3-2-2011

A Study of Gradient Climbing Technique Using Cluster Space Control of Multi-Robot Systems

Vincent Michael Howard
Santa Clara University

SANTA CLARA UNIVERSITY

Department of Mechanical Engineering

Date: March 2, 2011

I HEREBY RECOMMEND THAT THE THESIS PREPARED UNDER MY SUPERVISION

BY

Vincent Michael Howard

A Study of Gradient Climbing Technique Using Cluster Space Control of 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

A Study of Gradient Climbing Technique Using Cluster Space Control of Multi-robot Systems

By

Vincent Michael Howard

GRADUATE MASTERS THESIS

Submitted in Partial Fulfillment of the Requirements

for the Degree of Master of Science

in Mechanical Engineering

Santa Clara, California

A Study of Gradient Climbing Technique Using Cluster Space Control of Multi-robot Systems

Vincent Michael Howard

Department of Mechanical Engineering

Santa Clara University

2011

ABSTRACT

The design of the multi-robot system for distributed sensing and gradient climbing focuses on the capability to optimize the performance of tasks simultaneously. The strategy is to utilize the cluster's redundancy and flexibility to gain and maximize the overall coverage of surveying parameters so as to surpass the performance of any single robot. The collaborative nature of the cluster provides a more efficient and effective platform for collecting data and conducting fieldwork. The purpose of this study is to explore the existing cluster space control technique to show effective gradient-based navigation, particularly that of climbing a gradient in a sensed parameter field to the local maximum. In order to achieve positive results, we need to estimate the gradient direction based on real-time measurements captured by sensors on the distributed robotic network, and then maneuver the cluster to travel in the estimated direction. Verification and characterization of this technique has been performed through both simulation and hardware-in-the-loop experimentation. In these tests, the gradient controller enabled the cluster to sense and climb the gradient in a parameterized field using kayaks in a marine environment and utilizing wheeled robots in a land based system. The successful outcome of these demonstrations proves the value of the cluster space control technique and showcases how it can be used for efficiently locating minimum and maximum features in a parameter field.

Keywords: multi-robot, cluster control, gradient climbing, ASV

Acknowledgements

I would like to thank my advisor, Professor Christopher Kitts, for his guidance and encouragement throughout the process of this research. His dedication and insights have helped me overcome the many challenges encountered in this thesis.

The contributions by many people in the SCU Robotic System Lab both past and present have been invaluable. Much of the foundation of this thesis has been built directly on their dedicated work. They had spent many countless nights trouble-shooting, coding and researching. I am indebted to their devotion.

A special acknowledgement and thanks go to my friend and colleague, Thomas Adamek. He has made many instrumental contributions to this project. This would not have been achievable without his collaboration and commitment. Also, many thanks go to those who worked alongside on the test beds: Paul Mahacek and Steve Li.

I thank my parents for their support and reassurance throughout my academic career. None of it is realizable without them. Lastly, a special bone goes to Mr. Charly Howard with his gradient climbing skills to find the nearest steak.

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

Chapter 1 Introduction

1.1 Multiple Robot Systems

In the past decade, the advancement of design tools and communication networks have greatly improved the capabilities of autonomous mobile robots. A multi-robot platform can perform a wide range of applications, from surveillance to exploration in extreme environments over land, sea, air and space. The distributive properties of multi-robot systems allow them to complete tasks faster and more efficiently than single robots. Examples of tasks which could be performed with greater efficiency include mapping, collaborative manipulation and assembly of objects, and object tracking.

The key to designing a successful multi-robot system is to synchronize the co-ordination of individual robots and to work in collaboration to accomplish the mission. The strategy is to aim at a high level of co-operation among robots by dividing the workload which helps to increase productivity, speed, coverage, robustness, reliability, fault-tolerance, and in some cases cost efficiency.

In terms of increasing productivity and expanding the coverage of a parameter field in the environment, as in the case of a multi-robot mapping system, it can be more effective in mapping a vast area than a single robot [1]. Each robot in the system individually collects data in a small area and then collectively generates a global map. The increase in efficiency and speed are due to its distributive ability to perform tasks concurrently in various locations in real time. Also, a multi-robot system is more robust and fault tolerant compared to a single robot because the added redundancy eliminates a single point of failure.

The cost advantage of a multi-robot system maybe achieved by having simpler robots accomplish a designated task instead of using a single complex robot. Each robot in a multi-robot system can be designed to perform a very specific task, making them simpler to operate and more efficient. Therefore, a multi-robot system may be more cost effective than a single robot system.

There are many different types of control techniques in operating the multi-robot systems. One technique is to divide the workload distinctively. Figure 1.1 depicts the multi-robot Centibots system during a mapping and tracking demonstration [2]. This system consists of a hundred autonomous robots in which the first group was deployed to map an area and thereafter, a second group searched and tracked certain objects within the mapped area. The system demonstrated the ability of a multi-robot system to co-ordinate, communicate and distribute tasks effectively.

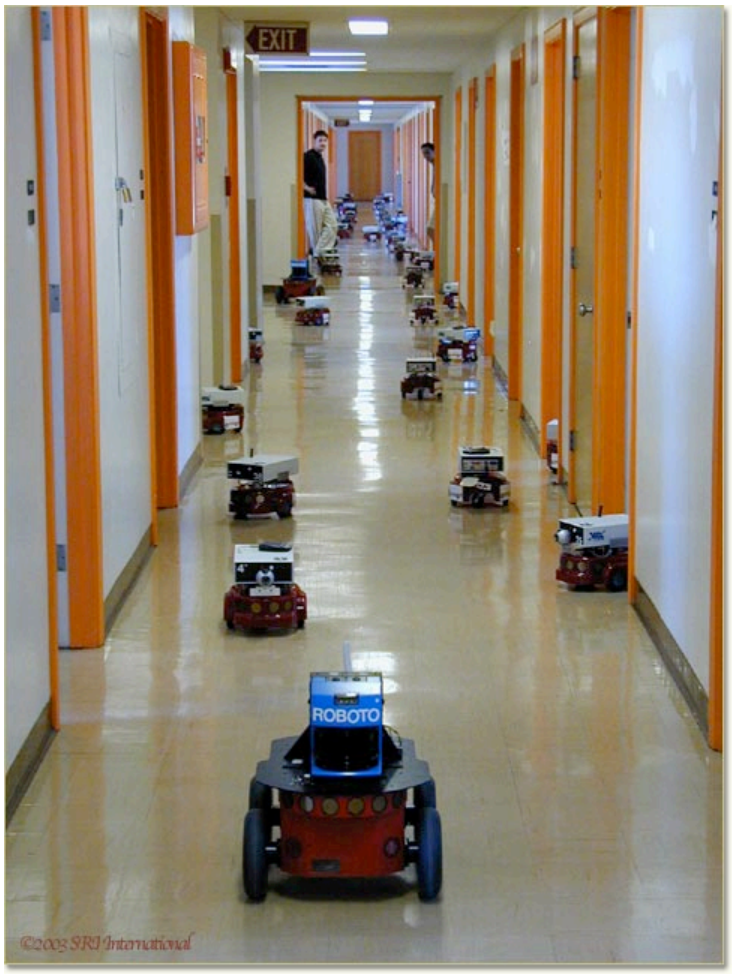

Figure 1.1 The Centibots Project [2]

Another control technique called “Follow the Leader” [26-28] is implemented in a decentralized configuration and is commonly used by many researchers because of its ease of use and expandability. This technique designates a leading robot, and the rest are followers. In conducting the operation, the follower robots organize in formation by keeping fixed distances and/or orientation from the leader. A feedback loop maintains the relative distance and orientation from the follower. A drawback of this technique is that it requires the leading robot to be constantly functioning and if it fails, the follower robots may cease to move.

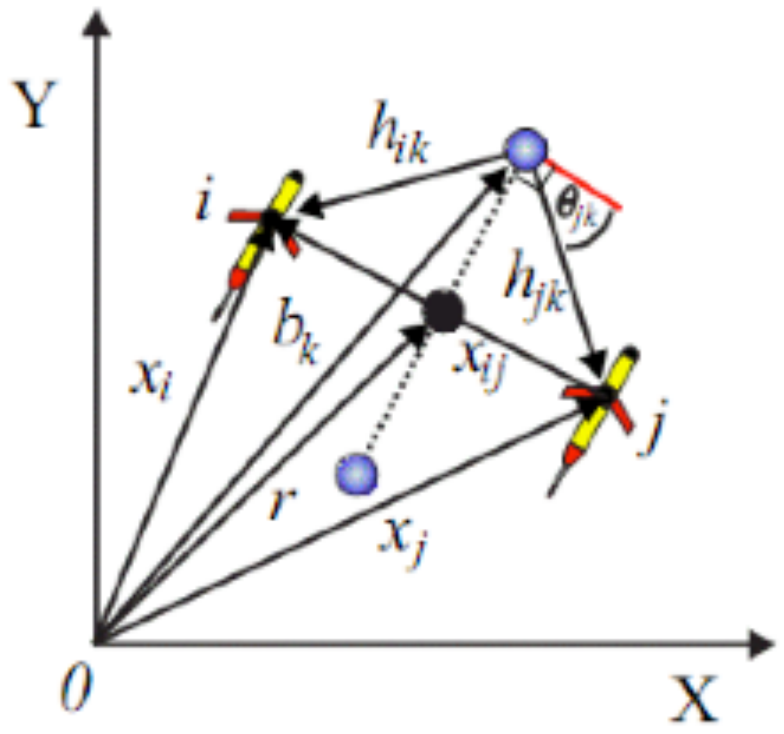

Figure 1.2 Two-Dimensional view of multi-AUV formation [5]

Another multi-robot platform, develop by the Monterrey Bay Aquarium Research Institute (MBARI), is the Autonomous Ocean Sampling Network (AOSN) II [3]. This is a multi-robot system to perform ocean sampling. The AOSN implements an adaptive formation control technique, which has been developed and tested on many autonomous underwater gliders [4]. The system employed three-dimensional multi-robot formation control. Non-holonomic robots that can encounter substantial disturbance forces such as ocean currents [5] are used. The size and shape of the three-dimensional cluster have actual vehicles and virtual reference points. Figure 1.2 depicts a simplified two-dimensional representation of the cluster formation. Successful demonstrations of the system were performed in the summer of 2003 in Monterey Bay.

1.2 Gradient Navigation Techniques

Gradient-based navigation is a system of self-regulating adjustment, and the closed-loop adaptation of the robotic system in response to the environment helps to achieve optimum performance in the parameter field. This phenomenon of gradient climbing often exists in biological organisms. For example the survival of aquatic organisms depend on finding areas with food sources. Schools of fish are known to climb gradients of nutrients in order to find the most concentrated sources of food [8] [25]. Moreover, birds, ants and bees are also known to follow natural coordinated collective behavior. Such behaviors demonstrate techniques for more effective searches for food sources and increased energy efficiency.

1.2.1 Previous Work on Gradient Climbing Techniques

The Autonomous Benthic Explorer (ABE) demonstrated gradient climbing with a single robot to find the deepest location in a lake [5]. This is an example of a single vehicle technique using the behavior model of the agellated bacteria that is similar to the Escherichia coli. Chemotaxis happens when individual bacterium move with respect to the chemicals that they sense in their environment. The "run and tumble" [6] method is the technique used to achieve chemotaxis [7]. The more concentrated the gradient, the longer the bacterium continues straight before the bacterium tumbles and changes direction. The opposite is true when the gradient is less concentrated; the bacteria will change directions more often. A limitation of this approach is that the convergence rate can be slow to the source [8].

Moths use anemotaxis to align themselves with the wind to find the source of pheromone. Similar to moths, an anemotaxis probe is a device that can be used for finding the location of an odor source. Such a probe can be used on a single robot by using a configuration of four anemometric sensors along with four gas sensors to determine the direction of the odor source [30-34]. The wind direction is measured by the anemometric sensor. The instantaneous gas-concentration gradient is measured by the gas sensor [9], [15]. The chemical plume is tracked by orienting the robot in the equivalent bearing as the incoming wind. When the plume is lost, the robot will move to and fro in the wind in order to locate the plume. The limitation of this technique is that it only covers a small area.

Another strategy to climb a gradient is casting. Casting is when the odor is no longer present, a local search is performed until it is relocated. The location of the maxima reading is estimated by using the location of the previous sensor reading. Silk moths have been observed to use surge-cast behavior [11]. Surge-cast is a combination of casting and upwind surge. When the moth is in contact with the plume, it continues to go on an upwind direction, and when it loses the plume it zig-zags to relocate the plume. Surge-cast and casting performance has been studied using simulations [12]. The surge-cast has a significantly better performance then the casting behavior [35].

Early work in using multiple robots for odor source tracking was first done by Hayes [10]. In his research, different phases were implemented in tracking odor sources. In the first phase the robots made contact with the plume. After contact was made with the plume, the robots tracked the chemical gradient to the source. The final phase is to locate the source of the chemical [10]. Hayes used both simulations and robotic experiments.

Another method using the nearest neighbor technique uses two vehicles that observe one another to climb up the gradient. This method is similar to the "follow the leader" approach. One vehicle follows the gradient, and the second vehicle will continue in the same direction using the shared information to converge on a point of high concentration [29].

In 2003, an extensive underwater operation that explored the areas of interest and used gradient climbing to locate and track features such as fronts and eddies was completed in Monterey Bay [13]. The systems were designed to integrate the detection and measurements of fields and features of particular interest, using a group of SLOCUM underwater gliders. The control technique implemented to climbing the parameter field is the virtual body and artificial potential (VBAP) multiple vehicle control system [5]. The gliders experienced communication disruptions and substantial disturbance forces. In spite of the complications encountered, the gliders were able to collect data and map the temperature gradients fields.

1.3 Project Statement

The purpose of this research was to demonstrate effective gradient-based navigation using the existing cluster space control technique in order to efficiently locate local maxima in a parameter field. In order to achieve this goal substantial effort was invested in the following tasks:

1. Derivation of the gradient field estimation function using three simultaneous samples of a distributed three-robot cluster.
2. Integration of the gradient estimate into the cluster space control architecture to enable gradient climbing,
3. Characterization of performance as a function of the cluster's spatial geometry using simulation of the control architecture,
4. Performance of field-testing to verify the technique using two experimental test beds: a set of three land rovers and a set of three robotic kayaks.

Gradient climbing of robots operating in the field and using simulated spatial gradients was successfully demonstrated in this work. It is worth mentioning that development of this research program was performed in collaboration with fellow graduate student, Thomas Adamek, who implemented a gradient-based technique in tracking gradient field contours of specified concentration levels [22].

1.4 Reader's Guide

There are five chapters in this thesis. The first chapter provides an introduction and background of both multi-robot systems and autonomous vehicles systems. It also covers different gradient climbing techniques and reviews several research projects. Lastly, it discusses the motivation and objectives for this thesis.

The second chapter discusses the cluster control and the formal gradient climbing techniques. It also includes the kinematic transforms and control framework of a three robot cluster. The third chapter illustrates the simulations of gradient cluster control. In addition, the performance of the gradient controller is characterized as a function of the cluster shape. The fourth chapter evaluates the three-ASV cluster space control through hardware experimentation. Finally, the fifth chapter summarizes the results of the thesis project. It also discusses the conclusions of the results and identifies the areas for future work.

Chapter 2 Cluster Space Based Gradient Climbing

2.1 Introduction

We first start off by describing the cluster space formation control technique. We present the cluster space representation, the associated kinematic transforms, the closed loop control architecture, and the singularities associated with a specific geometrical representation. Cluster space control allows for the simple and flexible control of many robots. Different control algorithms can be interfaced into this cluster control system. The second part of this chapter discusses how we use the cluster space technique to climb gradients with multi robot systems.

2.2 Review of the Cluster Space Control Technique

Controlling relative position and orientation of the system is a major challenge in multi-robot systems. Over the past few years, faculty and students in SCU's Robotic Systems Lab (RSL) developed the cluster space control technique to address this challenge [14]. The objective of the cluster control is to allow operators to specify simple motion and geometry directives for a formation and to have a formalized, automated process determine and execute the necessary robot-level drive commands that achieve this specification.

The core of the strategy is to consider a *n*-robot system as a single cluster entity. We specify the motion of the cluster as a function of independent cluster pose variables (and their time derivatives) such as cluster position and orientation, cluster shape, and the relative orientation of each robot with respect to the cluster. The controller computes its compensation commands in the cluster space, and these commands are converted to conventional robot space commands through kinematic transforms.

Previous researchers at the SCU Robotic Systems Lab (RSL) have published a framework to develop the cluster space approach with a system of *n* robots with each robot having its own degrees of freedom (DOF) [14]. Automated trajectory control [15-16], and versions of cluster-space-based regulated motion [17] have been successfully demonstrated for this framework. These demonstrations have included experiments with two-, three-, and four-robot planar land rover clusters [18-19], with a two-boat surface vessel system [20], and for robotics systems that are both holonomic and non-holonomic. RSL research has also demonstrated task- and application-specific controllers that hierarchically interact with multi-robot cluster space controllers in order to provide services such as escorting and patrolling [21].

The individual robots will ultimately actuate the control actions given by the cluster controller. The formal kinematic relationships of cluster space variables and robot space variables will be described below. The selection of cluster variables can be a function of the application, operator preferences, or the available communication and computation resources. The cluster reference frame is determined by the cluster space variable description. The references to individual robots in the cluster are in the cluster frame shown in Figure 2.1. The individual robot frame is with respect to the global frame. This transform in the form of a homogeneous transform [14].

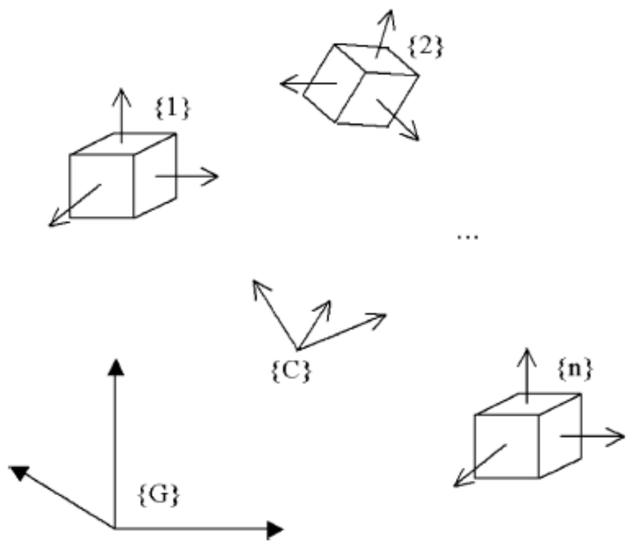

Figure 2.1 Robot pose using conventional vs. cluster space representation [14]

Cluster space pose variable is defined in equation (1),

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

where c_n represents variables such as position, orientation and geometry of the cluster. The robot space state variables are defined by,

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

where n is the number of robots, and the position and orientation of robot n is defined by $(x_n, y_n, \theta_n)^T$. A formal set of kinematic transformation ($KIN(^{G}\vec{R})$) relates the robot space variable to the cluster space variables in equation (3). The inverse kinematic $(INVKIN(\vec{C}))$ transforms cluster space variables to robot space variables in equation (4).

$$\vec{C} = \begin{pmatrix} c_1 \\ c_2 \\ . \\ c_{mn} \end{pmatrix} KIN(^{G}\vec{R}) \begin{pmatrix} g_1(r_1, r_2, ..., r_{mn}) \\ g_2(r_1, r_2, ..., r_{mn}) \\ . \\ g_{mn}(r_1, r_2, ..., r_{mn}) \end{pmatrix} \qquad (3)$$

$$^{G}\vec{R} = \begin{pmatrix} r_1 \\ r_2 \\ . \\ r_{mn} \end{pmatrix} INVKIN(\vec{C}) \begin{pmatrix} h_1(c_1, c_2, ..., c_{mn}) \\ h_2(c_1, c_2, ..., c_{mn}) \\ . \\ h_{mn}(c_1, c_2, ..., c_{mn}) \end{pmatrix} \qquad (4)$$

The relationship of velocities between cluster space and robot space ($\dot{\vec{C}}, {}^{G}\dot{\vec{R}}$) is considered by taking the derivative of equation (3). This is expressed as the Jacobian matrix J in equation (5). The robot velocities can be mapped to cluster velocities in a form of a linear time varying function using the Jacobian matrix.

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

Using a similar method the inverse mapping of cluster velocities to robot velocities can occur by using the inverse Jacobian matrix J^{-1} in equation (6).

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

The desired motions are specified and control compensation are computed in the cluster space. The compensation commands are transformed to robot space using the inverse Jacobian relationship. The robot-space velocities are transformed to actuator level commands using a vehicle-level inverse Jacobian relationship. Figure 2.2 show the control architecture for trajectory based cluster space control. The cluster can be sent trajectory commands for how the cluster should move, rotate, and transform its shape over time [24].

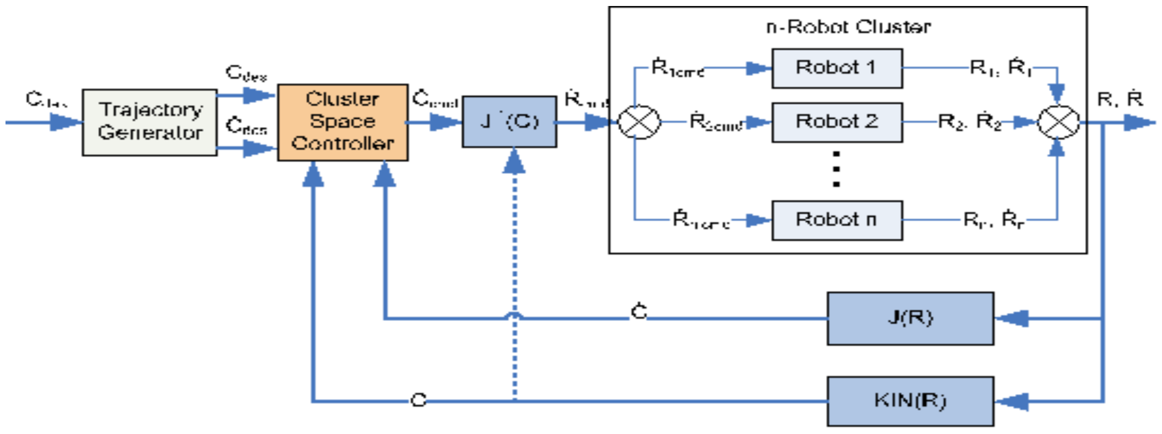

Figure 2.2 Trajectory cluster controller [9]

2.3 Three-Robot Cluster Definition

For this application of the cluster space framework, gradient climbing has been applied to the control of a simple three-robot cluster with differential drive robots operating in a planer formation. We review the selection of the cluster space variables with their resulting kinematic transforms for planer robots. This robotic cluster is used as part of the implementation of a gradient-based climbing cluster space controller.

