

12-2016

# Cluster Control of a Multi-Robot Tracking Network and Tracking Geometry Optimization

Jasmine Cashbaugh  
*Santa Clara University*

Follow this and additional works at: [http://scholarcommons.scu.edu/eng\\_phd\\_theses](http://scholarcommons.scu.edu/eng_phd_theses)



Part of the [Mechanical Engineering Commons](#)

---

## Recommended Citation

Cashbaugh, Jasmine, "Cluster Control of a Multi-Robot Tracking Network and Tracking Geometry Optimization" (2016). *Engineering Ph.D. Theses*. 6.  
[http://scholarcommons.scu.edu/eng\\_phd\\_theses/6](http://scholarcommons.scu.edu/eng_phd_theses/6)

This Dissertation is brought to you for free and open access by the Student Scholarship at Scholar Commons. It has been accepted for inclusion in Engineering Ph.D. Theses by an authorized administrator of Scholar Commons. For more information, please contact [rscroggin@scu.edu](mailto:rscroggin@scu.edu).

CLUSTER CONTROL OF A MULTI-ROBOT TRACKING NETWORK AND  
TRACKING GEOMETRY OPTIMIZATION

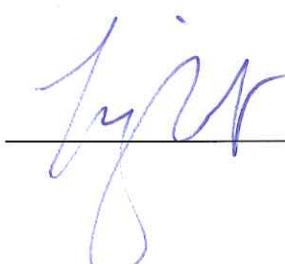
A DISSERTATION  
SUBMITTED TO THE DEPARTMENT OF MECHANICAL  
ENGINEERING  
AND THE COMMITTEE ON GRADUATE STUDIES  
OF SANTA CLARA UNIVERSITY  
IN PARTIAL FULFILLMENT OF THE REQUIREMENTS  
FOR THE DEGREE  
OF DOCTOR OF PHILOSOPHY

Jasmine Cashbaugh  
December 2016

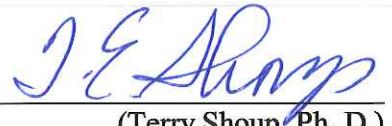
I certify that I have read this dissertation and that, in my opinion, it is fully adequate  
in scope and quality as a dissertation for the degree of Doctor of Philosophy.



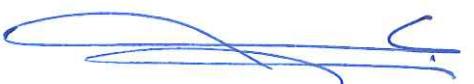
(Christopher A. Kitts, Ph. D.) Principal Advisor



(Timothy Hight, Ph. D.)



(Terry Shoup, Ph. D.)



(Maryam Khanbaghi, Ph. D.)



---

(Ignacio Mas, Ph. D.)

Approved for the University Committee on Graduate Studies.

## Abstract

The position of a moving object can be tracked in numerous ways, the simplest of which is to use a single static sensor. However, the information from a single sensor cannot be verified and may not be reliable without performing multiple measurements of the same object. When multiple static sensors are used, each sensor need only take a single measurement which can be combined with other sensor measurements to produce a more accurate position estimate. Work has been done to develop sensors that move with the tracked object, such as relative positioning, but this research takes this concept one step further; this dissertation presents a novel, highly capable strategy for utilizing a multi-robot network to track a moving target. The method optimizes the configuration of mobile tracking stations in order to produce the position estimate for a target object that yields the smallest estimation error, even when the sensor performance varies. The simulations and experiments presented here verify that the optimization process works in the real world, even under changing conditions and noisy sensor data. This demonstrates a simple, robust system that can accurately follow a moving object, as illustrated by results from both simulations and physical experiments. Further, the optimization led to a 6% improvement in the target location estimate over the non-optimized worst-case scenario tested with identical sensors at the nominal fixed radius distance of 2.83 m and even more significant improvements of over 90% at larger radial distances. This method can be applied to a wider variety of conditions than current methods since it does not require a Kalman filter and is able to find an optimal solution for the fixed radius case. To make this optimization method even more useful, it is proposed to extend the mathematical framework to  $n$  robots and extend the mathematical framework to three dimensions. It is also proposed to combine the effect of position uncertainty in the tracking system with position uncertainty of the tracking stations themselves in the analysis in order to better account for real-world conditions. Additionally, testing should be extended to different platforms with different sensors to further explore the applicability of this optimization method. Finally, it is proposed to modify the optimization method to compensate for the dynamics of the system so that sensor systems could move into an intercept course that would result in the optimal configuration about the tracked object at the desired time step. These proposals would result in a more applicable and robust system than is currently available.

## Acknowledgements

I would like to thank the people whose support made this dissertation possible. First, I'd like to thank my advisor, Christopher Kitts, for his help and advice throughout the research process. In addition to his technical help, he made the Robotic Systems Laboratory a fun and collaborative environment where there was always someone willing to help.

I would also like to thank my thesis committee for their support and feedback. Their expertise and knowledge were invaluable.

All of the members of the RSL have contributed in some manner to this thesis. In particular, I would like to acknowledge Anne Mahacek, Alicia Sherban, and Christian Zempel whose collaboration made it possible to set up the testbed used in this dissertation and turn it into a working system. I would also like to thank Thomas Adamek, Mike Rasay, and Mike Vlahos for their technical help throughout the process as well as Ethan Head for helping set up all of the computers.

Finally, I would like to thank my partner, Lloyd Droppers, who was always there to offer help and support. His help with testing procedures made work progress more smoothly as well as making it easier to explain the system to newcomers.

# Table of Contents

Abstract.....	iv
Acknowledgements.....	v
Table of Contents.....	vi
List of Figures .....	xii
List of Tables .....	xix
1. Introduction .....	1
1.1. Multi-Robot Overview .....	1
1.2. Motivation .....	1
1.3. Objective and Contributions .....	8
1.4. Reader's Guide .....	9
2. Multi-Robot Formation Control .....	10
2.1. Method Comparison .....	10
2.2. Cluster Control Overview.....	12
2.3. Cluster Space Controller.....	15
2.4. Description of a Two-Robot Cluster.....	16
2.4.1. Robot Space .....	16
2.4.2. Cluster Space .....	17
2.4.3. Forward Position Kinematics.....	18

2.4.4. Inverse Positon Kinematics.....	19
2.4.5. Forward and Inverse Velocity Kinematics.....	20
2.5. Description of a Three-Robot Cluster.....	22
2.5.1. Robot Space .....	22
2.5.2. Cluster Space .....	24
2.5.3. Forward Position Kinematics.....	25
2.5.4. Inverse Position Kinematics.....	26
2.5.5. Forward and Inverse Velocity Kinematics.....	28
3. Geometrical Optimization .....	31
3.1. Method Selection.....	31
3.2. Problem Setup.....	35
3.2.1. Sensor Constraints and their Covariance Matrices .....	37
3.2.2. Sensor Error Ellipses.....	38
3.3. Combining Error Ellipses .....	39
3.4. Configuration Optimization.....	41
3.5. Theoretical Fixed Radius Curves .....	42
3.5.1. Mathematical Simulation of Two Tracking Stations at a Fixed Radius with Identical Sensor Systems .....	44
3.5.2. Mathematical Simulation of Two Tracking Stations at a Fixed Radius with Different Sensor Systems.....	46

3.5.3. Mathematical Simulation of Three Tracking Stations at a Fixed Radius with Identical Sensor Systems .....	48
3.5.4. Mathematical Simulation of Three Tracking Stations at a Fixed Radius with Different Sensor Systems.....	50
3.5.5. Summary of Findings.....	52
3.6. Closed-Form Optimization Derivation.....	53
3.7. Two Robot Closed-Form Optimization.....	55
3.8. Three Robot Closed-Form Optimization.....	56
3.9. Closed-Form Theoretical Constrained Optimization.....	57
3.10. Formation Control .....	62
4. Vision Processing.....	64
4.1. Literature Survey .....	64
4.2. Available Data .....	65
4.3. The Influence of the Data Transmission Rate .....	67
4.4. Vision Data Simplification .....	68
4.5. Finding the Cluster-Level Position Estimate of the Pioneer.....	73
4.6. Tracking Control.....	75
5. Experimental Testbed.....	77
5.1. System Overview.....	77
5.2. Quadrotor Mobile Tracking Stations .....	78
5.3. Pioneer Tracked Object .....	79

5.4. Sensing System.....	80
5.5. Software.....	81
6. Stationary Results.....	84
6.1. Introduction .....	84
6.2. Two Quadrotor Results.....	85
6.2.1. Simulation Results .....	85
6.2.2. Pioneer Position Estimate .....	88
6.2.3. Ideal Object Position Estimate.....	90
6.3. Three Quadrotor Results.....	93
6.3.1. Simulation Results .....	93
6.3.2. Pioneer Position Estimate .....	97
6.3.3. Ideal Object Position Estimate.....	99
7. Controlled Physical Experimental Results .....	102
7.1. Introduction .....	102
7.2. Two Quadrotor Results.....	104
7.2.1. Stationary Pioneer.....	106
7.2.2. Moving Pioneer.....	109
7.3. Three Quadrotor Results.....	112
8. Simulations with Optimization-in-the-Loop .....	122
8.1. Two Quadrotor Simulations .....	123

8.2. Three Quadrotor Simulations .....	127
8.3. Exploration of the Target Location Estimate Improvement .....	130
9. Conclusions .....	134
9.1. Contributions .....	135
9.2. Future Work.....	137
References.....	139
Appendix A.....	147
A.1. Two Robot Forward Jacobian.....	147
A.2. Two Robot Inverse Jacobian .....	147
Appendix B .....	148
Appendix C .....	152
C.1. Raw Data .....	152
C.2. Plots .....	153
Appendix D.....	155
D.1. System Overview.....	155
D.2. AR.Drone.....	156
D.3. Sensing System.....	159
D.3.1. Hardware.....	159
D.3.2. Software .....	161
D.4. Networking.....	161

D.4.1. Hardware.....	161
D.4.2. Software .....	162
D.5. Characterization of Stationary Positioning.....	164
Appendix E .....	166
Appendix F .....	170
Appendix G.....	174
G.1. Calculating the Position Errors.....	174
G.2. Calculating the 60% Confidence Interval Error Ellipse Area.....	174

## List of Figures

Figure 2.1: Two robot cluster definition. ....	13
Figure 2.2: Cluster controller for n robots. ....	15
Figure 2.3: Two robots and their coordinates in robot space. ....	16
Figure 2.4: Two robots and their coordinates in cluster space. ....	18
Figure 2.5: Three robots and their coordinates in robot space. ....	23
Figure 2.6: Three robots rotation angle definitions. ....	24
Figure 2.7: Three robots and their coordinates in cluster space. ....	24
Figure 3.1: Genetic algorithm flowchart. ....	32
Figure 3.2: Particle swarm algorithm flowchart. ....	33
Figure 3.3: Hooke and Jeeves method flowchart adapted from [33]. ....	34
Figure 3.4: Terminology used to define the portion of a circle arc that describes the area of a sensor's valid sensor coverage area. ....	35
Figure 3.5: Example sensor error area. ....	36
Figure 3.6: Error ellipse of the example sensor. ....	39
Figure 3.7: Two error ellipses and their combined error ellipse. ....	41
Figure 3.8: Mathematical simulation of two sensors with a fixed radius of 2.83 m and identical sensors. The angle along the x-axis is the angle of separation between the two sensors. ....	44
Figure 3.9: Mathematical simulation of two sensors with a fixed radius of 30 m and identical sensors. The angle along the x-axis is the angle of separation between the two sensors. ....	45
Figure 3.10: Two sensors with a fixed radius of 2.83 m and different sensors properties. The angle along the x-axis is the angle of separation between the two sensors. ....	46
Figure 3.11: Two sensors with a fixed radius of 30 m and different sensors properties. The angle along the x-axis is the angle of separation between the two sensors. ....	47

Figure 3.12: Definition of the angle of separation for three mobile sensor stations. ....	48
Figure 3.13: Three sensors with a fixed radius of 2.83 m and identical sensor properties. The angle along the x-axis is the angle of separation between the two outer sensors. ....	49
Figure 3.14: Mathematical simulation results of three tracking stations with the same sensors at a fixed radius of 2.83 m from the tracked object. This plot shows the area of the combined error ellipses as a function of the angle of separation between the three mobile tracking stations where each angle of separation is varied separately. ....	50
Figure 3.15: Mathematical simulation results of three tracking stations with different sensors at a fixed radius of 2.83 m from the tracked object. This plot shows the area of the combined error ellipse as a function of the angle of separation between the three mobile tracking stations, as shown in Figure 3.12.....	51
Figure 3.16: Mathematical simulation results of three tracking stations with different sensors at a fixed radius of 2.83 m from the tracked object. This plot shows the area of the combined error ellipse as a function of the angle of separation between the three mobile tracking stations where each angle of separation is varied separately. ....	52
Figure 3.17: Optimal geometry cluster controller for $n$ robots. ....	63
Figure 4.1: Studio Diip's Fish on Wheels project together with its vision processing result [49]..	65
Figure 4.2: Camera locations on the AR.Drone version 1.0.....	66
Figure 4.3: Pioneer viewing options. Left: View from the top. Right: View from the side. ....	67
Figure 4.4: Flowchart of the vision processing algorithm.....	67
Figure 4.5: Image of the Pioneer inside the test area taken by the quadrotor's onboard forward-mounted camera, as displayed in Matlab.....	69
Figure 4.6: Progressive image simplification. Row 1 from left to right: Pioneer as seen from nominal flight distance by the human eye, Pioneer at nominal flight distance as seen by the	

quadrotor's forward-facing camera, Pioneer at nominal flight distance with a full pixel resolution of the "red" areas. Row 2 from left to right: Pioneer as seen from nominal flight distance where each square represents a grid of 5 by 5 pixels, Pioneer as seen from nominal flight distance where each square represents a grid of 10 by 10 pixels, Pioneer as seen from nominal flight distance where each square represents a grid of 20 by 20 pixels.....	70
Figure 4.7: The Pioneer as "seen" by the controllers from one meter away. ....	71
Figure 4.8: Distance explanation. ....	71
Figure 4.9: Kalman filter algorithm, adapted from [53].....	74
Figure 4.10: Optimal configuration of a two quadrotor tracking cluster. ....	75
Figure 4.11: Tracking and optimal geometry cluster controller. ....	76
Figure 5.1: Two mobile tracking stations testbed hardware layout. ....	77
Figure 5.2: Three mobile tracking stations testbed hardware layout. ....	78
Figure 5.3: AR.Drone 1.0 overview. ....	79
Figure 5.4: Pioneer 3-AT land rover overview. ....	80
Figure 5.5: UWB receiver and RFID tag with a quarter for scale. ....	80
Figure 5.6: UWB system setup. ....	81
Figure 5.7: Software layout. ....	82
Figure 6.1: Simulation results for two mobile tracking stations with identical sensors and a fixed radius. ....	87
Figure 6.2: Normalized simulation and theoretical results for two mobile tracking stations with identical sensors and a fixed radius. ....	87
Figure 6.3: Setup for testing the effect of the angle of separation on a system with two mobile tracking stations and the Pioneer as the tracked object. ....	88

Figure 6.4: Physical results with the Pioneer as tracked object and two AR.Drone 1.0 mobile tracking stations.....	89
Figure 6.5: Normalized results of the theoretical curve and physical experimental results with the Pioneer as the tracked object and two identical mobile tracking stations. ....	89
Figure 6.6: Pioneer land rover used as the tracked object. A front view (left) and side view (right) are shown.....	90
Figure 6.7: Front (top) and side (bottom) views of the Pioneer and ideal object. ....	91
Figure 6.8: Setup for two AR.Drone 1.0s as mobile tracking stations and an ideal object as the tracked object. ....	91
Figure 6.9: Results using two AR.Drone 1.0s as the mobile tracking station and an ideal object as the tracked object.....	92
Figure 6.10: Check this shit out.....	92
Figure 6.11: Simulation results for three mobile tracking stations with identical sensors and a fixed radius.....	95
Figure 6.12: Normalized simulation and theoretical results for three mobile tracking stations with identical sensors and a fixed radius.....	96
Figure 6.13: Setup for testing the effect of the angle of separation on a system with three mobile tracking stations and the Pioneer as the tracked object. ....	97
Figure 6.14: Physical results with the Pioneer as the tracked object and three AR.Drone 1.0s as the mobile tracking stations. ....	98
Figure 6.15: Normalized results of the theoretical curve and physical experimental results with the Pioneer as the tracked object and three identical mobile tracking stations.....	98
Figure 6.16: Results using three AR.Drone 1.0s as the mobile tracking stations and an ideal object as the tracked object. ....	100

Figure 6.17: Normalized results of the theoretical curve and physical experimental results with an ideal object as the tracked object and three identical mobile tracking stations.....	100
Figure 7.1: Pioneer user-input joystick control and position tracking.....	104
Figure 7.2: Cluster controller for two mobile tracking stations.....	105
Figure 7.3: Optimal two quadrotor configuration. ....	106
Figure 7.4: This plot shows the actual and estimate Pioneer positions throughout the first experiment with two mobile tracking stations and a stationary Pioneer. ....	107
Figure 7.5: This plot shows the actual and estimate Pioneer positions throughout the second experiment with two mobile tracking stations and a stationary Pioneer. ....	108
Figure 7.6: This plot shows the actual and estimate Pioneer positions throughout the first experiment with two mobile tracking stations and a moving Pioneer.....	110
Figure 7.7: This plot shows the actual and estimate Pioneer positions throughout the second experiment with two mobile tracking stations and a moving Pioneer.....	111
Figure 7.8: Three mobile tracking station cluster controller. ....	113
Figure 7.9: Optimal three quadrotor configuration.....	114
Figure 7.10: This plot shows the actual and estimated Pioneer positions throughout the first experiment with three mobile tracking stations and a stationary Pioneer.....	114
Figure 7.11: This plot shows the actual and estimated Pioneer positions throughout the second experiment with three mobile tracking stations and a stationary Pioneer.....	116
Figure 7.12: This plot shows the actual and estimated Pioneer positions throughout the third experiment with three mobile tracking stations and a stationary Pioneer.....	118
Figure 7.13: Actual and desired distance control variables for three mobile tracking station exploration test.....	119

Figure 7.14: Actual and desired angular control variables for three mobile tracking station exploration test.....	120
Figure 8.1: X estimation error for the two quadrotor simulation with slowly degrading identical sensors.....	123
Figure 8.2: At 20 seconds, the ideal configuration changes from 90 degrees (blue) to 180 degrees (red).....	125
Figure 8.3: X estimation error for the two quadrotor experiment with abruptly changing sensor properties.....	125
Figure 8.4: X estimation error for the three quadrotor experiment with an abrupt sensor failure at 20 seconds.....	127
Figure 8.5: Percent improvement of the optimal sensor configuration over the worst-case configuration of two and three identical sensors at the same fixed radius.....	131
Figure 8.6: Percent improvement of the optimal sensor configuration over the worst-case configuration of two and three sensors with extreme sensor properties at the same fixed radius.....	132
Figure 8.7: Percent improvement of the optimal sensor configuration over the worst-case configuration of two and three sensors with medium differences in sensor properties at the same fixed radius.....	132
Figure 0.1: The Pioneer on the left has a forward-back orientation while the Pioneer on the right has a right-left orientation.....	152
Figure 0.2: Raw data and exponential curve.....	154
Figure 0.1: Physical hardware layout.....	155
Figure 0.2: Testbed flowchart.....	156
Figure 0.3: Hardware layout diagram.....	156

Figure 0.4: AR.Drone components.....	157
Figure 0.5: AR.Drone coordinate frame.....	158
Figure 0.6: Drone coordinate frame. ....	159
Figure 0.7: UWB receiver and RFID tag.....	160
Figure 0.8: Layout of UWB system.....	161
Figure 0.9: Software layout.....	163
Figure 0.1: Block diagram of the two drone simulation. .....	167
Figure 0.2: Block diagram of the tracking algorithm used in the two drone simulation. ....	168
Figure 0.3: Block diagram of the PID controller used in the two drone simulation. ....	169
Figure 0.1: Block diagram of the three drone simulation.....	171
Figure 0.2: Block diagram of the tracking algorithm used in the three drone simulation. ....	172
Figure 0.3: Block diagram of the PID controller used in the three drone simulation.....	173
Figure 0.1: Matlab script used to calculate the 60% confidence interval error ellipse area. ....	174

## List of Tables

Table 2.1: Description of the variables in robot space for a two robot cluster.....	17
Table 2.2: Description of the variables in cluster space for a two robot cluster.....	18
Table 2.3: Two robot singularities. ....	22
Table 2.4: Description of the variables in robot space for a three robot cluster. ....	23
Table 2.5: Description of the variables in cluster space for a three robot cluster. ....	25
Table 2.6: Three robot singularities. ....	30
Table 3.1: Variable definitions for Eq. (3.1). ....	37
Table 3.2: Axes for combining error ellipses example.....	40
Table 3.3: Axes for the combining error ellipses example.....	41
Table 3.4: Inputs for a bounded 2D area for two sensors with identical properties.....	58
Table 3.5: Output for a bounded 2D area for two sensors with identical properties. ....	59
Table 6.1: Sensor and position errors determined from physical tests.....	86
Table 6.2: Variables used in the two mobile tracking station simulations. ....	86
Table 6.3: Angular variables used in the simulations with three mobile tracking stations.....	94
Table 6.4: Distance variables used in the simulations with three mobile tracking stations. ....	95
Table 7.1: Location estimate summary for the first stationary Pioneer test with two mobile tracking stations.....	106
Table 7.2: Control variable summary for the first stationary Pioneer test with two mobile tracking stations.....	107
Table 7.3: Location estimation summary for second stationary Pioneer test with two mobile tracking stations.....	108
Table 7.4: Control variable summary for second stationary Pioneer test with two mobile tracking stations.....	109

Table 7.5: Location estimation summary for the first moving Pioneer test with two mobile tracking stations.....	110
Table 7.6: Control variable summary for the first moving Pioneer test with two mobile tracking stations.....	110
Table 7.7: Location estimation summary for the second moving Pioneer test with two mobile tracking stations.....	111
Table 7.8: Control variable summary for the second moving Pioneer test with two mobile tracking stations.....	112
Table 7.9: First stationary Pioneer test with three mobile tracking stations location estimation summary. ....	115
Table 7.10: First stationary Pioneer test with three mobile tracking stations distance control variable summary. ....	115
Table 7.11: First stationary Pioneer test with three mobile tracking stations angular control variable summary. ....	115
Table 7.12: Second stationary Pioneer test with three mobile tracking stations location estimation summary. ....	116
Table 7.13: Second stationary Pioneer test with three mobile tracking stations distance control variable summary. ....	117
Table 7.14: Second stationary Pioneer test with three mobile tracking stations angular control variable summary. ....	117
Table 7.15: Stationary Pioneer exploration test with three mobile tracking stations location estimation summary. ....	118
Table 7.16: Stationary Pioneer exploration test with three mobile tracking stations distance control variables. ....	119

Table 7.17: Stationary Pioneer exploration test with three mobile tracking stations angular control variables .....	119
Table 8.1: Location estimation summary for two mobile tracking stations and slow sensor degradation.....	124
Table 8.2: Control variable summary for two mobile tracking stations and slow sensor degradation.....	124
Table 8.3: Location estimation for two mobile tracking stations and an abrupt change in sensor properties.....	126
Table 8.4: Control variable summary for two mobile tracking stations and an abrupt change in sensor properties.....	126
Table 8.5: Location estimation summary for simulation with three mobile tracking stations and an abrupt failure of sensor 1 at 20 seconds. ....	128
Table 8.6: Distance control variable summary for simulation with three mobile tracking stations and an abrupt failure of sensor 1 at 20 seconds. ....	128
Table 8.7: Angular control variable summary for simulation with three mobile tracking stations and an abrupt failure of sensor 1 at 20 seconds. ....	128
Table 8.8: Radii examined for the three target estimate improvement cases.....	130
Table 0.1: Raw data of the size of the Pioneer as “seen” by the quadrotor at various distances. .....	153
Table 0.2: $R^2$ values for the exponential curve fit.....	154
Table 0.1: Locations of the UWB reference tags. ....	160
Table 0.2: Locations of the UWB receivers. ....	160
Table 0.3: UWB system error analysis. ....	164

# **Chapter 1**

---

## **1. Introduction**

### **1.1. Multi-Robot Overview**

Robots have many uses in today's world; because of the robot's strength, speed, precision, repeatability, and ability to withstand extreme environments, they are used for a variety of purposes that would be dangerous or difficult for humans to perform [1]. Groups of cooperative robots are even better since they can cover more ground, provide validation for each other, or cover the area of a failed robot. Multi-robot systems can also perform new services by exploiting their ability to be physically distributed. This physical distribution of sensors in a multi-robot system can lead to greater accuracy of the fused sensor information obtained from the environment [2]. This advantage will be explored further in this dissertation by utilizing a multi-robot system to optimally track a mobile object by forming a distributed mobile tracking sensor network.

### **1.2. Motivation**

Tracking a moving object can be difficult due to terrain, lighting conditions, and the unpredictability of the tracked object. However, robots can fly above the terrain, be outfitted with sensors that mitigate the disadvantages of poor lighting conditions, and track an object until they are recalled. Cooperative groups of tracking robots can have an array of sensors that allow the robots to obtain different perspectives of a single scene using a few inexpensive robots rather than a single expensive robot.

Consequently, the literature proposes many methods for using groups of robots for localization and tracking purposes. While localization and tracking are not the same problem, they do share many elements in common since both strive to determine accurately the position of an object. In localization applications, sensors on the target object take relative measurements of environmental landmarks, allowing the target object to determine its own position estimate. In tracking applications, off-board sensor systems measure the relative position of the tracked object and determine a position estimate for that object. For the purposes of clarity in this dissertation, localization applications will be said to use beacons as landmarks for relative positioning estimates while tracking applications will be said to use sensor systems to determine positioning estimates for the tracked object.

In both localization and tracking applications, the accuracy of the position estimate is affected by the number of sensors/beacons that are able to provide relative target measurements. While a single sensor/beacon is the easiest system to implement, multiple measurements must be taken in order to ensure accuracy of the position information. Multiple sensors/beacons can allow more timely position verification, but introduce additional system complexities. For example, the properties of the sensors/beacons and their geometry with respect to the target affect the accuracy of the system. If identical sensors/beacons are too close together, they will supply nearly identical information, adding little to the knowledge base. If the sensors/beacons are too far apart, some important information may be missed. Thus, the best sensor/beacon spacing is somewhere between these two extremes. This dissertation details an online optimization process which identifies the optimal configuration geometry for multiple mobile sensor systems given possible changes in the number of sensors/beacons, sensor/beacon ranges, sensor/beacon operation, or other relevant parameters. The mathematical basis for this method is provided in this dissertation, along with simulation and

experimental validation of this technique. It is believed that this approach is new because it considers the sensor configuration as a whole rather than as the sum of its parts, providing a more comprehensive view of the system that is being optimized than that provided by other methods.

Previous work has explored many avenues for optimizing multi-sensor/beacon systems. In a localization application, the authors of [3] used a static array of acoustic beacons to determine the location of a mobile node using range information. The range information of the beacons formed intersecting circles, allowing the location of the mobile node to be determined quite accurately and the mobile node to closely follow the desired path. No optimization of the number or placement of sensors was performed in this set of experiments.

Chakrabarty *et al* [4] provided a mathematical basis for placing multiple beacons in an environment with one or more moving targets in order to minimize sensor cost while completely covering the sensor field. In this formulation, it is assumed that the beacons have different ranges and costs and that every grid in the 3D area through which the target(s) may move must be covered by a minimum number of beacons. The cost of the deployed beacons was minimized under the coverage constraints, resulting in the placement of specific beacon types at specific grid points.

Shang *et al* [5] present a method to minimize energy consumption without significantly impacting the positioning accuracy of a multi-sensor array by determining which sensor systems will participate in the positioning task using a neural network aggregation model. Only the sensor systems which are in range of the target transmit their positioning information; all other sensor systems are inactive and do not transmit data. This is taken a step further in [6] where every sensor system that is within range of the target is a candidate for participation in the

target position estimation task. The sensor systems are still static and those not participating in the tracking task are still inactive, but only the sensor system combination that yields the most accurate position estimate is used in the tracking process rather than every node within range of the target.

The energy cost of a wireless sensor network was further reduced in [7] which used a static wireless sensor network to track a single moving target constrained to move in 2D space. The sensors were ultrasonic and it was assumed that all sensors had the same sensing properties. A Monte Carlo method was used to determine which sensor systems to use in each time step to maximize tracking accuracy and minimize energy consumption subject to a constraint on the minimum number of sensor systems. In order to conserve energy, the minimum transmission energy consumption was used to determine which one of the active sensor systems was chosen as the data fusion center. All sensor systems not actively collecting data were inactive during the time step.

