y1  = u(2);
x2  = u(3);
y2  = u(4);
x3  = u(5);
y3 = u(6);

J = (2/3)*x1-(x2+x3)/3;
K = (2/3)*y1-(y2+y3)/3;
M = J^2+K^2;

R = (x1-x2)^2+(y1-y2)^2;
S = (x3-x1)^2+(y1-y3)^2;



J    =[[1/3                  0                   0           1/3
0                   0           1/3                 0
0];
       [0                   1/3                 0           0
1/3                 0           0                   1/3
0];
       [(2/3)*K/M           -(2/3)*J/M          0           -(1/3)*K/M
(1/3)*J/M           0           -(1/3)*K/M          (1/3)*J/M
0];
       [(2/3)*K/M           -(2/3)*J/M          1           -(1/3)*K/M
(1/3)*J/M           0           -(1/3)*K/M          (1/3)*J/M
0];
       [(2/3)*K/M           -(2/3)*J/M          0           -(1/3)*K/M
(1/3)*J/M           1           -(1/3)*K/M          (1/3)*J/M
0];
       [(2/3)*K/M           -(2/3)*J/M          0           -(1/3)*K/M
(1/3)*J/M           0           -(1/3)*K/M          (1/3)*J/M
1];
       [(x1-x2)/sqrt(R)     (y1-y2)/sqrt(R)     0           -(x1-
x2)/sqrt(R)         -(y1-y2)/sqrt(R)    0           0
0                   0];
       [-(x3-x1)/sqrt(S)    (y1-y3)/sqrt(S)     0           0
0                   0           (x3-x1)/sqrt(S)     -(y1-y3)/sqrt(S)
0];
       [ -(-2*x3*y2*x1+x1^2*y2+y2^2*y1-y2*y1^2-y2^2*y3-
y3*x1^2+y3*y1^2+y3^2*y2-y3^2*y1+x3^2*y2-x3^2*y1+y1*x2^2-
y3*x2^2+2*y3*x2*x1+2*x3*y1*x1-2*x1*y1*x2)/(y2^2-2*y2*y1+y1^2+x2^2-
2*x2*x1+x1^2)/(x1^2+y1^2-2*x3*x1+x3^2-2*y3*y1+y3^2)         (-y2^2*x3-
x3*x2^2-x2*x1^2+x1*x2^2-x3*y1^2-x3^2*x1+x3^2*x2+x3*x1^2-
2*y2*y1*x1+x1*y2^2+y3^2*x2+2*y3*y1*x1-y3^2*x1+y1^2*x2-
```

```
2*y3*y1*x2+2*x3*y2*y1)/(y2^2-2*y2*y1+y1^2+x2^2-
2*x2*x1+x1^2)/(x1^2+y1^2-2*x3*x1+x3^2-2*y3*y1+y3^2)      0       -(-
y2+y1)/(y2^2-2*y2*y1+y1^2+x2^2-2*x2*x1+x1^2)     (-x2+x1)/(y2^2-
2*y2*y1+y1^2+x2^2-2*x2*x1+x1^2)      0       (y1-y3)/(x1^2+y1^2-
2*x3*x1+x3^2-2*y3*y1+y3^2)       -(x1-x3)/(x1^2+y1^2-2*x3*x1+x3^2-
2*y3*y1+y3^2)      0]
       ];


Output = J;
```