Equation (7) is the three-robot cluster variables defined by figure 2.3. This three-robot kinematic formulation was developed in [14].

$$\vec{C} = (x_c, y_c, \theta, \phi_1, \phi_2, \phi_3, p, q, \beta)^T \tag{7}$$

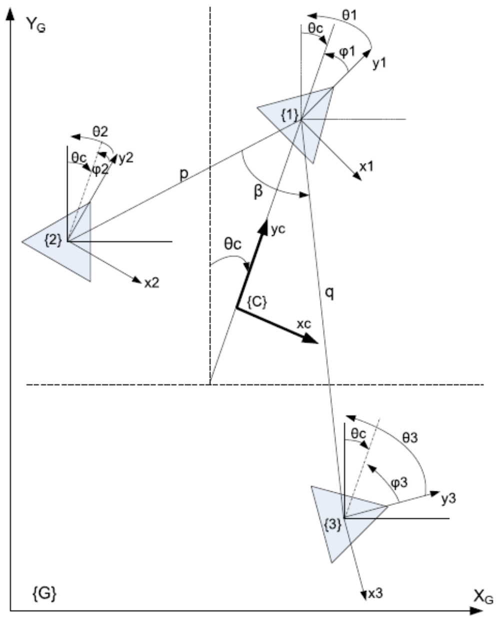

Figure 2.3 Three-robot cluster definition [9]

The local robot variables are defined by the x and y of the individual robot spatial position, and by θ, the heading relative to the global frame of the robot shown in equation (8):

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

The three-robot cluster [14] is controlled by nine cluster variables: x and y, cluster heading (θ_C), ϕ_1, ϕ_2, ϕ_3, P, Q and β. The cluster x and y position is the average of the three robot positions, as defined by equation (9).

$$x_c = \frac{x_1 + x_2 + x_3}{3}, y_c = \frac{y_1 + y_2 + y_3}{3} \tag{9}$$

The cluster heading, θ_C, is defined by equation (10).

$$\theta_c = arctan2\left(\frac{2x_1 - x_2 - x_3}{2y_1 - y_2 - y_3}\right) \tag{10}$$

The variable ϕ_i is the robot's heading in relationship to the cluster heading, defined by equation (11).

$$\phi_i = \theta_i + \theta_c \tag{11}$$

The variable P is the distance from robot 1 to robot 2, defined by equation (12).

$$P = \sqrt{(x_1 - x_2)^2 + (y_1 - y_2)^2} \tag{12}$$

The variable Q is the distance from robot 1 to robot 3, defined by equation (13).

$$Q = \sqrt{(x_1 - x_3)^2 + (y_1 - y_3)^2} \tag{13}$$

The angle β is the angle between robot 2 and 3, defined by equation (14).

$$\beta = a\cos\left(\frac{P^2 + Q^2 - (x_3 - x_2)^2 - (y_3 - y_2)^2}{2PQ}\right) \tag{14}$$

The inverse kinematics are defined by equations (15) to (24):

$$x_1 = x_c + \left(\frac{1}{3}\right) rsin\theta_c \tag{15}$$

$$y_1 = y_c + \left(\frac{1}{3}\right) rsin\theta_c \tag{16}$$

$$\theta_1 = \phi_1 - \theta_c \tag{17}$$

$$x_2 = x_c + \left(\frac{1}{3}\right) rsin\theta_c - Psin\left(\frac{\beta}{2} + \theta_c\right) \tag{18}$$

$$y_2 = y_c + \left(\frac{1}{3}\right) rsin\theta_c - Psin\left(\frac{\beta}{2} + \theta_c\right) \tag{19}$$

$$\theta_2 = \phi_2 - \theta_c \tag{20}$$

$$x_3 = x_c + \left(\frac{1}{3}\right) rsin\theta_c - Qsin\left(\frac{\beta}{2} + \theta_c\right) \tag{21}$$

$$y_3 = y_c + \left(\frac{1}{3}\right) rsin\theta_c - Qsin\left(\frac{\beta}{2} + \theta_c\right) \tag{22}$$

$$\theta_3 = \phi_3 - \theta_c \tag{23}$$

where,

$$r = \sqrt{(Q + Pcos\beta)^{\,2} + (Qsin\beta)^{\,2}} \tag{24}$$

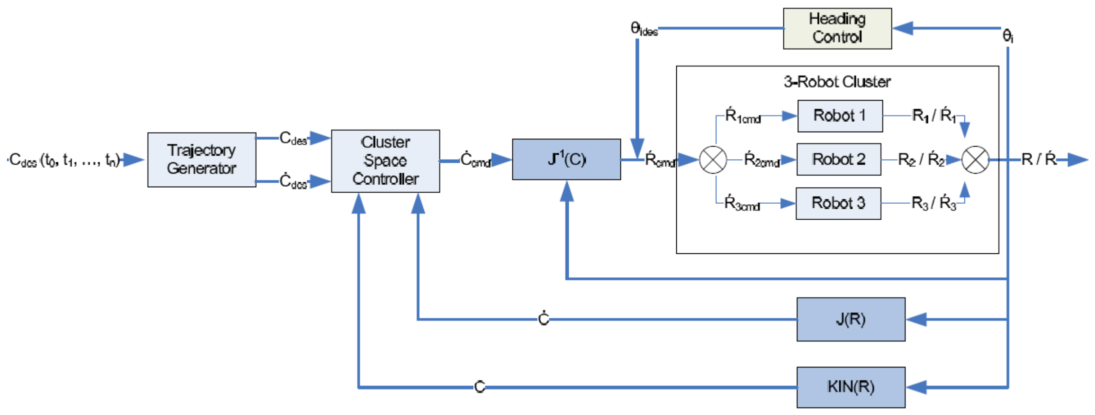

Figure 2.4 Three-robot PID controller cluster definition

Since the space is limited, the full algebraic expressions for $J^{-1}(\vec{C})$and $J(\vec{R})$ are not included here. They are presented in previous RSL research papers [21]. Given the three-robot control architecture is shown above in figure 2.4 for a trajectory controller [9]. The gradient controller will be presented later in the chapter.

There are cluster singularities where the cluster controller becomes unstable. When all the robots are in the same position the cluster heading becomes undefined. This is an unlikely situation in the field, but is possible in the simulation. To avoid this situation the P and Q are kept at a likely distance that the actual testbed would be set to such as 3-5 meters. The other singularity is when β is equal to 0 or 180 degrees. To avoid this the β is kept between 15 degrees and 165 degrees. A change of cluster variables can be used to allow for straight line configurations, where all three robots are aligned in a straight line. Therefore the use of specific set of cluster variables does not imply any loss of generality of the use of gradient-based navigation techniques with cluster space control.

2.4 Cluster-Based Gradient Climbing [23]

Applications in navigating a robotic cluster based on the gradient of a parameter field are useful. This functionality includes tasks such as finding the peaks or valleys in bathymetry. Similarly, point sources of pollutants can be located, and the minimum or maximum value of measurable, continuous phenomena can be tracked. As demonstrated by T. Adamek in [22], navigation may also be performed with respect to the contours of such fields in order to perform relevant applications, such as patrolling or collecting iso-parameter scientific data. In ship ports a gradient climbing robotic cluster can identify ships that release pollution into the marine environment. On land, gas plume sources can be identified, and traced.

In this thesis our purpose is to climb the gradient field. In order to accomplish this, the direction "up" the gradient field must be estimated by using real-time sensor measurements on the distributed robot network. The second task is to steer the cluster in the estimated direction.

The gradient field orientation estimation is demonstrated in figure 2.5. The red dots in the x-y plane represent the robots. The parameter field is represented by the dotted red contour lines within the x-y plane and also as an inclined plane above the x-y plane. The equivalent positions of the robots on the inclined parameter field are represented by the green dots. The Z_i value is the concentration value sensed by each robot, and it also represents the height of the parameter field. We create vectors $\vec{R}_{12}$ and $\vec{R}_{13}$ which point from the virtual robot 1 location to the virtual locations of robots 2 and 3, respectively.

The direction of the gradient field is computed by the cross product of $\vec{R}_{12}$ and $\vec{R}_{13}$; this results in the $\vec{N}$ as seen in Figure 2.5 and computed in equation (27). Projecting this vector into the x-y plane establishes the $\overrightarrow{\nabla f}$, as computed in equation (28), and which points in the direction of the maximum parameter increase. Given that we wish to drive the cluster in this direction, we require the globally referenced heading (θ_{Climb}) of this vector. As seen in figure 2.5 and conveyed in equation (30), this is easily computed. We assume that the parameter field is planar at the location of the cluster.

For the cluster space controller, we set the desired cluster heading to the gradient heading. A simple proportional control law, shown in equation (33), is used to rotate the cluster to this angular position. The integration of this control objective with the rest of the cluster space controller is shown in figure 2.6 [23]. As can be seen, cluster translational velocity is specified open loop as a constant, and each cluster shape variable is controlled with a proportional law.

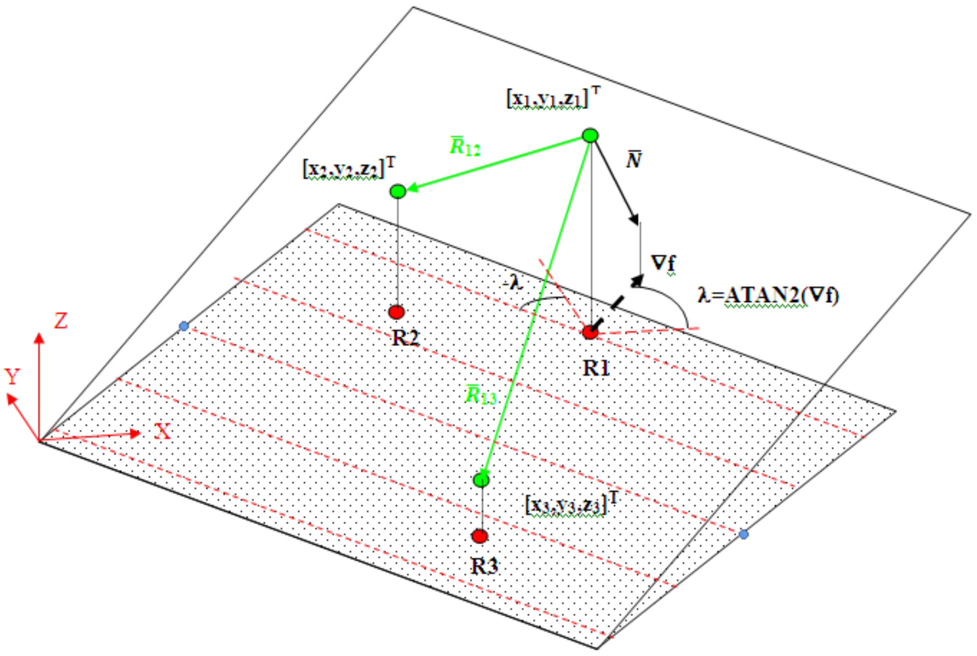

Figure 2.5 A simple 3-D robot cluster gradient-based description

$$\vec{R}_{12} = \begin{pmatrix} x_2 - x_1 \\ y_2 - y_1 \\ z_2 - z_1 \end{pmatrix} \tag{25}$$

$$\vec{R}_{13} = \begin{pmatrix} x_3 - x_1 \\ y_3 - y_1 \\ z_3 - z_1 \end{pmatrix} \tag{26}$$

$$\vec{N} = \vec{R}_{13} \times \vec{R}_{12} \tag{27}$$

$$\overrightarrow{\nabla f} = \begin{pmatrix} -N_y \\ N_x \end{pmatrix} \tag{28}$$

$$\theta_{Climb} = arctan2\left(\overrightarrow{\nabla \mathrm{f}}\right) \tag{29}$$

$$\theta_{Climb} = arctan2\left(\frac{\sqrt{N_x^2+N_y^2}+N_x}{-N_y}\right) \tag{30}$$

In order to orient the cluster up the gradient our control scheme uses the estimation of the parameter field's gradient and the measure cluster $\theta_{CActual}$ bearings. The control form to climb the gradient is shown in equations (31) to (33).

$$\theta_{Cdes} = \theta_{Climb} \tag{31}$$

$$\dot{\vec{\theta}} = K_{\theta_c}(\theta_{Cdes} - \theta_{CActual}) \tag{32}$$

$$\dot{\vec{\theta}} = K_{\theta_c}\left(arctan2\left(\overrightarrow{\nabla \mathrm{f}}\right) - \theta_{CActual}\right) \tag{33}$$

For the cluster controller, a simple proportional law is used where $\left(\dot{\theta}_{Cdesire}\right)$ is a simple proportional to function of the error in the desired heading $\left(\theta_{Cdesire}\right)$ as a set point. Integration of this gradient climbing control augmentation with the rest of the cluster space controller is depicted in figure 2.6 [23].

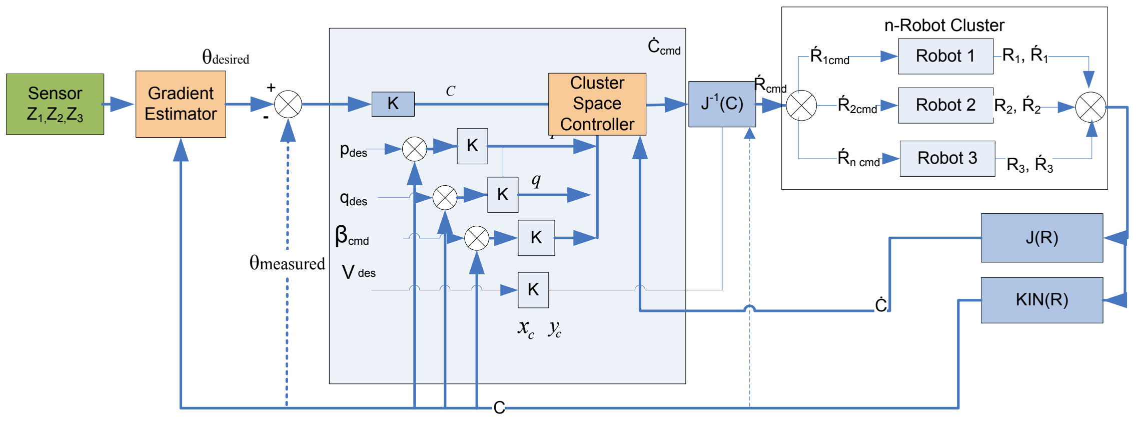

Figure 2.6 Gradient climbing controller

The cluster x and y velocities are calculated by using the measured cluster heading. This allows the cluster to turn into the gradient. In the experiments the desired P, Q, and β are held constant through out the individual runs. All ϕ's, the robot's heading in relationship to the cluster heading, are left uncontrolled. The actual $\theta_{heading}$, P, Q, and β are controlled using close loop PID control. The control inputs are feed into the inverse Jacobian. The inverse Jacobian outputs the desired x and y velocities for each robot in the cluster. Each individual robot has an independent controller for x and y velocities.

2.5 Gradient Singularities

In addition to the cluster singularities, there are five cases of gradient singularities. When the parameter field strength is infinite the slope is 90 degrees, and is unrealistic. When the parameter field strength is zero, there is no gradient field to measure. The last three gradient singularities are also cluster singularities. In the case that P or Q is zero, two or more of the robots are at the same position and the cluster loses a degree of freedom. This is avoided by using a collision avoidance control policy. The other two cases occur when β equals 0 and 180 degrees. In both incidences the three robots are lined up in a 1-D plane. The information is lost in the spatial dimension perpendicular to that line. We avoid this by selecting a desired value of β away from this configuration.

Chapter 3 Simulations

The cluster gradient climbing technique was verified using computer simulations of the controller. This chapter describes the simulation environment and simulation tests using a three-robot cluster. Results from the simulations verify the operation of the controller and are used to characterize the performance of the controller.

3.1 Simulation Environment

The simulations of the experimental tests were computed in Simulink. Simulink is a part of the software package Matlab that performs simulations for model-based dynamic systems. Simulink is a multi-domain environment for model base dynamic systems. Simulink has a graphical user interface with customizable block libraries to perform the simulations. The closed-loop controller used in the simulations is identical to the closed loop controller in the robotic experimental test set up. This simulation environment allows for the modeling of the dynamics of the three robot cluster, and characterizing the performance of the system.

The sensor generation function outputs the sensor data into the gradient controller. The gradient controller converts the global positions into the local cluster frame, and outputs the desired cluster heading. The cluster space controller generates the cluster space errors and computes the desired cluster velocities using a proportional control law. The cluster space variables are calculated from the forward kinematics. The VRML (Virtual Reality Markup Language) reads the forward kinematics to output a 3D view to the user, as seen in figure 3.1. The simulations are run in a continuous Simulink loop. The Simulink program for the gradient controller is given in Appendix B due to space. The simulation has three independent differential drive planer robots. The motor control signals were inputs into models of the robots. We assume a linear response function for the model of the robot's actuators. The actuator is assume to be a 1st order system. The model parameters were experimentally determined from the vehicle step response from previous work from the RSL [9].

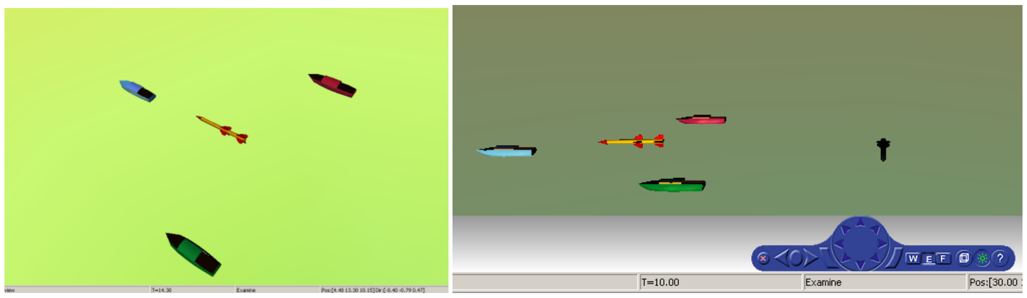

Figure 3.1 - Screen shot of virtual simulation tests (VRML)

3.2 Gradient Climbing Technique Verification

In this section the cluster based gradient climbing technique was verified using the Simulink simulator to climb a prescribed gradient. A simple fixed baseline test case was performed to determine functionality of the model as a precursor for different model properties, variables and future characterization. The tests were also performed to ensure that the frame transforms, kinematics, and gradient climbing technique were correct and the model behaved as designed. Simple planer equations without noise were used to verify the gradient climbing technique in the simulations. Figures 3.2, 3.3, and 3.4 show an example simulation of a 45-degree gradient climb. The cluster heading started 45 degrees away from the gradient. It had a steady state cluster heading error RMS less than 10^{-5} degrees. As can be seen in the figures, after an initial transient, the cluster maintains its shape while moving up the gradient. In this test the cluster configuration was set to $[P, Q, \beta] = [10\ m, 10\ m, 45°]$.

Other tests were conducted without noise, using different configurations of P and Q, β, and gradient plane slopes. The cluster formation is maintained and shows control of the system. The results were similar to the previous example. Without gradient sensor noise the heading error is expected to be negligible because the system can converge to a single heading. These tests verified the technique using ideal constant gradient planes without using gradient sensor noise.

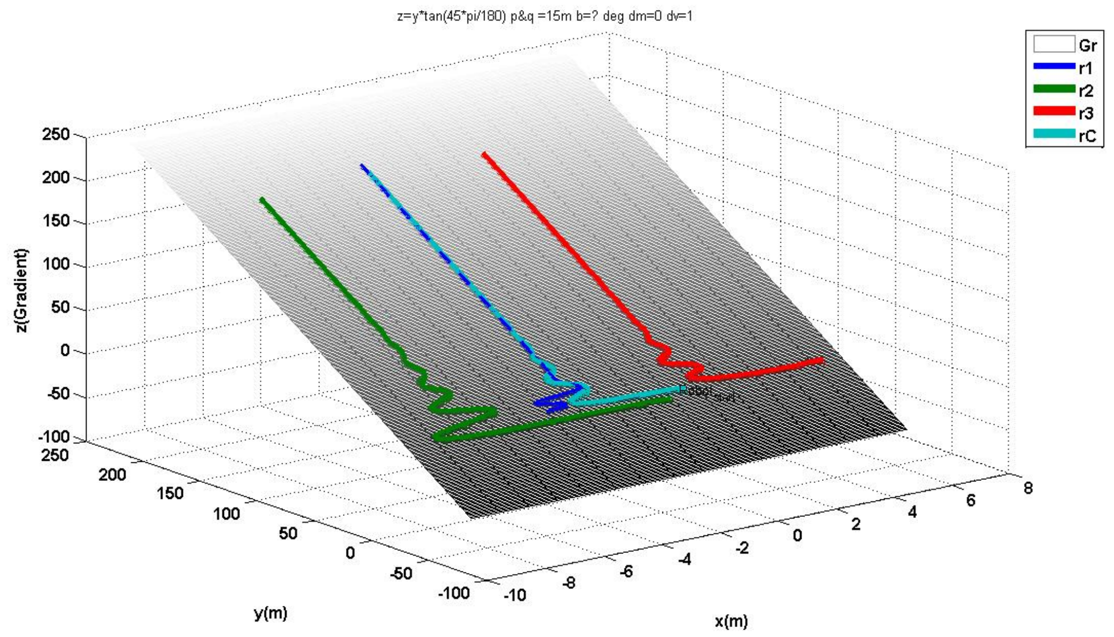

Figure 3.2 3D-plot of gradient climbing of a three-robot cluster without noise

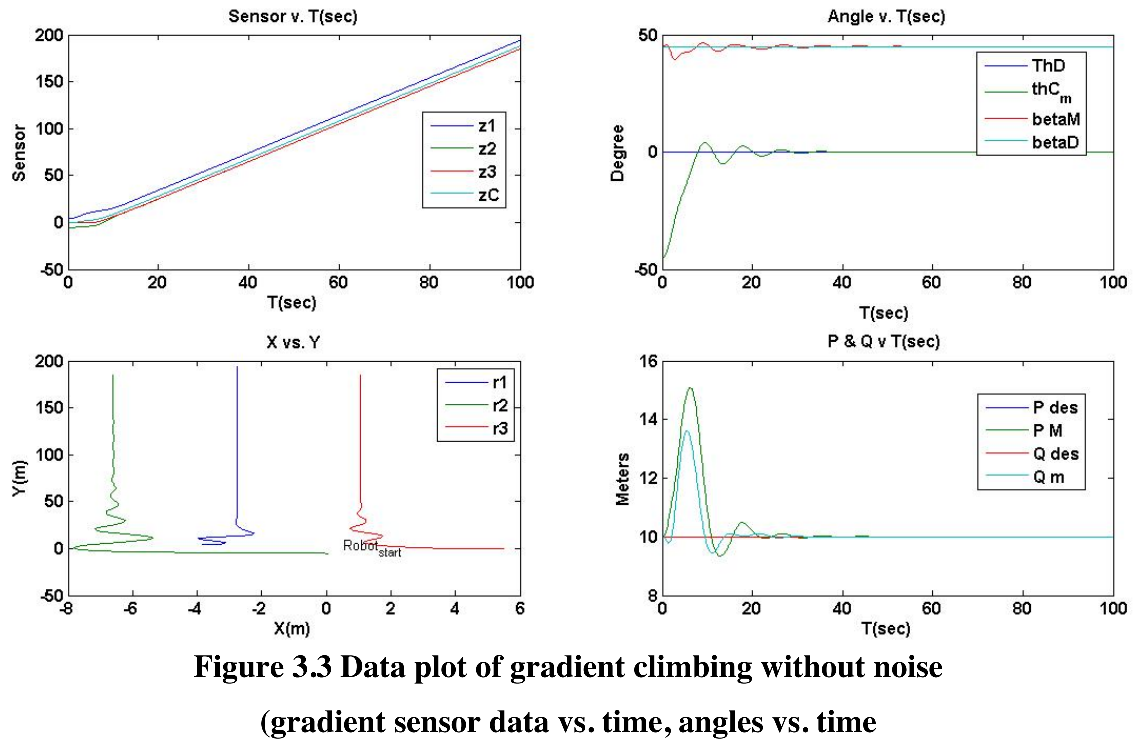

Figure 3.3 Data plot of gradient climbing without noise

(gradient sensor data vs. time, angles vs. time

X vs. Y, P&Q vs. time)

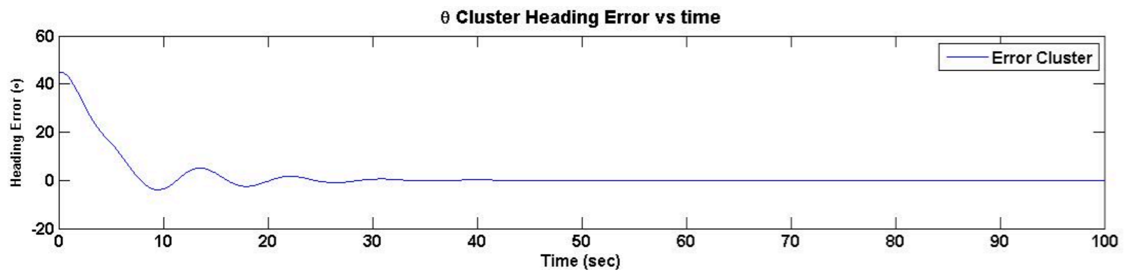

Figure 3.4 Cluster heading state error verse time of figure 3.2 and 3.3 (RMS steady state error $< 10^{-5}$ deg)

3.3 Test with Simulated Gradient Noise

The system was tested with more realistic simulations by adding gradient sensor noise. An independent Gaussian noise function with a mean of 0.0 units and variance of 1 unit was added into the sensor reading. All experiments with gradient sensor noise use this identical noise function. The gradient computation and cluster dynamics were left unchanged. The metric of studying the effect of sensor noise on the system was the RMS of the cluster-heading error. The gradient sensor noise affects the accuracy of obtaining the desired cluster heading. Increasing or decreasing the noise directly affects the stability of the system. Gradients with a constant plane without gradient sensor noise have a constant heading, but with sensor noise the instantaneous heading is changing as a function of the amount of noise. The effect of the sensor noise was quantified by using the RMS in the error of the desired cluster heading in simulations. Figures 3.5 to 3.7 shows a simple case of gradient climbing with sensor noise. The configuration was with a 45-degree plane, P and Q set to 15 meters, β set to 45 degrees, and the starting cluster heading aligned with the gradient. The RMS of the error in the desired cluster heading was 11.15 degrees with a variance of 2.17 degrees. The addition of sensor noise causes the cluster to drift perpendicular to the gradient due to the increase in the cluster heading error. In the next section we investigate the effect of sensor noise as a function of the shape of the cluster configuration.