A major issue when using static sensors to determine the location of a mobile object is that the mobile object may eventually leave the sensor range, resulting in loss of the mobile object. This can be avoided by moving the sensors to follow the tracked object. In [8], tracking experiments were performed using acoustic modems to measure ranges between vehicles. A leader-follower setup was used in which the lead vehicle was an underwater vehicle which acted as the target and the following vehicles were surface craft which acted as sensor systems. These sensor systems were able to remain with the target, providing it with more accurate position information than that obtained solely by the target vehicle, enabling greater navigation accuracy. A similar mix of surface craft and underwater vehicles were also used for a series of experiments in [9] where surface craft acted as sensor beacons for the localization of

underwater vehicles. Once the underwater vehicle calculated its own position, it broadcast this position estimate back to the surface vehicles. This allowed the sensor beacons to follow the underwater vehicles and try to form a right-angled triangle with the underwater vehicle at the vertex to minimize the estimation error.

Martínez and Bullo [10] used multiple identical sonar sensor systems to track a single target. The target was mobile and the sensor systems were either all static or all mobile, depending on the experiment. However, the target was constrained to a bounded area during both experimental cases and the sensor systems were constrained to the boundary of this area. An estimate of the target's position was found through fusion using an Extended Kalman Filter. For both the static and dynamic cases, the optimal sensor system position was defined as the position which yielded the lowest estimation error, found by minimizing the determinant of the Fisher information matrices for the sensor system estimation models. The resulting optimal sensor placement was an array wherein the sensor systems were evenly distributed about the target. Since the mobile sensor systems could react to changes in the target's position, the mobile sensor system experiments were found to consistently yield more accurate results.

Bahr *et al* [11] developed a method to minimize the localization uncertainty. This method involved two types of vehicles with mounted sensors: surface craft and underwater vehicles. All vehicles were equipped with acoustic range sensors, but only the surface craft knew their absolute position, allowing them to function as beacons. Using the ranging information and the positions of the beacons, the underwater vehicles, serving as the target vehicles, could determine their positions more accurately. All vehicles shared position and velocity information with one another on a fixed schedule. The optimization process chose the beacon configuration that minimized the trace of the difference between the covariance matrices before and after the

Extended Kalman Filter was applied and did not use knowledge of the underwater vehicles' trajectory.

The optimization of moving sensors is also useful in applications where the target positions are unknown or may change unpredictably. The authors of [12] explored this problem in a multi-target, multi-sensor environment where the sensor systems were mobile and had constraints on their movement and positions. Each sensor system tried to minimize the coverage requirements using its own constraints and knowledge of its neighbors' positions with each sensor system position determined individually. In [13], a swarm of mobile sensing robots were used to detect olfactory targets in a single target environment. The model did not penalize sensor overlap and assumed the mobile sensing robots had a limited sensing range and that neighboring coverage areas that touched had larger coverage areas than those that did not touch. Maximizing the coverage area was assumed to result in the best chance of tracking the olfactory plumes to their source. Thus, the optimal swarm formation was defined as the distance between sensor systems that resulted in the largest coverage area, found using Powell's conjugate gradient decent method.

The authors of [14] used mobile sensor systems, each with a single camera as the sensor, to track one or more moving targets. The mobile sensor systems were constrained to the maximum robot velocity and their positions were limited by a minimum standoff distance from the target. It was assumed that each mobile sensor system knew its own position. Dynamic models of the target's motion were obtained using an approximation of the target dynamics. The mobile sensor systems moved to minimize the target position estimate error at the next time instant based on the dynamic model.

In contrast to the previously presented methods, the method presented in this dissertation is intended for tracking purposes and assumes a single target and multiple sensor systems where the sensor systems reposition themselves, not only to follow the tracked object, but to follow the tracked object in the geometric configuration that results in the best position estimate at each time step. This methodology takes into account the sensor properties, which may change over time. It also allows for different sensors to be used during the same application. It does not require assumptions associated with the use of a Kalman filter and is shown to be computable for critical scenarios not covered by methods found in the literature, as discussed further in this work. Specifically, objective functions will be developed and implemented for two and three sensor systems to determine the optimal angular separation between tracking stations. This is defined as the angular separation that results in the estimate of the target object's location with the lowest estimation error. Thus, this optimization method is able to find an optimal geometric configuration under a wide range of conditions.

This dissertation also focuses on a proof of concept for tracking a moving object using a low cost testbed. The tracking system consisted of quadcopter vehicles controlled via a networked control system. The vehicles were positioned using the cluster space formation control approach and the quadcopters used only their onboard sensor capabilities to track a separately controlled robot via vision processing. This research represents an advancement from that found in the literature by using multiple mobile robots working together to track an object while maintaining the optimal geometric configuration. This optimal geometric configuration is defined as the configuration that minimizes the position estimation error and is found using the novel technique detailed in Chapter 3 and [15]. Cluster control, discussed in the next chapter, is used to maintain this optimal geometry throughout the tracking process. This method is applicable whether the sensors have identical or disparate properties and, if the optimization process is

included in the control loop, can adapt to changing conditions where sensor performance is a function of position or compromised due to a malfunction. This method was experimentally verified and the mobile sensor systems were found to maintain the desired geometric configuration with respect to the tracked object for the duration of the experiment, yielding an accurate estimate of the target's position.

### **1.3. Objective and Contributions**

The objective of this dissertation is to find and implement, via real-time formation control, the optimal tracking configuration for both two and three robot clusters, as well as a generic method that could be applied to a cluster of  $n$  robots. Previous work at SCU's RSL has found a two-dimensional optimal tracking configuration through experimental methods [16] [17]. This dissertation extends this work by providing a mathematical basis for determining the optimal tracking configuration for a general  $n$  robot cluster. The theory for two and three robot clusters is experimentally verified with two and three quadrotor clusters and a land-based Pioneer as well as an ideal object.

This dissertation also seeks to provide a proof of concept for the tracking mission by demonstrating that two and three quadrotor robot clusters can follow a land-based Pioneer robot using only sensor data. That is, the cluster has no knowledge of the Pioneer's position other than that provided by the quadrotors' onboard sensors. The Pioneer is not stationary during these tests, but travels throughout the test area in a random pattern. The pattern is not known to the cluster ahead of time and is not programmed into its behavior in any way. The cluster will be controlled to maintain the optimal configuration throughout the tracking procedure.

Finally, this dissertation will examine the effect of an optimization-in-the-loop that allows the cluster to dynamically re-optimize its configuration due to changes in the sensor properties. These changes include the number of available sensors, sensor failures, and changes in sensor sensitivity. These explorations will shed more light on some of the issues that will be faced in a real-world implementation of robotic clusters in emergency situations and will show that this methodology is feasible outside of a laboratory setting.

## 1.4. Reader's Guide

This dissertation consists of nine chapters that detail the concept of two and three tracking robot clusters. The first chapter (of which this paragraph is a part), provides an overview of the dissertation motivation and objectives. The modeling framework for cluster control is covered in Chapter 2 while the geometrical optimization is discussed in Chapter 3. Chapter 4 details the vision processing algorithm used in this research and Chapter 5 provides an overview of the experimental testbed. Chapter 6 details the results of stationary verification of the mathematical algorithm and Chapter 7 provides the controlled experimental results. Chapter 8 describes the simulation results with an optimization-in-the-control-loop as well as an exploration of the percent improvement of the target location estimate under various conditions. Finally, the conclusions are presented in Chapter 9.

# **Chapter 2**

---

## **2. Multi-Robot Formation Control**

### **2.1. Method Comparison**

There are many methods that can be used to control a group of robots. The methods considered for this application include swarm control, leader-follower formation control, and cluster control. The following paragraphs provide an overview of swarm control and leader-follower formation control and why they were not chosen for this application. The rest of this chapter details cluster control and its application in this dissertation.

Swarm control is a popular method in the literature. [19] described swarm control as utilizing a decentralized local control and local communication which results in the emergence of global behavior as the consequence of self-organization. For example, in [20], a robotic swarm was used to track odor plumes to their source by establishing a cohesive sensor network across the swarm. Each robot could measure the distance to its neighbors and to any obstacle it might encounter as well as send and receive small messages to its neighbors. The robots individually measured their own odor concentration and shared this information with their neighbors. In this manner, the robots each acted as temporal filters by taking new measurements each time step and the swarm acted as a spatial filter by comparing many measurements taken from different positions. The robots then moved together in the direction of the swarm consensus and maintained their formation while tracking the odor plume. The more robots that participated in the swarm, the more accurate the picture of the gradient they were able to obtain.

The authors of [21] utilized a swarm of simple robots that collaborated with one another to search an area. While this method was successful under the right conditions, a number of shortcomings were found to this swarm approach. The swarm was unable to dynamically adjust its velocity, which could lead to non-optimal tracking or even loss of the target. The success of the method was dependent on the initialization conditions and the accuracy of the fitness criteria used. Additionally, the swarm sometimes experienced premature convergence and stagnation in local optima and failed to perform well with multi-objective, dynamic, uncertain, and time dependent problems.

Leader-follower formation control strategies are also a popular control method in the literature for multi-robot applications. This method usually requires the specification of a bearing and distance between the lead robot and the following robot, which lends itself to tracking applications as the follower robots already “track” the lead robot. This circumstance was utilized in [22], [23], and [24] where the tracked object was treated as a virtual leader. Each follower robot could only measure its relative distance and bearing to the lead robot. This information was then used to position the robot to “track” the lead robot along its trajectory. In [24], the lead robot even shared its control vector, orientation, and velocity information with the following robots. While this does lead to more accurate tracking, it is not always feasible outside of the laboratory.

In [25], a leader-follower control strategy was followed in order to organize a group of  $n$  robots into a rigid formation while maintaining a reference velocity. The reference velocity was assumed to be constant or piecewise constant and only two robots, the leaders, knew the reference velocity. These leader robots only knew the relative position of each other while the following robots could measure the relative position of their two neighbors. The follower robots

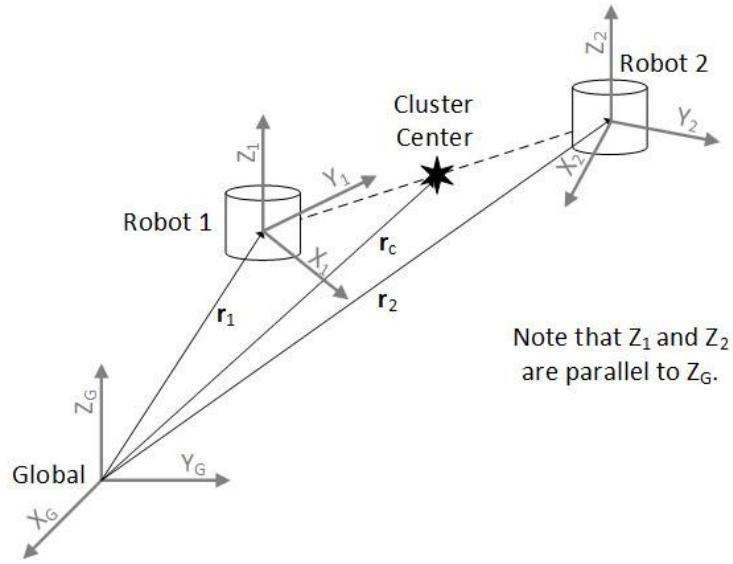
then positioned themselves at a set distance from their neighbors, the group leaders, and tried to match their velocity. There were two major issues with this implementation. First, the geometry of the group was not able to be completely specified; different configurations could meet the required objectives equally well. Second, the robot positions did not have to be unique in order to meet these requirements so some robots were collocated. This is feasible in simulation, but would not be possible in a physical experiment.

## 2.2. Cluster Control Overview

Cluster control, developed at Santa Clara University's Robotics Systems Laboratory (SCU's RSL) and presented in [1], was used to control the multi-robot systems used in this dissertation. This method allows for the control of a group of robots without specifying the behavior of each robot individually. Instead, the position and geometry of a group of  $n$  robots is specified by the user while the controller calculates the individual robot commands. This technique is an operational space approach that envisions the multi-robot cluster as a virtual, full degree-of-freedom, articulating mechanism [1] and will be described further in this chapter. While, in theory, any number of robots could be used in cluster control, clusters of two or three aerial robots, each with four independent degrees of freedom, were used to demonstrate the work described here.

Cluster control makes use of two static spaces termed robot space and cluster space. Robot space state variables are the conventional position and velocity variables used to describe the motion state of a mobile robot with respect to a global frame [1]. For the cluster of two aerial vehicles shown in Figure 2.1, the pose vector,  $R$ , consists of the three dimensional positions ( $x_i$ ,  $y_i$ ,  $z_i$ ) and the yaw angle,  $\theta_i$ , for each of the two vehicles, where  $i = 1, 2$ . This definition is provided in the following equation:

$$R = \begin{bmatrix} x_1 \\ y_1 \\ z_1 \\ \theta_1 \\ x_2 \\ y_2 \\ z_2 \\ \theta_2 \end{bmatrix} \quad (2.1)$$



**Figure 2.1: Two robot cluster definition.**

To represent the state in cluster space, a cluster frame is assigned to the group of robots with an explicit designation of its position and orientation with respect to the robots. For the example shown in Figure 2.1, the frame is centered between the robots with its  $\hat{Y}$  unit vector oriented up, parallel to  $\hat{Z}_G$ . The cluster space pose vector,  $C$ , consists of the position and orientation of the cluster frame, shape variables that collectively describe the location of the robots with respect to the cluster frame, and individual orientation variables describing the relative orientation of each robot with respect to the cluster frame. For the system in Figure 2.1, the cluster frame is located by the variables  $(x_c, y_c, z_c)$  and oriented by the yaw and roll angles,  $\alpha$

and  $\beta$  respectively; the separation distance between robots,  $p$ , is the shape variable, and the relative robot orientation variables are  $\phi_1$  and  $\phi_2$ . This definition is provided in Eq. (2.2).

$$C = \begin{bmatrix} x_c \\ y_c \\ z_c \\ \alpha \\ \beta \\ \varphi_1 \\ \varphi_2 \\ p \end{bmatrix} \quad (2.2)$$

The position vectors in each space,  $R$  and  $C$ , can be related through a set of kinematic equations, as can the velocities,  $\dot{R}$  and  $\dot{C}$ . The forward position kinematic relationships, discussed in detail in Sections 2.4.3 and 2.5.3, allow the cluster space positions to be computed based on knowledge of robot space positions. These equations can be solved for the robot space positions to produce inverse position kinematic equations, allowing robot space positions to be computed based on knowledge of cluster space positions. A Jacobian transform can be used to transform  $\dot{R}$  to  $\dot{C}$ , as shown in Eq. (2.3), where the Jacobian is a matrix of the partial derivatives of the forward position kinematic equations. The inverse velocity relationship is provided in Eq. (2.4), allowing  $\dot{R}$  to be computed from a specified  $\dot{C}$ . It is interesting to note that the Jacobian and its inverse are both instantaneous linear transforms that are functions of the pose of the group of robots.

$$\dot{C} = J(R) * \dot{R} \quad (2.3)$$

$$\dot{R} = J^{-1}(C) * \dot{C} \quad (2.4)$$

### 2.3. Cluster Space Controller

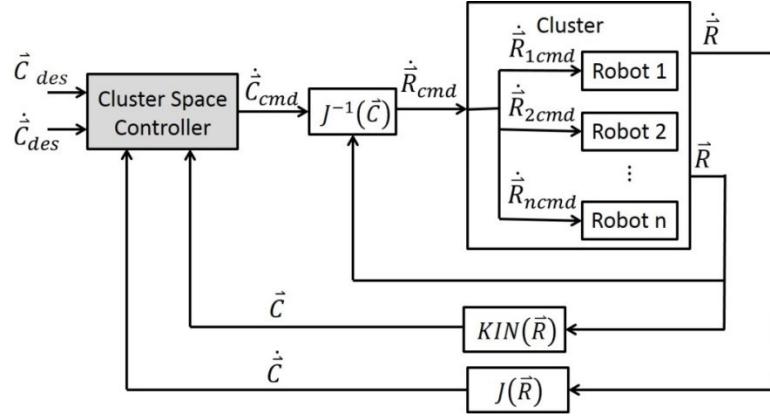


Figure 2.2: Cluster controller for  $n$  robots.

A typical control implementation for a cluster space controller is shown in Figure 2.2. In this architecture, the controller accepts control specifications as cluster space variables, an abstraction that was found to be beneficial since it promotes simple human interaction for human-based control as well as a convenient level of abstraction for higher-level automated controllers. Control compensations are also computed in cluster space, which generally leads to well-behaved cluster space motions even though the motions of individual robots may be quite complex due to the nonlinear nature of the kinematic relationships. The diagram in Figure 2.2 employs a resolved rate control approach in which control commands are cluster velocity set-points that are converted to individual robot velocity set-points through the inverse Jacobian. As the robots execute these individual velocity commands, their positions and velocities can be collectively converted to cluster space for use by the cluster controller. In practice, SCU's RSL has made great use of this resolved rate control approach given the RSL's use of many commercially available robots that are naturally commanded through velocity set-points. It is possible, however, to implement full dynamic control in which the controller computes forces and torques. In this case, these controller commands are transformed to robot-specific control

forces and torques through the use of a Jacobian transpose transform [26]. In the experiments presented later in this dissertation, a resolved rate controller is used which does not make use of velocity feedback due to the slow speed of the system.

## 2.4. Description of a Two-Robot Cluster

This section provides a detailed description of a two robot cluster. In the work presented here, the two robots used are quadrotor aerial robots with four degrees of freedom:  $x$ ,  $y$ ,  $z$ , and  $\theta$  (yaw). The following subsections present a description of robot space, cluster space, and the positon kinematics and Jacobians used to translate between the two spaces.

### 2.4.1. Robot Space

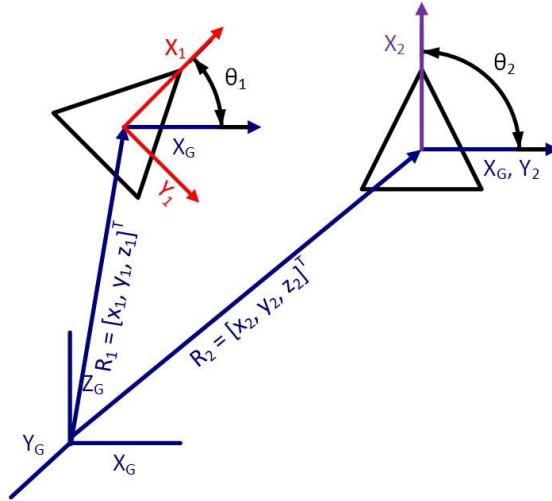


Figure 2.3: Two robots and their coordinates in robot space.

Cluster control works by transforming a set of positions between two different coordinate frames in order to perform calculations in the simplest frame. The first coordinate frame is called robot space and is probably the easiest to visualize; it is illustrated in Figure 2.3 while the variables are defined in Table 2.1. Here, each robot has its own ( $x$ ,  $y$ ,  $z$ ) location in global space and its own yaw angle,  $\theta$ , with respect to the global z-axis [1]. The robot positions are not

defined with respect to each other in any way. This makes robot space the ideal frame in which to give robot commands. Each robot receives an individual command dependent solely on its own position in space.

**Table 2.1: Description of the variables in robot space for a two robot cluster.**

Variable	Description
$x_1$	Distance (in meters) from the first robot to the origin in the x direction.
$y_1$	Distance (in meters) from the first robot to the origin in the y direction.
$z_1$	Distance (in meters) from the first robot to the origin in the z direction.
$\theta_1$	The yaw (in radians) of the first robot about the z-axis.
$x_2$	Distance (in meters) from the second robot to the origin in the x direction.
$y_2$	Distance (in meters) from the second robot to the origin in the y direction.
$z_2$	Distance (in meters) from the second robot to the origin in the z direction.
$\theta_2$	The yaw (in radians) of the second robot about the z-axis.

#### 2.4.2. Cluster Space

In contrast, all robot positions are defined relative to the cluster center in cluster space, illustrated in Figure 2.4 and defined in Table 2.2. Here, the cluster has its own ( $x$ ,  $y$ ,  $z$ ) location, yaw (rotation about the z-axis, designated by  $\alpha$ ), and roll (rotation about the y-axis, designated by  $\beta$ ) with respect to the global coordinate frame. Each of the robot positions are then defined with respect to this cluster center by a distance,  $p/2$ , from the center, a vector direction for that distance,  $\pm y_c$ , and a yaw angle,  $\varphi_i$ , with respect to the cluster yaw [1]. Since the robot positions are defined with respect to the cluster center in this coordinate frame, it is the ideal frame in which to give commands to the cluster. This allows the user to specify the location of the cluster center and the cluster geometry without specifying commands for each robot. For two robots, this does not save much work, but the benefits increase as the amount of robots used in the cluster increases.

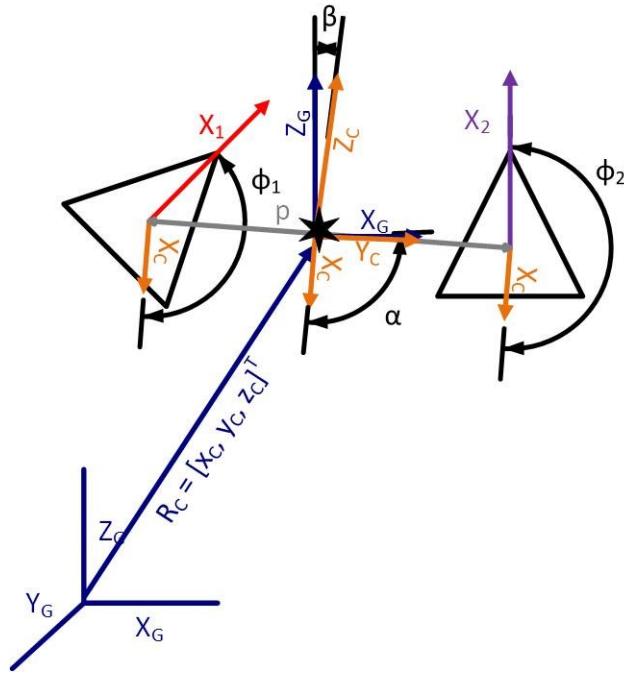


Figure 2.4: Two robots and their coordinates in cluster space.

Table 2.2: Description of the variables in cluster space for a two robot cluster.

Variable	Description
x	Distance (in meters) from the cluster center to the origin in the x direction.
y	Distance (in meters) from the cluster center to the origin in the y direction.
z	Distance (in meters) from the cluster center to the origin in the z direction.
$\alpha$	The cluster angle of rotation (in radians) about the z-axis, yaw.
$\beta$	The cluster angle of rotation (in radians) about the y-axis, roll.
$\varphi_1$	The heading of quadrotor 1 (in radians) with respect to the cluster x-axis.
$\varphi_2$	The heading of quadrotor 2 (in radians) with respect to the cluster x-axis.
p	Distance (in meters) of the quadrotor centers.

#### 2.4.3. Forward Position Kinematics

A key aspect of cluster control is determining the kinematic equations that transform the variables from robot space to cluster space. While there are multiple ways to do this from a mathematical standpoint, the goal is to keep the equations as simple as possible so that

calculations can be performed quickly and to limit the impact of singularities on the controller.

The impact of the singularities will be discussed in more detail in Section 2.4.5.

The kinematic equations used in this research are found in Eq. (2.5) to (2.12) below. The input variables in these equations are all variables defined in robot space while the outputs are all defined in cluster space. The robot space variables are defined in Table 2.1 while the cluster space variables are defined in Table 2.2.

$$x_c = \frac{1}{2}(x_1 + x_2) \quad (2.5)$$

$$y_c = \frac{1}{2}(y_1 + y_2) \quad (2.6)$$

$$z_c = \frac{1}{2}(z_1 + z_2) \quad (2.7)$$

$$\alpha = \text{atan2}(\hat{x}_c \cdot \hat{y}_G, \hat{x}_c \cdot \hat{x}_G)^1 \quad (2.8)$$

$$\beta = \text{atan2}(\hat{y}_c \cdot \hat{z}_G, \hat{z}_c \cdot \hat{z}_G) \quad (2.9)$$

$$\phi_1 = \theta_1 - \alpha \quad (2.10)$$

$$\phi_2 = \theta_2 - \alpha \quad (2.11)$$

$$p = \sqrt{(x_1 - x_2)^2 + (y_1 - y_2)^2 + (z_1 - z_2)^2} \quad (2.12)$$

This results in the following position vector in cluster space defined in Eq. (2.2).

#### 2.4.4. Inverse Position Kinematics

Similarly, inverse position kinematics are required to transform the cluster space variables into robot space variables. As for the forward position kinematics, there are multiple ways to do this

---

<sup>1</sup> Atan2 is a Matlab function that returns a signed value of the inverse tangent function [27]. The use of atan2 allows the user to obtain an angle measurement between  $[-\pi, \pi]$  in the quadrant corresponding to the provided input values. This simplifies computations by allowing the use of a single function rather than the two step process of computing an angle from the inverse tangent and then finding the appropriate quadrant for that angle.

from a mathematical standpoint, but the equations should be simple and the singularities should have minimal impact on the controller. Again, the impact of the singularities will be discussed in more detail in Section 2.4.5.

The inverse kinematic equations used in this dissertation are found in Eq. (2.13) to (2.20). These equations feature cluster space input variables and robot space output variables. Both sets of variables are defined as in the previous section.

$$x_1 = x_c - \frac{1}{2}p \sin \alpha \cos \beta \quad (2.13)$$

$$y_1 = y_c + \frac{1}{2}p \cos \alpha \cos \beta \quad (2.14)$$

$$z_1 = z_c + \frac{1}{2}p \sin \beta \quad (2.15)$$

$$\theta_1 = \phi_1 + \alpha \quad (2.16)$$

$$x_2 = x_c + \frac{1}{2}p \sin \alpha \cos \beta \quad (2.17)$$

$$y_2 = y_c - \frac{1}{2}p \cos \alpha \cos \beta \quad (2.18)$$

$$z_2 = z_c - \frac{1}{2}p \sin \beta \quad (2.19)$$

$$\theta_2 = \phi_2 + \alpha \quad (2.20)$$

This results in the position vector in robot space defined by Eq. (2.1).

#### 2.4.5. Forward and Inverse Velocity Kinematics

However, both the position and the velocity of each variable are necessary for cluster control. In order to find the velocities, Jacobians are used. The forward Jacobian is a matrix of the partial derivatives of Eq. (2.5) to (2.12) with respect to the robot space variables while the inverse Jacobian is a matrix of the partial derivatives of Eq. (2.13) to (2.20) with respect to the cluster

space variables. The forward and inverse Jacobians are shown symbolically in Eq. (2.21) and (2.22), respectively, and shown in more detail in Appendix A.

$$J = \begin{bmatrix} \frac{\delta x_c}{\delta x_1} & \frac{\delta x_c}{\delta y_1} & \frac{\delta x_c}{\delta z_1} & \frac{\delta x_c}{\delta \theta_1} & \frac{\delta x_c}{\delta x_2} & \frac{\delta x_c}{\delta y_2} & \frac{\delta x_c}{\delta z_2} & \frac{\delta x_c}{\delta \theta_2} \\ \frac{\delta y_c}{\delta x_1} & \frac{\delta y_c}{\delta y_1} & \frac{\delta y_c}{\delta z_1} & \frac{\delta y_c}{\delta \theta_1} & \frac{\delta y_c}{\delta x_2} & \frac{\delta y_c}{\delta y_2} & \frac{\delta y_c}{\delta z_2} & \frac{\delta y_c}{\delta \theta_2} \\ \frac{\delta z_c}{\delta x_1} & \frac{\delta z_c}{\delta y_1} & \frac{\delta z_c}{\delta z_1} & \frac{\delta z_c}{\delta \theta_1} & \frac{\delta z_c}{\delta x_2} & \frac{\delta z_c}{\delta y_2} & \frac{\delta z_c}{\delta z_2} & \frac{\delta z_c}{\delta \theta_2} \\ \frac{\delta \alpha}{\delta x_1} & \frac{\delta \alpha}{\delta y_1} & \frac{\delta \alpha}{\delta z_1} & \frac{\delta \alpha}{\delta \theta_1} & \frac{\delta \alpha}{\delta x_2} & \frac{\delta \alpha}{\delta y_2} & \frac{\delta \alpha}{\delta z_2} & \frac{\delta \alpha}{\delta \theta_2} \\ \frac{\delta \beta}{\delta x_1} & \frac{\delta \beta}{\delta y_1} & \frac{\delta \beta}{\delta z_1} & \frac{\delta \beta}{\delta \theta_1} & \frac{\delta \beta}{\delta x_2} & \frac{\delta \beta}{\delta y_2} & \frac{\delta \beta}{\delta z_2} & \frac{\delta \beta}{\delta \theta_2} \\ \frac{\delta \phi_1}{\delta x_1} & \frac{\delta \phi_1}{\delta y_1} & \frac{\delta \phi_1}{\delta z_1} & \frac{\delta \phi_1}{\delta \theta_1} & \frac{\delta \phi_1}{\delta x_2} & \frac{\delta \phi_1}{\delta y_2} & \frac{\delta \phi_1}{\delta z_2} & \frac{\delta \phi_1}{\delta \theta_2} \\ \frac{\delta \phi_2}{\delta x_1} & \frac{\delta \phi_2}{\delta y_1} & \frac{\delta \phi_2}{\delta z_1} & \frac{\delta \phi_2}{\delta \theta_1} & \frac{\delta \phi_2}{\delta x_2} & \frac{\delta \phi_2}{\delta y_2} & \frac{\delta \phi_2}{\delta z_2} & \frac{\delta \phi_2}{\delta \theta_2} \\ \frac{\delta p}{\delta x_1} & \frac{\delta p}{\delta y_1} & \frac{\delta p}{\delta z_1} & \frac{\delta p}{\delta \theta_1} & \frac{\delta p}{\delta x_2} & \frac{\delta p}{\delta y_2} & \frac{\delta p}{\delta z_2} & \frac{\delta p}{\delta \theta_2} \end{bmatrix} \quad (2.21)$$

$$J^{-1} = \begin{bmatrix} \frac{\delta x_1}{\delta x_c} & \frac{\delta x_1}{\delta y_c} & \frac{\delta x_1}{\delta z_c} & \frac{\delta x_1}{\delta \alpha} & \frac{\delta x_1}{\delta \beta} & \frac{\delta x_1}{\delta \phi_1} & \frac{\delta x_1}{\delta \phi_2} & \frac{\delta x_1}{\delta p} \\ \frac{\delta y_1}{\delta x_c} & \frac{\delta y_1}{\delta y_c} & \frac{\delta y_1}{\delta z_c} & \frac{\delta y_1}{\delta \alpha} & \frac{\delta y_1}{\delta \beta} & \frac{\delta y_1}{\delta \phi_1} & \frac{\delta y_1}{\delta \phi_2} & \frac{\delta y_1}{\delta p} \\ \frac{\delta z_1}{\delta x_c} & \frac{\delta z_1}{\delta y_c} & \frac{\delta z_1}{\delta z_c} & \frac{\delta z_1}{\delta \alpha} & \frac{\delta z_1}{\delta \beta} & \frac{\delta z_1}{\delta \phi_1} & \frac{\delta z_1}{\delta \phi_2} & \frac{\delta z_1}{\delta p} \\ \frac{\delta \theta_1}{\delta x_c} & \frac{\delta \theta_1}{\delta y_c} & \frac{\delta \theta_1}{\delta z_c} & \frac{\delta \theta_1}{\delta \alpha} & \frac{\delta \theta_1}{\delta \beta} & \frac{\delta \theta_1}{\delta \phi_1} & \frac{\delta \theta_1}{\delta \phi_2} & \frac{\delta \theta_1}{\delta p} \\ \frac{\delta x_2}{\delta x_c} & \frac{\delta x_2}{\delta y_c} & \frac{\delta x_2}{\delta z_c} & \frac{\delta x_2}{\delta \alpha} & \frac{\delta x_2}{\delta \beta} & \frac{\delta x_2}{\delta \phi_1} & \frac{\delta x_2}{\delta \phi_2} & \frac{\delta x_2}{\delta p} \\ \frac{\delta y_2}{\delta x_c} & \frac{\delta y_2}{\delta y_c} & \frac{\delta y_2}{\delta z_c} & \frac{\delta y_2}{\delta \alpha} & \frac{\delta y_2}{\delta \beta} & \frac{\delta y_2}{\delta \phi_1} & \frac{\delta y_2}{\delta \phi_2} & \frac{\delta y_2}{\delta p} \\ \frac{\delta z_2}{\delta x_c} & \frac{\delta z_2}{\delta y_c} & \frac{\delta z_2}{\delta z_c} & \frac{\delta z_2}{\delta \alpha} & \frac{\delta z_2}{\delta \beta} & \frac{\delta z_2}{\delta \phi_1} & \frac{\delta z_2}{\delta \phi_2} & \frac{\delta z_2}{\delta p} \\ \frac{\delta \theta_2}{\delta x_c} & \frac{\delta \theta_2}{\delta y_c} & \frac{\delta \theta_2}{\delta z_c} & \frac{\delta \theta_2}{\delta \alpha} & \frac{\delta \theta_2}{\delta \beta} & \frac{\delta \theta_2}{\delta \phi_1} & \frac{\delta \theta_2}{\delta \phi_2} & \frac{\delta \theta_2}{\delta p} \end{bmatrix} \quad (2.22)$$

In order to convert the velocity from robot space to cluster space, the Eq. (2.3) is used. This means that in order to find the velocity in cluster space, both the position and velocity in robot space must be known. Similarly, the equation to convert the velocity from cluster space to robot

space is given in Eq. (2.4). Both the position and velocity in cluster space must be known in order to obtain the velocity in robot space [1].

A singularity is defined in [28] as:

$$|J| = 0 \quad (2.23)$$

The singularities for both the forward and inverse Jacobians were calculated to have the values shown in Table 2.3. These values all mean the same thing: the robots cannot be in the same location at the same time nor can they be on top of one another. Since the first condition is physically impossible and the second is undesirable, these singularities were determined to be acceptable since they would be unlikely to occur in practice.

**Table 2.3: Two robot singularities.**

Jacobian	Singularity	Physical Description
Forward	$x_1 = x_2$ and $y_1 = y_2$ at the same time	The robots cannot be on top of each other.
Forward	$x_1 = x_2$ , $y_1 = y_2$ , and $z_1 = z_2$ at the same time	The robots cannot be co-located.
Inverse	$p = 0$	The robots cannot be co-located.
Inverse	$\beta = \pm \pi/2$	The robots cannot be on top of each other.

## 2.5. Description of a Three-Robot Cluster

Although two robots were used in the majority of the simulations and experiments detailed here, three robots were also used in this dissertation. The following sections provide a detailed description of a three robot cluster and some of the challenges faced by this setup.

### 2.5.1. Robot Space

Robot space is defined the same way for both a two and three robot cluster, with the addition of single robot, and is illustrated in Figure 2.5 using the variables defined in Table 2.4. Each robot

still has its own  $(x, y, z)$  location in global space and its own yaw angle,  $\theta$ , with respect to the global z-axis [1], just as before. This means that robot space is still the ideal frame in which to give robot commands since each robot receives an individual command dependent solely on its own position in space.

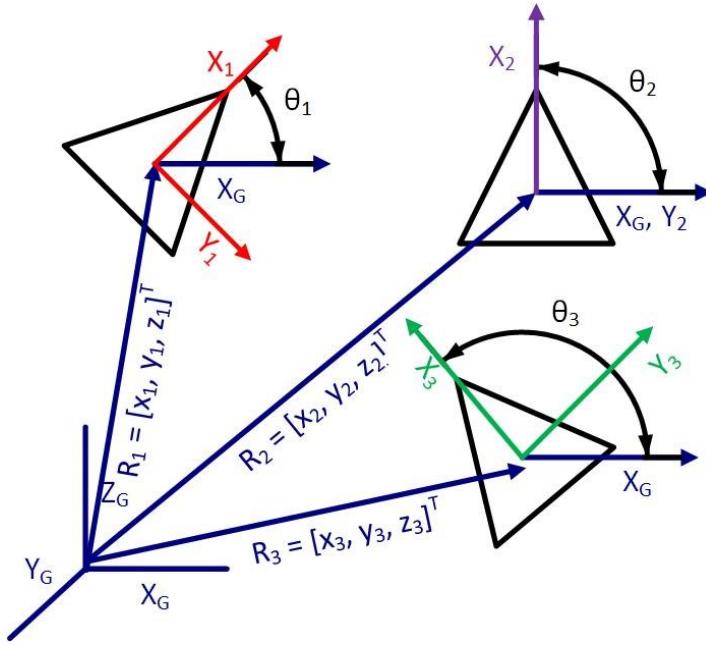


Figure 2.5: Three robots and their coordinates in robot space.

Table 2.4: Description of the variables in robot space for a three robot cluster.

Variable	Description
$x_1$	Distance (in meters) from the first robot to the origin in the x direction.
$y_1$	Distance (in meters) from the first robot to the origin in the y direction.
$z_1$	Distance (in meters) from the first robot to the origin in the z direction.
$\theta_1$	The yaw (in radians) of the first robot about the z-axis.
$x_2$	Distance (in meters) from the second robot to the origin in the x direction.
$y_2$	Distance (in meters) from the second robot to the origin in the y direction.
$z_2$	Distance (in meters) from the second robot to the origin in the z direction.
$\theta_2$	The yaw (in radians) of the second robot about the z-axis.
$x_3$	Distance (in meters) from the third robot to the origin in the x direction.
$y_3$	Distance (in meters) from the third robot to the origin in the y direction.
$z_3$	Distance (in meters) from the third robot to the origin in the z direction.
$\theta_3$	The yaw (in radians) of the third robot about the z-axis.

### 2.5.2. Cluster Space

Cluster space is a bit more complicated for a three robot cluster; the variables are illustrated in Figure 2.6 and Figure 2.7 and defined in Table 2.5. The cluster still has its own ( $x$ ,  $y$ ,  $z$ ) location, yaw (rotation about the  $z$ -axis, designated by  $\alpha$ ), roll (rotation about the  $y$ -axis, designated by  $\beta$ ), and pitch (rotation about the  $x$ -axis, designated by  $\gamma$ ) with respect to the global coordinate frame. Each of the robot positions are then defined with respect to this cluster center by the inverse kinematics found in Section 2.5.4 and a yaw angle,  $\phi_i$ , with respect to the cluster yaw [1]. Cluster space is still the ideal frame in which to give commands to the cluster. For three robots, the amount of work saved due to cluster control is noticeable.

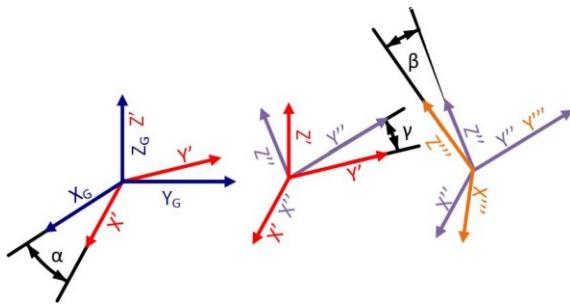


Figure 2.6: Three robots rotation angle definitions.

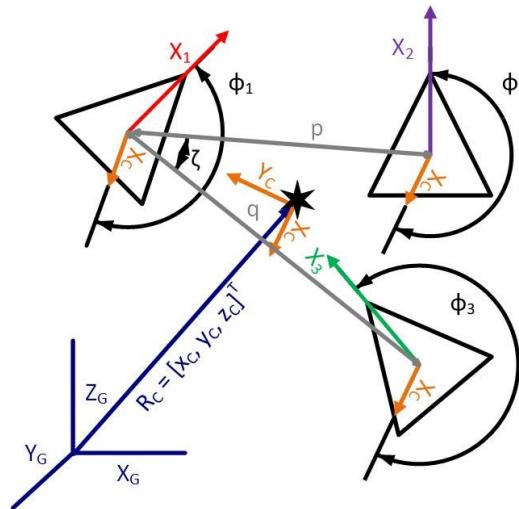


Figure 2.7: Three robots and their coordinates in cluster space.

**Table 2.5: Description of the variables in cluster space for a three robot cluster.**

Variable	Description
x	Distance (in meters) from the cluster center to the origin in the x direction.
y	Distance (in meters) from the cluster center to the origin in the y direction.
z	Distance (in meters) from the cluster center to the origin in the z direction.
$\alpha$	The cluster angle of rotation (in radians) about the z-axis, yaw.
$\beta$	The cluster angle of rotation (in radians) about the y-axis, roll.
$\gamma$	The cluster angle of rotation (in radians) about the x-axis, pitch.
$\varphi_1$	The heading of quadrotor 1 (in radians) with respect to the cluster x-axis.
$\varphi_2$	The heading of quadrotor 2 (in radians) with respect to the cluster x-axis.
$\varphi_3$	The heading of quadrotor 3 (in radians) with respect to the cluster x-axis.
$\zeta$	The angle (in radians) between p and q.
p	Distance (in meters) of the centers of quadrotors 1 and 2.
q	Distance (in meters) of the centers of quadrotors 1 and 3.

### 2.5.3. Forward Position Kinematics

The three robot forward position kinematic equations transform the variables from robot space to cluster space. There are even more ways to do this from a mathematical standpoint than for a two robot system, but the goal is the same: keep the equations as simple as possible and limit the impact of singularities on the controller. The impact of the singularities will be discussed in more detail in Section 2.5.5.

The kinematic equations used for a three robot cluster are found in Eq. (2.24) to (2.35) below and are a modified version of those found in [18]. The input variables in these equations are all variables defined in robot space while the outputs are defined in cluster space. The robot space variables are defined in Table 2.4 while the cluster space variables are defined in Table 2.5.

$$x_c = \frac{1}{3}(x_1 + x_2 + x_3) \quad (2.24)$$

$$y_c = \frac{1}{3}(y_1 + y_2 + y_3) \quad (2.25)$$

$$z_c = \frac{1}{3}(z_1 + z_2 + z_3) \quad (2.26)$$

$$\alpha = atan2(-\hat{y}_c \cdot \hat{x}_G, \hat{y}_c \cdot \hat{y}_G) \quad (2.27)$$

$$\beta = atan2(\hat{y}_c \cdot \hat{z}_G, \sqrt{(\hat{y}_c \cdot \hat{x}_G)^2 + (\hat{y}_c \cdot \hat{y}_G)^2}) \quad (2.28)$$

$$\gamma = atan2(-\hat{x}_c \cdot \hat{z}_G, \hat{z}_c \cdot \hat{z}_G) \quad (2.29)$$

$$\phi_1 = \theta_1 - \alpha \quad (2.30)$$

$$\phi_2 = \theta_2 - \alpha \quad (2.31)$$

$$\phi_3 = \theta_3 - \alpha \quad (2.32)$$

$$p = \sqrt{(x_1 - x_2)^2 + (y_1 - y_2)^2 + (z_1 - z_2)^2} \quad (2.33)$$

$$q = \sqrt{(x_1 - x_3)^2 + (y_1 - y_3)^2 + (z_1 - z_3)^2} \quad (2.34)$$

$$\zeta = \text{acos} \left( \frac{\left( \begin{bmatrix} x_1 - x_2 \\ y_1 - y_2 \\ z_1 - z_2 \end{bmatrix} \cdot \begin{bmatrix} x_1 - x_3 \\ y_1 - y_3 \\ z_1 - z_3 \end{bmatrix} \right)}{pq} \right) \quad (2.35)$$

This results in the following position vector in cluster space:

$$C = [x_c \ y_c \ z_c \ \alpha \ \beta \ \gamma \ \phi_1 \ \phi_2 \ \phi_3 \ p \ q \ \zeta]^T \quad (2.36)$$

#### 2.5.4. Inverse Position Kinematics

The inverse position kinematics that are required to transform the cluster space variables into robot space variables are defined in Eq. (2.37) to (2.59). Since there are multiple ways to do this from a mathematical standpoint, the equations should be simple and the singularities should

have a minimal impact on the controller. These equations, a modified version of the inverse position kinematics found in [18], feature cluster space input variables and robot space output variables. Both sets of variables are defined as in the previous section.

$$B = \sqrt{(q + p \cos \zeta)^2 + (p \sin \zeta)^2} \quad (2.37)$$

$$e = \sqrt{q^2 + p^2 - 2pq \cos \zeta} \quad (2.38)$$

$$\text{angle2} = \text{acos} \left( \frac{p^2 + e^2 - q^2}{2pe} \right) \quad (2.39)$$

$$\text{angle3} = \text{acos} \left( \frac{q^2 + e^2 - p^2}{2qe} \right) \quad (2.40)$$

$$a = p \cos(\text{angle2}) \quad (2.41)$$

$$b = p \sin(\text{angle2}) \quad (2.42)$$

$$c = q \cos(\text{angle3}) \quad (2.43)$$

$$d = \frac{e}{2} - a \quad (2.44)$$

$$\text{zeta1} = \text{atan2}(a, b) \quad (2.45)$$

$$\text{zeta2} = \text{atan2}(d, b) \quad (2.46)$$

$$\text{zeta3} = \text{atan2}(c, b) - \text{zeta2} \quad (2.47)$$

$$x_1 = x_c - \frac{1}{3}B \sin \alpha \cos \beta \quad (2.48)$$

$$y_1 = y_c + \frac{1}{3}B \cos \alpha \cos \beta \quad (2.49)$$

$$z_1 = z_c + \frac{1}{3}B \sin \beta \quad (2.50)$$

$$\theta_1 = \phi_1 + \alpha \quad (2.51)$$

$$x_2 = x_c + p \sin(zeta1 + zeta2) (\cos \alpha \cos \gamma - \sin \alpha \sin \beta \sin \gamma) \\ - \sin \alpha \cos \beta \left( \frac{1}{3}B - p \cos(zeta1 + zeta2) \right) \quad (2.52)$$

$$y_2 = y_c + p \sin(zeta1 + zeta2) (\sin \alpha \cos \gamma + \cos \alpha \sin \beta \sin \gamma) \\ + \cos \alpha \cos \beta \left( \frac{1}{3}B - p \cos(zeta1 + zeta2) \right) \quad (2.53)$$

$$z_2 = z_c - p \sin(zeta1 + zeta2) \cos \beta \sin \gamma \\ + \sin \beta \left( \frac{1}{3}B - p \cos(zeta1 + zeta2) \right) \quad (2.54)$$

$$\theta_2 = \phi_2 + \alpha \quad (2.55)$$

$$x_3 = x_c - q \sin(zeta3) (\cos \alpha \cos \gamma - \sin \alpha \sin \beta \sin \gamma) \\ - \sin \alpha \cos \beta \left( \frac{1}{3}B - q \cos(zeta3) \right) \quad (2.56)$$

$$y_3 = y_c - q \sin(zeta3) (\sin \alpha \cos \gamma + \cos \alpha \sin \beta \sin \gamma) \\ + \cos \alpha \cos \beta \left( \frac{1}{3}B - q \cos(zeta3) \right) \quad (2.57)$$

$$z_3 = z_c + q \sin(zeta3) \cos \beta \sin \gamma + \sin \beta \left( \frac{1}{3}B - q \cos(zeta3) \right) \quad (2.58)$$

$$\theta_3 = \phi_3 + \alpha \quad (2.59)$$

This results in the following position vector in robot space:

$$R = [x_1 \ y_1 \ z_1 \ \theta_1 \ x_2 \ y_2 \ z_2 \ \theta_2 \ x_3 \ y_3 \ z_3 \ \theta_3]^T \quad (2.60)$$

### 2.5.5. Forward and Inverse Velocity Kinematics

In order to find the velocities necessary for cluster control, the Jacobians, both forward and inverse, are used. The forward Jacobian is a matrix of the partial derivatives of Eq. (2.24) to (2.35) with respect to the robot space variables while the inverse Jacobian is a matrix of the partial derivatives of Eq. (2.37) to (2.59) with respect to the cluster space variables. The forward and inverse Jacobians are shown symbolically in Eq. (2.61) and (2.62), respectively, and explained in more detail in a supplement, see reference [29], due to their length.



In order to convert the velocity from robot space to cluster space, Eq. (2.3) is used, as in the two robot case. Likewise, in order to convert the velocity from cluster space to robot space, Eq. (2.4) is used [1].

The singularities for both the forward and inverse three robot Jacobians were calculated to have the values shown in Table 2.6. The first row states that the robots cannot all be in the same place at the same time while the second row states that Robots 1 and 2 cannot be in the same place at the same time, both of which are physically impossible. The next two rows state that the robots cannot form a line, which is not a desired configuration. The final row states that the cluster cannot be in a vertical plane, rotated  $\pi$  radians about the global z-axis. In all of the experiments discussed here,  $\beta$  was set to 0 radians. Thus, these singularities were deemed acceptable since they were either physically impossible or not part of a desired configuration.

**Table 2.6: Three robot singularities.**

<b>Jacobian</b>	<b>Singularity</b>	<b>Physical Description</b>
Forward	$x_1 = x_2 = x_3, y_1 = y_2 = y_3$ , and $z_1 = z_2 = z_3$ at the same time	The robots cannot be co-located.
Inverse	$p = 0$	Robot 1 and Robot 2 cannot be co-located.
Inverse	$\zeta = 0$	The robots cannot form a line with Robot 1 on one end.
Inverse	$\zeta = \pm \pi$	The robots cannot form a line with Robot 1 in the middle.
Inverse	$\alpha = \pm \pi$ and $\beta = \pi/2$ at the same time	The cluster cannot be on its edge and rotated $\pi$ radians about the global z-axis.

# **Chapter 3**

---

## **3. Geometrical Optimization**

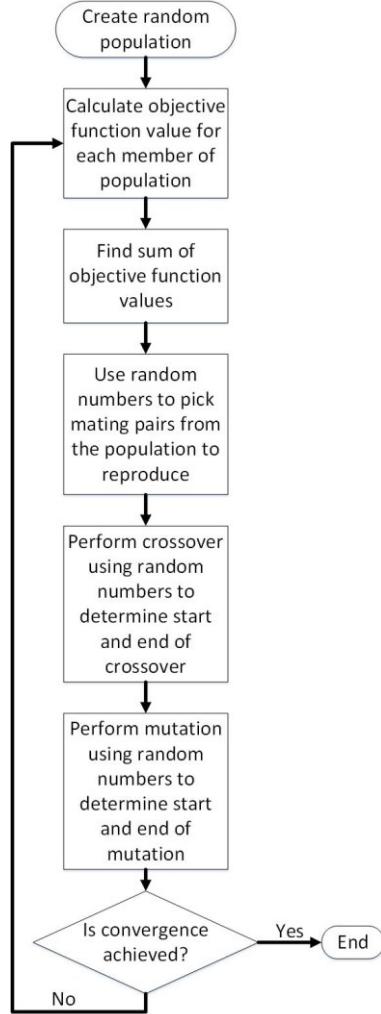
One of the main goals of this research is to find the optimal sensor configuration for a variety of sensor options. The sensor options examined in this chapter include: two identical sensors at a fixed radius from the tracked object, two different sensors at a fixed radius from the tracked object, three identical sensors at a fixed radius from the tracked object, and three different sensors at a fixed radius from the tracked object. Each of these sensor options share the same mathematical basis, which is described in this chapter.

### **3.1. Method Selection**

There are a variety of optimization algorithms found in the literature. However, not every optimization method is right for every application. This section provides a brief overview of some of the optimization methods that were considered for this dissertation as well as the reasoning for why each method was rejected for this application. The method that was selected for this work is also described, though a detailed description is left for Section 3.9.

Evolutionary algorithms are popular optimization methods currently used in a wide variety of applications. One of the more popular types of evolutionary algorithms is the genetic algorithm (GA). The GA is a discrete variable, random search method that does not require the use of gradients [30]. The lack of required gradients makes this method attractive when the governing equations are not known or are very complex. Additionally, the GA can deal with non-continuous functions since it only uses function values rather than the functions themselves [30], adding to its popularity. In this method, the inputs to the cost function are represented by binary strings which are then combined into a single string to create a single candidate solution

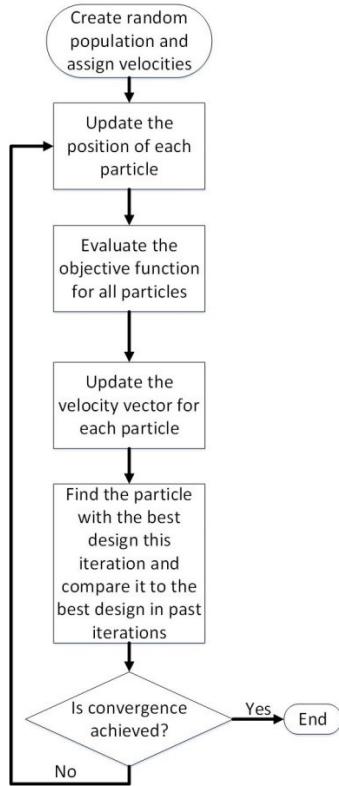
[30]. A basic flowchart of GA is shown in Figure 3.1. However, this method requires a large number of function evaluations and, thus, can quickly become computationally expensive [30].



**Figure 3.1: Genetic algorithm flowchart.**

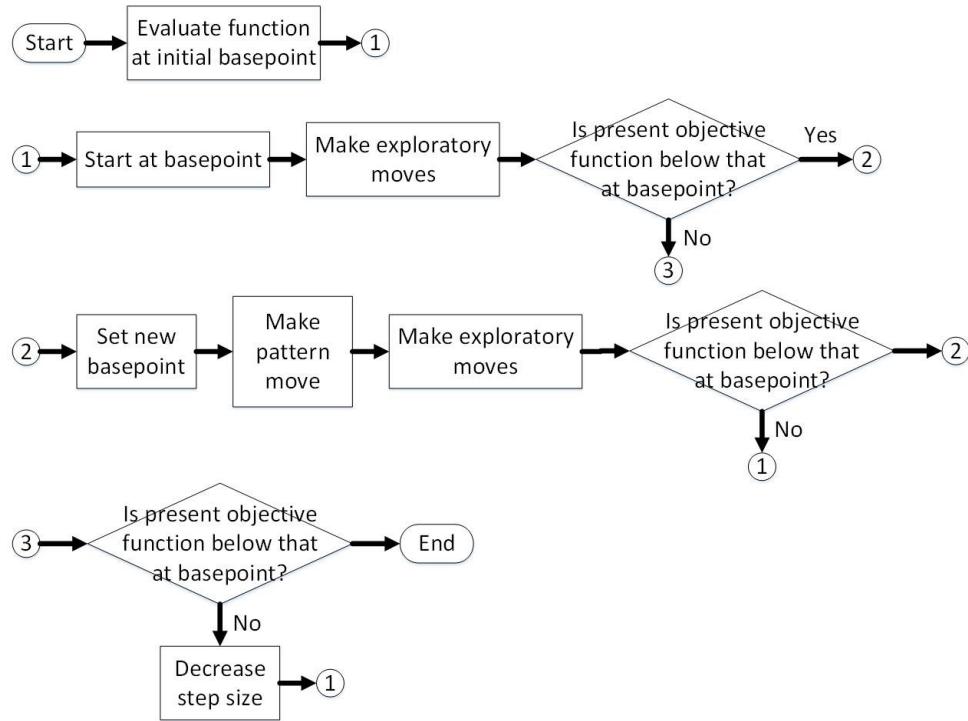
Another popular evolutionary algorithm is particle swarm, which is similar to the motion of a flock of birds. This method is a continuous variable method that has been modified to use discrete variables and begins with a population of potential solutions, called particles [31], and has often been used to train neural networks since its development in 1995 [32]. Each particle is

also assigned a random velocity, which is used to update the particles' position each time step [31]. A flowchart of this algorithm can be seen in Figure 3.2.



**Figure 3.2: Particle swarm algorithm flowchart.**

There are three parameters in particle swarm optimization that are problem dependent: the inertia parameter and the two trust parameters. The inertia parameter determines how aggressive the search will be, with higher values covering a wider search area than lower values. The trust parameters determine how much confidence the particle has in itself and in the swarm for the first and second parameters, respectively [31]. The higher the value, the more similar the next solution will be to the current solution. Although this method uses function values only, it also requires a very high number of function evaluations [31] and, thus, is also too computationally intensive for this application.