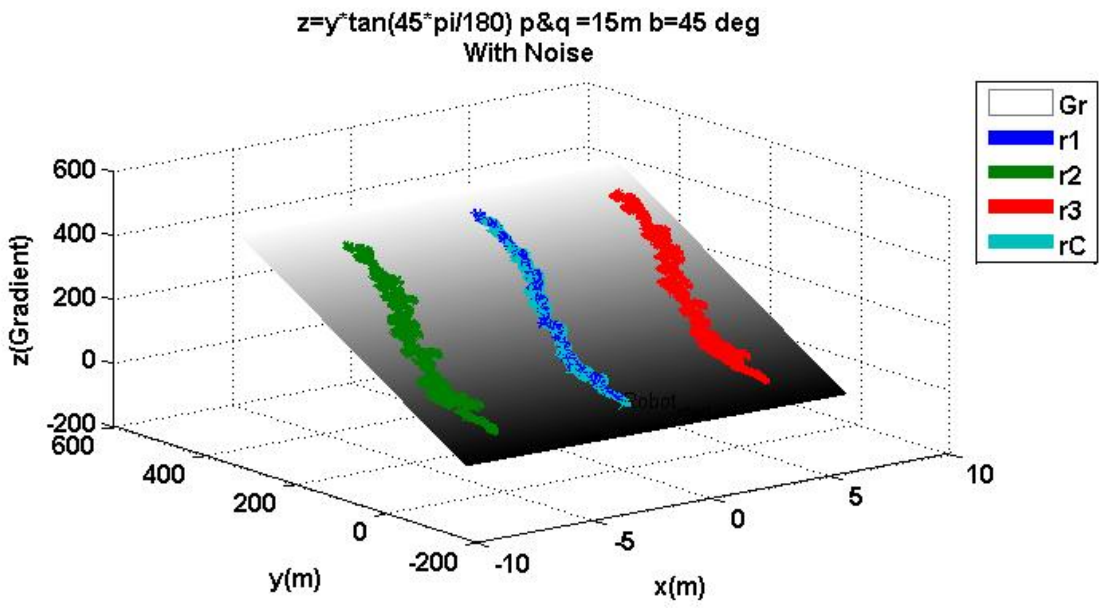

Figure 3.5 3D Plot of $z = y\tan\left(\frac{45\pi}{180}\right)$ with gradient sensor noise

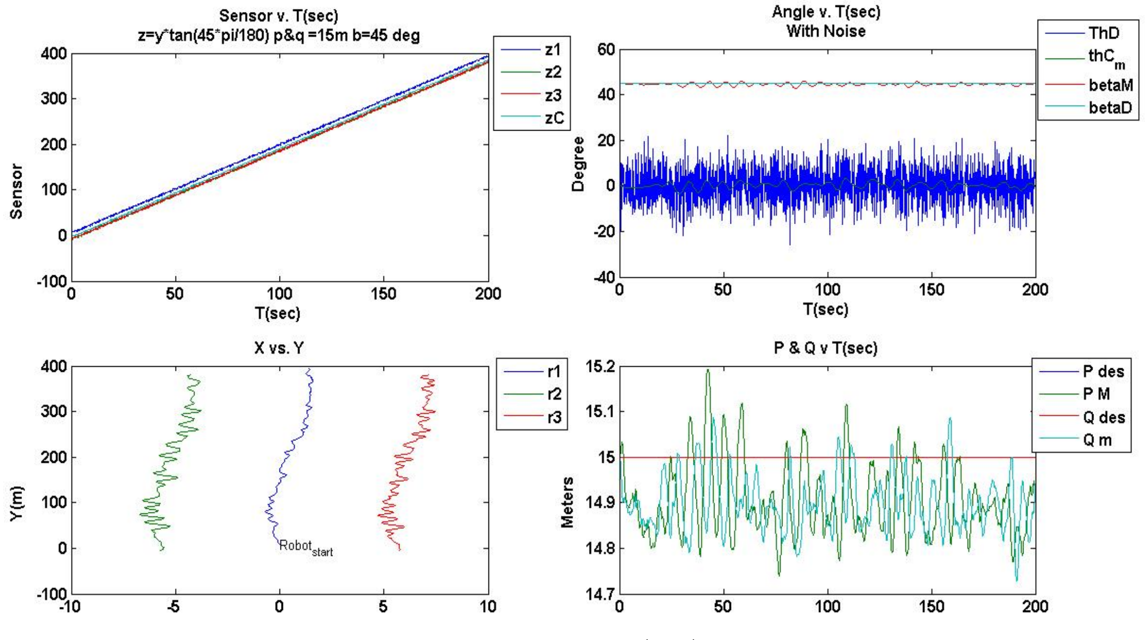

Figure 3.6 Quad plot of $z = y\tan\left(\frac{45\pi}{180}\right)$ with gradient noise

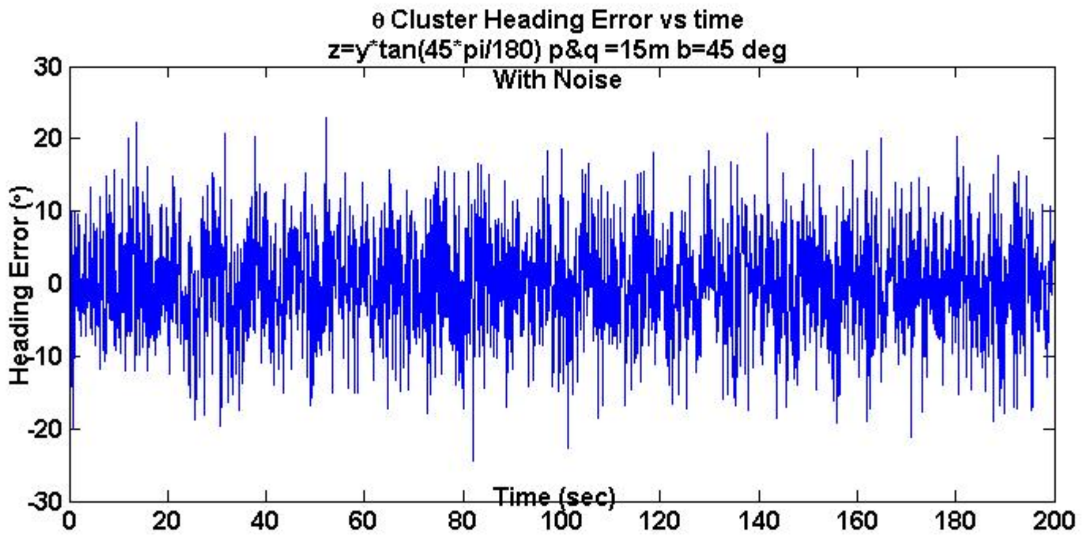

Figure 3.7 Plot of cluster heading error vs. time for $z = y\tan\left(\frac{45\pi}{180}\right)$ with gradient noise (RMS error of 11 deg)

3.4 Simulation Experiments

In this section a series of simulation experiments were performed to test the effects of the cluster shape with gradient sensor noise on the RMS cluster heading error. One cluster variable was independently changed for each simulation run. The variables changed were β, P and Q, and the slope of the gradient plane. Each simulation ran for 200 seconds. The slope of the gradient was set to 45 degrees when not varied. Similarly P and Q were set to 15 meters, and β was set to 45 degrees when not varied. The variables β and P and Q were chosen to be similar to field conditions. The noise functions were same as described in Chapter 3.3.

3.4.1 Effects of Varying Only β

Figures 3.8 to 3.10 shows results of varying only β while including the noise function. In these simulation runs the gradient plane was set to 45 degrees. P and Q were set to 15 meters. These simulations ran for 200 seconds. Figure 3.8 shows the results with β set to 5 degrees. This value of β is near a gradient singularity, so we expect the system to be unstable. This simulation run had a high RMS cluster heading error of 63 degrees as expected for being to close a singularity. Figure 3.9 shows the results with β set to 30 degrees. This value of β is further away from the singularity and we expect a smaller error than the pervious run. The RMS heading error was 11 degrees. With β set to 135 degrees, the results shown in figure 3.10 has the low RMS heading error of 3.3 degrees because the value of β is further away from any singularity.

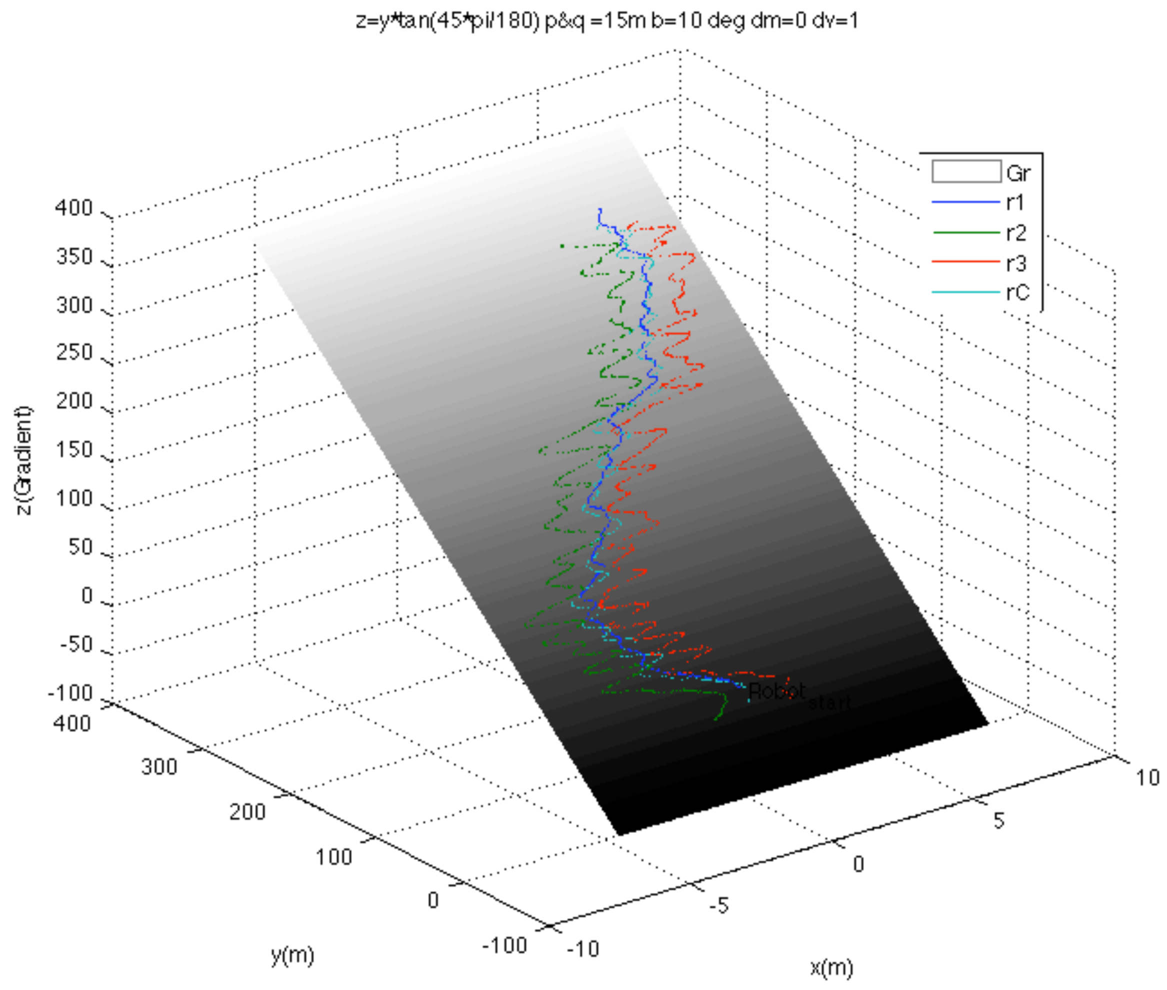

Figure 3.8 Simulation results for β=5 degrees

(Gradient plane = 45 deg, P&Q =15m)

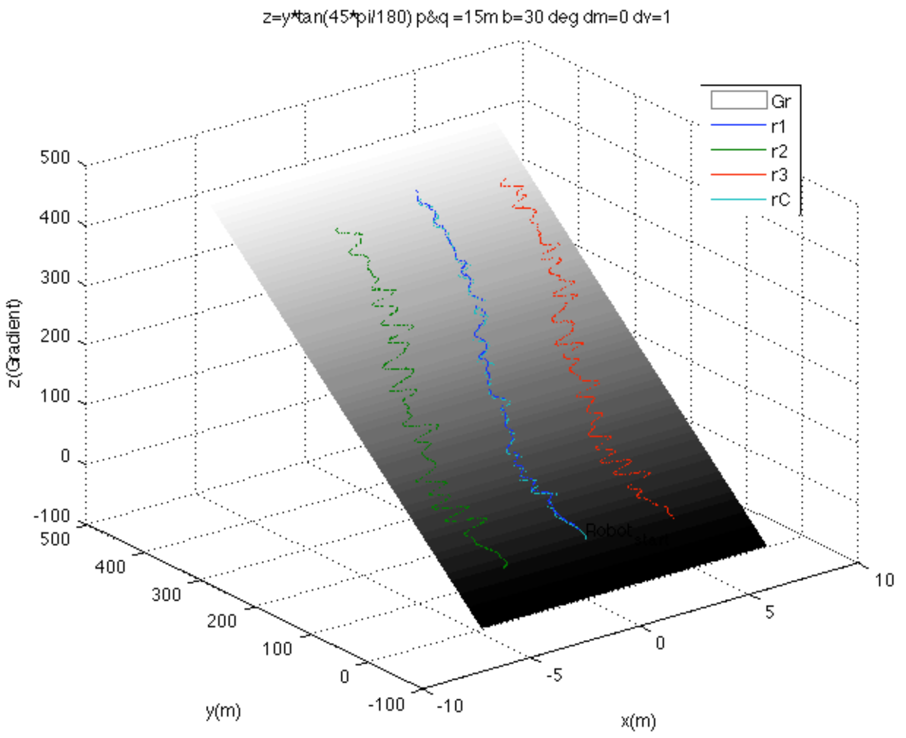

Figure 3.9 Simulation results for β=30 degrees

(Gradient plane= 45 deg, P&Q = 15m)

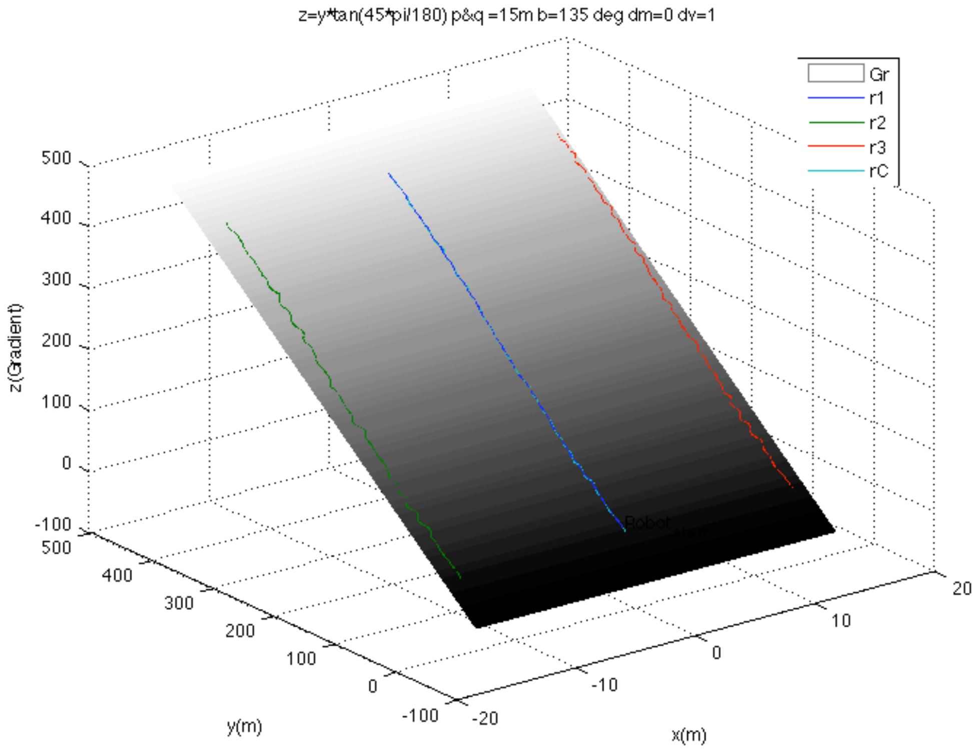

Figure 3.10 Simulation results for β=135 degrees

(Gradient plane=45 deg, P&Q = 15m)

To further develop this relationship, a set of simulation experiments was performed with the values of P and Q, and the gradient slope held constant. Only the values for β were varied. In this set of simulation experiments the gradient slope was set to 45 degrees and P and Q were set 15 meters. The value of β was varied from 5 degrees to 165 with 10-degree increments. A total of 17 simulations were conducted. The data from these simulations are shown in the first graph of figure 3.17. With small β angles, the RMS cluster error heading drops as the β angle increases. This results because increasing β moves the cluster away from the gradient singularity. For β values of 30 degrees to 150 degrees there is a region of stability, where the RMS error is constant. For β values larger than 150 degrees the RMS cluster error heading increases as β increases. This results because increasing β in this region moves the cluster closer to the gradient singularity. The two β singularities are 0 degrees and 180 degrees as mentioned in chapter 2. Thus we conclude that moving the value β away from the singularities increases the stability of the system.

3.4.2 Effects of Varying Only P and Q

Figures 3.11 to 3.13 show the results of the effects of varying only P and Q with the noise function. In these simulations the gradient plane was set to 45 degrees and the value of β was set to 45 degrees. These simulations also ran for 200 seconds. The results of figure 3.11 show the results with P and Q set to 5 meters. With this case, the shortest distance between the robots, the RMS cluster heading error was 22 degrees. Figures 3.12 and 3.13 show the results for larger values of P and Q of 10 and 20 meters respectively. As the distance between the robots increases, the heading error decreases. When P and Q are set to 10 meters the heading error is 11 degrees. When P and Q are set to 20 meters the heading error is 5.5 degrees.

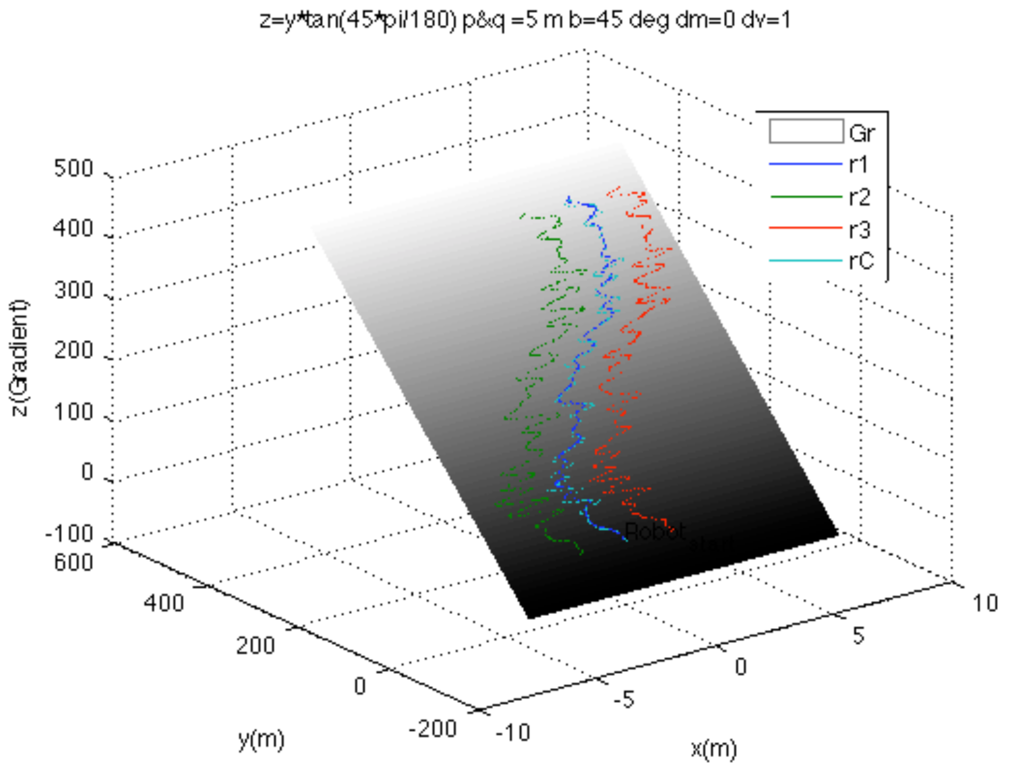

Figure 3.11 Simulation results for P&Q =5 m (Gradient plane 45, β =45 deg)

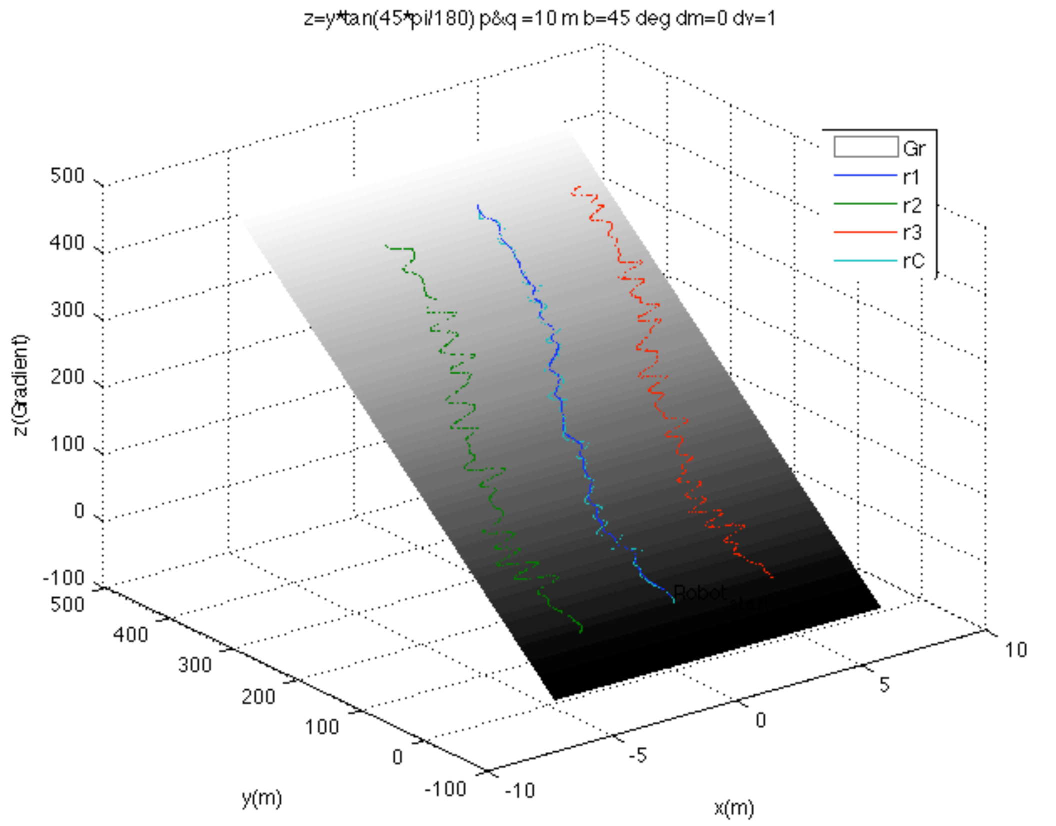

Figure 3.12 Simulation results for P&Q =10 m (Gradient plane 45, β =45 deg)

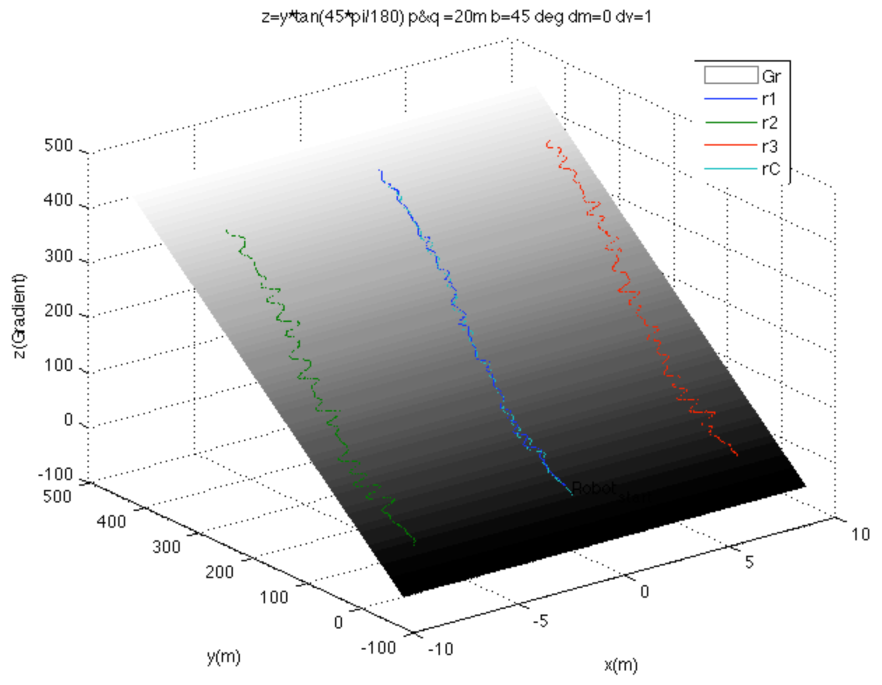

Figure 3.13 Simulation results for P&Q =20 m (Gradient plane 45, β =45 deg)

To further develop this relationship a set of simulation experiments was performed with a constant gradient slope of 45 degrees, and a constant value of β of 45 degrees. The values for P and Q were varied from 4 meters to 40 meters, with 2-meter increments. There were a total of 19 simulation experiments. The RMS cluster error heading decreased linearly for increasing values of P and Q. The results of these simulations are shown in the second graph of figure 3.17. As expected, the larger clusters have a larger gradient sensor difference in the gradient field between the robots resulting in a deviation due to the noise function. Therefore we conclude that with a smaller spatial distance between the robots, the noise function becomes more important in determining an accurate gradient heading.

3.4.3 Effects of Varying Only the Gradient Slope

Figures 3.14 to 3.16 show the results of effects of varying only the gradient slope. In these simulation experiments the value for P and Q was set to 15 meters, and the value for β was set to 45 degrees. These simulations also ran for 200 seconds. The results shown in figure 3.14, with a gradient slope set to 5 degrees, have a RMS cluster heading error of 63 degrees. Increasing the gradient slope to 30 degrees (figure 3.15) results in the heading error decreasing to 13 degrees. A low heading error in the simulation experiments of 4.2 degrees was found when the gradient angle was set to 60 degrees (figure 3.16).

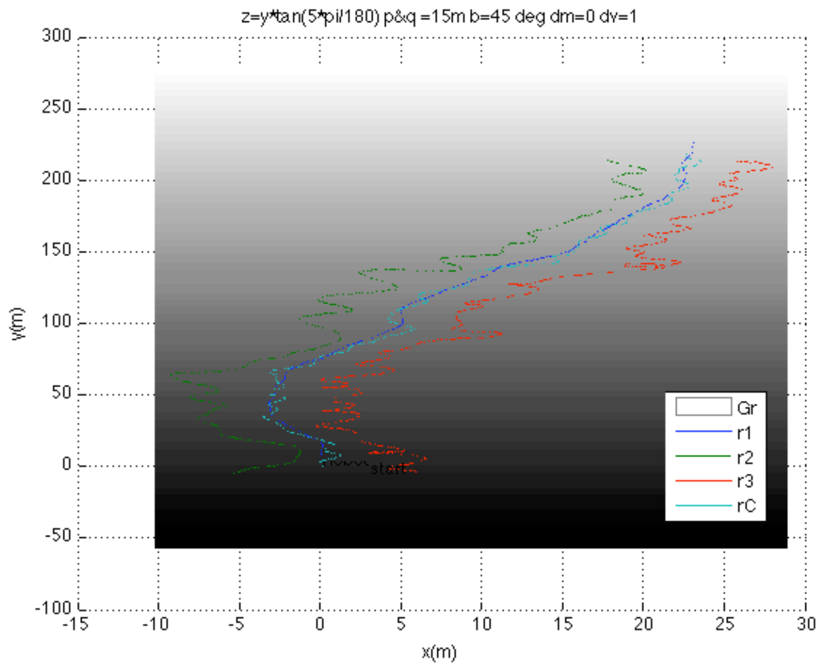

Figure 3.14 Simulation results for Gradient plane 5 degree (P&Q=15 m, β =45 deg)

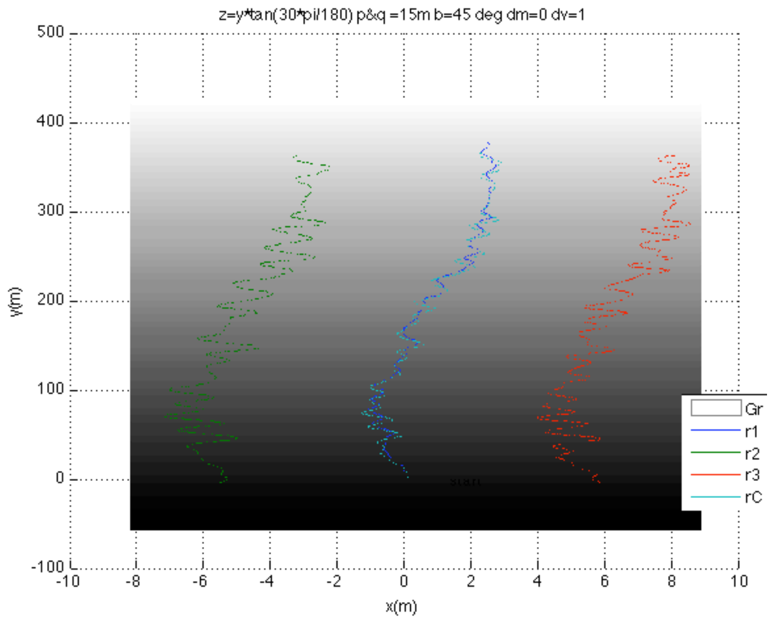

Figure 3.15 Simulation results for Gradient plane 30 degree (P&Q=15 m, β=45 deg)

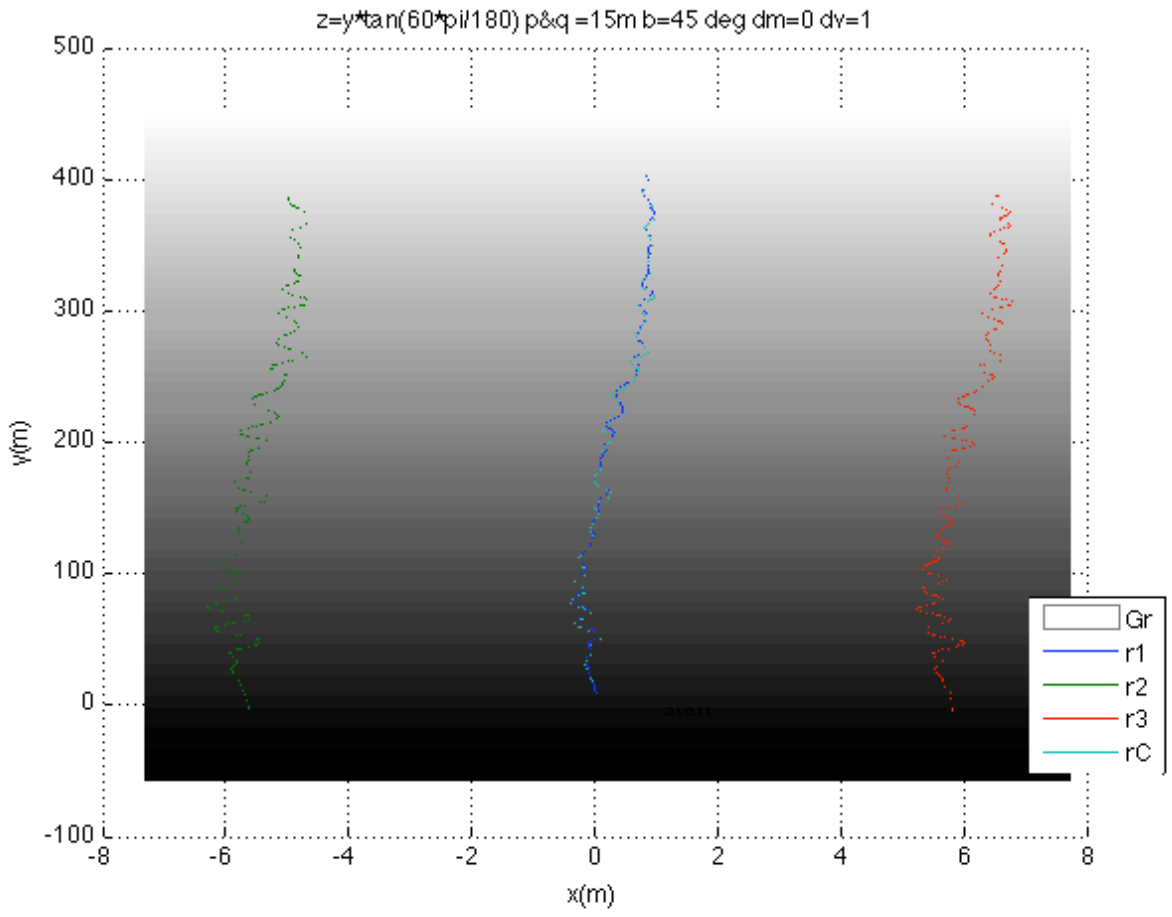

Figure 3.16 Simulation results for Gradient plane 60 degree (P&Q=15 m, β=45 deg)

Additional simulation experiments were performed with the value of P and Q, and the value of β held constant. Only the values for the gradient slope were varied. P and Q were set to 15 meters, and β was set to 45 degrees. The gradient slope was varied from 15 degrees to 75 degrees, with 5-degree increments. A total of 13 simulation experiments were conducted. As the gradient slope increased, the RMS cluster heading error linearly decreased. These results are shown in the third graph of figure 3.17. At a gradient field of 90 degrees there is an expected singularity because the gradient field becomes undefined. Therefore we conclude that for small gradient angles or gradient angles approaching 90 degrees are more sensitive to the noise function.

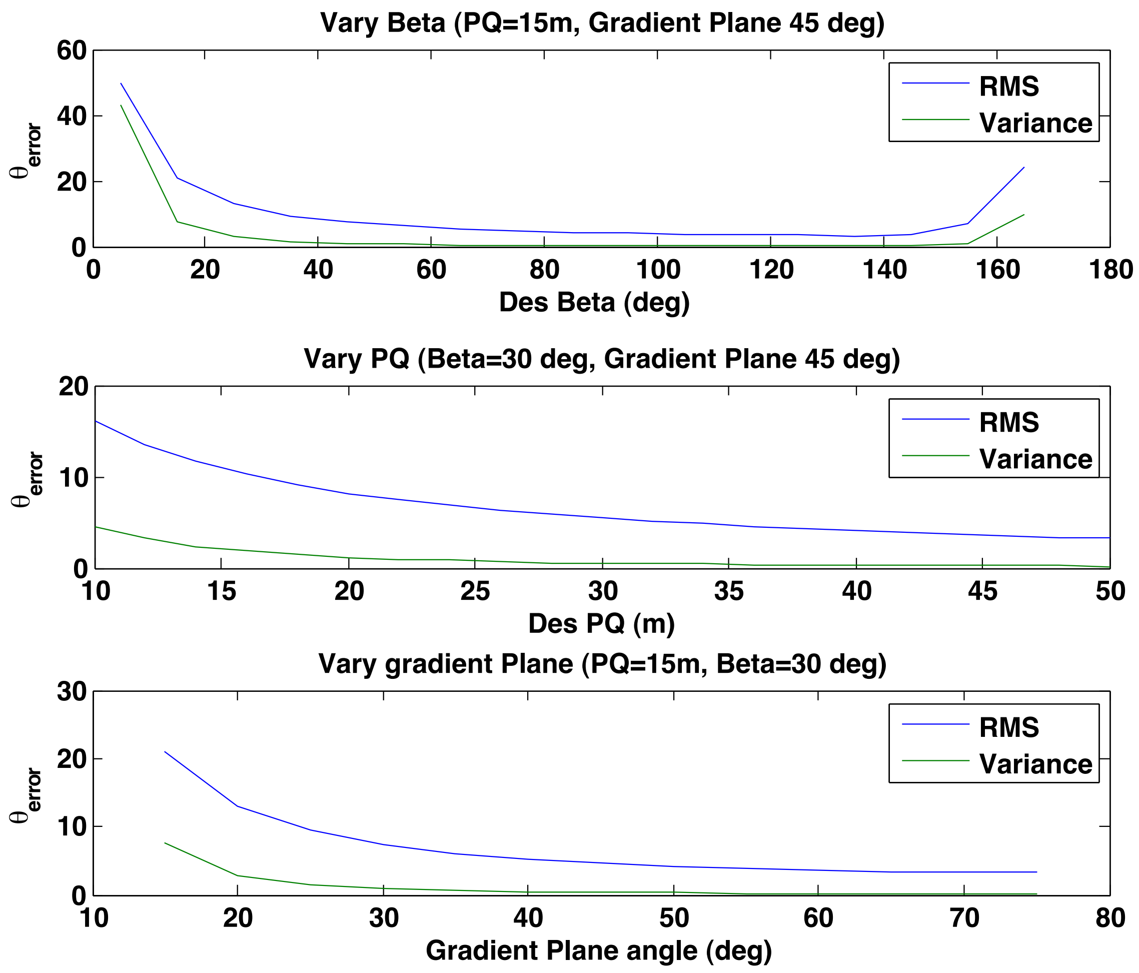

Figure 3.17 Graph of three simulation experiments, varying β, varying P and Q, and varying the gradient plane

3.4.4 Discussion of Simulation Results

We performed a series of simulation experiments to understand the effect of the RMS cluster heading error on variances in the value of the state variables. The first simulation experiments only varied the value of β. Moving the value β away from the singularities of 0 degrees and 180 degrees increases the stability of the system. The second set of simulation experiments only varied the values of P and Q. The RMS error decreased with a larger spatial distances between the robots. Moreover, as the height difference between the three robots increased, the RMS heading error also decreased. The third set of simulation experiments only varied the gradient slope angle. The RMS error decreased for slopes away from slopes of 0 degrees or 90 degrees. Clusters with β angles away from singularities, and with larger P and Q value have more accurate heading estimations. These sets of experiments give guidelines for configuring the cluster shape for the most optimal performance based on field conditions.

3.5 Tests with Gaussian gradients

The previous sections covered climbing with constant gradient slopes to demonstrate functionality and characterize performance of the gradient climbing controller in the simplest environment as possible. The actual gradients in the field are not constant. In this section the behavior of the gradient climbing controller is characterized with Gaussian gradients fields.

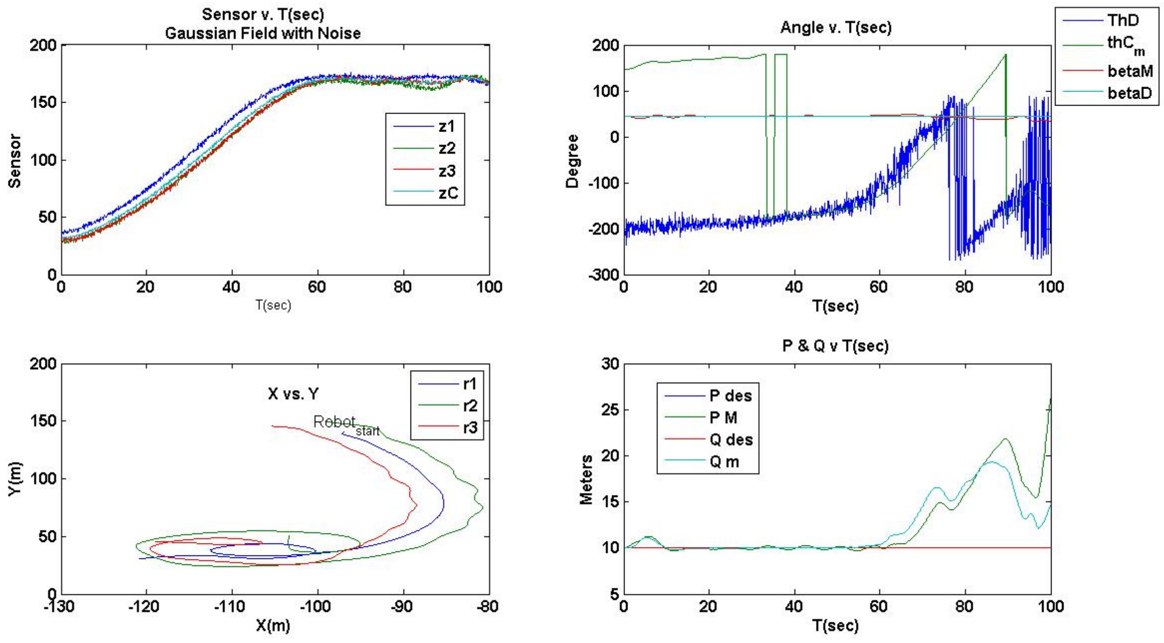

Figure 3.18 Data plot of a Gaussian field with noise

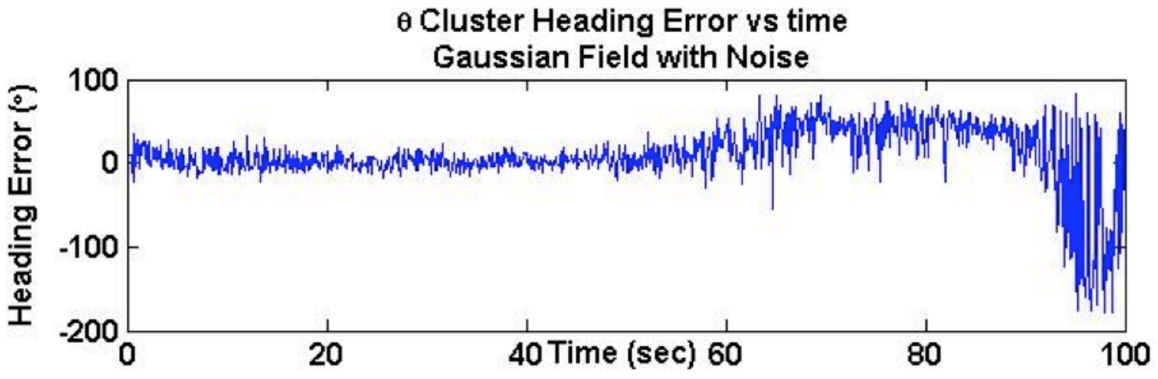

Figure 3.19 Plot of cluster heading error vs. time of a Gaussian field with noise