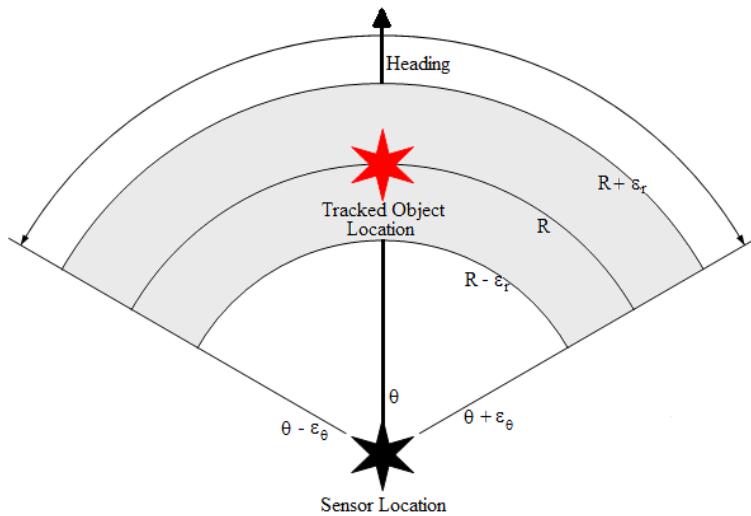
**Figure 3.3: Hooke and Jeeves method flowchart adapted from [33].**

The Hooke and Jeeves method is a pattern search method that does not require function derivatives in order to find an optimal solution [33]. For the equations developed in this dissertation, finding the derivatives is a non-trivial problem so it is preferred to use a method that does not require their derivation. The Hooke and Jeeves method has the added benefit that its computation time scales linearly with the number of inputs [33], the space complexity is  $O(n)$ , so the number of robots can easily be increased, making this method a good choice for use in this dissertation. This method was chosen over other heuristic methods such as the Nelder-Mead Simplex method and the Rosenbrock pattern search [34] because it was easy to find an already implemented Hooke and Jeeves method in Matlab and easy to modify for constraints. Further, both the Nelder-Mead Simplex and the Rosenbrock pattern search methods have a complexity space of  $O(n^2)$  [34], making it more difficult to increase the number of robots. A flow

chart of the Hooke and Jeeves algorithm is shown in Figure 3.3 and will be discussed further in Section 3.9.

### 3.2. Problem Setup

The methodology presented here involves fusing the sensor measurements from multiple mobile sensor systems to obtain a more accurate position estimate than is achievable by the individual sensor systems. The angle of separation between sensors is optimized to find the best fused sensor system estimate given the position constraints on the mobile sensor systems. In order to achieve this optimization, the sensor properties themselves must be modeled mathematically.



**Figure 3.4: Terminology used to define the portion of a circle arc that describes the area of a sensor's valid sensor coverage area.**

It is assumed that a sensor will not necessarily report the exact position of an object, but will instead report the position with a certain degree of error. The area in which the object's position may be reported is described by a portion of a circle arc, as shown in Figure 3.4, known as the valid sensor coverage area. The position of the object from the sensor has a mean radial error of

$\varepsilon_r$  and a mean heading error of  $\varepsilon_\theta$ . The distances and angles are measured with respect to the sensor and the radial and heading errors are assumed to have a normal distribution about their given means. Thus, if the tracked object is at an angle of  $\theta$  degrees and a distance of  $R$  m from the sensor, the position of the object could be reported anywhere within the circle arc described in Figure 3.4. The distances and angles are measured with respect to the sensor. Thus, the inner radius of the circle arc is  $R - \varepsilon_r$  and the outer radius of the circle arc is  $R + \varepsilon_r$ . The line of symmetry of the circle arc is the same as the sensor heading,  $\theta$ . This means that the circle arc is bounded between  $\theta - \varepsilon_\theta$  and  $\theta + \varepsilon_\theta$  degrees.

For example, let a sensor have a heading of 90 degrees and a radius from the tracked object of 5 meters. The sensor has an  $\varepsilon_r$  of 1 meter and an  $\varepsilon_\theta$  of 10 degrees. Thus, the circle arc radius varies from 4 to 6 meters with an angle of 80 to 100 degrees. This sensor's valid sensor coverage area is shown in Figure 3.5.

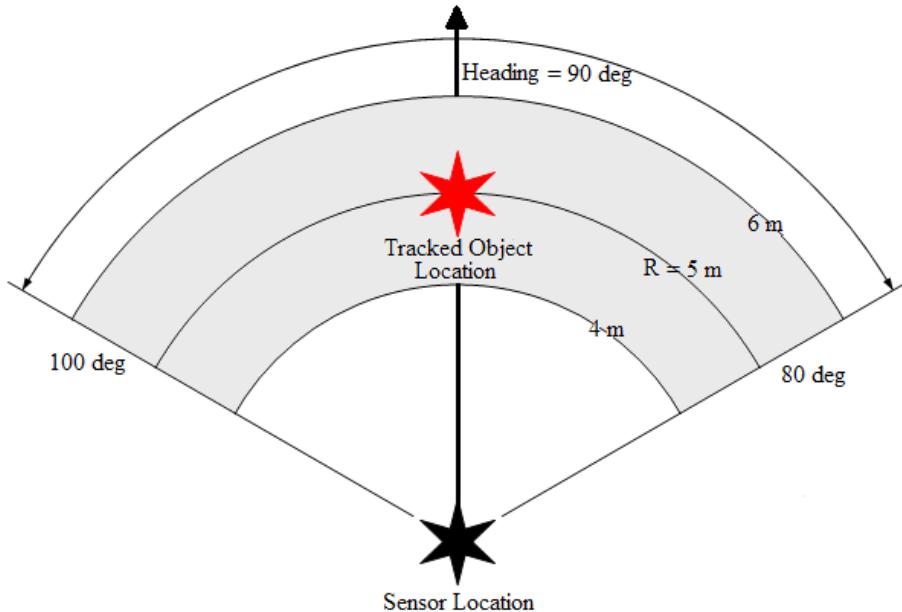


Figure 3.5: Example sensor error area.

### 3.2.1. Sensor Constraints and their Covariance Matrices

Once the sensor parameters and constraints are known, the corresponding covariance matrix can be calculated using Eq. (3.1) below with the variables defined in Table 3.1 [34].

$$cov(x, y) = \frac{1}{n-1} \sum_{i=1}^n \begin{bmatrix} m_{xx} & m_{xy} \\ m_{xy} & m_{yy} \end{bmatrix} \quad (3.1)$$

**Table 3.1: Variable definitions for Eq. (3.1).**

Variable	Definition
$m_{xx}$	$(r_i \cos t_i + x - \mu_x)(r_i \cos t_i + x - \mu_x)$
$m_{yy}$	$(r_i \sin t_i + y - \mu_y)(r_i \sin t_i + y - \mu_y)$
$m_{xy}$	$(r_i \cos t_i + x - \mu_x)(r_i \sin t_i + y - \mu_y)$
$r_i$	Radius of the arc at point $i$
$t_i$	Angle of point $i$ with respect to the sensor
$x$	X position of the sensor
$y$	Y position of the sensor
$\mu_x$	Mean x value
$\mu_y$	Mean y value

This results in the following error covariance matrix where  $\sigma$  is the standard deviation:

$$cov(x, y) = \begin{bmatrix} \sigma_x^2 & \sigma_{xy} \\ \sigma_{xy} & \sigma_y^2 \end{bmatrix} \quad (3.2)$$

This covariance matrix is then used to define the corresponding error ellipse as described in the next section.

An example of a covariance matrix calculation uses the sensor defined in the previous section. Here,  $x$  is 3 m,  $y$  is 1 m,  $\mu_x$  is 3 m,  $\mu_y$  is 6 m,  $r_i$  ranges from 4 to 6 m, and  $t_i$  ranges from 80 to 100 degrees. Substitute these values into Eq. (3.1) to obtain the following covariance matrix:

$$cov(x, y) = \begin{bmatrix} 0.2613 & 0 \\ 0 & 0.3636 \end{bmatrix} \quad (3.3)$$

### 3.2.2. Sensor Error Ellipses

The semi-major and semi-minor axes of the error ellipses for each sensor are derived from the eigenvalues of their covariance matrices as shown in the following equations, adapted from Schubert and Kirchner [36]:

$$a = \sqrt{\chi_2^2 \cdot \lambda_1} \quad (3.4)$$

$$b = \sqrt{\chi_2^2 \cdot \lambda_2} \quad (3.5)$$

Here,  $a$  is the semi-major axis of the error ellipse,  $b$  is the semi-minor axis of the error ellipse,  $\lambda_i$  are the eigenvalues of the covariance matrix found in Eq. (3.2), and  $\chi_2^2$  is the chi-squared distribution with two degrees of freedom. In the work presented here, a 60% probability distribution was desired so the corresponding chi-squared value of 1.833 was found in [37]. The angle of rotation for the error ellipse was used solely for visualization purposes and was also determined from the covariance matrix. It is described by the following equation found in [38]:

$$\omega = \frac{1}{2} \tan^{-1} \left( \frac{2\sigma_{xy}}{\sigma_y^2 - \sigma_x^2} \right) \quad (3.6)$$

This angle is the counterclockwise rotation angle of the error ellipse with respect to the y-axis.

For example, using the ellipse from the previous sections, we know that the covariance matrix is given in Eq. (3.3). The eigenvalues of the covariance matrix are found by solving for  $\lambda$  in the following equation [39]:

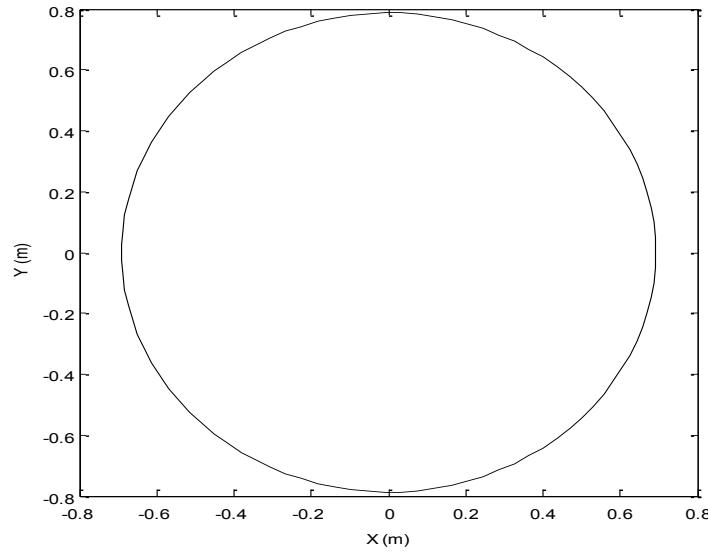
$$|cov(x, y) - \lambda \times I_{2 \times 2}| = 0 \quad (3.7)$$

This results in  $\lambda_1 = 0.2609$  and  $\lambda_2 = 0.3378$ . Thus, the semi-major and semi-minor axes are defined as follows:

$$\text{semi-minor: } x = \sqrt{1.833 \cdot 0.2609} = 0.6915 \quad (3.8)$$

$$\text{semi-major: } y = \sqrt{1.833 \cdot 0.3378} = 0.7869 \quad (3.9)$$

Next, substitute the values of the covariance matrix from Eq. (3.3) into Eq. (3.6) to obtain an angle of rotation of 0 degrees. The resultant covariance error ellipse is shown in Figure 3.6.



**Figure 3.6: Error ellipse of the example sensor.**

### 3.3. Combining Error Ellipses

Once the error ellipse for each sensor in the experiment was determined, the next step was to find a combined error ellipse for all sensors in the system. [38] showed that the combined error ellipse can be found as follows:

$$cov_{comb}(x, y) = \left( \sum_{i=1}^n (cov_i(x, y))^{-1} \right)^{-1} \quad (3.10)$$

The semi-major and semi-minor axes for this combined ellipse were then found using Eq. (3.4) and (3.5) as for the individual matrices. In all cases, the error ellipses were assumed to be centered on the tracked object.

Now, assume two sensors with the same error characteristics as in the examples from the previous sections. The tracked object is located at (0, 0). Sensor 1 has a heading of 135 degrees and is located at (3.5355, -3.5355) while Sensor 2 has a heading of 45 degrees and is located at (-3.5355, -3.5355). This results in the following covariance matrices:

$$cov_1(x, y) = \begin{bmatrix} 0.3124 & -0.0512 \\ -0.0512 & 0.3124 \end{bmatrix} \quad (3.11)$$

$$cov_2(x, y) = \begin{bmatrix} 0.3124 & 0.0512 \\ 0.0512 & 0.3124 \end{bmatrix} \quad (3.12)$$

From these covariance matrices, the semi-major and minor axes are found as described in Eq. (3.4) and (3.5) while the rotation angles are found as described in Eq. (3.6). The results are shown in the table below.

**Table 3.2: Axes for combining error ellipses example.**

	<b>Sensor 1</b>	<b>Sensor 2</b>
Semi-Minor Axis (m)	x = 0.6919	x = 0.6919
Semi-Major Axis (m)	y = 0.8164	y = 0.8164
Rotation Angle (rad)	$\theta = -0.7854$	$\theta = 0.7854$

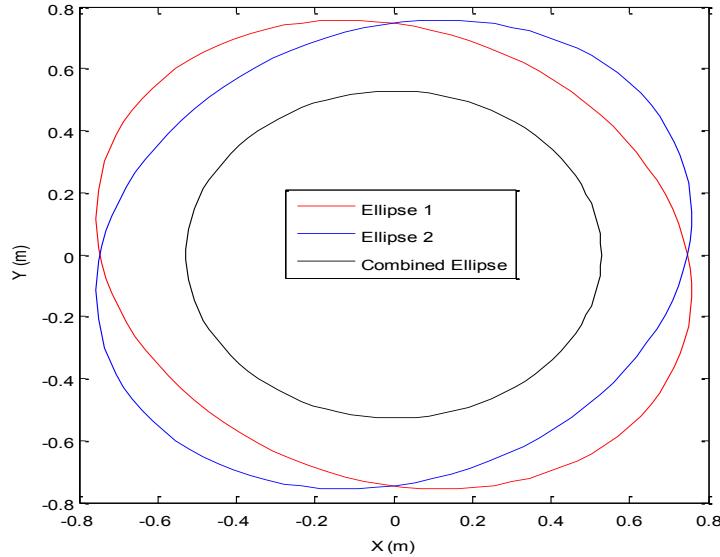
Next, the covariance matrix of the combined error ellipse is found using Eq. (3.10). The result is shown below:

$$cov_{comb}(x, y) = \begin{bmatrix} 0.1520 & 0 \\ 0 & 0.1520 \end{bmatrix} \quad (3.13)$$

The combined ellipse is also centered at the location of the tracked object. The semi-minor and semi-major axes and rotation angle for the combined error ellipse is found in the same manner as for each of the individual error ellipses. The results are shown in Table 3.3. A plot of both the error ellipses and the combined error ellipse is shown in Figure 3.7.

**Table 3.3: Axes for the combining error ellipses example.**

	<b>Combined Ellipse</b>
Semi-Minor Axis (m)	$x = 0.5278$
Semi-Major Axis (m)	$y = 0.5278$
Rotation Angle (rad)	$\theta = 0$



**Figure 3.7: Two error ellipses and their combined error ellipse.**

### 3.4. Configuration Optimization

Finally, the optimal geometric tracking configuration is found by minimizing the area of the combined ellipse, found by Eq. (3.14).

$$Area = \pi ab \quad (3.14)$$

Here,  $a$  is the semi-major axis and  $b$  is the semi-minor axis [36]. In the example carried out in this chapter, the area of the combined error ellipse is found to be  $0.8752 \text{ m}^2$  using Eq. (3.14).

Physically, the smaller the area for the combined error ellipse, the closer the estimated location of the tracked object will be to the actual location. A closed-form expression for the area, with inputs of radii from the target, the tracking stations' headings, the radial errors, and the angular errors will be discussed in Sections 3.6 through 3.8.

### 3.5. Theoretical Fixed Radius Curves

For the work presented in this dissertation, the target and the mobile sensor systems were constrained to move at a maximum speed of  $0.315 \text{ m/s}$ . Each time step in the following simulations and experiments was  $0.125 \text{ s}$  long so a maximum distance of less than  $0.04 \text{ m}$  could occur in each time step. Since this distance was on par with the error in the Ultrawide Band system of  $(x, y, z) + (\pm 0.05, \pm 0.07, \pm 0.39) \text{ m}$  used to provide the robot locations, the simplifying assumption was made that the optimal sensor system configuration could be determined as a static configuration for each time step with minimal loss of accuracy. An additional simplifying assumption was made that the cluster center and the tracked object were in the same plane. In reality, the mobile sensor systems operated at a slightly higher altitude than the tracked object, but since the mean plane inclination angle was only  $2.9$  degrees, this was also considered to be a valid assumption.

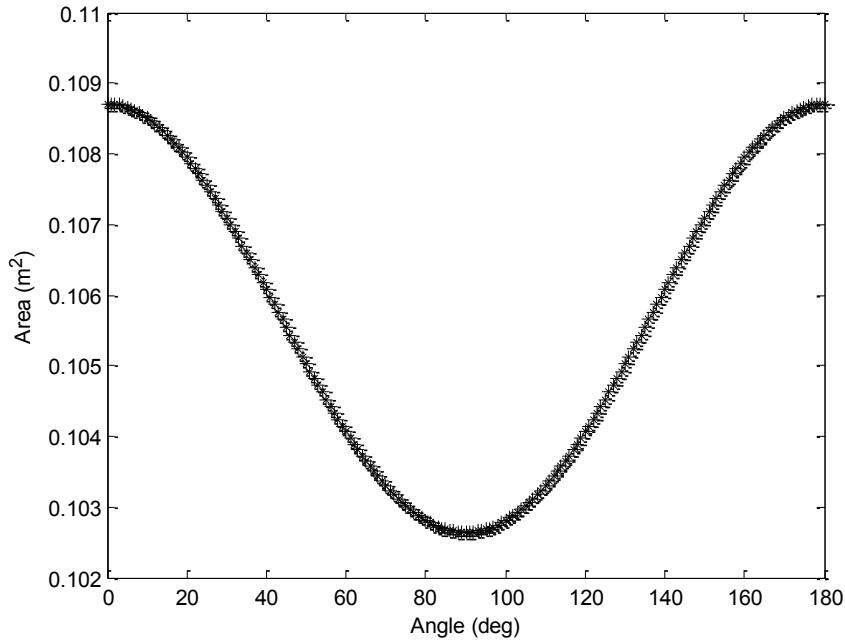
To test the optimization theory for fidelity and accuracy, two test cases were used; both were constrained to a fixed radius of  $2.83 \text{ m}$  from the tracked object. This distance was chosen because the optimal viewing distance for the quadrotors used in the physical experiments was

between 1.7 m and 3.3 m. A viewing distance of 2.83 m was within the optimal viewing distance and allowed the quadrotors to be placed 2 m away from the target and 4 m apart. Case 1 featured two identical sensor systems while Case 2 featured one sensor system with a small angular error but a large radial error and one sensor system with a large angular error but a small radial error. Both cases had been examined from a geometric perspective and Case 1 had been explored experimentally in work at Santa Clara University [40]. The smallest estimation error was found from a geometric perspective by finding the angle of separation between the sensor systems which resulted in the smallest area of overlapping valid sensor coverage areas when both sensors were pointed at the same target. It was found that Case 1 had an optimal sensor system separation of  $\pm 90$  degrees, as predicted by geometric considerations and [40]. The angle of separation has the same general configuration at 90 degrees and -90 degrees; however, the positions of the individual sensor systems are reversed. Mathematically, these configurations are identical. This optimal configuration also matched that used by researchers in [9]. Case 2 was found to have an optimal separation angle of  $\pm 180$  degrees as predicted by geometric considerations. Again, these two configurations are mathematically identical. This constitutes sufficient validation to test the theory both in simulation and physical experiments.

Note that a fixed radius was used for both tests because it allowed for the examination of the angle of separation between sensor systems without additional effects from changing multiple variables. A radius of 2.83 m was chosen for this initial test because it was both a distance that was practical for later experimental work at Santa Clara University and showed clear differentiation in the results at each separation angle. The examination of the effects of changing multiple parameters will be examined in future work.

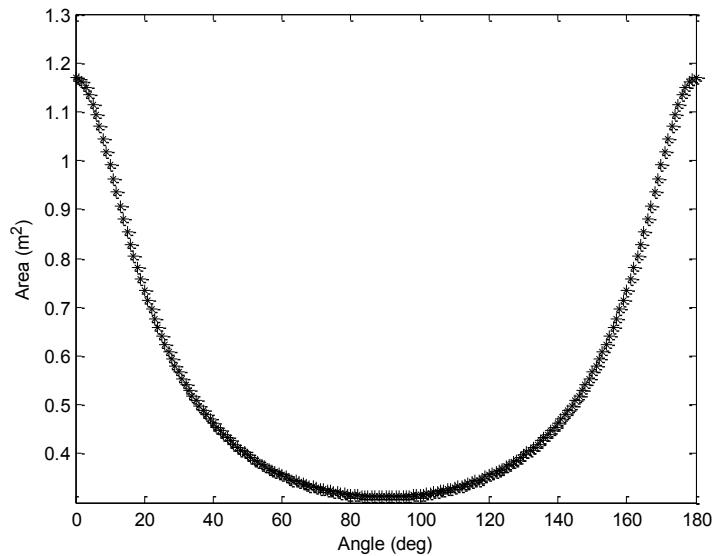
### 3.5.1. Mathematical Simulation of Two Tracking Stations at a Fixed Radius with Identical Sensor Systems

The equations developed in this chapter were used to mathematically simulate two tracking stations at a fixed radius of 2.83 m from the tracked object. The sensors for each tracking station were identical, as in Case 1, and were given an angular error,  $\varepsilon_\theta$ , of 5.7 degrees and a radial error,  $\varepsilon_r$ , of 0.4 m to match the quadrotor sensor parameters. In the simulation, the tracking stations were separated by 0 through 180 degrees and the resulting combined error ellipse area was found at each point. The curve formed by the area of the combined error ellipse at each angle of separation, shown in Figure 3.8, was found to have an ideal angle of separation of 90 degrees, as expected from [40] and Case 1.



**Figure 3.8: Mathematical simulation of two sensors with a fixed radius of 2.83 m and identical sensors. The angle along the x-axis is the angle of separation between the two sensors.**

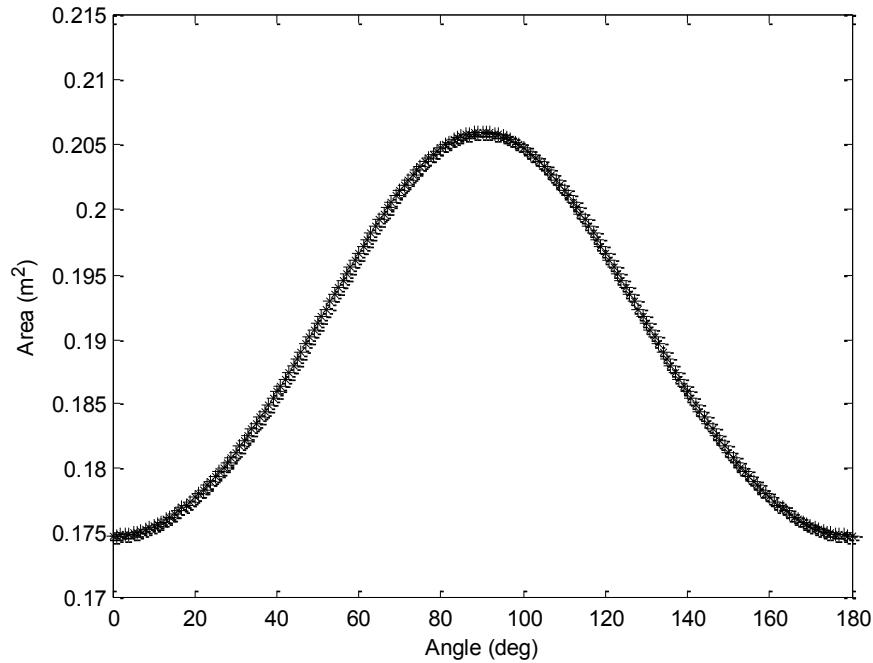
This simulation was then repeated at a fixed radius of 30 m to test whether the same ideal angle of separation would be found at a much greater radius from the tracked object. The sensor error parameters remained the same: an angular error of 5.7 degrees and a radial error of 0.4 m. Again, the angle of separation was varied from 0 to 180 degrees and the combined ellipse area was found for each angle. The resulting curve can be seen in Figure 3.9 and illustrates the same ideal angle of separation of 90 degrees. There are two notable differences between this curve and that shown in Figure 3.8. First, the combined ellipses have much greater area in Figure 3.9, as expected due to the much larger valid sensor coverage areas at greater distances. Secondly, the curve is much more rounded at greater distances and has a more nearly flat bottom. This is also expected since the valid sensor coverage areas are much wider than they are long so their overlapping coverage areas are very similar between 60 and 120 degrees. However, this curve illustrated that the ideal angle of separation between two identical sensor systems is 90 degrees for a variety of ranges.



**Figure 3.9: Mathematical simulation of two sensors with a fixed radius of 30 m and identical sensors. The angle along the x-axis is the angle of separation between the two sensors.**

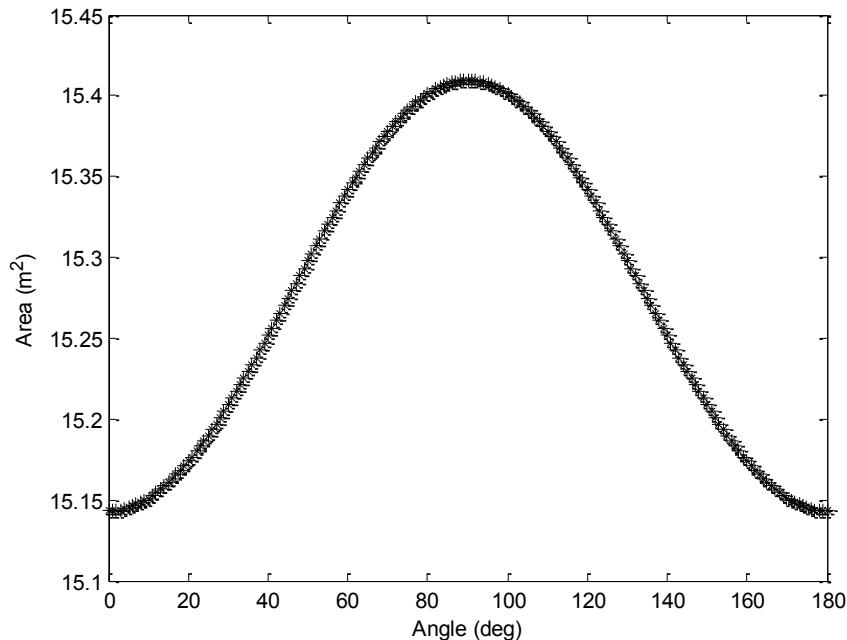
### *3.5.2. Mathematical Simulation of Two Tracking Stations at a Fixed Radius with Different Sensor Systems*

The equations presented in Sections 3.2 through 3.4 were also used to mathematically simulate the effect of two tracking stations with different sensor parameters at a fixed radius of 2.83 m. Sensor 1 was assigned an angular error of 5.7 degrees and a radial error of 0.8 m while Sensor 2 was given an angular error of 10.1 radians and a radial error of 0.4 m. Again, the tracking stations were separated by 0 to 180 degrees with the resulting combined error ellipse area calculated at each point. Figure 3.10 shows the resultant curve, which was found to have an ideal angle of separation at 0 degrees and 180 degrees. A separation angle of 0 degrees is physically impossible since the sensor systems cannot be collocated, but a separation of 180 degrees represents the same configuration obtained from Case 2 as described above.



**Figure 3.10:** Two sensors with a fixed radius of 2.83 m and different sensors properties. The angle along the x-axis is the angle of separation between the two sensors.

Two tracking stations at a fixed radius of 30 m with different sensors were also mathematically simulated in order to verify that the same ideal angle of separation was valid. Sensor 1 was assigned an angular error of 5.7 degrees and a radial error of 3.2 m while Sensor 2 was assigned an angular error of 0.8 radians and a radial error of 0.4 m. These values were large enough that a greater magnitude change was necessary in order to create the significantly different sensors assumed in this scenario. The resulting curve can be seen in Figure 3.11. This is the same shape as seen in Figure 3.10 with the same ideal angles of separation of 0 degrees and 180 degrees. The only difference is that the area of the combined covariance ellipse is greater in magnitude at a distance of 30 m. This is expected due to the larger size of the valid sensor coverage areas themselves and the larger magnitude of the sensor errors. This simulation again confirms that this methodology applies to variety of sensor ranges.



**Figure 3.11:** Two sensors with a fixed radius of 30 m and different sensors properties. The angle along the x-axis is the angle of separation between the two sensors.

### 3.5.3. Mathematical Simulation of Three Tracking Stations at a Fixed Radius with Identical Sensor Systems

The equations in Sections 3.2 through 3.4 were also used to mathematically simulate three tracking stations at a fixed radius of 2.83 m from the tracked object. The sensor parameters were matched to the quadrotor sensor parameters: the angular error was 5.7 degrees and the radial error was 0.4 m for each sensor. In all cases with three tracking stations, the first tracking station was placed directly in front of and facing the tracked object. The remaining two tracking stations were positioned symmetrically on either side of the first tracking station. The angle of separation was defined as the angle between the second and third tracking stations, as shown in Figure 3.12. In this simulation, the combined error ellipse area was found for angles of separation between 0 and 360 degrees. The ideal angle of separation was found to be 120 degrees or 240 degrees, which both have the same effective angle of separation between Sensors 2 and 3, although an angular separation of 120 degrees is easier to use in practice. The results of the mathematical simulation are shown in Figure 3.13.

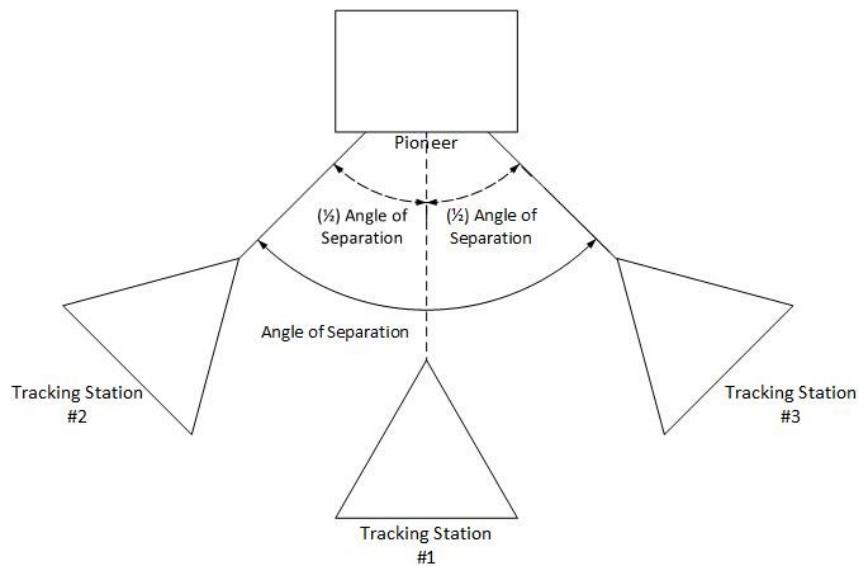
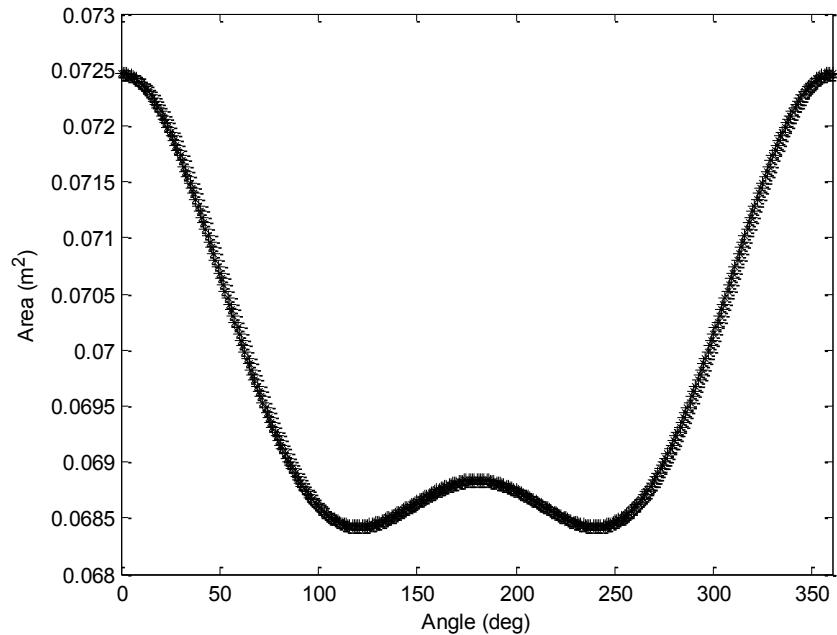


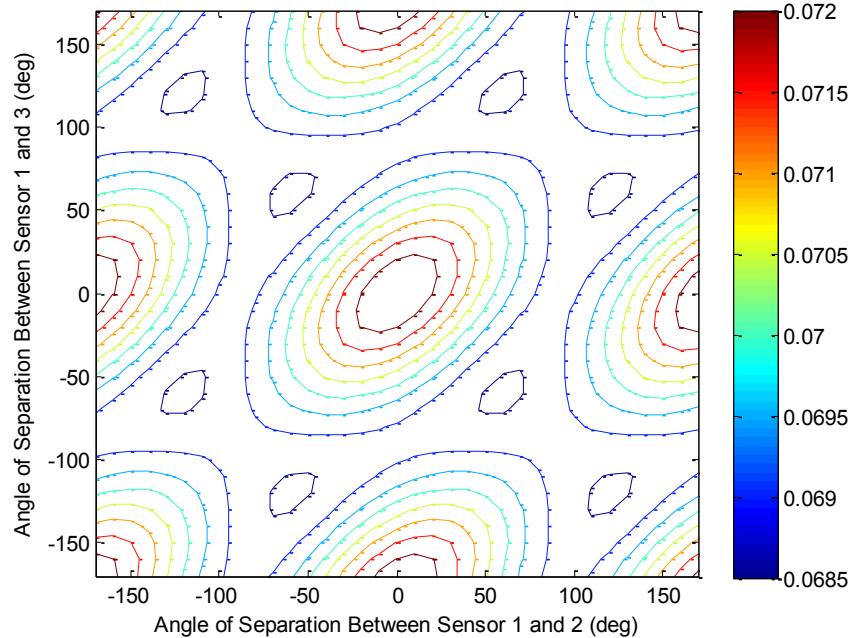
Figure 3.12: Definition of the angle of separation for three mobile sensor stations.



**Figure 3.13: Three sensors with a fixed radius of 2.83 m and identical sensor properties. The angle along the x-axis is the angle of separation between the two outer sensors.**

A variation of this simulation was also performed at a fixed radius of 2.83 m where each sensor system angle was varied separately, allowing for asymmetric angles of separation. This produced the contour plot shown in Figure 3.14. Here, there were two angles of separation: the angle between Sensor 2 and the static Sensor 1 and the angle between Sensor 3 and the static Sensor 1. These angles of separation were independent of one another. The lowest area of the combined ellipse occurred in eight places, marked by the smallest blue circles in Figure 3.14. These areas corresponded to the following angle of separation couplets that represent (Sensor 2, Sensor 3) in degrees: (240, -240), (-240, 240), (120, -120), (-120, 120), (120, 240), (-120, -240), (240, 120), and (-240, -120). Each of these combinations yielded an angle of separation between Sensors 2 and 3, as defined in Figure 3.12, of  $\pm 240$  degrees and represent the same effective

geometric configuration. This confirms that the angle of separation was the global ideal and not simply an artifact of the definition of the angle of separation.

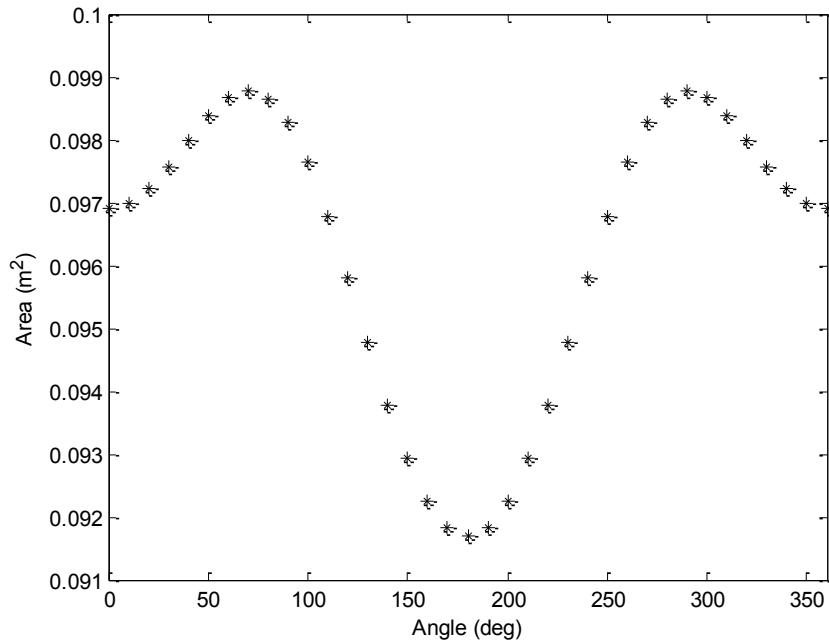


**Figure 3.14:** Mathematical simulation results of three tracking stations with the same sensors at a fixed radius of 2.83 m from the tracked object. This plot shows the area of the combined error ellipses as a function of the angle of separation between the three mobile tracking stations where each angle of separation is varied separately.

### 3.5.4. Mathematical Simulation of Three Tracking Stations at a Fixed Radius with Different Sensor Systems

Finally, the equations presented in Sections 3.2 through 3.4 were used to mathematically simulate three tracking stations with different sensor systems at a fixed radius of 2.83 m from the tracked object. The angle of separation was again defined as in Figure 3.12 and only symmetric configurations were examined. Here, Sensor 1 was given an angular error of 5.7 degrees and a radial error of 0.4 m, Sensor 2 was given an angular error of 5.7 degrees and a radial error of 0.8 m, and Sensor 3 was given an angular error of 10.1 degrees and a radial error

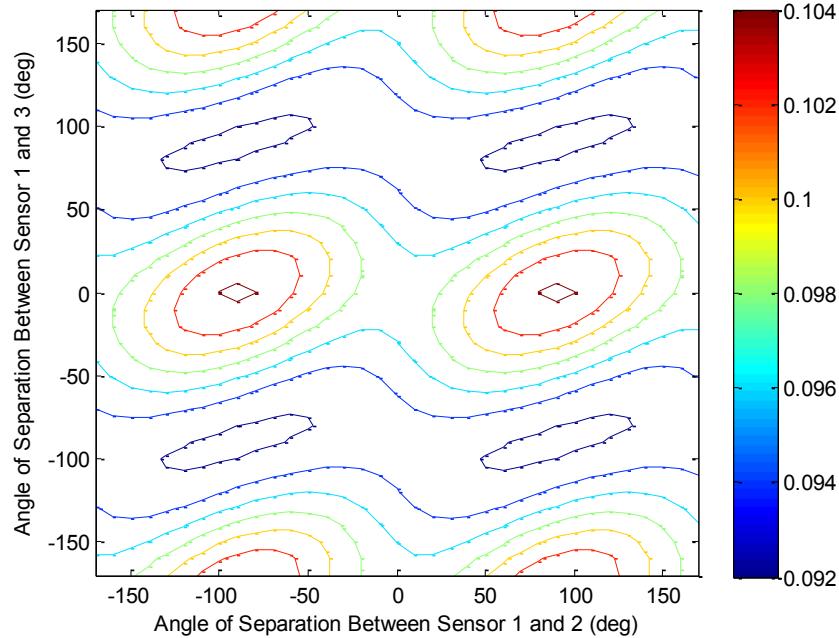
of 0.4 m. Again, the tracking stations were separated by 0 through 360 degrees and the resulting area of the combined error ellipse was found. The ideal of separation was found to consist of a single value: 180 degrees. Figure 3.15 shows the mathematical simulation results.



**Figure 3.15: Mathematical simulation results of three tracking stations with different sensors at a fixed radius of 2.83 m from the tracked object. This plot shows the area of the combined error ellipse as a function of the angle of separation between the three mobile tracking stations, as shown in Figure 3.12.**

A variation of this simulation was performed where each sensor system angle was varied independently to allow for asymmetric results. The same fixed radius of 2.83 m was used for all three sensor systems in this simulation, and the resulting contour plot can be seen in Figure 3.16. As in Section 3.5.3, the two independent angles of separation were defined as the angle between Sensor 2 and Sensor 1 and the angle between Sensor 3 and Sensor 1. The minimum of this plot occurred in four places, marked by the darkest blue circles in Figure 3.16, and were centered on the following couplets: (-90 -90), (90, 90), (-90, 90), and (90, -90). The first two

couplets are not physically possible as the two sensor systems cannot be collocated, but the second two couplets both represent a separation angle of 180 degrees, as found in Figure 3.15. This further confirms that the ideal angle of separation was not merely an artifact of the definition of the angle of separation.



**Figure 3.16: Mathematical simulation results of three tracking stations with different sensors at a fixed radius of 2.83 m from the tracked object. This plot shows the area of the combined error ellipse as a function of the angle of separation between the three mobile tracking stations where each angle of separation is varied separately.**

### 3.5.5. Summary of Findings

The mathematical simulations in this section confirm that the ideal angle of separation calculations hold true for two and three robot configurations. This methodology can accommodate a variety of sensor system ranges and cases of both identical and non-identical sensor systems. Specifically, the two robot results demonstrate that the ideal angle of separation is more heavily dependent on the relative sensor performance than on the radius

between the sensor systems and the tracked object. The three drone case with identical sensor systems verified that the ideal angle of separation of sensor systems is not an artifact of the definition of the angle of separation, but is truly a property of the sensor systems themselves. The three drone case with non-identical sensor systems confirmed that the sensor properties affect the ideal angle of separation calculations.

### 3.6. Closed-Form Optimization Derivation

In this section, the equations from Sections 3.2 through 3.4 are combined to yield a single closed-form expression that can be formally optimized. Rather than calculate separate covariance matrices for each sensor heading, a single covariance matrix was calculated for each sensor at a heading of 0 degrees with a single fixed radius. This covariance matrix was then rotated to the desired heading using the matrix rotation formula shown in Eq. (3.15).

$$\begin{bmatrix} \sigma_x^2 & \sigma_{xy} \\ \sigma_{xy} & \sigma_y^2 \end{bmatrix}_{\theta} = \begin{bmatrix} \cos(\theta) & -\sin(\theta) \\ \sin(\theta) & \cos(\theta) \end{bmatrix} \begin{bmatrix} \sigma_x^2 & \sigma_{xy} \\ \sigma_{xy} & \sigma_y^2 \end{bmatrix} \begin{bmatrix} \cos(\theta) & \sin(\theta) \\ -\sin(\theta) & \cos(\theta) \end{bmatrix} \quad (3.15)$$

The matrix formula used in Eq. (3.15) is a standard formula in dynamics and can be used in this application since  $x$  and  $y$  were independent variables.

Next, the sensor covariance matrix at a heading of 0 degrees was calculated symbolically. It was assumed that  $x$  and  $y$  were products of the independent variables  $r$  (radius from the tracked object) and  $t$  (heading to the tracked object), which were assumed to have a uniform distribution. Thus:

$$x = r_i \cos(t_i) + x_{tsn} - \mu_x \quad (3.16)$$

$$y = r_i \sin(t_i) + y_{tsn} - \mu_y \quad (3.17)$$

Here,  $x_{tsn}$  is the x location of tracking station  $n$ ,  $\mu_x$  is the mean x position of tracking station  $n$ ,  $y_{tsn}$  is the y location of tracking station  $n$ , and  $\mu_y$  is the mean y position of tracking station  $n$ . At a heading of 0 degrees,  $x_{tsn} - \mu_x$  simplifies to  $R_i$ , the radius from the tracking station to the tracked object, and  $y_{tsn} - \mu_y$  simplifies to 0. This is due to the symmetric distribution of the valid sensing area about the target object and results in the following simplified equations:

$$x = r_i \cos(t_i) + R_i \quad (3.18)$$

$$y = r_i \sin(t_i) \quad (3.19)$$

However, since the variance of  $Z + q$  when  $q$  is a constant and  $Z$  is a variable is the same as the variance of  $Z$  by the properties of variance [41], the equation for  $x$ , for the purpose of calculating the variance only, can be further reduced to:

$$x = r_i \cos(t_i) \quad (3.20)$$

At a fixed heading of 0 degrees, the values of  $r_i$  and  $t_i$  are dependent on the radius of the tracked object from the tracking station, the radial error, and the angular error of the tracking station. Consequently, the values of the covariance matrix can be found as follows:

$$\begin{aligned} \sigma_x^2 &= E(x_1^2)E(x_2^2) - (E(x_1))^2(E(x_2))^2 \\ &= E(r^2)E(\cos^2(\theta)) - (E(r))^2(E(\cos(\theta)))^2 \\ &= \left( \int_{R_i-\varepsilon_r}^{R_i+\varepsilon_r} \left( \frac{r^2}{R_i + \varepsilon_r - (R_i - \varepsilon_r)} \right) dr \right) \left( \int_{-\varepsilon_\theta}^{\varepsilon_\theta} \left( \frac{\cos^2(\theta)}{\varepsilon_\theta - (-\varepsilon_\theta)} \right) d\theta \right)^2 \\ &\quad - \left( \int_{R_i-\varepsilon_r}^{R_i+\varepsilon_r} \left( \frac{r}{R_i + \varepsilon_r - (R_i - \varepsilon_r)} \right) dr \right)^2 \left( \left( \int_{-\varepsilon_\theta}^{\varepsilon_\theta} \left( \frac{\cos(\theta)}{\varepsilon_\theta - (-\varepsilon_\theta)} \right) d\theta \right) \right)^2 \\ &= \left( R_i^2 + \frac{\varepsilon_r^2}{3} \right) \frac{1}{2\varepsilon_\theta} \left( \varepsilon_\theta + \frac{1}{2} \sin(2\varepsilon_\theta) \right) - \frac{R_i^2}{\varepsilon_\theta^2} \sin^2(\varepsilon_\theta) \end{aligned} \quad (3.21)$$

In this equation,  $x_1 = r$  and  $x_2 = \cos(\theta)$ .

$$\begin{aligned}
\sigma_y^2 &= E(y_1^2)E(y_2^2) - (E(y_1))^2(E(y_2))^2 \\
&\quad E(r^2)E(\sin^2(\theta)) - (E(r))^2(E(\sin(\theta)))^2 \\
&= \left( \int_{R_i-\varepsilon_r}^{R_i+\varepsilon_r} \left( \frac{r^2}{R_i + \varepsilon_r - (R_i - \varepsilon_r)} \right) dr \right) \left( \int_{-\varepsilon_\theta}^{\varepsilon_\theta} \left( \frac{\sin^2(\theta)}{\varepsilon_\theta - (-\varepsilon_\theta)} \right) d\theta \right) \\
&\quad - \left( \int_{R_i-\varepsilon_r}^{R_i+\varepsilon_r} \left( \frac{r}{R_i + \varepsilon_r - (R_i - \varepsilon_r)} \right) dr \right)^2 \left( \left( \int_{-\varepsilon_\theta}^{\varepsilon_\theta} \left( \frac{\sin(\theta)}{\varepsilon_\theta - (-\varepsilon_\theta)} \right) d\theta \right) \right)^2 \\
&= \left( R_i^2 + \frac{\varepsilon_r^2}{3} \right) \frac{1}{2\varepsilon_\theta} \left( \varepsilon_\theta - \frac{1}{2} \sin(2\varepsilon_\theta) \right)
\end{aligned} \tag{3.22}$$

In Eq. (3.22),  $y_1 = r$  and  $y_2 = \sin(\theta)$ .

$$\begin{aligned}
\sigma_{xy} &= E(xy) - E(x)E(y) \\
&= \int_{R_i-\varepsilon_r}^{R_i+\varepsilon_r} \int_{-\varepsilon_\theta}^{\varepsilon_\theta} \frac{1}{2\varepsilon_\theta} \frac{1}{2\varepsilon_r} (r^2 \sin(\theta) \cos(\theta) + R_i r \sin(\theta)) d\theta dr \\
&\quad - \left( \int_{R_i-\varepsilon_r}^{R_i+\varepsilon_r} \int_{-\varepsilon_\theta}^{\varepsilon_\theta} \frac{1}{2\varepsilon_\theta} \frac{1}{2\varepsilon_r} (r \cos(\theta) + R_i) d\theta dr \right) * \\
&\quad \left( \int_{R_i-\varepsilon_r}^{R_i+\varepsilon_r} \int_{-\varepsilon_\theta}^{\varepsilon_\theta} \frac{1}{2\varepsilon_\theta} \frac{1}{2\varepsilon_r} (r \sin(\theta)) d\theta dr \right) \\
&= 0 - R_i * 0 \\
&= 0
\end{aligned} \tag{3.23}$$

These equations allow for the direct calculation of the covariance matrix for each sensor system as shown in Eq. (3.2).

### 3.7. Two Robot Closed-Form Optimization

Next, the individual covariance matrices were rotated as shown in Eq. (3.15) and substituted into Eq. (3.10) to obtain the combined covariance matrix. The resulting equation allows for the direct calculation of this combined covariance matrix using only the tracking stations' radii, headings, and associated errors, reducing the number of calculations necessary. For the case with two tracking stations, the combined covariance matrix can be directly calculated using the following equation:

$$cov_{comb}(x, y) = \frac{(\sigma_x)_1^2 (\sigma_y)_1^2 (\sigma_x)_2^2 (\sigma_y)_2^2}{AC - B^2} \begin{bmatrix} C & -B \\ -B & A \end{bmatrix} \quad (3.24)$$

Here,  $A$ ,  $B$ , and  $C$  are defined as follows:

$$A = (\sigma_x)_2^2 (\sigma_y)_2^2 \left( \sin^2(\theta_1) (\sigma_x)_1^2 + \cos^2(\theta_1) (\sigma_y)_1^2 \right) + (\sigma_x)_1^2 (\sigma_y)_1^2 \left( \sin^2(\theta_2) (\sigma_x)_2^2 + \cos^2(\theta_2) (\sigma_y)_2^2 \right) \quad (3.25)$$

$$B = \sin(\theta_1) \cos(\theta_1) (\sigma_x)_2^2 (\sigma_y)_2^2 \left( (\sigma_y)_1^2 - (\sigma_x)_1^2 \right) + \sin(\theta_2) \cos(\theta_2) (\sigma_x)_1^2 (\sigma_y)_1^2 \left( (\sigma_y)_2^2 - (\sigma_x)_2^2 \right) \quad (3.26)$$

$$C = (\sigma_x)_2^2 (\sigma_y)_2^2 \left( \cos^2(\theta_1) (\sigma_x)_1^2 + \sin^2(\theta_1) (\sigma_y)_1^2 \right) + (\sigma_x)_1^2 (\sigma_y)_1^2 \left( \cos^2(\theta_2) (\sigma_x)_2^2 + \sin^2(\theta_2) (\sigma_y)_2^2 \right) \quad (3.27)$$

Finally, the eigenvalues were found for Eq. (3.24) and the corresponding semi-major and semi-minor axes were found using Eq. (3.4) and (3.5). These values were then substituted into Eq. (3.14) to obtain a single objective function for two tracking stations that could be minimized in order to find the optimal tracking configuration.

$$Area = \frac{\pi \chi_2^2 (\sigma_x)_1^2 (\sigma_y)_1^2 (\sigma_x)_2^2 (\sigma_y)_2^2}{2(AC - B^2)} * \sqrt{A + C + \sqrt{(A - C)^2 + 4B^2}} * \sqrt{A + C - \sqrt{(A - C)^2 + 4B^2}} \quad (3.28)$$

$A$ ,  $B$ , and  $C$  are defined in Eq. (3.25) through (3.27) above and  $\chi_2^2$  is the chi-squared distribution with two degrees of freedom as used in Eq. (3.4) and (3.5).

### 3.8. Three Robot Closed-Form Optimization

The same process was followed to obtain the closed-form single objective function for the three robot case: combine the individual sensor covariance matrices using Eq. (3.10) to obtain the following combined error covariance matrix:

$$cov_{comb}(x, y) = \frac{(\sigma_x)_1^2 (\sigma_y)_1^2 (\sigma_x)_2^2 (\sigma_y)_2^2 (\sigma_x)_3^2 (\sigma_y)_3^2}{GH - K^2} \begin{bmatrix} H & -K \\ -K & G \end{bmatrix} \quad (3.29)$$

$G$ ,  $H$ , and  $K$  are defined as follows:

$$\begin{aligned} G = & (\sigma_x)_2^2 (\sigma_y)_2^2 (\sigma_x)_3^2 (\sigma_y)_3^2 \left( \sin^2(\theta_1) (\sigma_x)_1^2 + \cos^2(\theta_1) (\sigma_y)_1^2 \right) + \\ & (\sigma_x)_1^2 (\sigma_y)_1^2 (\sigma_x)_3^2 (\sigma_y)_3^2 \left( \sin^2(\theta_2) (\sigma_x)_2^2 + \cos^2(\theta_2) (\sigma_y)_2^2 \right) + \\ & (\sigma_x)_1^2 (\sigma_y)_1^2 (\sigma_x)_2^2 (\sigma_y)_2^2 \left( \sin^2(\theta_3) (\sigma_x)_3^2 + \cos^2(\theta_3) (\sigma_y)_3^2 \right) \end{aligned} \quad (3.30)$$

$$\begin{aligned} K = & \sin(\theta_1) \cos(\theta_1) (\sigma_x)_2^2 (\sigma_y)_2^2 (\sigma_x)_3^2 (\sigma_y)_3^2 \left( (\sigma_y)_1^2 - (\sigma_x)_1^2 \right) + \\ & \sin(\theta_2) \cos(\theta_2) (\sigma_x)_1^2 (\sigma_y)_1^2 (\sigma_x)_3^2 (\sigma_y)_3^2 \left( (\sigma_y)_2^2 - (\sigma_x)_2^2 \right) + \\ & \sin(\theta_3) \cos(\theta_3) (\sigma_x)_1^2 (\sigma_y)_1^2 (\sigma_x)_2^2 (\sigma_y)_2^2 \left( (\sigma_y)_3^2 - (\sigma_x)_3^2 \right) \end{aligned} \quad (3.31)$$

$$\begin{aligned} H = & (\sigma_x)_2^2 (\sigma_y)_2^2 (\sigma_x)_3^2 (\sigma_y)_3^2 \left( \cos^2(\theta_1) (\sigma_x)_1^2 + \sin^2(\theta_1) (\sigma_y)_1^2 \right) + \\ & (\sigma_x)_1^2 (\sigma_y)_1^2 (\sigma_x)_3^2 (\sigma_y)_3^2 \left( \cos^2(\theta_2) (\sigma_x)_2^2 + \sin^2(\theta_2) (\sigma_y)_2^2 \right) + \\ & (\sigma_x)_1^2 (\sigma_y)_1^2 (\sigma_x)_2^2 (\sigma_y)_2^2 \left( \cos^2(\theta_3) (\sigma_x)_3^2 + \sin^2(\theta_3) (\sigma_y)_3^2 \right) \end{aligned} \quad (3.32)$$

The eigenvalues from Eq. (3.29) were substituted into Eq. (3.14) to obtain a single objective function that could be minimized to find the optimal tracking configuration with three tracking stations as seen below:

$$\begin{aligned} Area = & \frac{\pi \chi_2^2 (\sigma_x)_1^2 (\sigma_y)_1^2 (\sigma_x)_2^2 (\sigma_y)_2^2 (\sigma_x)_3^2 (\sigma_y)_3^2}{2(GH - K^2)} \\ & * \sqrt{G + H + \sqrt{(G - H)^2 + 4K^2}} * \\ & \sqrt{G + H - \sqrt{(G - H)^2 + 4K^2}} \end{aligned} \quad (3.33)$$

Here,  $G$ ,  $H$ , and  $K$  are as defined in Eq. (3.30) through (3.32) and  $\chi_2^2$  is the chi-squared distribution with two degrees of freedom as used in Eq. (3.4) and (3.5).

### 3.9. Closed-Form Theoretical Constrained Optimization

The closed-form area equations presented in the last two sections were optimized using a constrained Hooke and Jeeves method. While this method may produce a locally optimal

solution rather than a globally optimal solution [33], the scenarios examined in this research featured locally optimal solutions that were also globally optimal solutions, so this was not an issue.

The inputs for the objective functions in Eq. (3.28) and (3.33) were the sensor radial error and sensor angular error, which were both held constant, and the radii and heading between each sensor and the tracked object. The last two inputs were varied while the heading was allowed to take any value since  $\theta \pm 360$  degrees is equivalent to  $\theta$ . The radii were constrained to be within the observable range of the sensors. In this case, these bounds were set at a minimum of 1.7 m and a maximum of 4 m to match the quadrotor camera viewing distances. The Hooke and Jeeves method was modified to include these bounds by placing a penalty of 100 times the objective function output on any input that violated these bounds. The Matlab script [42] used in this optimization can be found in Appendix B.