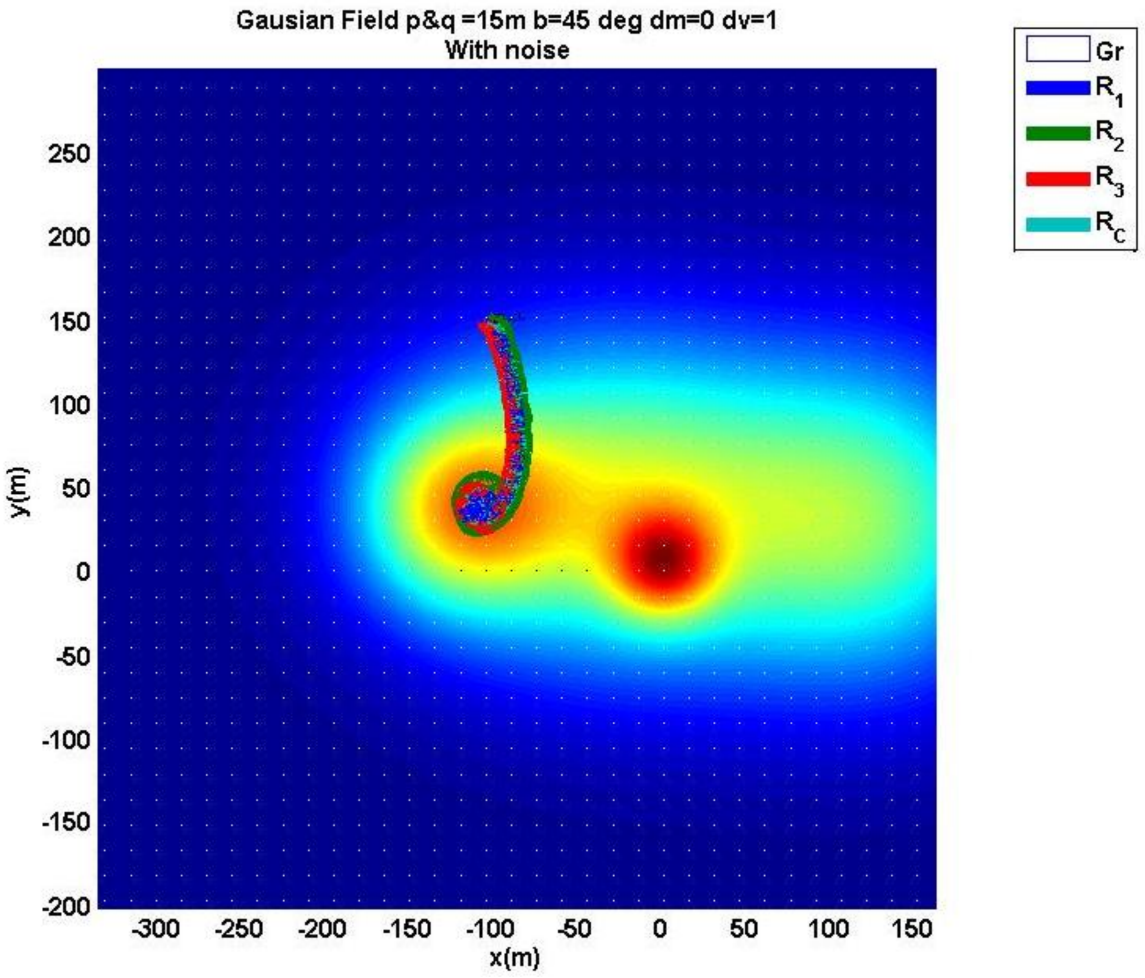

Figure 3.20 3D Plot of a Gaussian field with noise

A simulation experiment was performed using a field of Gaussian gradients. The simulations were performed without modification to the gradient computation. The cluster was started from a position of low concentration away from the top of the field, allowing the cluster to climb the gradient. At the top of the gradient the cluster attempted to hover about the maximum point of the gradient. These tests were run with constant desired values of P and Q, and β angles to maintain a shape of $[P, Q, \beta] = [10\,m, 10\,m, 45°]$. Figures 3.18 to 3.20 show the results of a simulation of climbing a Gaussian field with the noise function. In this field there were two peaks of concentration. Each peak was a sum of multiple Gaussians. The cluster succeeded in finding the local maximal peak. After 60 seconds of simulation time, the cluster arrived to the top of the local Gaussian peak. At the top of the peak the instantaneous vector is constantly rotating. The cluster center rotated about the top of the peak to stay on the peak. As expected the desired heading of the cluster was constantly changing in order to keep the cluster on the peak; therefore the cluster heading error became large at the top of the local Gaussian.

Chapter 4 Field Robotic Experimental Testbed

4.1 System Overview

The robotic tests were conducted on three different mobile robotic platforms. Each system is comprised of three independently controlled robots with a differential drive. The Pioneer robots are an outdoor all terrain land system in the field. The kayak system is a marine field setup. The BoeBot system is an indoor small-scale test bed. The BoeBot system uses light sensors to climb a light gradient. These test beds were used in order to verify the technique using realistic robots in field conditions. The BoeBot system is the first attempt of a gradient climbing robotic test bed. This chapter covers the kayak and the Pioneer test beds. The BoeBot results and description are given in the Appendix A because the data from this test bed is limited. In all experimental test bed systems, we used Simulink in Matlab to run the gradient cluster controller.

The software architecture is similar on the Pioneer and kayak test bed systems. The control law is calculated in Simulink. A series of software layers take the telemetry data from the robot and feed it to Simulink. The software layers also take the output motor velocities from Simulink and relay the data to the individual ASV robots. RBNB Data Turbine and CASADE are the software layers that transfer the realtime telemetry and commands between the Simulink controller and the communication system to the robots. RBNB Data Turbine is a channel-based architecture to relay real-time data between distributed applications. Simulink continually polls RBNB Data Turbine for the latest telemetry data and sends the motor command to RBNB Data Turbine. CASADE is software, which accepts data from multi-sources, developed by the Robotics Systems Laboratory in Santa Clara University. It allows users to have a general framework for system specific plug-ins to control the flow of telemetry. Serial Port Turbine transfers serial data to and from Data Turbine.

Each individual ASV robot has its own data structure in Data Turbine. Each robot receives telemetry data from on board sensors and then sends the data to the off board controller. Figure 4.1 diagrams the software flow. The microprocessor on board is a set of 2-basicX processors to send sensor and receive velocity data. The data is sent over a 900 MHz wireless modem (Ricochet). The Ricochet uses a P2P (peer to peer) architecture. The controller computer has one Ricochet modem to receive and send data from the robots in the cluster.

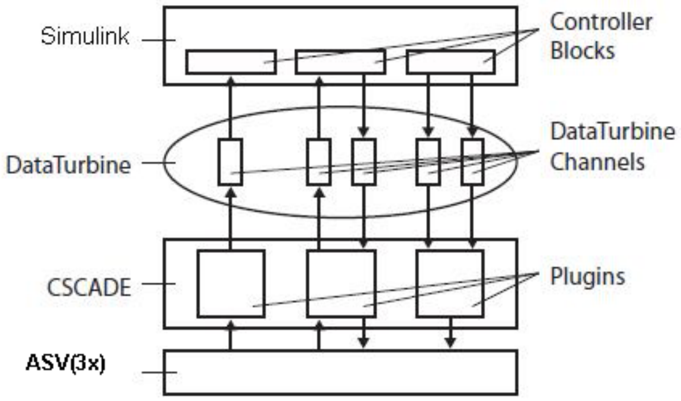

Figure 4.1 Software architecture diagram

4.2 Kayak Robotic Test-Bed Platform and Results

The ASV testbed was chosen to demonstrate the gradient climbing control technique in field marine applications. The robotic kayaks have significant physical real-world vehicle dynamics and are significantly affected by environmental forces such as wind and current. The controller may encounter these conditions in a field environment. The testbed is pictured in figure 4.2.

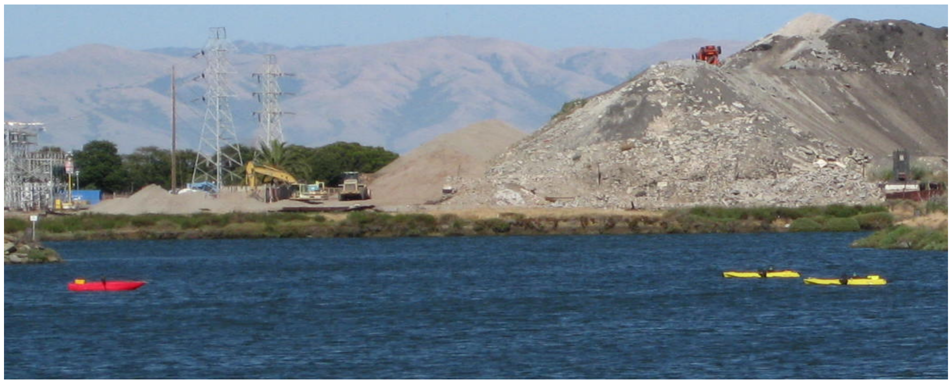

Figure 4.2 Outdoor kayak setup in the Redwood City Marina

4.2.1 Kayak System Overview

The kayaks were selected for their wide ultra-stable flat hull and low cost. Each kayak has two trolling motors for left and right propulsion. This allows for differential drive. The sensor package is the same as the Pioneer setup with a GPS and compass. The wireless communication is also the same as the Pioneer.

Two BasicX microprocessors make up the onboard computing. The BasicX-24p was programmed with BASIC with multi-thread capability. The number of I/O ports limits the BasicX. Figure 4.3 and 4.4 shows the architecture and data flow. The expanded bus architecture can be found in previous work [20].

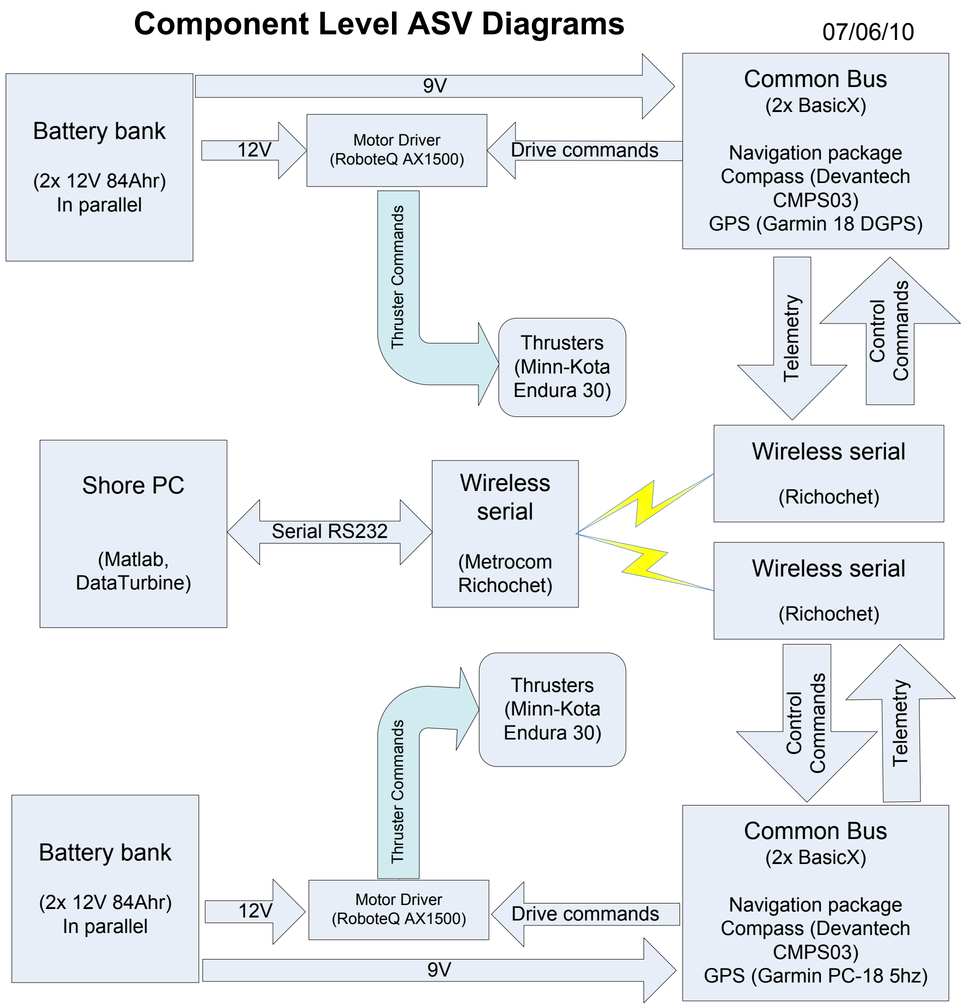

Figure 4.3 Architecture diagram of a single ASV robot

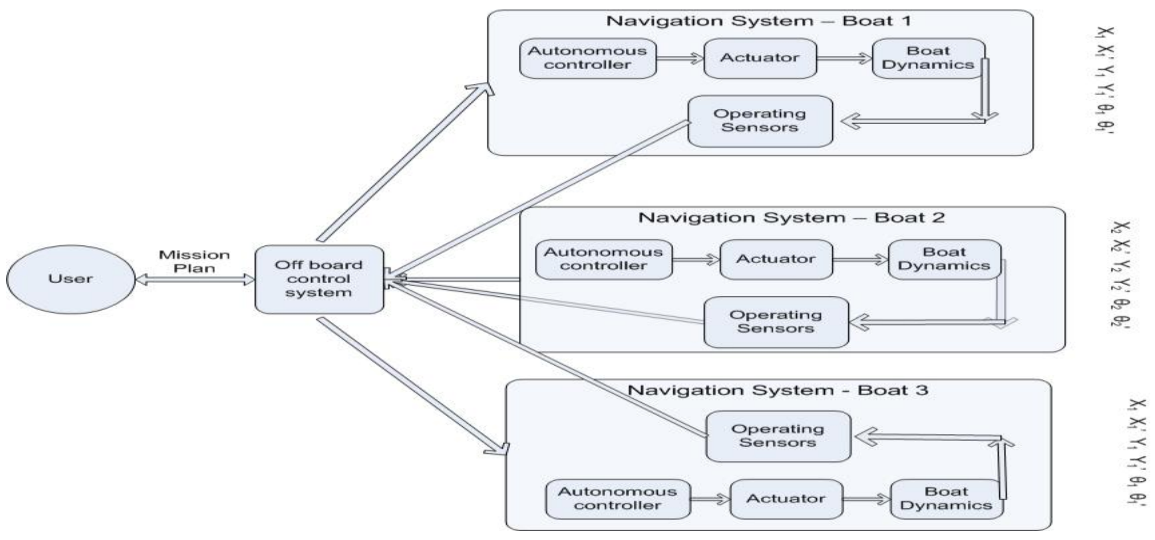

Figure 4.4 Functional block diagram of the robot cluster

4.2.2 Kayak Testbed Results

The experimental test in the field was done with a simulated gradient field. Figure 4.2 shows the experimental setup in the Redwood City Marina (CA). The first test demonstrates the functionality of the testbed in the field. For the first test, a three-dimensional virtual gradient field was created in Matlab as seen in figure 4.5. The robot cluster was started on the virtual gradient at a cluster heading away from the gradient. The cluster then proceeded to climb the gradient by turning up the gradient.

The equation of the gradient was $z = -x - y$, resulting in a slope that points with a bearing of $-135°$. For the first test the cluster was commanded to climb up the gradient, while maintaining a shape of $[P, Q, \beta] = [20\,m, 20\,m, 60°]$. The cluster was started at $[P, Q, \beta] = [20.2\,m, 23.2\,m, 52°]$, with a cluster heading of $38.9°$, which is 173° away from the gradient. The starting cluster center sensor value was -154 units, and the ending cluster sensor value was -17.8 units as depicted in figure 4.6. After the cluster oriented up the gradient, the RMS steady state cluster heading error was 10.6° from data in Figure 4.7. In this test, kayak 3 had issues with non-linear compass data, therefore the robot could not hold a straight line. In this simple case the cluster climbed the gradient.

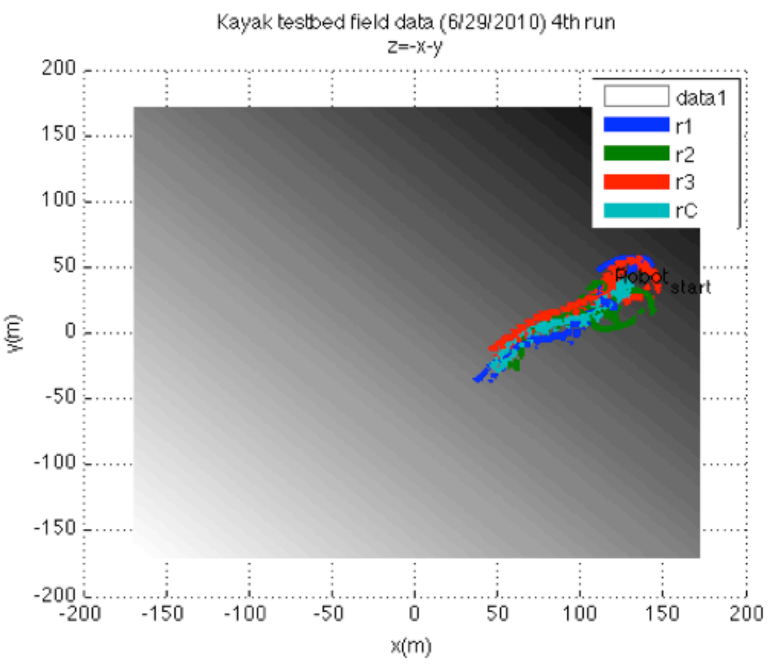

Figure 4.5 3D plot of kayak gradient climbing at Redwood City, CA $(z = -x - y)$

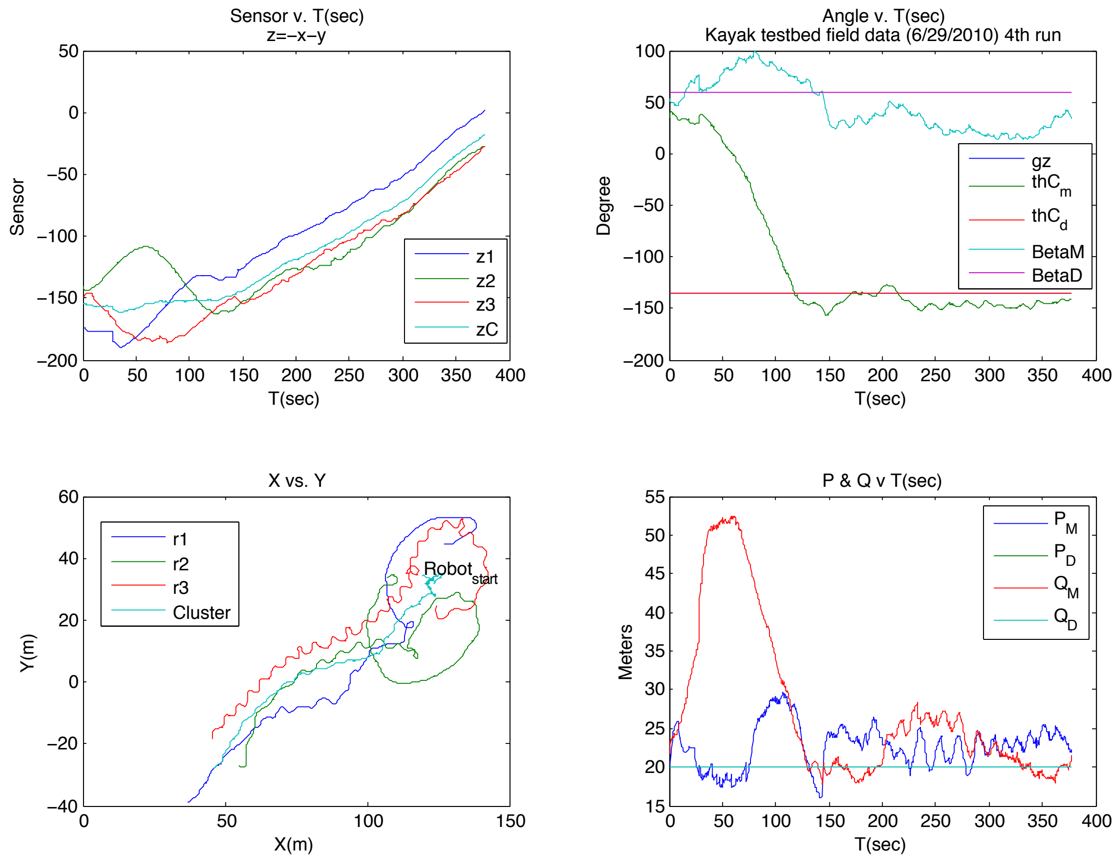

Figure 4.6 Data of kayak gradient climbing at Redwood City, CA $(z = -x - y)$

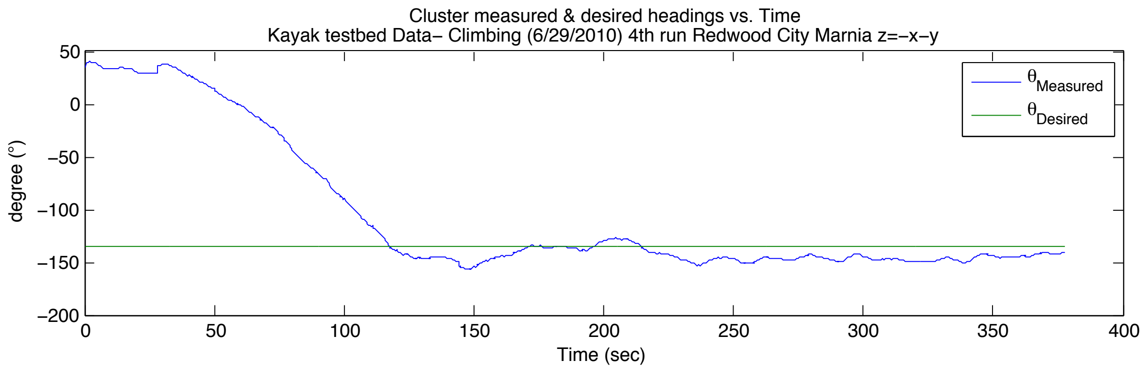

Figure 4.7 Cluster $\theta_{Measured}$ & $\theta_{Desired}$ of kayak field test with a RMS steady state cluster heading error of $10.6°$. $(z = -x - y)$

Figures 4.8, 4.9 and 4.10 show a kayak cluster run in the Redwood City Marina with an inverted parabola as the gradient. Gradients in field will have a Gaussian shape. The kayak cluster test was setup to be turned away from the gradient at a low concentration point. This allows the cluster to demonstrate the ability to find and turn up the gradient. The cluster detected the gradient and turned towards the local maximum.