**Table 3.4: Inputs for a bounded 2D area for two sensors with identical properties.**

Input	Value
<b>Radius of sensor 1 (m)</b>	4
<b>Sensor 1 radial error (m)</b>	0.4
<b>Heading of sensor 1 (deg)</b>	0
<b>Sensor 1 angular error (deg)</b>	5.7
<b>Radius of sensor 2 (m)</b>	4
<b>Sensor 2 radial error (m)</b>	0.4
<b>Heading of sensor 2 (deg)</b>	0
<b>Sensor 2 angular error (deg)</b>	5.7

To show how this optimization method works, two identical sensors with restricted two-dimensional boundaries were examined. As in Section 3.5.1, the sensors were given radial and angular errors of 0.4 m and 5.7 degrees, respectively. The initial input for each sensor was a radius of 4 m and a heading of 0 degrees. This resulted in the inputs shown in Table 3.4. Since only two sensors were used, Eq. (3.28) was the objective function. The results are shown in Table 3.5.

**Table 3.5: Output for a bounded 2D area for two sensors with identical properties.**

Input	Value
Radius of sensor 1 (m)	1.7
Sensor 1 radial error (m)	0.4
Heading of sensor 1 (deg)	135
Sensor 1 angular error (deg)	5.7
Radius of sensor 2 (m)	1.7
Sensor 2 radial error (m)	0.4
Heading of sensor 2 (deg)	-135
Sensor 2 angular error (deg)	5.7

This means that the sensors yield the best results when they are as close as possible to the tracked object and have a separation angle of 90 degrees, matching the result obtained in Section 3.5.1 as expected. The algorithm can then be incorporated into the cluster controller as described in Section 3.10.

Three identical sensors with restricted two-dimensional boundaries were also examined. All sensors were given radial and angular errors of 0.4 m and 5.7 degrees, respectively, and an

initial radius of 4 m and a heading of 0 degrees. Eq. (3.33) was the objective function and resulted in an ideal angle of separation of 120 degrees, also matching the results in Section 3.5.3 as expected.

The Hooke and Jeeves method has the added benefit that its computation time scales linearly with the number of inputs [33]; the space complexity is  $O(n)$ , so the number of robots can easily be increased. Specifically, the two robot case presented here computed on a conventional Pentium-class workstation with a 2.10 GHz processor and 4.00 GB of RAM in about 0.14 second while the result for the three robot case computed in 0.33 second. Both of these computation times are more than fast enough to correct a system under disturbances such as loss of a sensor system, changing sensor properties, or changing radii. If faster response times are desired, it is possible to pre-compute the optimal angle of separation for likely scenarios, reserving online computations for unforeseen changes.

Compared to the symmetric, non-optimized worst case examined by this research, the optimization method presented here results in a 6% target estimation improvement for both the two and three fixed radius, identical sensor cases. This represents a significant improvement in the estimation of a target's location.

The method of sensor placement examined in this research was also compared to existing sensor placement optimization methods. Although similar to the method presented in [11], it cannot be directly compared since [11] assumed a covariance matrix produced by a Kalman filter. The method presented here does not use a Kalman filter, so the algorithm developed in [11] does not apply. Instead, this method is compared with the method presented in [43] and [44] where the determinate of the error covariance matrix was minimized. In this method, the global covariance matrix was defined as follows:

$$\det(P_{global}) = \det\left(\left(\sum_{i=1}^n P_i^{-1}\right)^{-1}\right) \quad (3.34)$$

Here,  $n$  is the total number of sensor systems and  $P$  is the error covariance matrix. Since  $P_{global}$  is defined as in Eq. (3.10), a direct comparison between the methods can be found, for two mobile tracking stations, by taking the determinant of Eq. (3.24) in Section 3.7.

$$\begin{aligned} & \det\left(\frac{(\sigma_x)_1^2(\sigma_y)_1^2(\sigma_x)_2^2(\sigma_y)_2^2}{AC - B^2} \begin{bmatrix} C & -B \\ -B & A \end{bmatrix}\right) \\ &= \frac{((\sigma_x)_1^2(\sigma_y)_1^2(\sigma_x)_2^2(\sigma_y)_2^2)^2}{AC - B^2} \end{aligned} \quad (3.35)$$

In the case of a fixed radius and identical sensor systems examined here, Eq. (3.21) and (3.22) show that  $(\sigma_x)_1^2 = (\sigma_x)_2^2$  and  $(\sigma_y)_1^2 = (\sigma_y)_2^2$  for the fixed radius, identical sensor system case. Finally, the assumption can be made that  $\theta_1 = -\theta_2$  since any symmetric separation of angles yields the same result in the method presented here. Thus,  $AC - B^2$  reduces to:

$$AC - B^2 = 2\sigma_x^2\sigma_y^2(\sigma_x^2 + \sigma_y^2) \quad (3.36)$$

Equation (3.36) consists only of  $\sigma_x^2$  and  $\sigma_y^2$  terms, which were shown in Section 3.6 to always have the same value. Thus, no minimum can be found for the fixed radius, identical sensor system case using the determinate method presented in [43] and [44].

The fixed radius, identical sensor system, three mobile tracking station case was also compared to the determinate method by taking the determinate of Eq. (3.29) from Section 3.8.

$$\begin{aligned} & \det\left(\frac{(\sigma_x)_1^2(\sigma_y)_1^2(\sigma_x)_2^2(\sigma_y)_2^2(\sigma_x)_3^2(\sigma_y)_3^2}{GH - K^2} \begin{bmatrix} H & -K \\ -K & G \end{bmatrix}\right) \\ &= \frac{((\sigma_x)_1^2(\sigma_y)_1^2(\sigma_x)_2^2(\sigma_y)_2^2(\sigma_x)_3^2(\sigma_y)_3^2)^2}{GH - K^2} \end{aligned} \quad (3.37)$$

Again, the  $\sigma_x^2$  and  $\sigma_y^2$  terms are the same for each sensor with identical properties at a fixed radius. Using the definition of the separation of angles presented here,  $\theta_1 = 0$  degrees and  $\theta_2 = -\theta_3$  since, once again, any symmetric angle distribution yields the same results. Thus,  $GH-K^2$  reduces to the following:

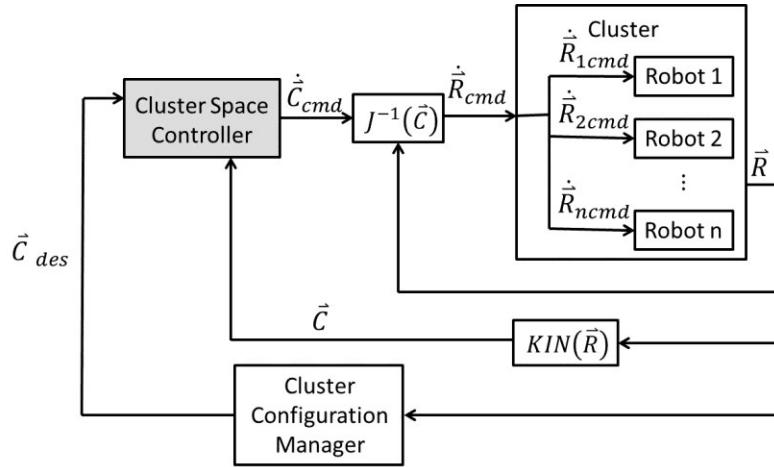
$$GH - K^2 = (\sigma_x^2)^4 (\sigma_y^2)^4 * \left( 2(\sigma_x^2)^2 + 5\sigma_x^2 \sigma_y^2 + 2(\sigma_y^2)^2 \right) \quad (3.38)$$

Even in the case of three mobile tracking stations, the determinate method presented in [43] and [44] fails to yield a result. Thus, the method presented here can be applied in cases where the determinate method fails.

### 3.10. Formation Control

This aspect of formation control can be integrated into the cluster controller shown in Figure 2.2 in order to maintain the optimized formation under changing circumstances. The resultant controller is shown in Figure 3.17. This controller performs similarly to the generic cluster controller discussed in Section 2.3. The desired cluster positon is compared to the actual cluster position, in cluster space, and passed through a PID controller to obtain the cluster command velocity, also in cluster space. This is passed through the inverse Jacobian in order to obtain the command velocity in robot space. The velocity is then broken down into the command velocities for each individual robot and implemented. The resultant robot positions are measured and fed through the forward kinematics in order to obtain the cluster position in cluster space for comparison with the desired cluster positon. The difference is that in the optimal geometry cluster controller, the measured robot positions are also passed through the cluster

configuration manager which determines the desired cluster position. This control method will be expanded further in the following chapter.



**Figure 3.17:** Optimal geometry cluster controller for  $n$  robots.

# **Chapter 4**

---

## **4. Vision Processing**

Robots need to gather information about the world around them in order to successfully track an object. This is accomplished through the use of sensors, here, cameras, which take in a wide array of information about the world around the robot. The information is then processed and formatted in a manner that the controller can interpret and used to determine the cluster's desired location. The specific steps in this process are detailed in this chapter.

### **4.1. Literature Survey**

There are a wide variety of ways to extract location information from camera data. For example, the authors of [45] used a background subtraction method combined with depth segmentation in order to determine the location of a human in an indoor environment. Once background subtraction was performed, the foreground image blobs were run through a size filter and all blobs that did not match human size and aspect parameters were rejected. Next, depth was used to filter the remaining background from the human image by rejecting all pixels that did not cluster at the corresponding depth. This method was able to locate a human in an indoor environment quite accurately. However, it required the use of a depth sensor, which was not available on the AR.Drone 1.0 quadrotors.

In [46], the authors used a single commercially available camera mounted on a quadrotor to detect moving targets. In this method, static objects in the camera images were used as points of reference. The dynamic pixels in each image were then grouped into dynamic objects which could be tracked by the quadrotor. While this method resulted in accurate tracking, it was deemed to be more complex than required for the application presented in this dissertation.

Another method of vision processing is filtering the camera image for a single color. This method works best when the tracked object is a different color than its surroundings. In [47], color segmentation was used to identify a hand in a camera image. These hand images were classified into one of six gestures and used to communicate preplanned trajectories to a group of robots. Studio Diip also used this method to track its goldfish for the Fish on Wheels project [48]. This project was an effort to demonstrate the possibilities of computer vision using existing simple hardware [49] and located a goldfish against the plain white background of the floor of a fish tank. Its motion was then used to guide the motion of the tank. For example, if the fish swam to the right of the tank, the tank rolled to the right.



Figure 4.1: Studio Diip's Fish on Wheels project together with its vision processing result [49].

The work presented in this dissertation uses a similar method of vision processing; it tracks the target object by locating the center of the red blob in each video image. The specifics of this implementation will be further discussed in the following sections of this chapter.

## 4.2. Available Data

The AR.Drones come equipped with two onboard cameras that can be viewed in real-time through the onboard wireless connection. The first camera is mounted on the “nose” of the

quadrotor and is a 93° wide-angle diagonal lens. The second camera is mounted on the bottom of the quadrotor and is a 64° diagonal lens [50]. These cameras can be seen in Figure 4.2. The camera image is in RGB color and is updated at a frequency of about 30 Hz [51]. It is possible to obtain data from both cameras at once, but the data obtained from a single camera angle was more than sufficient for the purposes of this study. Initially, the bottom-mounted camera was thought to provide the best data. The top of the Pioneer is dark and has distinctive electronics mounted on it, making it easy to find against the light background of the test area. However, the test area also features low-hanging lights that limit the height of the quadrotor to less than two meters. At this height, the angle of the camera does not allow much margin for error in tracking the Pioneer; the camera can only see approximately 0.3 m to either side of the Pioneer. This gives the advantage to the front-mounted camera which has a wider angle and allows the quadrotor to be positioned as far away from the Pioneer as necessary in order to obtain a sufficient view. In practice, this distance is two to three meters away. Sample images of the Pioneers from both angles can be seen in Figure 4.3.

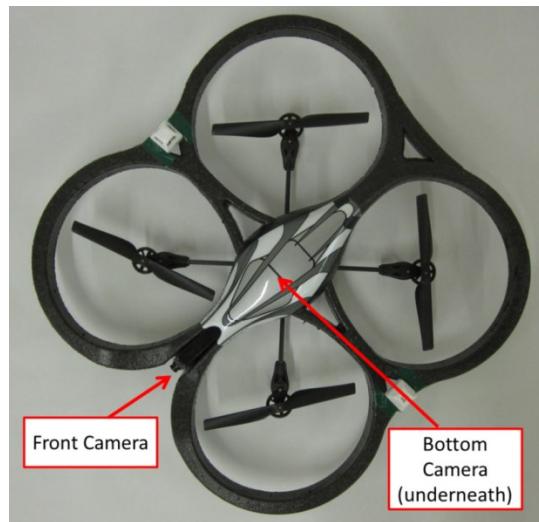
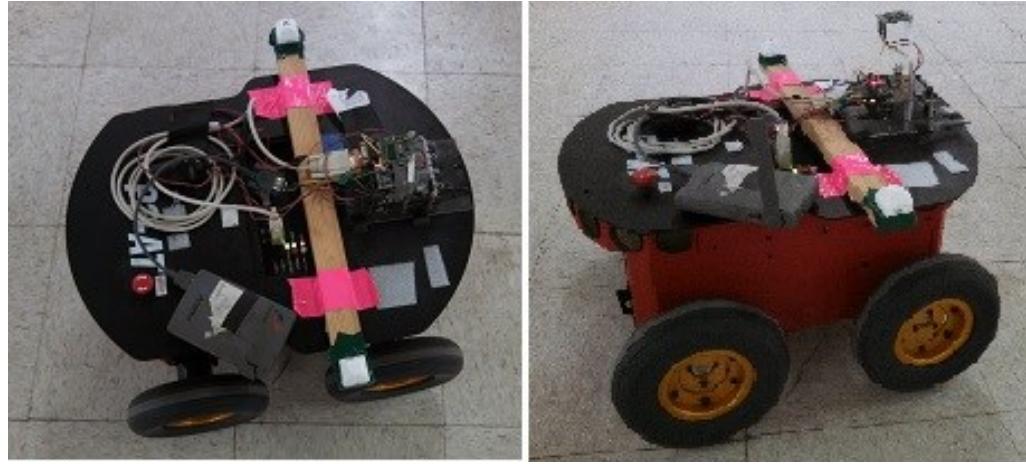
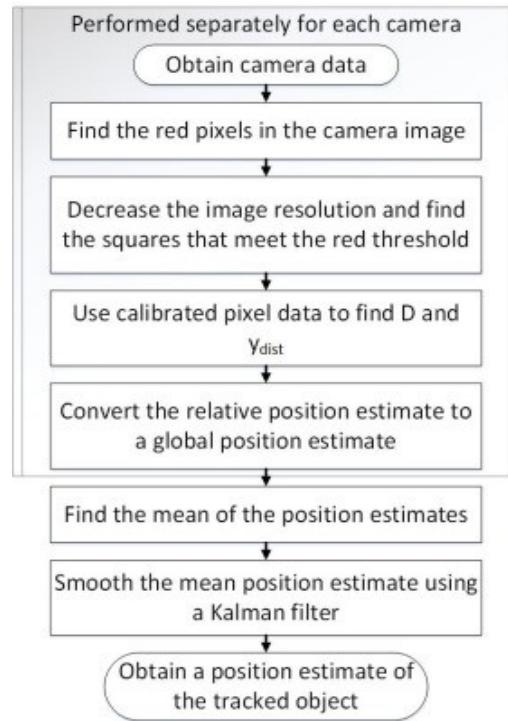


Figure 4.2: Camera locations on the AR.Drone version 1.0.



**Figure 4.3: Pioneer viewing options. Left: View from the top. Right: View from the side.**

### 4.3. The Influence of the Data Transmission Rate



**Figure 4.4: Flowchart of the vision processing algorithm.**

The camera data is updated at a rate of about 30 Hz [51], faster than the 8 Hz rate at which the controller sends out a new command. The camera data also contains far more information than

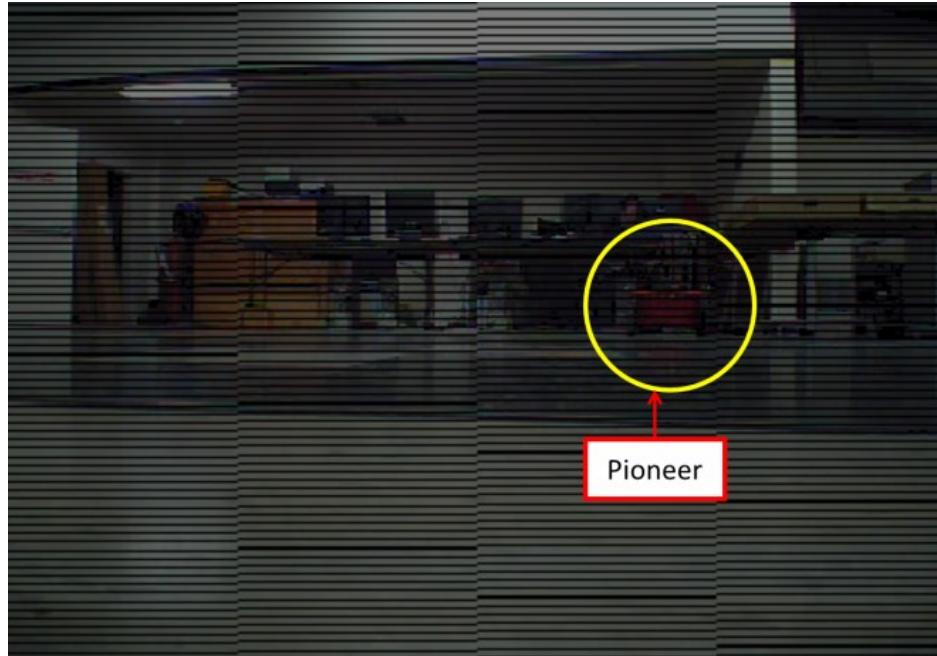
is necessary for this application where the crucial information is the Pioneer's location relative to the quadrotor. Further, transmitting the RGB image over DataTurbine, a modified version of open-source software used at SCU's RSL, takes approximately five minutes for a single still image. This is far too long for the purpose of creating a controller, so the camera data was simplified to identify only the Pioneer using the method shown in Figure 4.4.

#### 4.4. Vision Data Simplification

The camera data was initially simplified by considering what made the Pioneer stand out from its surroundings and made it recognizable as the object to be tracked. As stated above, the test area featured a light-colored floor, a dull-colored ceiling, and was surrounded by few brightly colored-objects. An initial view of the Pioneer as seen from the quadrotor is shown below in Figure 4.5. As can be seen in this image, the Pioneer is a bright red while very few of the surrounding objects have the same color. Thus, the computer finds the Pioneer based on the amount of pure red in the image. The computer represents each pixel in the image as an RGB value, but the pixels with the highest *R* values do not necessarily correspond to the areas that are the brightest red. To filter out the high red values that actually represent colors such as white or tan, the following equation was used:

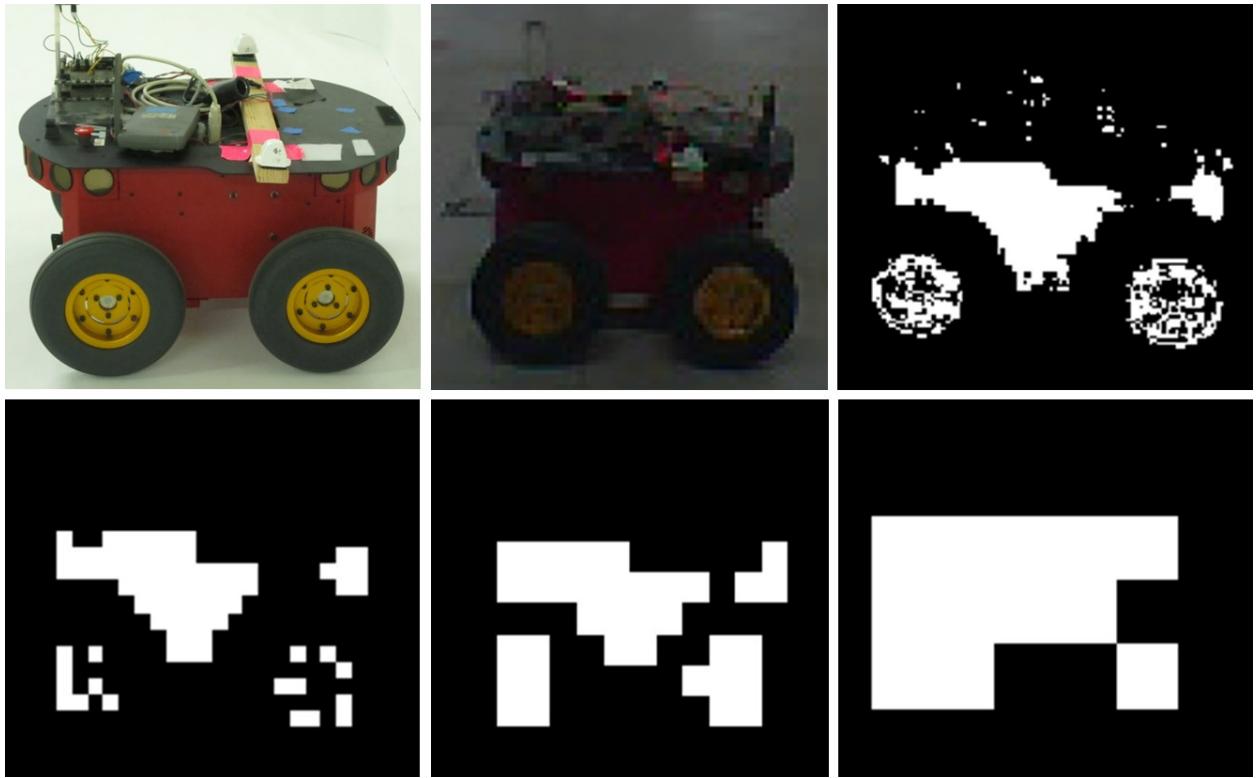
$$distance = \sqrt{(green\ value)^2 + (blue\ value)^2} \quad (4.1)$$

Once this distance was found, only pixels that had both a red value greater than 35 and a distance less than 35 were labeled as "red" pixels. In other words, only pixels with colors very close to pure red were labeled "red". These pixels were given the maximum red value of 255 while the red value of all other squares was set to zero.



**Figure 4.5: Image of the Pioneer inside the test area taken by the quadrotor's onboard forward-mounted camera, as displayed in Matlab.**

The image was then progressively simplified in order to find an image that conveyed the necessary information and could be transmitted at a fast enough rate from the quadrotor to the controller. The steps of this simplification process can be seen in Figure 4.6. The resolution level finally chosen was a grid 16 squares long and 12 squares tall where each square represents a 20x20 grid of pixels. The sum of red values for each pixel in a square was given a threshold level of 15,000 and any square with a sum above this threshold was given a value of one. If a square did not meet this threshold value, it was given a value of zero. Thus, only 24 bytes of data needed to be sent from Java to Matlab via DataTurbine rather than the 76,800 bytes necessary for the full color image. These 24 bytes consist of binary values that are assembled into a corresponding matrix and are then displayed in Matlab as shown in the black and white images in Figure 4.6 and Figure 4.7. The values of one correspond to the white areas while the values of zero correspond to the black areas, with the white areas representing the Pioneer.



**Figure 4.6: Progressive image simplification.** Row 1 from left to right: Pioneer as seen from nominal flight distance by the human eye, Pioneer at nominal flight distance as seen by the quadrotor's forward-facing camera, Pioneer at nominal flight distance with a full pixel resolution of the “red” areas. Row 2 from left to right: Pioneer as seen from nominal flight distance where each square represents a grid of 5 by 5 pixels, Pioneer as seen from nominal flight distance where each square represents a grid of 10 by 10 pixels, Pioneer as seen from nominal flight distance where each square represents a grid of 20 by 20 pixels.

Once the Pioneer was “seen” by the camera, the Pioneer’s location with respect to the quadrotor was determined using two distances: the downrange distance in front of the camera,  $D$ , and the lateral distance from the center of the image,  $y_{dist}$ . These distances can be seen in Figure 4.8. The number of pixels representing the Pioneer was used to find the downrange distance  $D$  while the lateral distance  $y_{dist}$  was found from the centroid of the white squares in the image.

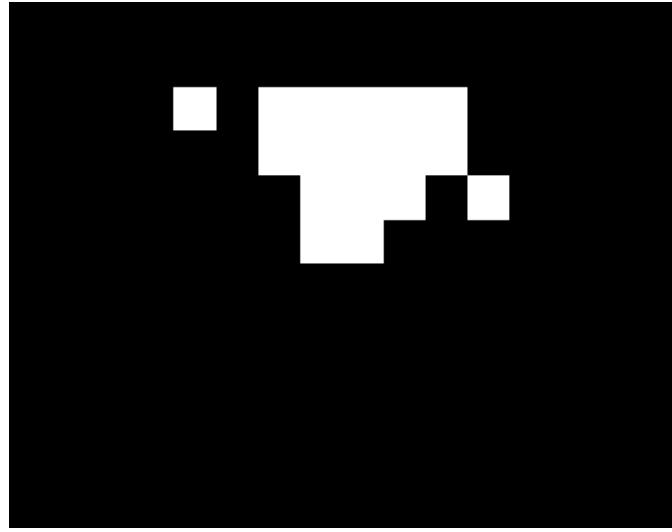


Figure 4.7: The Pioneer as "seen" by the controllers from one meter away.

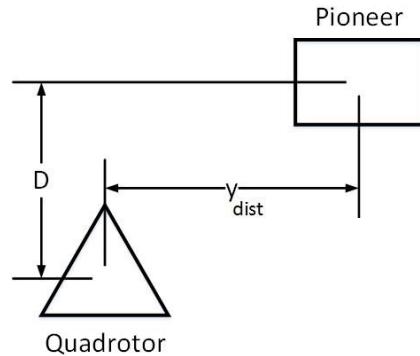


Figure 4.8: Distance explanation.

$D$  is found using the following equation, found by curve fitting experimental data (see Appendix C):

$$D = \ln\left(\frac{23}{n}\right) \quad (4.2)$$

Here,  $n$  is the number of white pixels in the image. The use of the relative size of a known object at various distances to determine the distance of an object from the observer is similar to how

the human eye determines distances [52]. If the quadrotor does not “see” the Pioneer, a value of  $D = 0$  is used.

The first step in calculating  $y_{dist}$  is to find the center of the Pioneer by taking the mean of the indices where the white pixels occur. Using Figure 4.7 as an example, the mean position is (4.1, 8.9). The first value is ignored since the physical space used in the experiments does not allow the quadrotors much ability to move up and down. The second value, however, is used to determine how far to the right or left the Pioneer is with respect to the camera. A value of eight means that the camera is perfectly aligned with the Pioneer, a value of zero to eight means that the camera is to the left of the Pioneer, and a value of eight to 16 means that the camera is to the right of the Pioneer.

Next, the distance per pixel is calculated using the distance  $D$ , as shown in Eq. (4.3). This value is then used in Eq. (4.4) to find  $y_{dist}$ .

$$(distance \text{ } per \text{ } pixel) = \frac{D \sin\left(\frac{53.13\pi}{180}\right)}{16} \quad (4.3)$$

$$y_{dist} = (y_{center} - 8) \times (distance \text{ } per \text{ } pixel) \quad (4.4)$$

In Eq. (4.4),  $y_{center}$  is the y value of the white area’s centroid. Physically, this represents the signed distance, in pixels, of the object from the image center times the physical distance represented by each pixel. Together,  $D$  and  $y_{dist}$  specify the location of the Pioneer with respect to the camera.

## 4.5. Finding the Cluster-Level Position Estimate of the Pioneer

In multi-quadrotor clusters, each quadrotor is equipped with its own camera and finds its own relative estimate of the Pioneer position. This relative estimate is then converted to a global estimate by adding the location of the quadrotor to the relative Pioneer position, converted into the global coordinate frame, as shown in Eq. (4.5) below.

$$\begin{bmatrix} x_{new} \\ y_{new} \end{bmatrix} = \begin{bmatrix} \cos \theta_i & \sin \theta_i \\ \sin \theta_i & -\cos \theta_i \end{bmatrix} \begin{bmatrix} D \\ y_{dist} \end{bmatrix} + \begin{bmatrix} x_i \\ y_i \end{bmatrix} \quad (4.5)$$

Here,  $\theta_i$  is the yaw angle of the quadrotor in the global frame and  $(x_i, y_i)$  is the location of the quadrotor in the global frame. The mean of these individual estimates is then passed through a Kalman filter and used to find the new cluster center. A flowchart of the Kalman filter algorithm can be seen in Figure 4.9 where  $x$  is the state estimate  $[x \ y \ \dot{x} \ \dot{y}]^T$  and is updated by  $A$ , defined in Eq. (4.6), which assumes a constant velocity and updates the position based on the distance travelled in a single time step of 0.125 second.  $P$  is the estimate covariance,  $Q$  is the process noise covariance and is defined in Eq. (4.7) based on a 10% process error. A larger error was used for the velocity in the  $y$  direction since this was the only axis the Pioneer could move along and had a greater uncertainty as the Pioneer sometimes stopped or reversed direction.  $K$  is the Kalman gain;  $C$  is defined in Eq. (4.8) and is used to measure only the position.  $R$  is the measurement noise covariance matrix defined in Eq. (4.9). The noise for position was based on the maximum error found during experimentation while the noise for velocity assumed a worst case scenario and doubled the maximum position errors found during experimentation. Finally,  $z$  is the measured stated positon and  $I$  is the identity matrix.

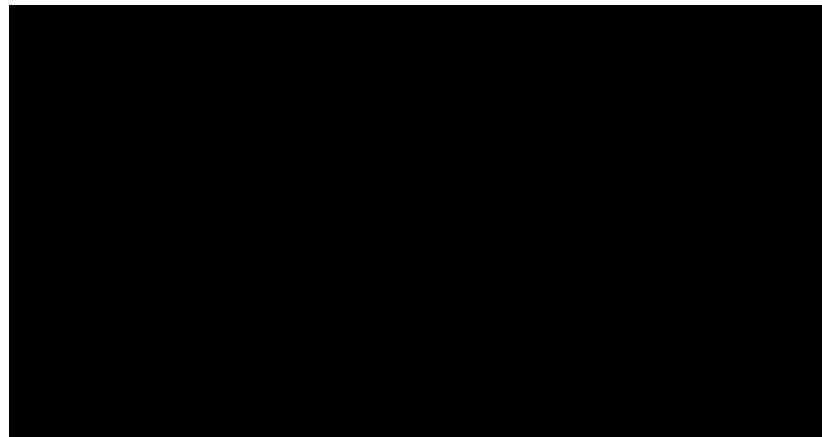
$$A = \begin{bmatrix} 1 & 0 & 0.125 & 0 \\ 0 & 1 & 0 & 0.125 \\ 0 & 0 & 1 & 0 \\ 0 & 0 & 0 & 1 \end{bmatrix} \quad (4.6)$$

$$Q = \begin{bmatrix} 0.005 & 0 & 0 & 0 \\ 0 & 0.005 & 0 & 0 \\ 0 & 0 & 0.005 & 0 \\ 0 & 0 & 0 & 0.05 \end{bmatrix} \quad (4.7)$$

$$C = \begin{bmatrix} 1 & 0 & 0 & 0 \\ 0 & 1 & 0 & 0 \\ 0 & 0 & 0 & 0 \\ 0 & 0 & 0 & 0 \end{bmatrix} \quad (4.8)$$

$$R = \begin{bmatrix} 0.37 & 0 & 0 & 0 \\ 0 & 0.27 & 0 & 0 \\ 0 & 0 & 0.74 & 0 \\ 0 & 0 & 0 & 0.54 \end{bmatrix} \quad (4.9)$$

This method allows for the use of different sensors on each quadrotor and means that if one of the quadrotors or its sensor is lost, the cluster can continue to track the Pioneer.



**Figure 4.9: Kalman filter algorithm, adapted from [53].**

In a two quadrotor cluster, the cluster center is specified to have the same x position as the Pioneer and a y position a fixed distance behind the Pioneer. The  $p$ ,  $\phi_i$ , and standoff distance are

specified to allow optimal viewing of the Pioneer by each quadrotor, as determined using the method described in Chapter 3. This setup is shown in Figure 4.10.

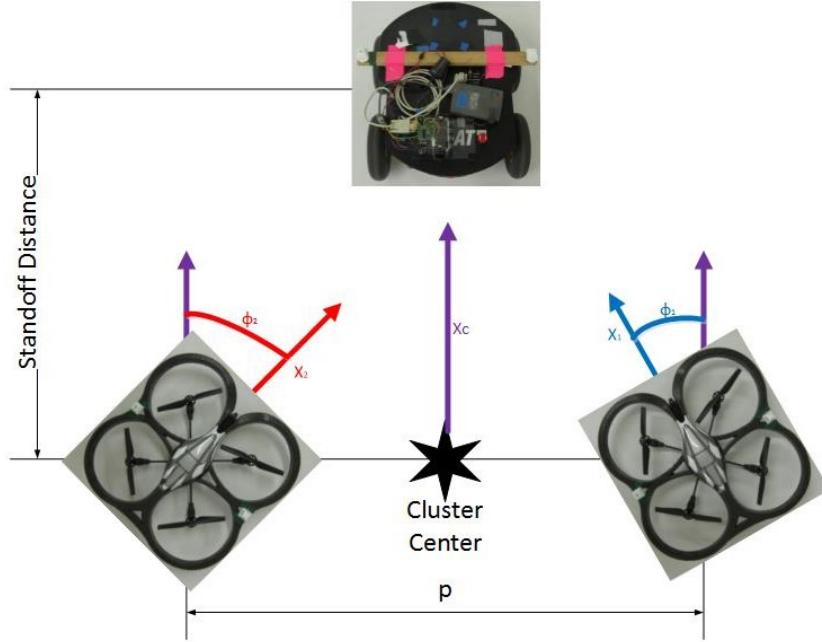


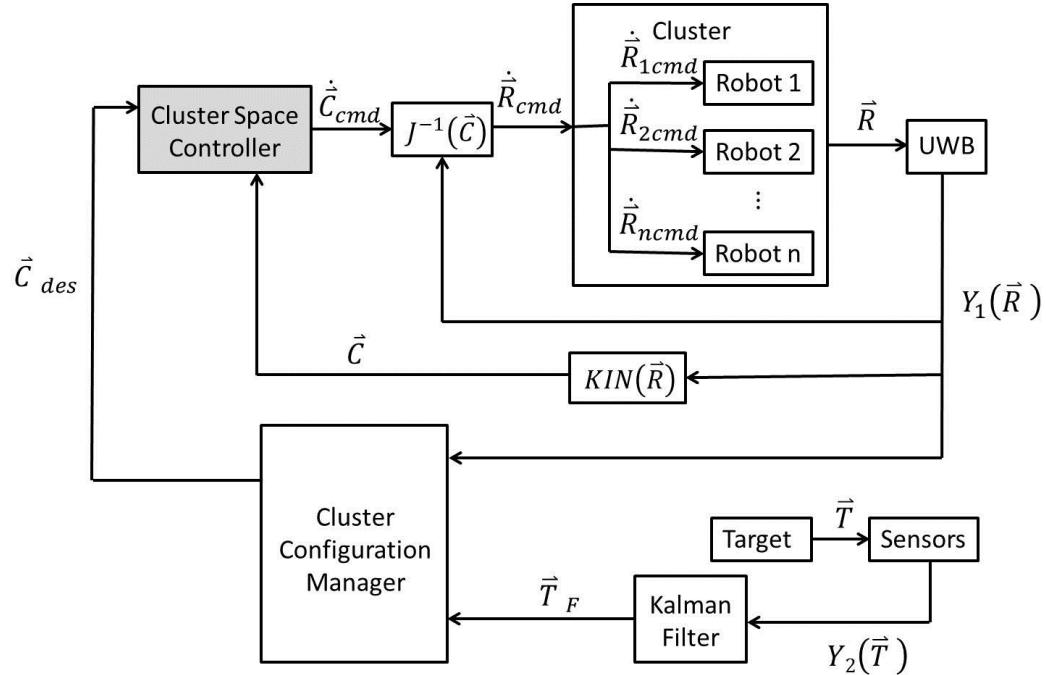
Figure 4.10: Optimal configuration of a two quadrotor tracking cluster.

## 4.6. Tracking Control

A final addition to the cluster controller discussed in Sections 2.3 and 3.10 is the development of a tracking module. The block diagram of this cluster controller is shown in Figure 4.11. Again, this controller works similarly to the generic cluster controller, the difference being the addition of a tracked object that is not under the control of the cluster controller. Here, the desired and actual cluster positions, in cluster space, are compared and passed through a PID controller in order to obtain the command velocity in cluster space. This command velocity is then passed through the inverse Jacobian in order to obtain the command velocity in robot space, which is then broken down into the individual robot components and implemented by the robots. The actual robot positions are measured using an UltraWide Band (UWB) network and radio

frequency identifier (RFID) tags while the target is measured using the robots' onboard sensors.

The actual robot positon, in robot space, is fed through the forward kinematics in order to obtain the actual cluster position in cluster space, just as in the generic cluster controller.



**Figure 4.11: Tracking and optimal geometry cluster controller.**

However, the position of the tracked object is fed through a Kalman filter, as described in Section 0. This filtered position estimate, along with the actual robot positions in robot space, are then passed through the cluster configuration manager in order to obtain the optimal tracking geometry. The addition of the target position information allows the cluster configuration manager to adapt more easily to changes in the tracked object's position, allowing this methodology to track moving objects as well as stationary objects.

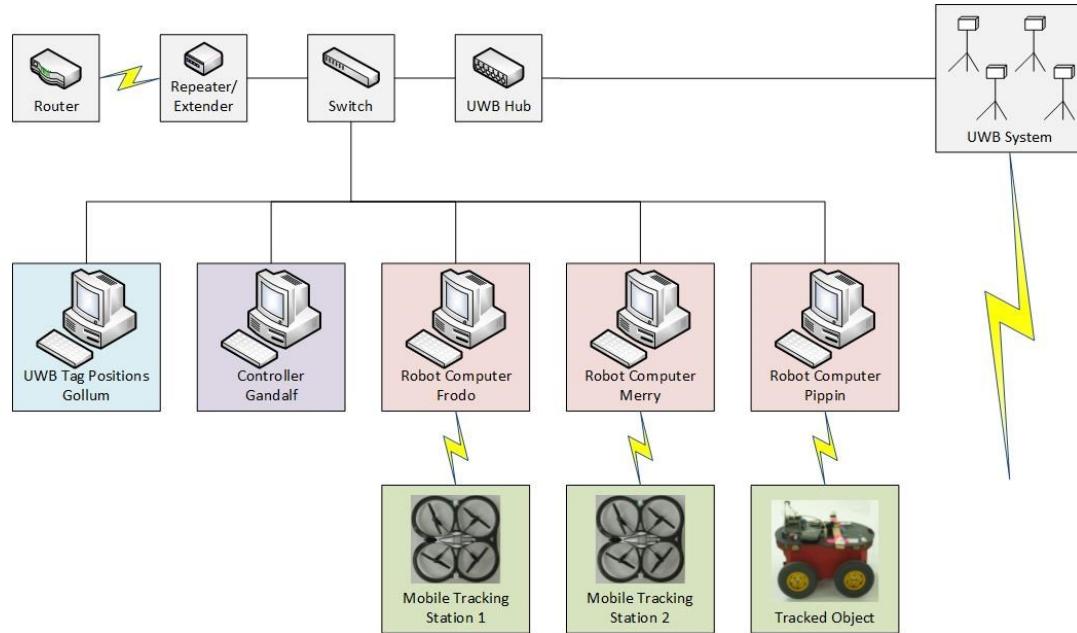
# Chapter 5

---

## 5. Experimental Testbed

This chapter provides an overview of the testbeds used in this research. The first testbed consists of a two quadrotor, single Pioneer system while the second testbed consists of a three quadrotor, single Pioneer system. A more detailed description of the system can be found in Appendix D and in [54].

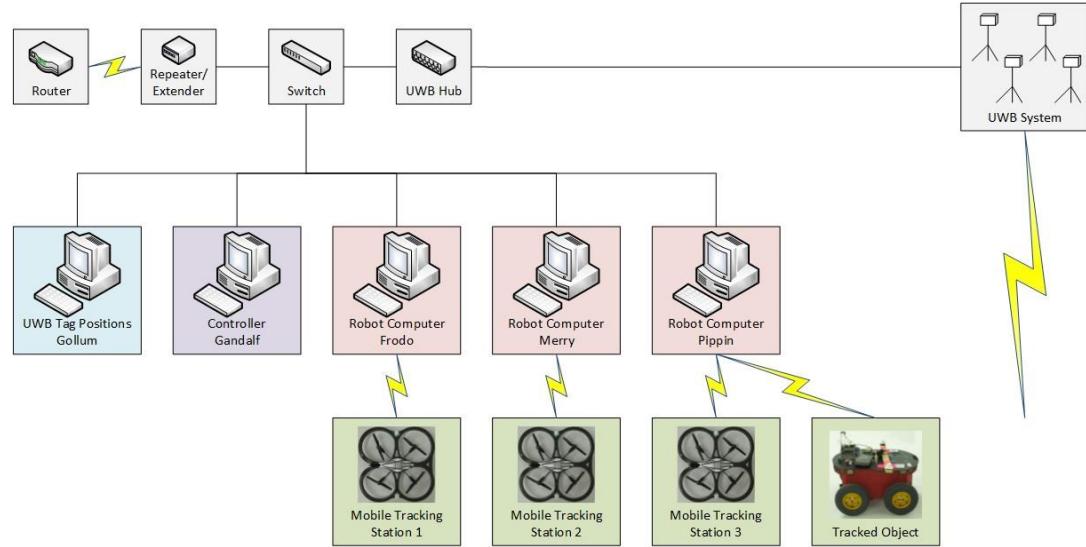
### 5.1. System Overview



**Figure 5.1: Two mobile tracking stations testbed hardware layout.**

The testbed used for this research was developed in conjunction with fellow graduate students [55][56][57] and was designed to serve as a proof-of-concept testbed. A physical layout of the hardware used can be seen in Figure 5.1 and Figure 5.2. There are four main components of both testbeds: the quadrotor mobile tracking stations, the Pioneer tracked object, the sensing

system, and the software. Each of these components will be discussed in detail in the following sections.



**Figure 5.2: Three mobile tracking stations testbed hardware layout.**

## 5.2. Quadrotor Mobile Tracking Stations

Parrot's AR.Drone 1.0 quadrotors were used as the mobile tracking stations in this dissertation and can be seen in Figure 5.3. These quadrotors are a hobby class aerial vehicle designed to be controlled via mobile phone [50]. They have a total of four degrees of freedom (DOF): three translational and one rotational. Drone movement is constrained translationally along the robot's local x, y, and z axis and rotationally about the z axis, yaw. Rotation about the x axis, roll, is coupled with translation about the y axis while rotation about the y axis, pitch, is coupled with translation along the x axis so these are not true degrees of freedom.

The quadrotor has a maximum speed of 5 m/s and an approximate running time of 15 minutes with no payload. The forward-facing camera, which acts as the mobile sensor in this testbed, is a 93 degree wide angle diagonal lens [50]. The robot communicates over its own WiFi

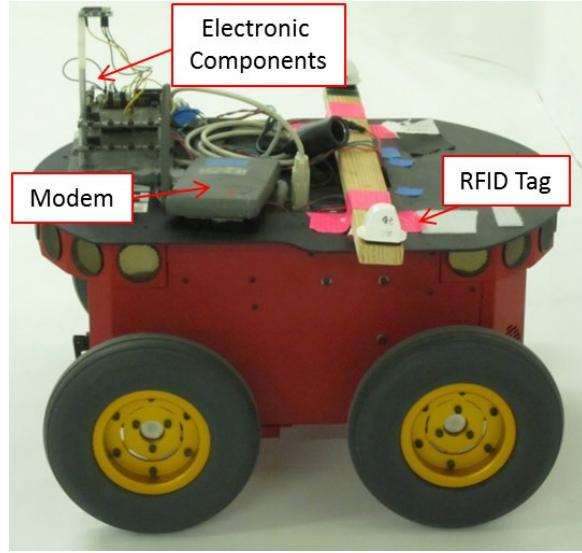
network; thus, each robot communicates with a separate computer. In the experiments with two mobile tracking stations, Frodo and Merry [58] (see Figure 5.1) are used as the tracking station computers while Pippin [58] is used solely to communicate with the Pioneer. In the experiments with three tracking stations, the only change is that Pippin is used to control two robots: one quadrotor and one Pioneer. Despite communicating through the same computer, the robots do not share information with each other.



Figure 5.3: AR.Drone 1.0 overview.

### 5.3. Pioneer Tracked Object

A Pioneer 3-AT land rover, shown in Figure 5.4, was used as the tracked object in both testbeds. Only one Pioneer was used in each of the experiments, creating a single target environment. This robot has a maximum speed of 0.7 m/s and a maximum running time of three hours [59]. The Pioneer has two degrees of freedom: movement along its longitudinal axis and rotation about its z axis. An attached modem allows the Pioneer to receive commands from the control computer.



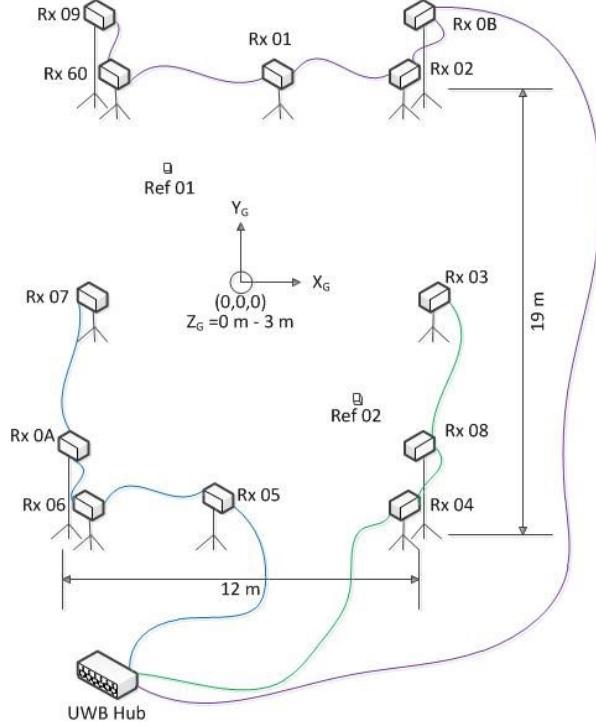
**Figure 5.4: Pioneer 3-AT land rover overview.**

## 5.4. Sensing System

The location of each of the robots, both the quadrotors and the Pioneer, were measured using a Sapphire Dart Ultra Wideband (UWB) system, consisting of a series of receivers placed around the perimeter of the test area at various heights and RFID tags, two of which are used as reference tags in the test area and two of which are placed on each robot. A picture of a receiver and an RFID tag is shown in Figure 5.5. Eleven receivers, with three separate “daisy chain” connections back to the UWB hub, are spaced around the perimeter of the 12 m by 19 m test area. This placement is shown in Figure 5.6.



**Figure 5.5: UWB receiver and RFID tag with a quarter for scale.**



**Figure 5.6: UWB system setup.**

The RFID tags transmit at 25 Hz while the receivers triangulate the position of each tag [60].

Each robot has two RFID tags attached: one on its extreme right and one on its extreme left. The average position of these two tags is used to calculate the position of the robot's center and heading. An error analysis for these tags can be found in the detailed system description in Appendix D.

## 5.5. Software

DataTurbine was used to make various data accessible on all of the networked computers used in these testbeds. DataTurbine software allows the user to upload data to a data engine and download the data to any computer than can connect to the same data engine [61]. In this testbed, DataTurbine is used across a network of computers with a single instance of DataTurbine run on Pippin, the Pioneer computer, and all data is sent to or received from this

single instance. Figure 5.7 illustrates the software layout used in this testbed. Data from the sensor system is received on Gollum [58] and imported into Matlab and then uploaded to DataTurbine and made available to the other computers in the network.

The control computer, Gandalf [58], downloads the robot position data and camera data from DataTurbine into Simulink, via Matlab and jmatlab (a software bridge between DataTurbine and Matlab developed at SCU's RSL [62]), where it runs the controller. The controller calculates the desired movement of each robot and then uploads individual robot commands to DataTurbine. In the case of the Pioneer, these robot commands were not determined by the controller, but by user-input joystick command.

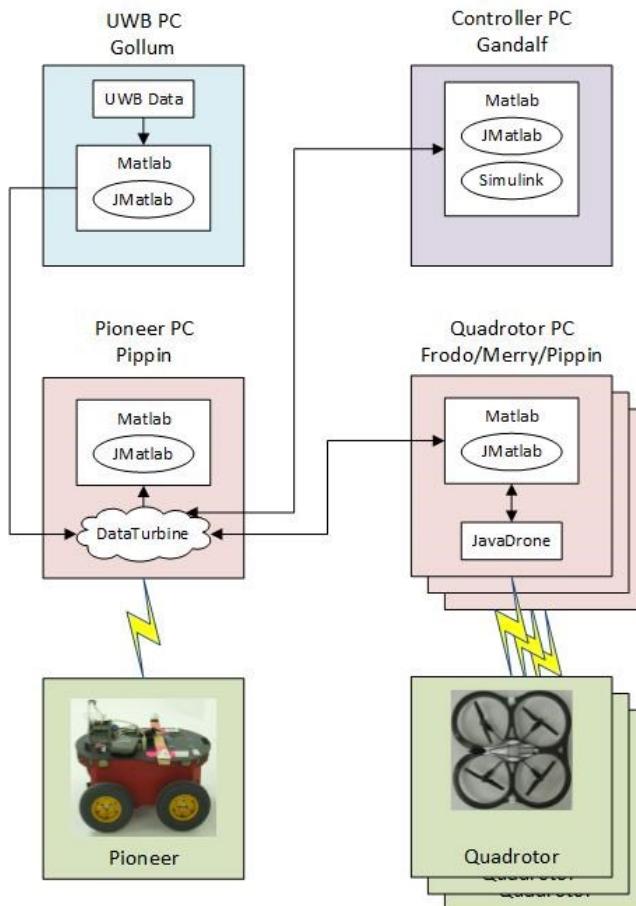


Figure 5.7: Software layout.

The individual robot computers download their commands from DataTurbine into Matlab, again via jmatlab, and send these commands to the robot. For the Pioneer, these commands are sent through DataTurbine over a modem. For the quadrotors, Javadrone is used to send the commands. Javadrone is an open source software package that provides a link between the AR.Drone onboard software and Java [63]. This software package is also used to retrieve camera data from the quadrotors and upload it to DataTurbine so that it is available to the control computer.

# **Chapter 6**

---

## **6. Stationary Results**

This chapter presents the physical proof of concept for angle of separation optimization presented in Chapter 3. The theoretical curves found in Section 3.5 are compared to experimental results achieved by three types of experiments: simulation, using a Pioneer as the tracked object, and using an ideal object as the tracked object. In all experiment types, the quadrotors were stationary in order to test only the effect of the angle of separation without control system inaccuracies adding a source of error.

### **6.1. Introduction**

The first experiment type tested the optimization theory in simulation to get an idea of what the real-world application of this theory would look like. Full explanations of the Simulink models used for the simulations can be found in Appendices E and F for two and three mobile tracking stations, respectively. In both simulations, the actual quadrotor sensor properties and limitations were used. In an effort to make the simulation as realistic as possible, the actual errors in the UWB system, maximum ( $x, y, z$ ) errors of  $(\pm 0.35, \pm 0.32, \pm 0.85)$  m, were also added to the robot position information. This step was taken to confirm that the errors in a physical system would not be greater than the improved position estimates achieved by changing the angle of separation between the mobile tracking stations.

Next, two series of physical experiments were performed and compared to the mathematical results. Both series of physical experiments were performed with stationary quadrotors; to mimic actual flight conditions, the quadrotors were statically mounted at their nominal flight height. Configurations with separation angles ranging from 10 degrees to 180

degrees in increments of 10 degrees were evaluated at a fixed radius of 2.83 m. Data was collected for two minutes at each location and the mean total distance between the actual Pioneer position and the estimated Pioneer position were measured.

## 6.2. Two Quadrotor Results

Results for two mobile tracking stations and a single tracked object have already been examined at Santa Clara University [16][17] and this work extends the technique both through a more rigorous mathematical approach and through its application to a new testbed. In all experiments, whether simulated or physical, AR.Drone 1.0 quadrotors and their forward-facing onboard cameras were used as the two mobile tracking stations while a single tracked object was used. The results were expected to follow the shape of the theoretical curve found in Figure 3.8 in Chapter 3.

### 6.2.1. *Simulation Results*

The following simulation results were obtained using the simulation presented in Appendix E. The sensor errors, matched to the physical errors observed during testing ( $\varepsilon_\theta = 5.7$  degrees and  $\varepsilon_r = 0.4$  m), and the UWB system errors position information are given in Table 6.1. Eighteen tests were performed using this simulation: one test every 10 degrees from 10 to 180 degrees. In each test, the angle of separation was changed by moving the mobile tracking stations farther apart. In order to keep them at a fixed radius of 2.83 m from the tracked object, the distance of the cluster center from the tracked object also had to be changed. The heading of each quadrotor was also changed with each test so that the quadrotors faced the tracked object from each position. The variables used in these tests can be seen in Table 6.2 where  $d$  is the distance between the cluster center and the tracked object.

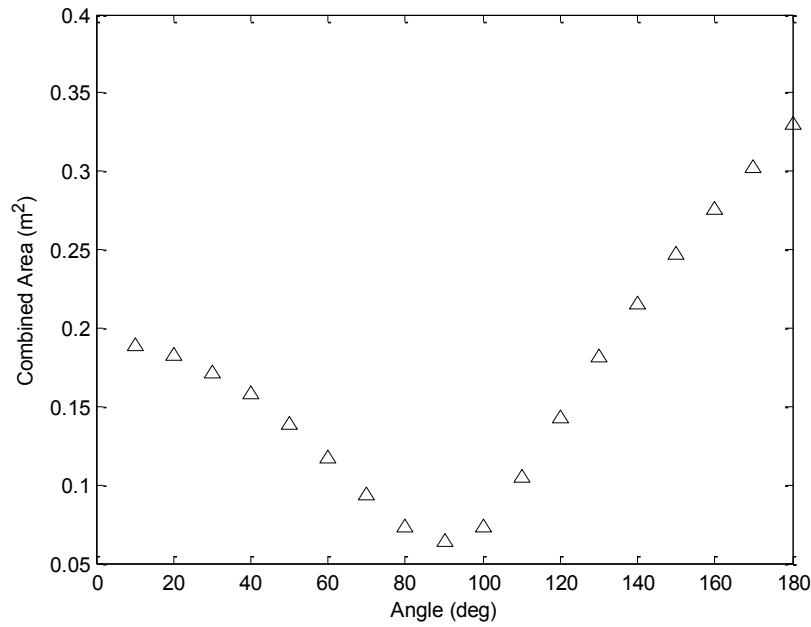
**Table 6.1: Sensor and position errors determined from physical tests.**

Variable	Value
$D$	$\pm 0.4$ m
$y_{center}$	$\pm 0.04$ m
$x$	$\pm 0.35$ m
$y$	$\pm 0.32$ m
$z$	$\pm 0.85$ m

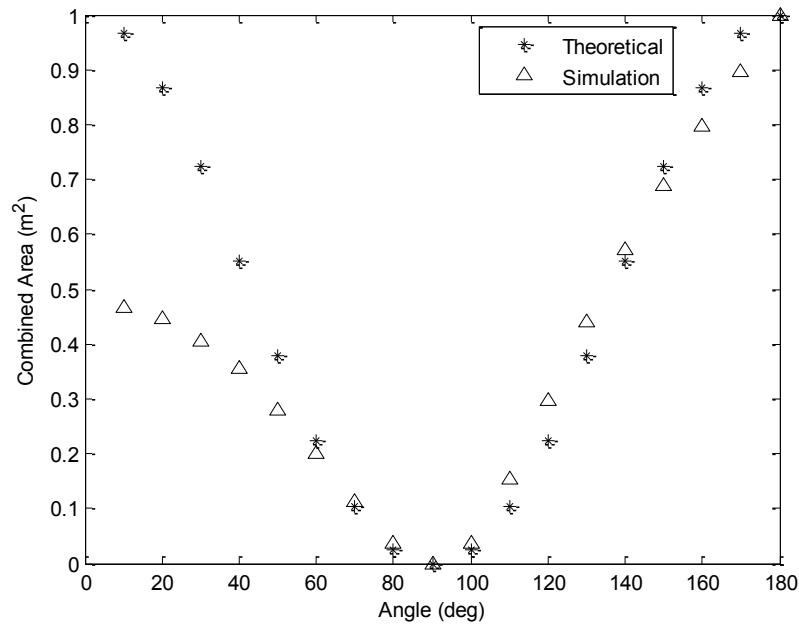
**Table 6.2: Variables used in the two mobile tracking station simulations.**

Test	Angle of Separation (deg)	z (m)	$\alpha$ (deg)	$\beta$ (deg)	$\varphi_1$ (deg)	$\varphi_2$ (deg)	p (m)	d (m)
1	10	1	-90	0	-175	175	0.49	2.82
2	20	1	-90	0	-170	170	0.98	2.79
3	30	1	-90	0	-165	165	1.46	2.73
4	40	1	-90	0	-160	160	1.94	2.66
5	50	1	-90	0	-155	155	2.39	2.56
6	60	1	-90	0	-150	150	2.83	2.45
7	70	1	-90	0	-145	145	3.25	2.32
8	80	1	-90	0	-140	140	3.64	2.17
9	90	1	-90	0	-135	135	4.00	2.00
10	100	1	-90	0	-130	130	4.34	1.82
11	110	1	-90	0	-125	125	4.64	1.62
12	120	1	-90	0	-120	120	4.90	1.42
13	130	1	-90	0	-115	115	5.13	1.20
14	140	1	-90	0	-110	110	5.32	0.97
15	150	1	-90	0	-105	105	5.47	0.73
16	160	1	-90	0	-100	100	5.57	0.49
17	170	1	-90	0	-95	95	5.64	0.25
18	180	1	-90	0	-90	90	5.66	0.00

The area of the 60% confidence interval error covariance matrix was then found for each of these tests, as in the theoretical results. The method used to calculate this area can be found in Appendix G. Figure 6.1 shows the results of these calculations for each simulation while Figure 6.2 shows the normalized results on the same plot as the normalized results of Figure 3.8.