In the field test described below, the cluster was commanded to climb a Gaussian peak. The Gaussian field equation was $z = -\frac{x^2+y^2}{100} + 100$. The peak value of this Gaussian field is 100 units. The Gaussian peak is centered around the origin of the global frame. This gradient field was chosen to duplicate a point source that could potentially be found in a natural environment. For the first test the cluster was commanded to climb up the gradient, while maintaining a shape of $[P, Q, \beta] = [20\,m, 20\,m, 60°]$. The cluster was started at $[P, Q, \beta] = [21.2\,m, 27.4\,m, 65.5°]$, with a cluster heading of 75.5°. The starting cluster sensor value was 12 units, and the ending cluster sensor value was 92.8 units. After the cluster oriented up the gradient, the RMS steady state cluster heading error was 9.52° as calculated from data in figure 4.9. After 202.5 seconds in the run the cluster center reached the maximum sensor value of 98.7 units. After passing the peak of the Gaussian, the cluster θ_{Desire} value changed 180° to turn back to the peak. This caused the cluster heading error to sharply increase. After passing the peak, the cluster turned back towards the peak to minimize the cluster heading error. In this test, kayak 3 had issues with non-linear compass data, therefore the robot could not hold a straight line. Since, the GPS has an error of 3 meters, P and Q were kept at no less then 12 meters.

Despite the poor compass data, the kayak cluster was able to climb to the peak and hover about it.

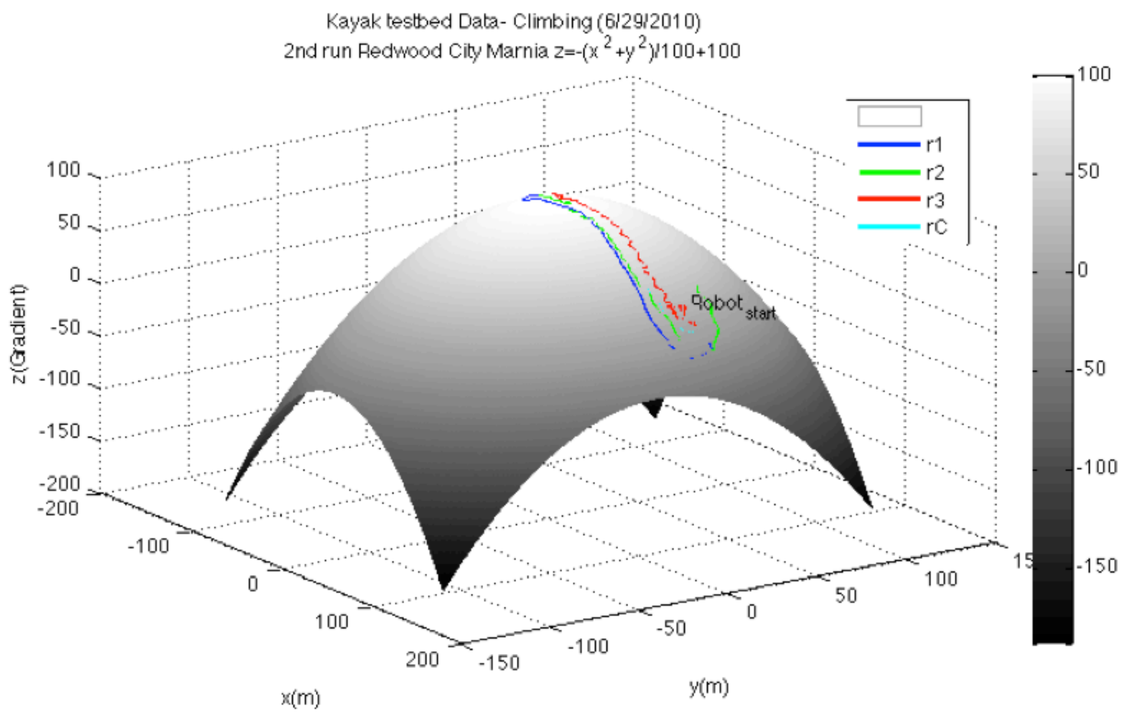

Figure 4.8 3D plot of kayak field run of $z = -\frac{x^2+y^2}{100} + 100$

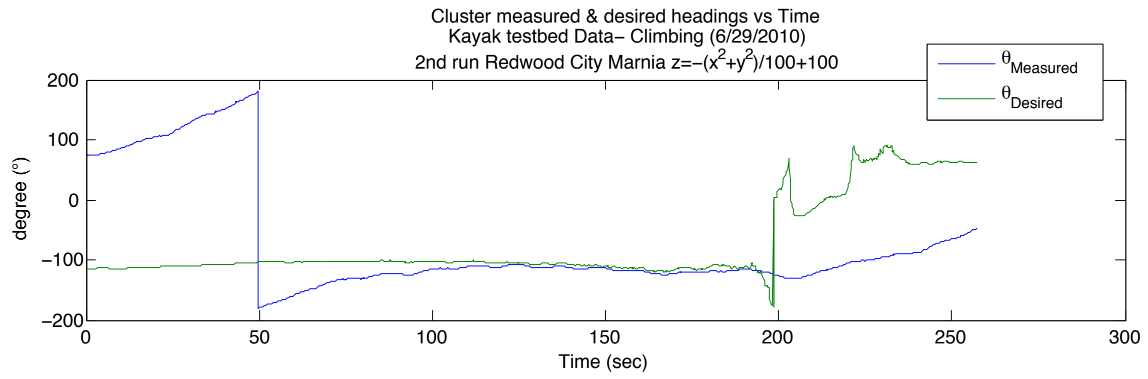

Figure 4.9 Cluster $\theta_{Measured}$ & $\theta_{Desired}$ of kayak field test $z = -\frac{x^2+y^2}{100} + 100$ with a RMS steady state cluster heading error of $9.52°$.

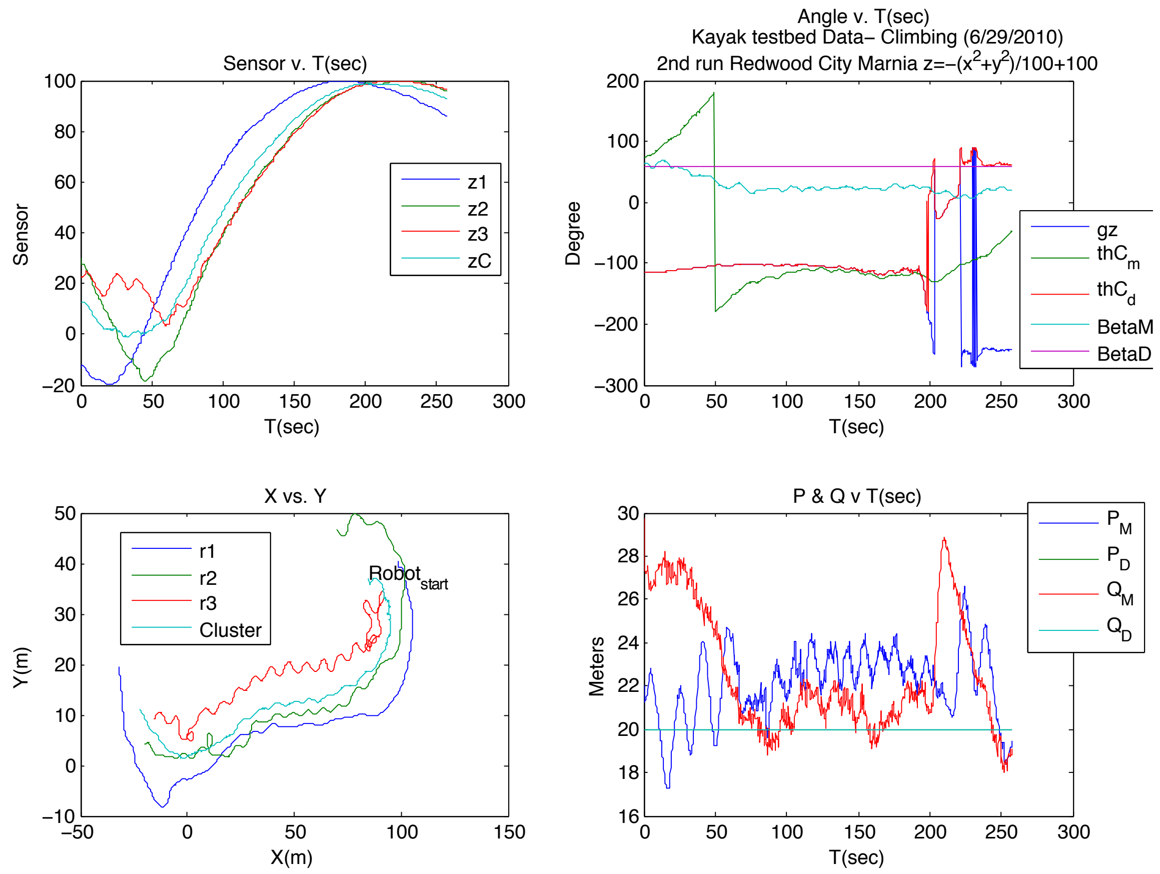

Figure 4.10 Data of kayak field run of $z = -\frac{x^2+y^2}{100} + 100$

4.3 Pioneer test-bed Platform and Results

To have a more complete understanding of the gradient cluster control system, the Pioneer rovers shown in Figure 4.11 were chosen to demonstrate the gradient climbing technique in real robotic systems. The rovers can be modeled as a first order system with little disturbance forces from the outside environment, which makes it easier to apply and study the control technique.

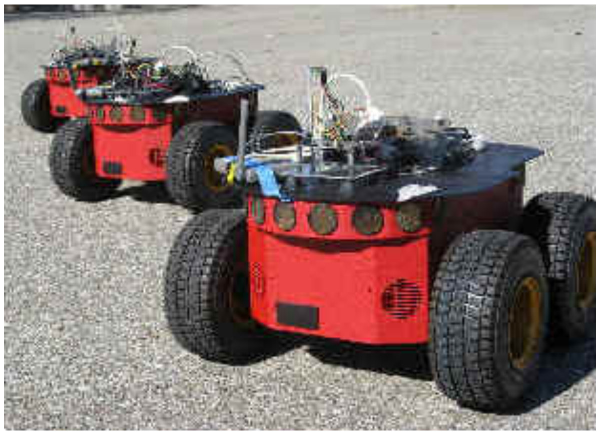

Figure 4.11 Pioneer robotic test bed platform

4.3.1 Pioneer Test Bed

The Pioneer test bed is based on the mobile robots platform Pioneer 3-AT, an all terrain robot with a four-wheel differential drive as seen in Figure 4.11. It is capable of linear translation speeds up to 0.8m/s and rotational speeds 300°/s. It sends telemetry data and receives commands using a 900 MHz radio link. The communication link preserves data integrity, but it does not guarantee packet delivery. Students at SCU have developed custom sensors and communication subsystems for this robot. The sensors include a Garmin 18-5hz differential GPS unit, a digital Devantech compass and a Ricochet 128Kbits/s radio modem. The BasicX microcontrollers control the subsystem, which is linked through RS-232 interfaces. The system is capable of outputting telemetry at a 5Hz rate with a range of approximately 2 miles in clear and ideal conditions. The system architecture of the Pioneer system is similar to the kayak system.

4.3.2 Pioneer Results

The experimental test in the field was done with a simulated gradient field. The experimental setup was performed at Santa Clara University's baseball field. The first test demonstrated the functionality of the test bed in the field. For the first test, a three-dimensional virtual gradient field was created in Matlab, seen in figure 4.12. The robot cluster was started on the virtual gradient at a cluster heading away from the gradient. The cluster then proceeded to climb the gradient by turning up the gradient.

The equation of the gradient was $z = (\square + 2y)$. For the first test the cluster was commanded to climb up the gradient, while maintaining a shape of $[P, Q, \beta] = [15\ m, 15\ m, 60°]$. The cluster was started at $[P, Q, \beta] = [13.2\ m, 10.9\ m, 63.4°]$, with a cluster heading of $-155°$. The starting cluster center sensor value was -38.1 units, and the ending cluster sensor value was 64.7 units as depicted in Figure 4.13. After the cluster oriented up the gradient, the RMS steady state cluster heading error was $1.62°$ as calculated using data in figure 4.14. In this simple case the robotic cluster climbed the gradient. As a comparison to kayak cluster, the Pioneer cluster was able to have a lower RMS steady state cluster heading error. The error of the Pioneer system was $1.62°$ and that of the kayak was $10.6°$ due to less dynamic forces acting upon the Pioneer cluster.

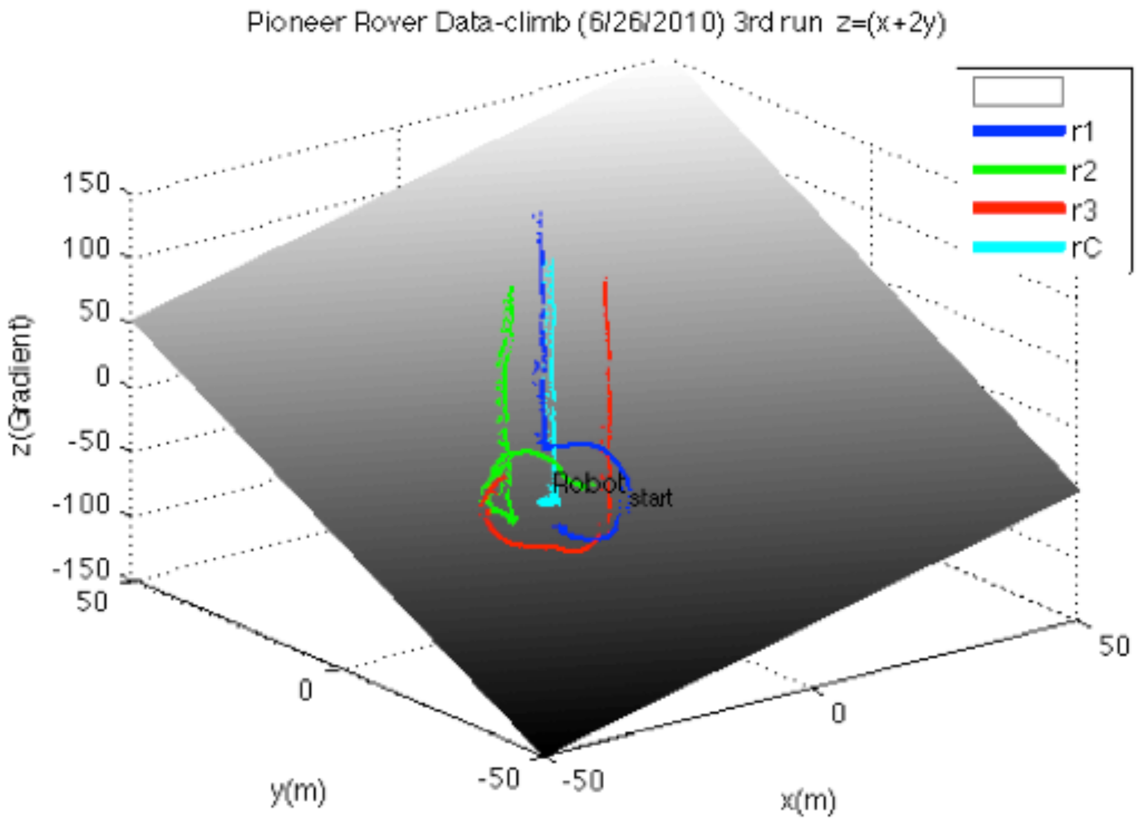

Figure 4.12 3D Pioneer rover data climb $z = (x + 2y)$.

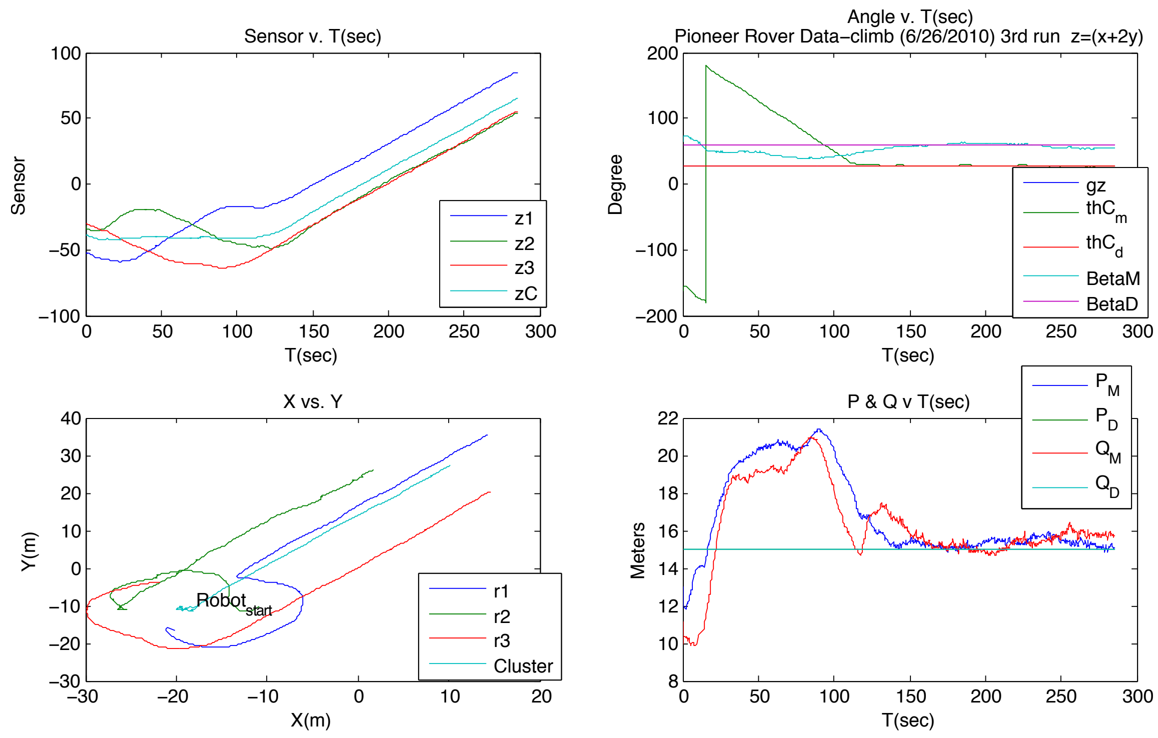

Figure 4.13 Pioneer rover data climb $z = (x + 2y)$.

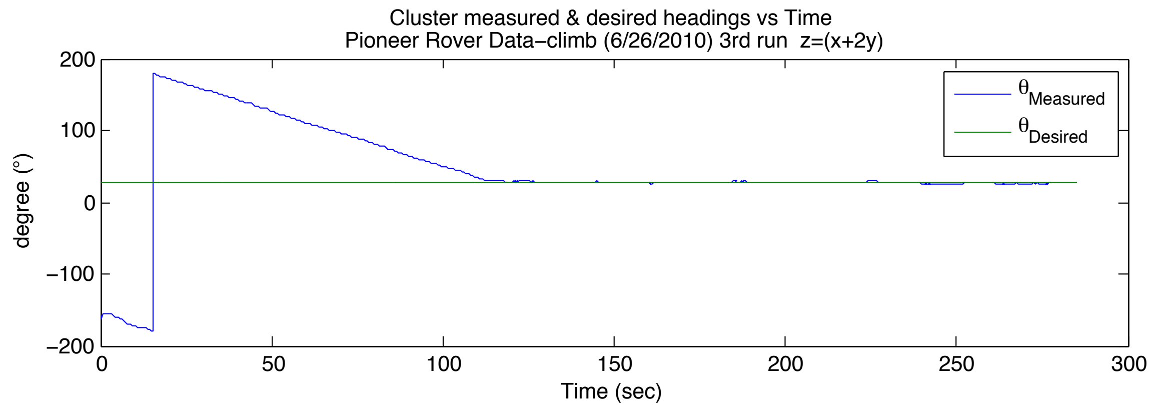

Figure 4.14 Pioneer rover data climb $z = (x + 2y)$ with a RMS steady state cluster heading error of $1.62°$

A secound field test was also run at the baseball field at Santa Clara University. The cluster was commanded to climb a Gaussian peak as depicted in figures 4.15, 4.16, and 4.17. The Gaussian field equation was $z = -\frac{x^2+y^2}{100} + 100$. The peak value of this Gaussian field is 100 units. The Gaussian is centered around the origin of the global frame. This gradient field was chosen to replicate a point source that could potentially be found in a natural environment. For the first test the cluster was commanded to climb up the gradient, while maintaining a shape of $[P, Q, \beta] = [10\,m, 10\,m, 60°]$. The cluster was started at $[P, Q, \beta] = [4.74\,m, 9.25\,m, 135°]$, with a cluster heading of $155°$. The starting cluster sensor value was 74.44 units, and the ending cluster sensor value was 99.1 units. After the cluster oriented up the gradient, the RMS steady state cluster heading error was 7.13° as calculated from data in figure 4.17. After 247.5 seconds in the run the cluster center reached the maximum sensor value of 99.6 units. After passing the peak of the Gaussian, the cluster θ_{Desire} value turned 180° to turn back to the peak. This caused the cluster heading error to increase. As the cluster turned back towards the peak the cluster heading error decreased. The cluster was able to climb to the peak and hover about it. As comparison the Pioneer cluster was able to have a lower RMS steady state cluster heading error of 7.13° than the kayak cluster error of 9.52° due to less dynamic forces acting upon the Pioneer cluster.

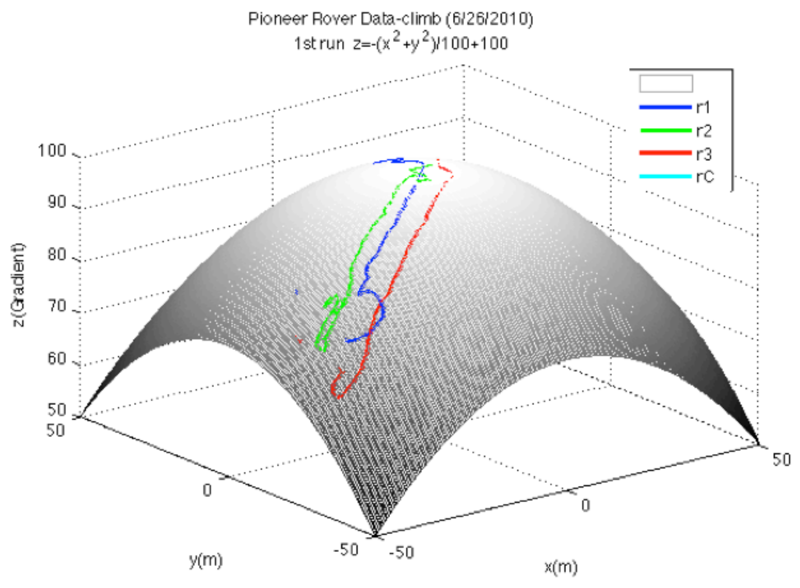

Figure 4.15 3D plot of a Pioneer field run $z = -\frac{x^2+y^2}{100} + 100$

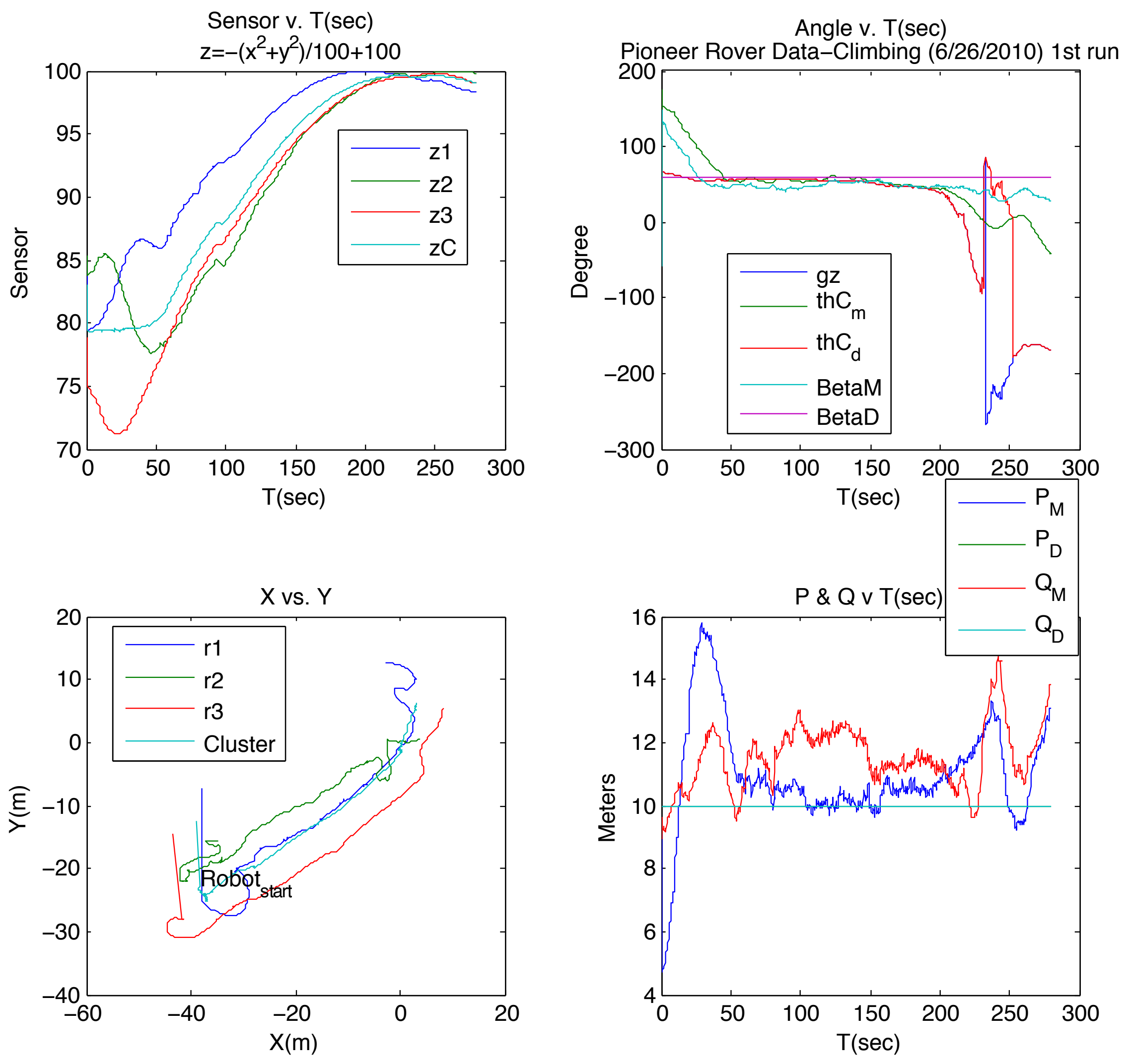

Figure 4.16 Data plot of a Pioneer field run of $z = -\frac{x^2+y^2}{100} + 100$

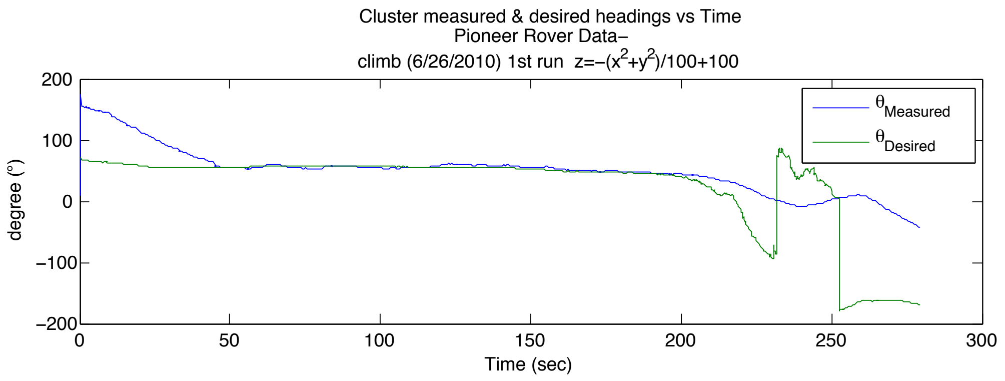

Figure 4.17 Plot of $\theta_{C\ Measure}$ & $\theta_{C\ Desired}$ vs. time, with a steady state RMS cluster heading error of $7.13°$.

4.4 Summary of the Robotic Test Platforms

The Pioneer test bed is a land based system, while the kayak system is a marine system capable in both fresh and salt water. The Pioneer test bed demonstrates the gradient controller working in outdoor land environments. The dynamics response of the Pioneer robot is linear with no dynamic forces. The kayaks demonstrate the gradient controller working with dynamic forces such as wind and surface currents in the test environment. On both systems the controller enabled the cluster to climb to a Gaussian peak and hover about it. The tracking errors of the two robotic test beds were maintained within acceptable accuracy margins.

Chapter 5 Conclusion

5.1 Summary

The cluster control technique was successfully demonstrated to autonomously climb an unknown gradient field in land and marine environments. Gradient estimation was based on three spatially distributed field samples, each at the location of a robot in the cluster. The estimated gradient direction was used to steer the cluster in order for the cluster to climb up the gradient field to the local maxima. This was demonstrated in a simulation environment as well as with a simulated gradient field while using real robotic systems. These real systems included a three robot cluster of wheeled land rovers as well as a three robot cluster of robotic kayaks. The kayak experiments demonstrated the controller's ability to function given the challenges of significant plant dynamics and external disturbance forces. This research gives initial guidance for designers and field operators to effectively use this technique in the field from the initial characterization of performance based on the sensor noise and cluster configurations. This work suggests the possibility of further expandability.

5.2 Future Work

The work in this thesis of cluster space gradient climbing is promising for future developments. This work can be expanded in at least three different areas, using larger sensor robotic clusters, three dimensional sensor platforms, and dynamic gradient fields.

Expanding the cluster kinematic and the gradient characterization may help to deal with smaller Gaussian gradients and other irregularities. With larger robotic clusters, fault detection is important to drop malfunctioning robots from the cluster. This makes the gradient cluster controller more robust to deal with irregularities of the sensor networks. A promising technique known as "cluster of cluster" control is in development for hierarchical system control. A six-robot cluster has shown promising results. Multiple clusters in the field could map multiple Gaussians at once in a given environment.

Using three dimensional test platforms such as blimps, helicopter or AUVs (autonomous underwater vehicle) allows for the characterization of plumes in the water column or in the atmosphere. Expanding the gradient control to a three dimensional environment would allow for characterization of the plumes in the water. For example in large industrial gas leaks, the gas cloud could be tracked with robotic blimps.

Chemical and temperature gradient fields are also time varying. Expanding the work into dynamic gradient fields will allow tracking time and space varying gradients. Developing an algorithm to optimize the cluster variables for climbing gradient fields will further the studies in spatial relationships of the robots in the gradient field. The dynamic gradient controller may detect the size of the Gaussian, and adjust the cluster variables accordingly. Incorporating other environmental sensors such as wind direction, anemometers and current sensors can also help follow a time and varying gradient.

This work establishes a base line for the gradient climbing cluster space control technique. There are areas for further development in 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. 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," in *Proceedings 2004 Sixteenth Innovative Applications of Artificial Intelligence Conference (IAAI-2004),* 2004, pp. 1022-1023.

[3] J. Bellingham, (2009, Feb. 10) *Autonomous Ocean Sampling Network* [Online]. Available: http://www.mbari.org/aosn/

[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," in *Proceedings of IEEE Autonomous Underwater Vehicles 2004: Workshop on Multiple AUV Operations (AUV04)*, June 2004, Sebasco, ME.

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

[6] Ralf Bachmayer, Naomi Ehrich Leonard, "Experimental Test-Bed for Multi-Vehicle Control," in *Navigation and Communication 12th International Symposium on Unmanned Untethered Submersible Technology*, Durham, NH., 2001.

[7] J. Adler. "Chemotaxis in bacteria," *Science*, vol. 153, no. 3737, pp 708-716, Aug. 1966.

[8] Jongeun Choi, Songhwai Oh and Roberto Horowitz, "Cooperatively Learning Mobile Agents for Gradient Climbing," in *Proceedings of the 46th IEEE Conference on Decision and Control,* New Orleans, LA, Dec. 12-14, 2007.

[9] I. Mas, J. Acain, O. Petrovic, and C. Kitts, "Error characterization in the vicinity of singularities in multi-robot cluster space control," in *2008 IEEE Int. Conf. Robot. Biomimetics*, Bangkok, Thailand, 2008, pp. 1911-1917.

[10] A. T. Hayes, A Martinoli and R M Goodman, "Distributed Odor Source Localization," *IEEE Sensors*, vol. 2, no. 3, pp. 260-271, 2002.

[11] 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.

[12] 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.

[13] R. Vincent, (2003), *AOSN Charter* [Online]. Available: http://www.princeton.edu/~dcsl/aosn/documents/AOSN_Charter.doc

[14] C. Kitts, and I. Mas, "Cluster Space Specification and Control of Mobile Multi-Robot Systems," *IEEE/ASME Transactions on Mechatronics*, vol. 14, no. 2, pp. 207–218, 2009.

[15] P. Connolley, "Design and implementation of a cluster space trajectory controller for multiple holonomic robots," M.S. thesis, Dept. Mech. Eng., Santa Clara Univ., Santa Clara, CA, Jun. 2006.

[16] T. To, "Automated cluster space trajectory control of two non-holonomic robots," M.S. thesis, Dept. Comput. Eng., Santa Clara Univ., Santa Clara, CA, Jun. 2006.

[17] R. Ishizu, "The design, simulation and implementation of multi-robot collaborative control from the cluster perspective," M.S. thesis, Dept. Electr. Eng., Santa Clara Univ., Santa Clara, CA, Dec. 2005.

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

[19] P. Mahacek, I. Mas, O. Petrovic, J. Acain, and C. Kitts, "Cluster space control of a 2-robot autonomous surface vessels system," *Marine Technol. Soc. J.*, vol. 43, no. 1, pp. 13-20, 2009.

[20] P. Mahacek, "Design and Cluster Space Control of Two Autonomous Surface Vessels" M.S. thesis, Dept. Mech. Eng., Santa Clara Univ., Santa Clara, CA. 2009.

[21] I. Mas, S. Li, J. Acain, and C. Kitts, "Entrapment/Escorting and Patrolling Missions in Multi-Robot Cluster Space Control," in *IEEE/RSJ International Conference on Intelligent Robots and Systems*, St. Louis, MO., 2009, pp. 5855-5861.

[22] T. Adamek, "Cluster Space Gradient Contour Tracking for Mobile Multi-robot Systems," Kitts, Adv. M.S.Thesis, Dept. Mech Eng, Santa Clara Univ., Santa Clara, CA, 2010, In Draft

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

[24] M. S. Agnew, P. DalCanto, C. Kitts and S. Li, "Cluster Space Control of Aerial Robots," in *2010 IEEE Conference on Advanced Intelligent Mechatronics*, Montreal Canada, July 2010.

[25] A. E. Magurran, T. J. Pitcher, "Foraging, timidity and shoal size in minnows and goldfish," *Behavioral Ecology and Sociobiology,* vol. 12, no. 2, pp. 147-152, 1983.

[26] J. Fredslund, 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, Oct 2002

[27] 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, Oct 2002

[28] Kumar, "Controlling formations of multiple mobile robots," in *IEEE International Conference on Robotics & Automation Leuven.*, Belgium, May 1998. pp. 2864 – 2869.

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

[30] H. Ishidaa, K. Suetsugua, T. Nakamotoa, T. Moriizumia, "Plume-Tracking Robots: A New Application of Chemical Sensors Faculty of Engineering," Tokyo Institute of Technology, Ookayama, Meguro-ku, Tokyo, Rep. 152, Dec. 2001. Available online

[31] Willis, M. A., E. A. Arbas., "Odor-modulated upwind flight of the sphinx moth, Manduca sexta L. J. Comp." *Physiol.* ser. A vol. 169 pp. 427–440, 1991

[32] Kaissling, *Orientation and Communication in Anthorpods*, Berlin: Birkhäuser, 1997

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

[34] Wei Li, Jay A. Farrell, Shuo Pang, et al. "Moth-Inspired Chemical Plume Tracing on an Autonomous Underwater Vehicle." *IEEE Transactions on Robotics*, vol. 22 no. 2, pp. 292-307, April 2006.

[35] Thomas Lochmatter, Alcherio Martinoli, "Tracking Odor Plumes in a LaminarWind Field with Bio-inspired Algorithms" in *Experimental Robotics The Eleventh International Symposium* Berlin, 2009 pp. 473-482.

Appendix A BoeBot Test Bed

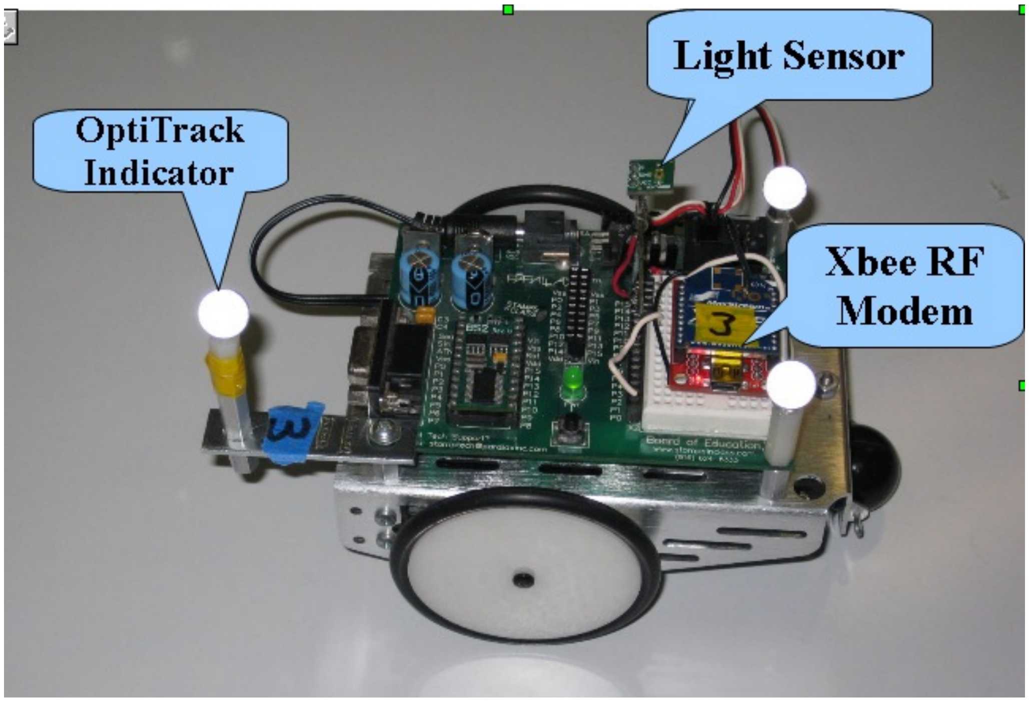

Figure A.1 – BoeBot

BoeBot Description

Three small differential drive robots (BoeBots) are used to detect a point source and climb to a level of higher concentration in the form of a gradient. Each robot has one light sensor that measures the intensity and passes the data through the xBees (2.4 Ghz radio) to Matlab and Simulink for processing. With this information the slope and direction of the gradient can be calculated. Once this information is processed it is sent back to the robots as serial data commanding the motors with a PWM signal. The signal is then used to guide and steer the cluster towards the point of highest concentration.

Mechanical Description

The configuration of the robot is a modular setup up with the BoeBots piggy backing an xBee RF transmitter for serial communication and data transfer. Each robot has the science payload consisting of a TEMT6000 light sensor powered with a 3.3-volt regulator shown in figures A.1 and A.2. The sensor passes the data through the xBee.

The BoeBots have the stamp as the interface. The motor commands are also sent using the xBee.

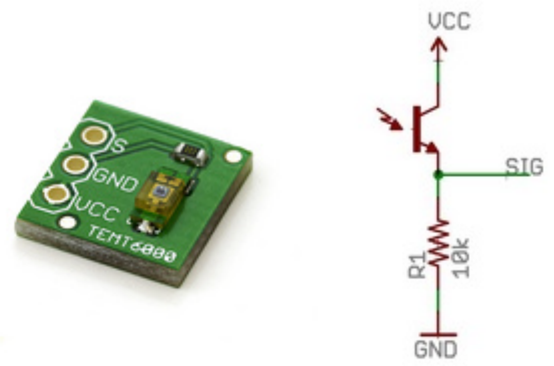

Figure A.2 TEMT6000 Light sensor used in the BoeBot test bed, and circuit diagram.

The position of the robot is given by the Omni track infrared system. Each robot is placed on a 4 feet x 8 feet level sheet in a pre-determined cluster configuration. The point source and light is an overhead lighting projector system with the light sensors facing upwards towards the ceiling collecting the measured light intensity at the given location. The non-negligible dynamics are the dynamics of the two motors, the sense and response time of the sensors and, the processing speed of the stamp. The orders of the dynamics are all first order resulting in a fourth order system.

The OptiTrack system has built-in support for industry standard VRPN and Trackd protocols making integrating real time tracking data with applications relatively straightforward. The included network transport allows for convenient cross-platform communication. The sensor test bed consists of an environment sensor of a single sensor chosen from light, navigation sensor array (Optitrack), a ZigBee 802.15.4 radio modem, and differential drive actuators on three robots. The system architecture is shown in figure A.3.

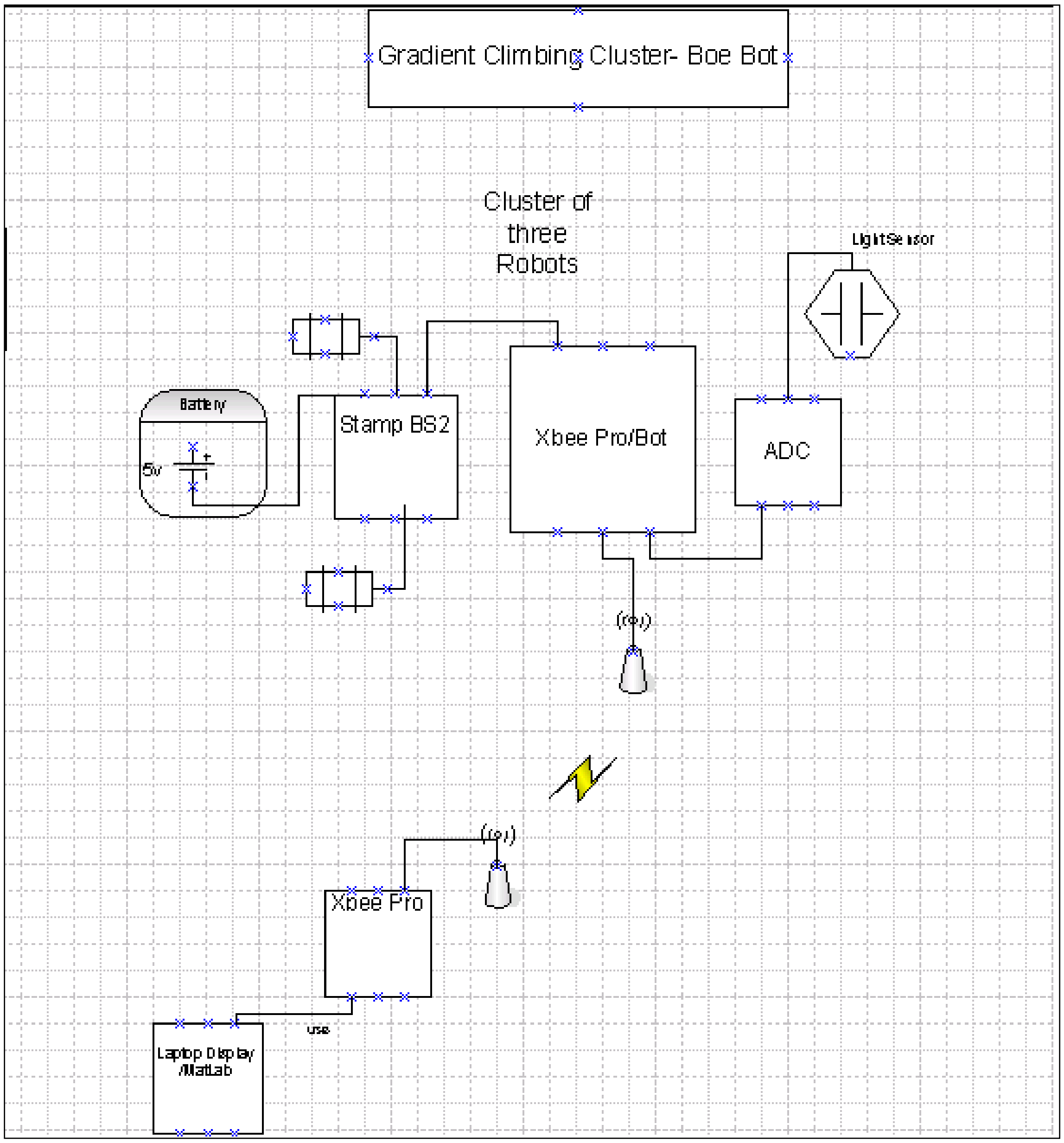

Figure A.3 BoeBot system architecture

A.1.1 BoeBot Test Bed and Results

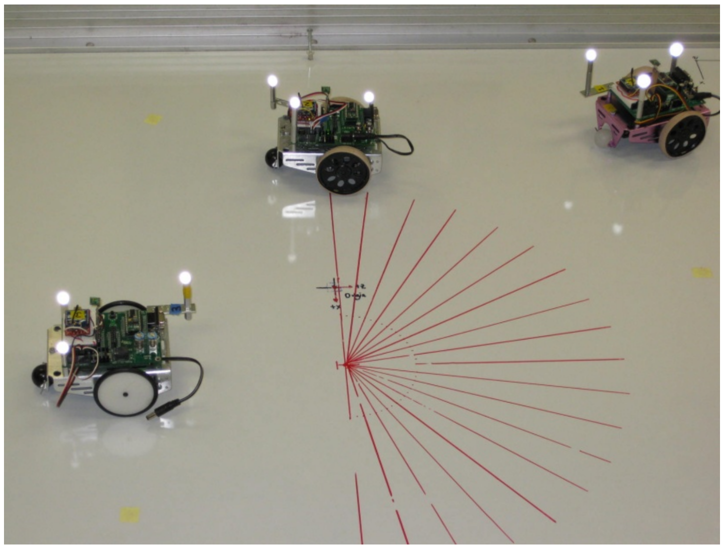
Figure A.4 BoeBot test bed setup

The BoeBot (figure A.4) system is comprised of three Parallax BoeBot robots, having OptiTrack for positioning and a digital projector. Each robot has a light sensor to measure the intensity of the light on the robot. An IR filter on the sensor reduced the noise added by the strobe of the OptiTrack system. A digital projector mounted from the ceiling projected the light gradient onto the robot workspace. Each robot also had 4 IR reflective balls mounted to the robot so the OptiTrack could recognize the robots as rigid bodies. The OptiTrack system was comprised of six cameras setup around the room to track simultaneously all three robots. OptiTrack calculates the heading of the robots. An IR filter was placed on each light sensor to remove any noise from the OptiTrack system. The drive system has two continuous servos, and used differential drive for steering. The motor and light sensor data was transmitted using xBee (2.4 GHz radio) and Data turbine with CASADE into Matlab. The projected gradients were generated using built-in plot functions in Matlab. The robots ran on a table in the room. The advantage of the BoeBot system is that the telemetry data is reliable and accurate to -/+ 1 cm. The disadvantages of the system is that the amount of workspace is limited, and there are areas of dead zones where Optirack cannot give reliable telemetry data.

A.1.2 BoeBot Results

The test results shown in figures A.5 and A.6 show the BoeBots following a light gradient. The cluster center started from the position (0.3345 m, .2542 m) with a gradient concentration of 2.17 units. The gradient projected on the workspace surface was a 2-D parabola with the brightest in the center of the parabola. The parabola was projected with a computer projector mounted from the ceiling. The image was displayed on the table. The parabola image was created using the Gimp's gradient generation toolkits. The center of the parabola was projected around the point (0.6 m, -0.3 m). The cluster ended at (0.7487 m, -.4312 m), 0.2 meters away from the center of the parabola with a gradient concentration of 5.8 units, with a difference of concentration 3.63 units. The cluster detected the gradient, and proceeded to turn left towards the gradient with a forward velocity. Robots 1 and 2 started outside the image, so that the readings of these robots started high. Through experimental testing, the highest sensor value with the projector set to a white image on the workspace was 6.5 units. At the top of the gradient the difference in sensor values is more then the noise.

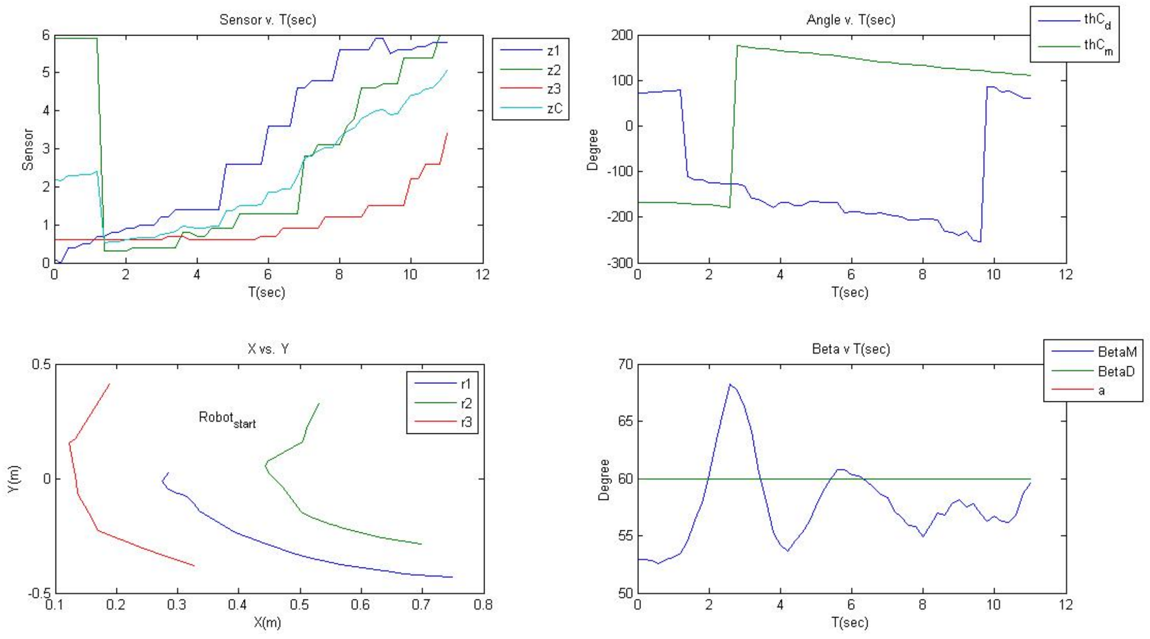

Figure A.5 Data of BoeBot run with a parabola gradient

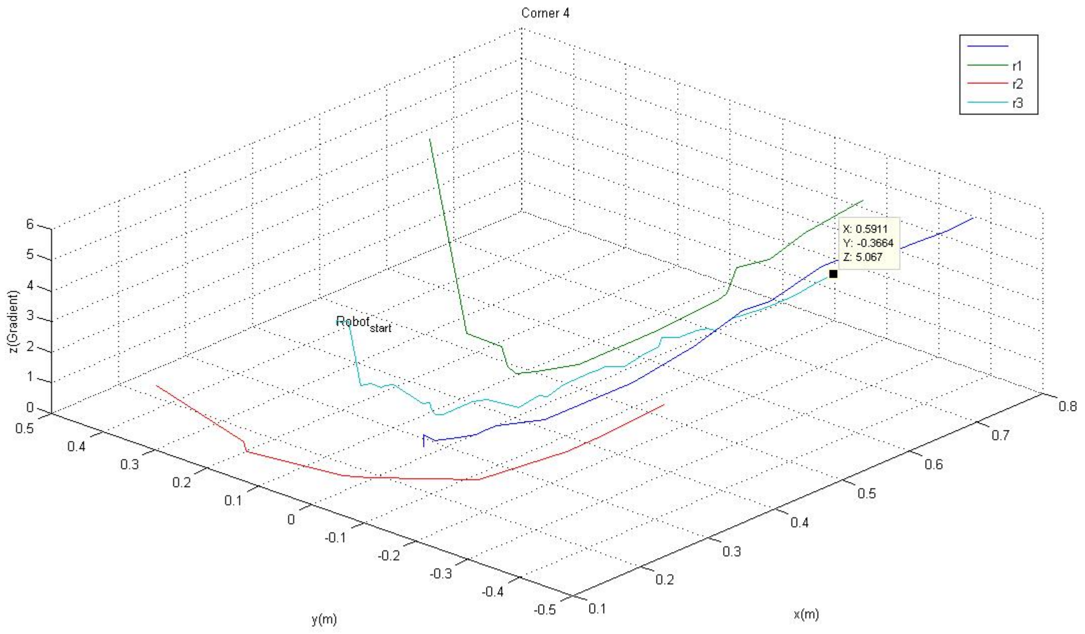

Figure A.6 3D plot of BoeBot run with a parabola gradient

The BoeBot test bed is limited by the infrastructure needed for the OptiTrack system, and is not mobile. The small size of the robots allows for easier debugging. The robots demonstrate the gradient controller working with active sensor data.

Appendix B Simulink and Matlab Code

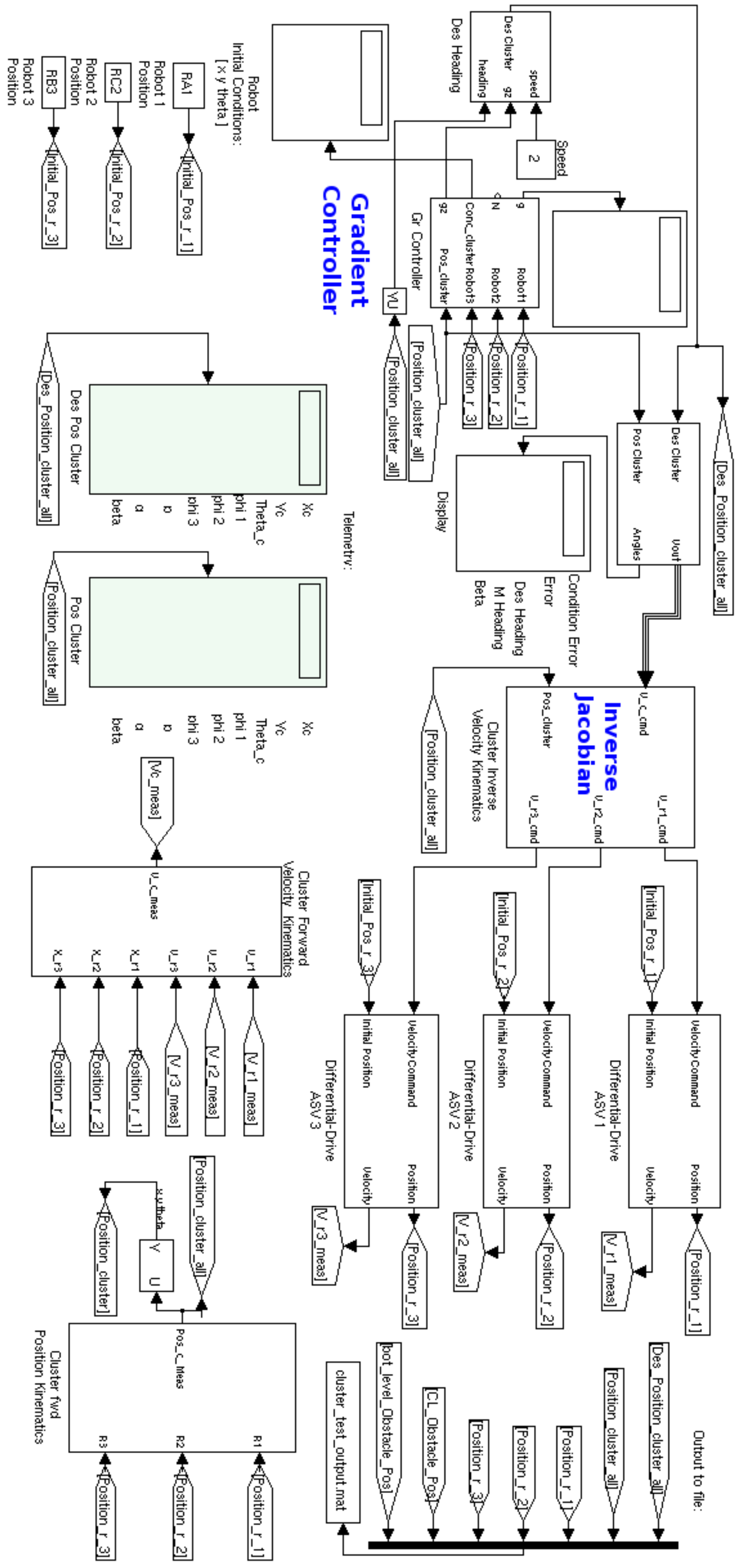

Figure B.1 Simulink program of the gradient cluster controller

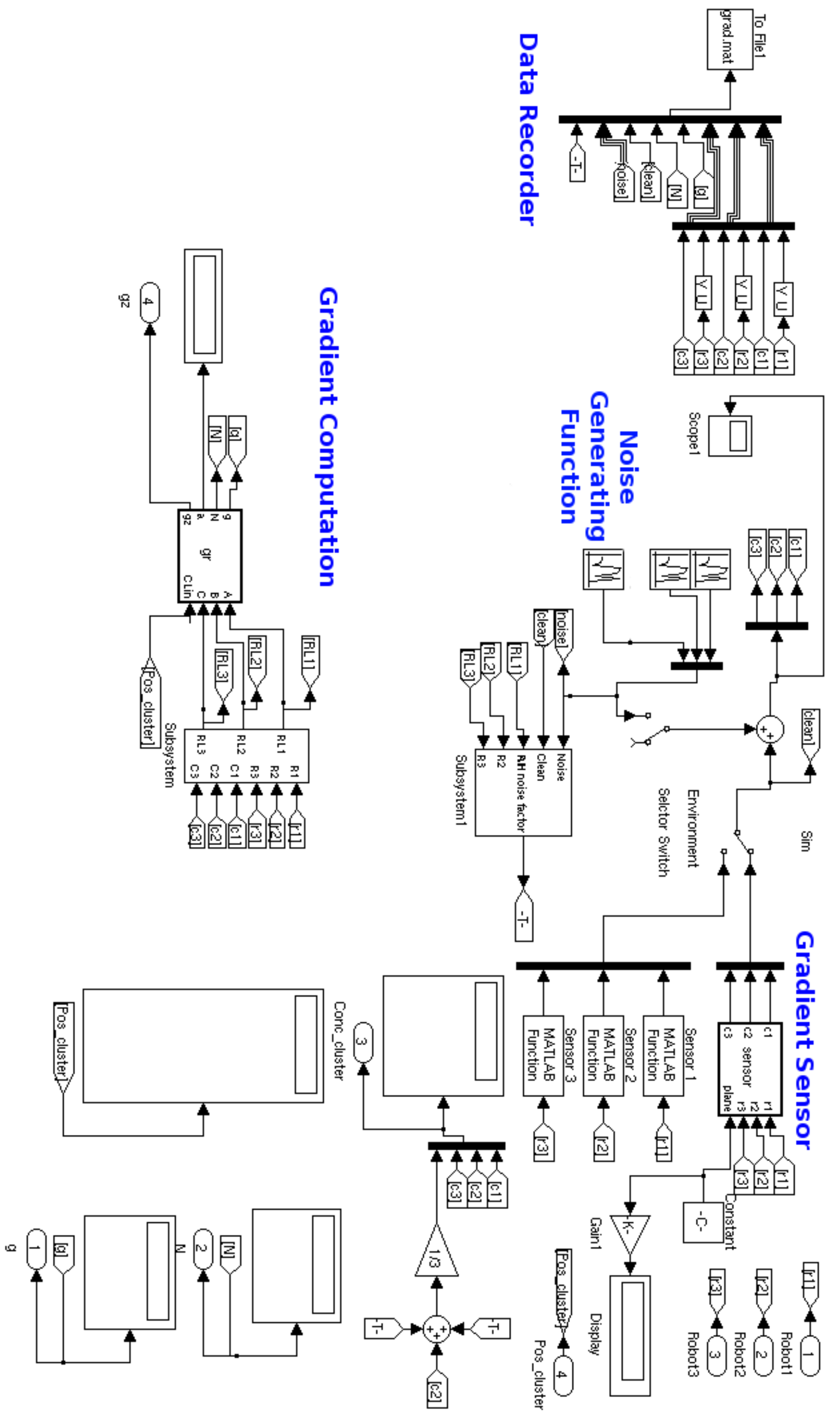

Figure B.2 Simulink program of the gradient climbing controller

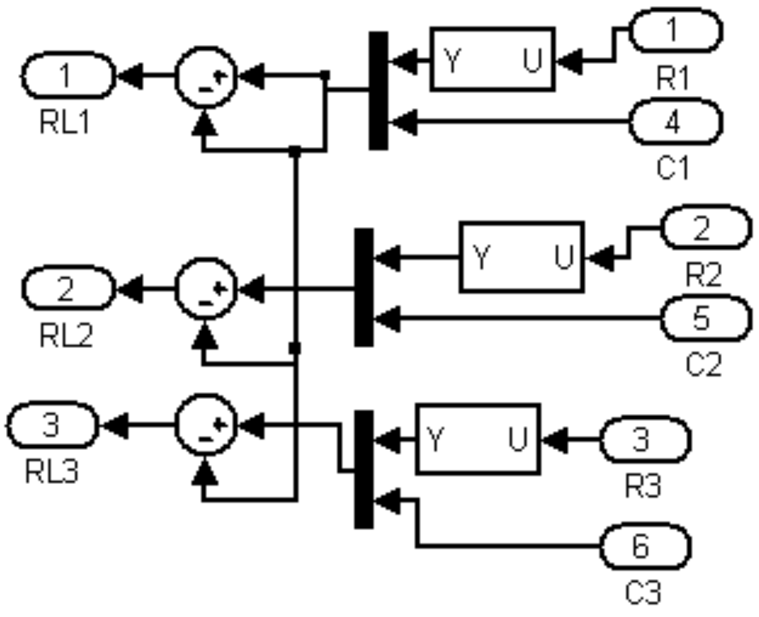

Figure B.3 Transformation for the gradient controller into the local frame

Gradient climbing computation

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


N=cross(B,C)*sign(CLin(9));
a=0;
N=N/sqrt(N(1)^2+N(2)^2+N(3)^2);
gz=atan2(N(2),N(1));
gxy=atan2(N(3),cos(gz)*N(2)+sin(gz)*N(1))-pi/2;
a=gxy*180/pi;

gy=atan2(N(3),N(1));
gx=atan2(N(3),N(2));
gz=atan2(N(2),N(1));

gz=-pi/2-atan2(N(2),N(1));
gz=gz;
g=[gx;gy;gz]*180/pi;
end
```

(gz is the final heading output given to the cluster controller as cluster desired heading)

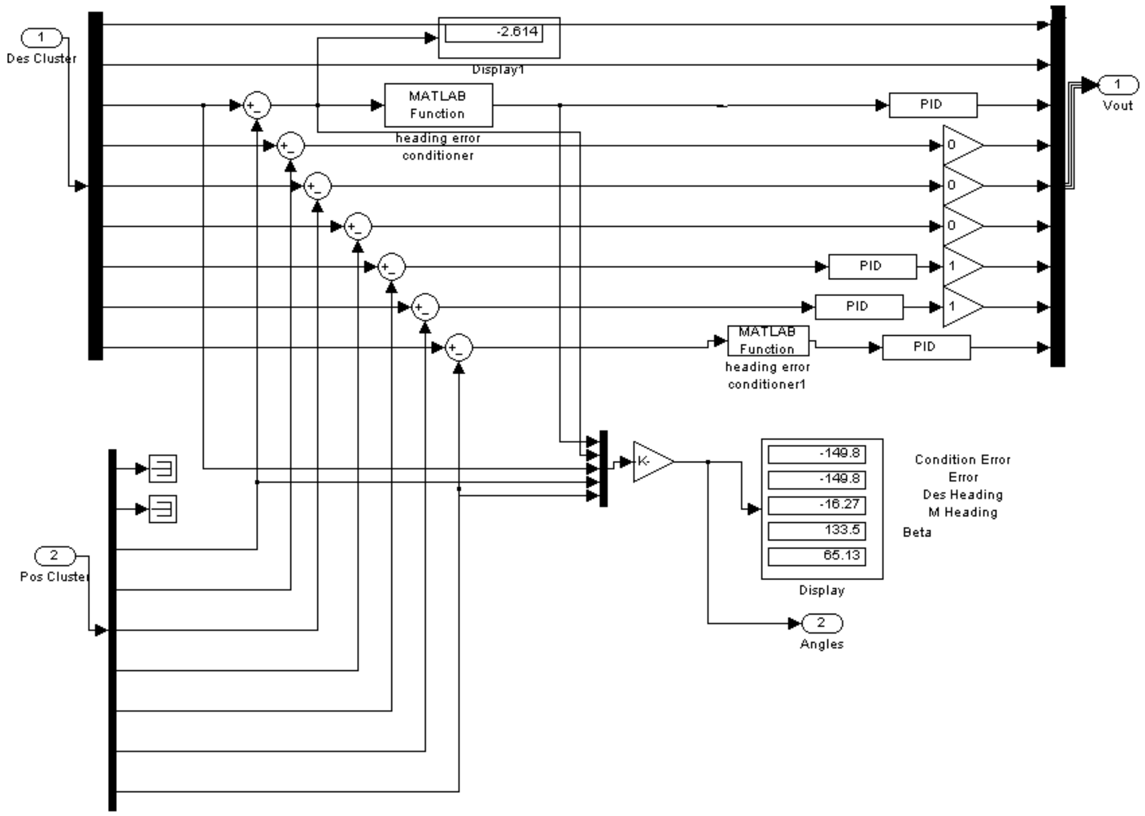

Figure B.4 Calculations of cluster errors

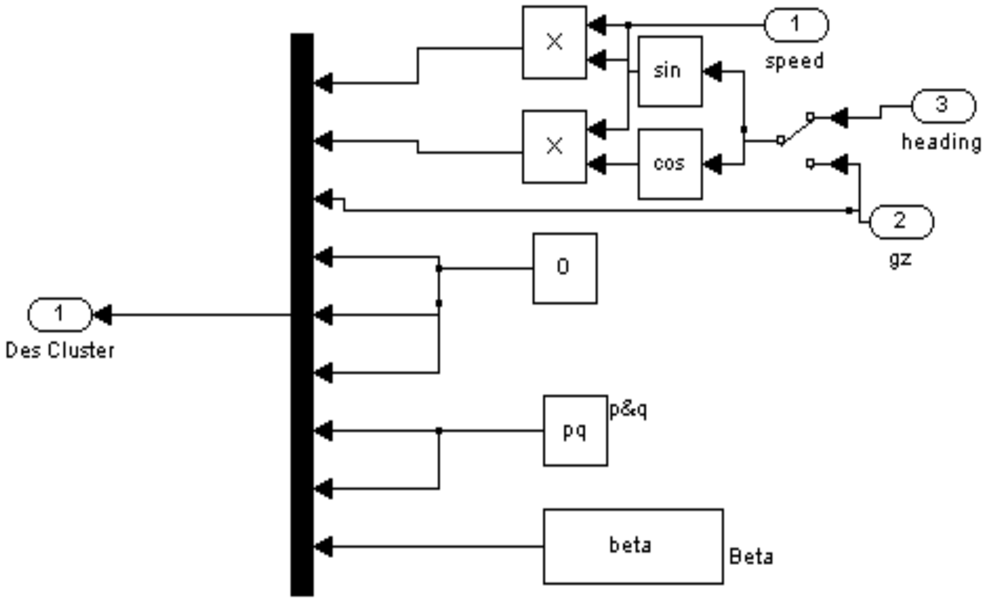

Figure B.5 Desired cluster variables

Centroid & start location

Centroid_Start.m

```
function [RA1,RC2,RB3]=Centroid_Start(Rc,beta,pq,thCs)
%Rc->Cluster Pos. [x y]
% beta-> rads
% pq ->meters
% Centroid Start
% finding the cluster space robots using


%define triangle
r=pq/2*sec(beta/2);

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];
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)
```

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

end
```

Gradient Equation

```
function [c1,c2,c3] = sensor(r1,r2,r3,plane)
% 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));
%c=100*(sin((x+y)/1000-3)+cos(y/100-3));
%c=-(x.^2+y.^2)+200;
%c=(x+y);
%c=[0;0;0]
c=tan(plane)*y;

c1=c(1);
c2=c(2);
c3=c(3);
end
```

Appendix C Inverse Jacobian m files

three_bots_centroid_inv_jacobian_matrix_exact.m

```
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;
```