**Figure 6.1:** Simulation results for two mobile tracking stations with identical sensors and a fixed radius.



**Figure 6.2:** Normalized simulation and theoretical results for two mobile tracking stations with identical sensors and a fixed radius.

The theoretical and simulation results match well, both reaching a minimum at 90 degrees and following a parabolic shape. The notable difference between these results is that the simulation results show less combined error ellipse area change per change in angle of separation than predicted by theory. This is thought to be due to the errors in the positioning system which are not taken into account in the theoretical model.

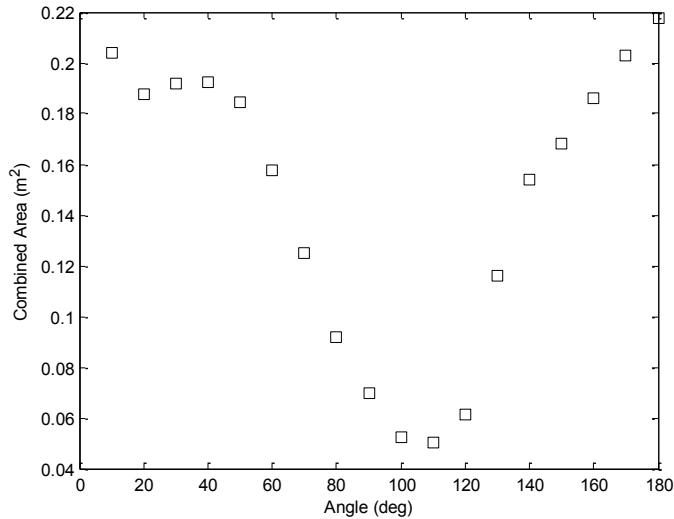
### 6.2.2. Pioneer Position Estimate

The same test was then performed using the physical testbed where the Pioneer itself was used as the tracked object and two quadrotors were used as the mobile tracking stations. Here, the angle of separation was also varied from 10 degrees to 180 degrees in 10 degree increments. A circle with a radius of 2.83 m, the ideal viewing distance for the quadrotor camera, was marked on the floor, as were the placements for the mobile tracking stations. To simulate flight conditions, the quadrotors were statically mounted at their nominal flight height. This setup can be seen in Figure 6.3. It is important to note that both the quadrotors and the Pioneer were stationary during these tests. The experiments were intended to determine only the effect of the placement of the quadrotors, not the efficacy of the control system.

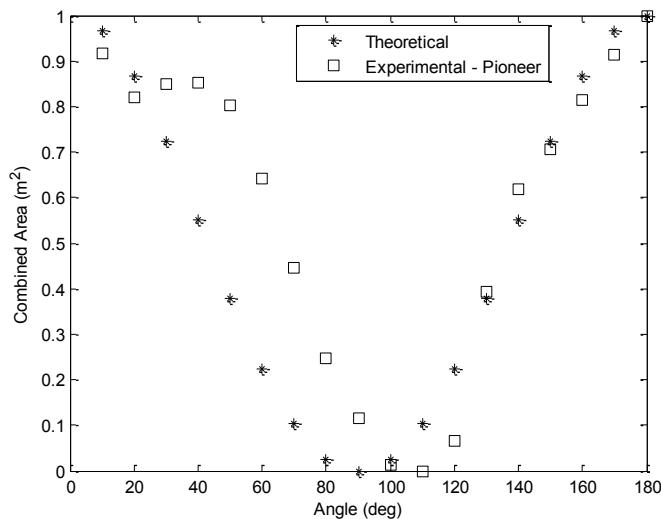


**Figure 6.3: Setup for testing the effect of the angle of separation on a system with two mobile tracking stations and the Pioneer as the tracked object.**

The formulas from Appendix G were used to calculate the 60% confidence interval error covariance matrix for these tests. The results themselves can be seen in Figure 6.4 while the normalized results can be seen compared to the normalized theoretical curve (see Figure 3.8) in Figure 6.5.



**Figure 6.4: Physical results with the Pioneer as tracked object and two AR.Drone 1.0 mobile tracking stations.**



**Figure 6.5: Normalized results of the theoretical curve and physical experimental results with the Pioneer as the tracked object and two identical mobile tracking stations.**

The theoretical shape is the same as for the mathematical results. Again, the general shape of the curve followed the theory. However, the angles below 90 degrees yielded a smaller change in the total distance error each time the angle of separation was changed than the theory predicted. This is believed to be because the RFID system error has a larger relative effect on the position estimations at small distances between the sensor systems. Additionally, the angle of separation with the minimum mean total distance error was found to be 110 degrees rather than the predicted 90 degrees. The cause of this deviation was posited to be the shape of the Pioneer itself. Figure 6.6 shows that the Pioneer features large wheels that can obscure large portions of the body of the Pioneer.



**Figure 6.6: Pioneer land rover used as the tracked object. A front view (left) and side view (right) are shown.**

#### *6.2.3. Ideal Object Position Estimate*

To determine whether the Pioneer's large wheels were responsible for the difference between the theory and the experimental results, a second series of tests was performed with two tracking stations and a uniform object that looked the same when viewed from any angle. A red ball with an apparent surface area similar to the side of the Pioneer was chosen as the uniform object. Figure 6.7 shows the Pioneer from the front and side next to this uniform object. The red

ball was placed in the center of the circle and held in place by a bespoke stand that did not obscure the red ball from any viewing angle. The setup for this series of tests can be seen in Figure 6.8.

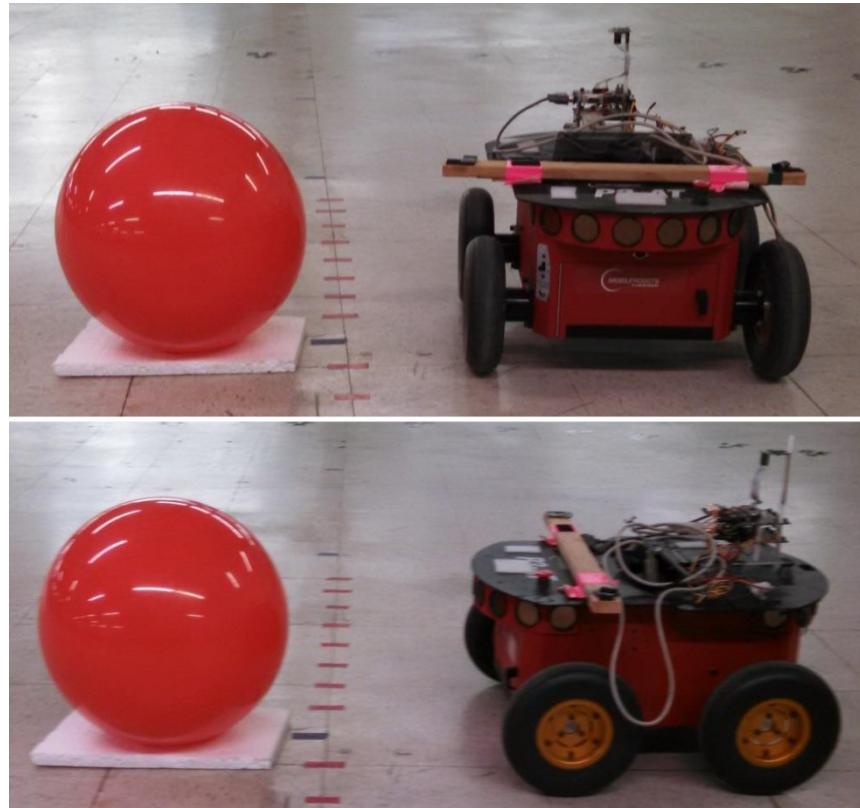


Figure 6.7: Front (top) and side (bottom) views of the Pioneer and ideal object.

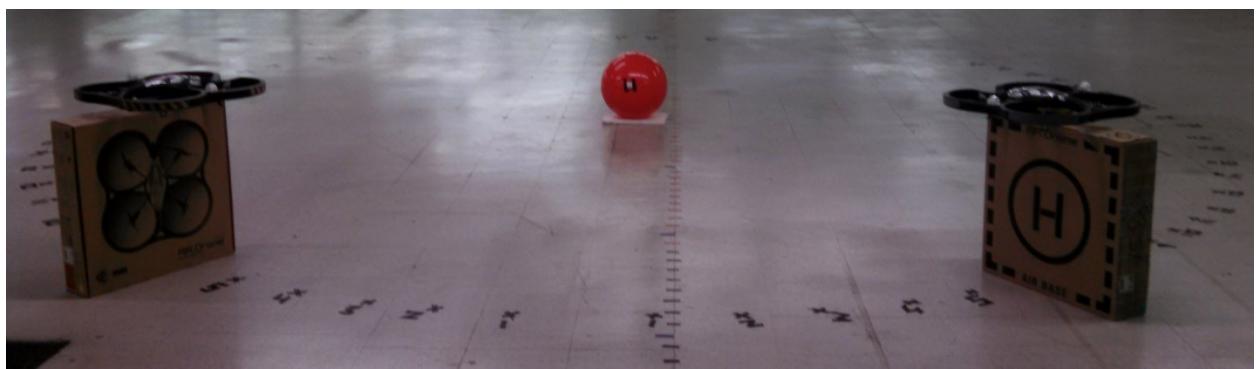
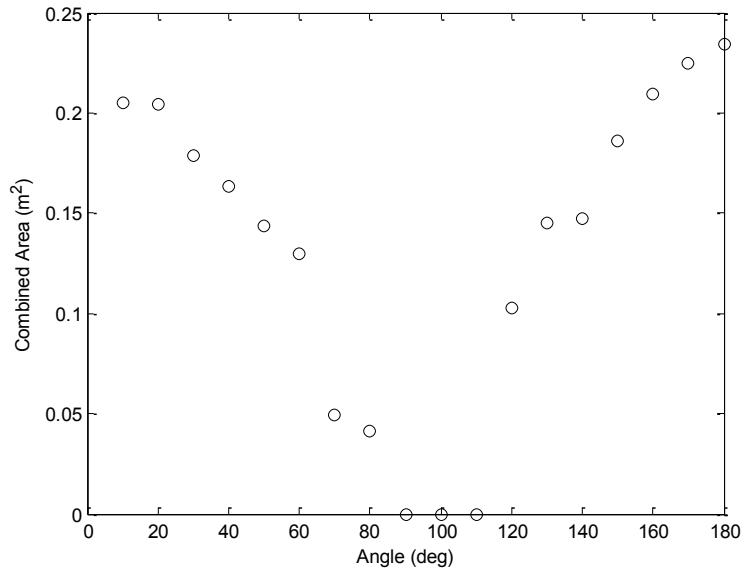
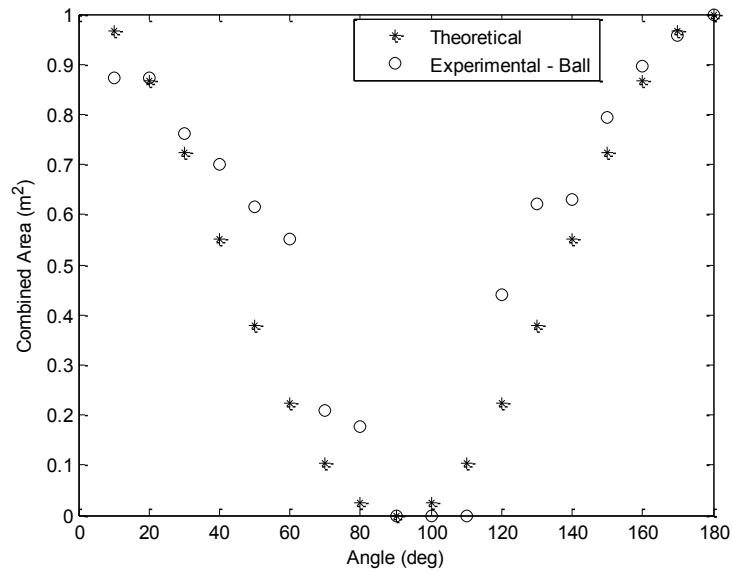


Figure 6.8: Setup for two AR.Drone 1.0s as mobile tracking stations and an ideal object as the tracked object.

The 60% confidence error covariance ellipse was found for each of these tests using the formulas from Appendix G. The results can be seen in Figure 6.9 while Figure 6.10 shows the normalized results along with the normalized theoretical results of Figure 3.8.



**Figure 6.9: Results using two AR.Drone 1.0s as the mobile tracking station and an ideal object as the tracked object.**



**Figure 6.10: Check this shit out.**

Here, the experimental results exhibit the same shape as the theoretical curve with the same lower slope at angles of separation below 90 degrees. However, the angle of separation that produced the minimum mean total distance error was found to be 100 degrees which is much closer to the theoretical minimum of 90 degrees, suggesting that the shift observed in the first series of physical experiments was mainly due to the shape of the Pioneer itself. The remaining deviation from theory is thought to be caused by the lack of uniform lighting in the test area and an exploration of this theory is suggested for future work.

### 6.3. Three Quadrotor Results

The optimization of the angle of separation in a cluster of three mobile tracking stations has not been examined at Santa Clara University before. This optimization is an extension of the rigorous mathematics developed to optimize the angle of separation in a cluster of two mobile tracking stations and is discussed more extensively in Chapter 3. The results presented here are a proof-of-concept for extending this technique to include clusters of three or more mobile tracking stations. Both the simulated and physical experiments presented here used the AR.Drone 1.0 quadrotors and their forward-facing onboard cameras as the three mobile tracking stations and a single tracked object. In this series of tests, it was expected that the results would follow those shown in Figure 3.13.

#### 6.3.1. *Simulation Results*

The simulation results presented here were found using the simulation discussed in Appendix F. As in the simulation for two mobile tracking stations, the physical errors and UWB system positioning errors were matched to those observed during physical tests. These errors are shown in Table 6.1. Since the results from three mobile tracking stations are symmetric about 180 degrees, only angles below 180 degrees were examined. This resulted in a total of 17 tests:

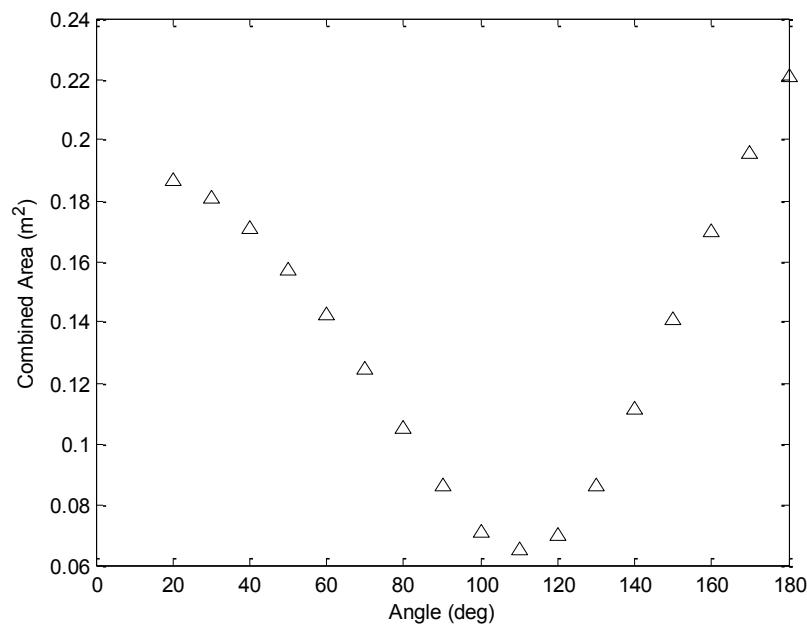
one test every 10 degrees from 20 degrees to 180 degrees. An angle of separation of 10 degrees was not tested because the quadrotors could not physically group so close together due to their hull diameter. For three mobile tracking stations, the angle of separation, defined as in Figure 3.12, was varied by changing  $\zeta$  in the cluster definition. Keeping the quadrotors at a fixed radius was a bit more complicated:  $p$ ,  $q$ , and the distance between the cluster center and the tracked object had to be changed for each angle of separation. The heading of Robot 1 remained constant since it did not move during the course of testing, but the headings of Robots 2 and 3 also were changed so that they always faced the tracked object. The resulting variable values can be seen in Table 6.3 and Table 6.4.

**Table 6.3: Angular variables used in the simulations with three mobile tracking stations.**

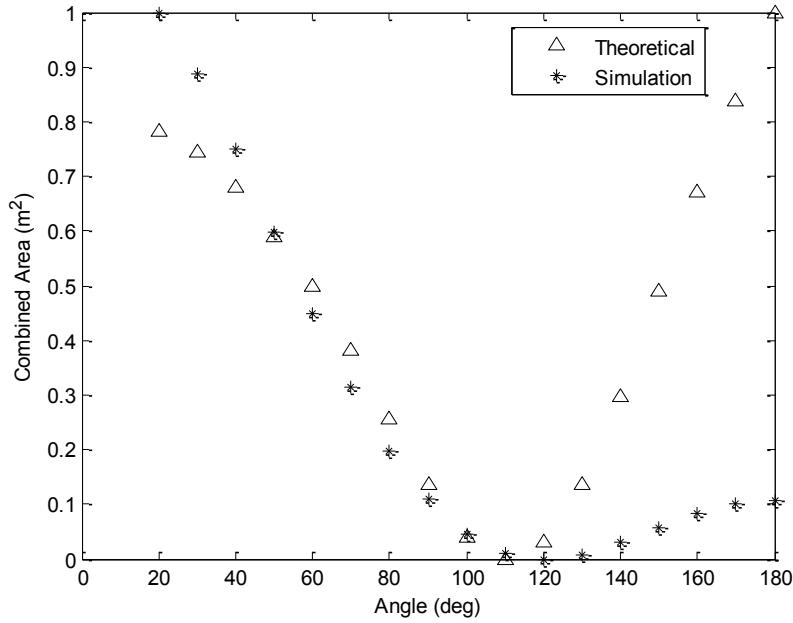
Test	Angle of Separation (deg)	$\alpha$ (deg)	$\beta$ (deg)	$\gamma$ (deg)	$\varphi_1$ (deg)	$\varphi_2$ (deg)	$\varphi_3$ (deg)	$\zeta$ (deg)
1	20	180	0	0	-90	-100	-80	170
2	30	180	0	0	-90	-105	-75	165
3	40	180	0	0	-90	-110	-70	160
4	50	180	0	0	-90	-115	-65	155
5	60	180	0	0	-90	-120	-60	150
6	70	180	0	0	-90	-125	-55	145
7	80	180	0	0	-90	-130	-50	140
8	90	180	0	0	-90	-135	-45	135
9	100	180	0	0	-90	-140	-40	130
10	110	180	0	0	-90	-145	-35	125
11	120	180	0	0	-90	-150	-30	120
12	130	180	0	0	-90	-155	-25	115
13	140	180	0	0	-90	-160	-20	110
14	150	180	0	0	-90	-165	-15	105
15	160	180	0	0	-90	-170	-10	100
16	170	180	0	0	-90	-175	-5	95
17	180	180	0	0	-90	180	0	90

**Table 6.4: Distance variables used in the simulations with three mobile tracking stations.**

Test	z (m)	p (m)	q (m)	d (m)
1	1	0.49	0.49	2.80
2	1	0.74	0.74	2.77
3	1	0.98	0.98	2.72
4	1	1.23	1.23	2.65
5	1	1.46	1.46	2.58
6	1	1.70	1.70	2.49
7	1	1.94	1.94	2.39
8	1	2.17	2.17	2.28
9	1	2.39	2.39	2.16
10	1	2.61	2.61	2.03
11	1	2.83	2.83	1.89
12	1	3.04	3.04	1.74
13	1	3.25	3.25	1.59
14	1	3.45	3.45	1.43
15	1	3.64	3.64	1.27
16	1	3.82	3.82	1.11
17	1	4.00	4.00	0.94



**Figure 6.11: Simulation results for three mobile tracking stations with identical sensors and a fixed radius.**



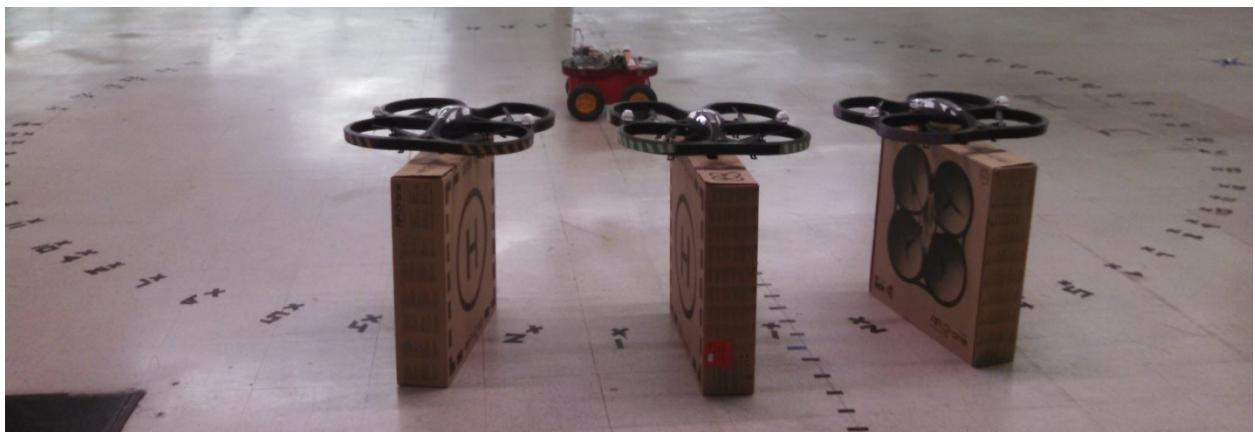
**Figure 6.12: Normalized simulation and theoretical results for three mobile tracking stations with identical sensors and a fixed radius.**

The formulas shown in Appendix G were used to calculate the area of the 60% confidence interval error covariance ellipse. The simulation results are shown in Figure 6.11 while Figure 6.12 shows the normalized simulation results on the same plot as the normalized results of Figure 3.13.

The theoretical and simulation results match fairly well. The change in the combined area ellipse per change in angle of separation is a bit less than predicted by theory in the first two tests and more than predicted by theory at angles above 120 degrees. The greater than expected change in angles above 120 degrees is thought to be due positioning error which was not taken into account by the theory. The simulation found a minimum at 110 degrees rather than 120 degrees. However, the difference between these two values was less than  $0.005 \text{ m}^2$ , so this may be due to a particularly large error in a single reading.

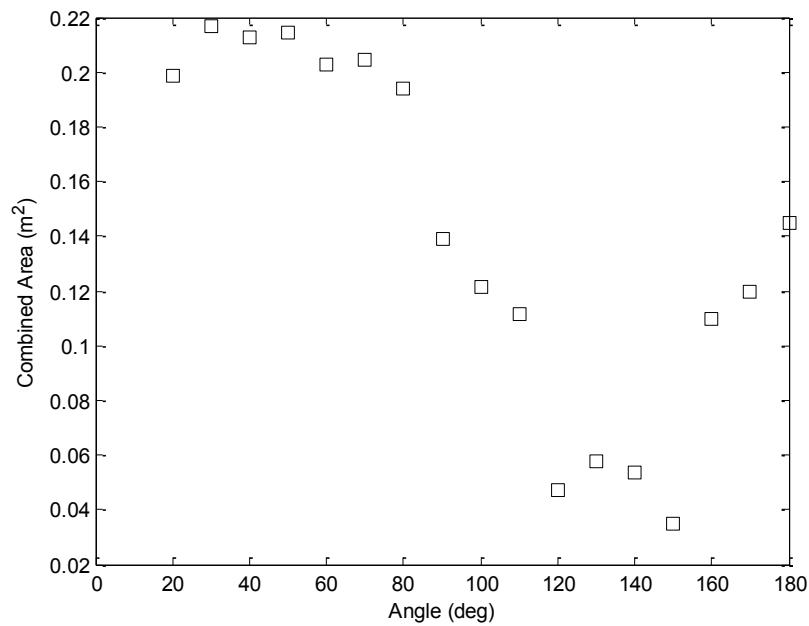
### 6.3.2. Pioneer Position Estimate

The same series of experiments were then performed using the physical testbed: three AR.Drone 1.0 quadrotors were used as the mobile tracking stations and a Pioneer was used as the tracked object. The same circle from the two mobile tracking station tests was used where the first robot was placed perpendicular to the side of the Pioneer, the second robot was placed to the right of the first robot, and the third robot was placed to the left of the first robot. This setup can be seen in Figure 6.13. As in the case for two tracking stations, the robots were not moving in order to test only the effect of the changing angle of separation without induced noise from the control system.

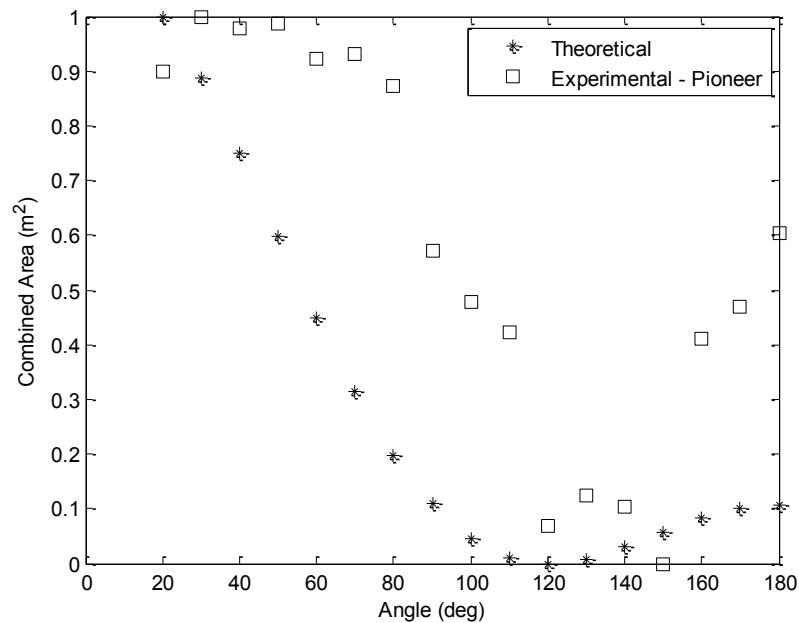


**Figure 6.13: Setup for testing the effect of the angle of separation on a system with three mobile tracking stations and the Pioneer as the tracked object.**

The area of the 60% confidence error ellipse was calculated using the formulas discussed in Appendix G. Figure 6.14 shows the results of this series of experiments and Figure 6.15 shows the normalized results on the same plot as the normalized theoretical results from Figure 3.13.



**Figure 6.14: Physical results with the Pioneer as the tracked object and three AR.Drone 1.0s as the mobile tracking stations.**



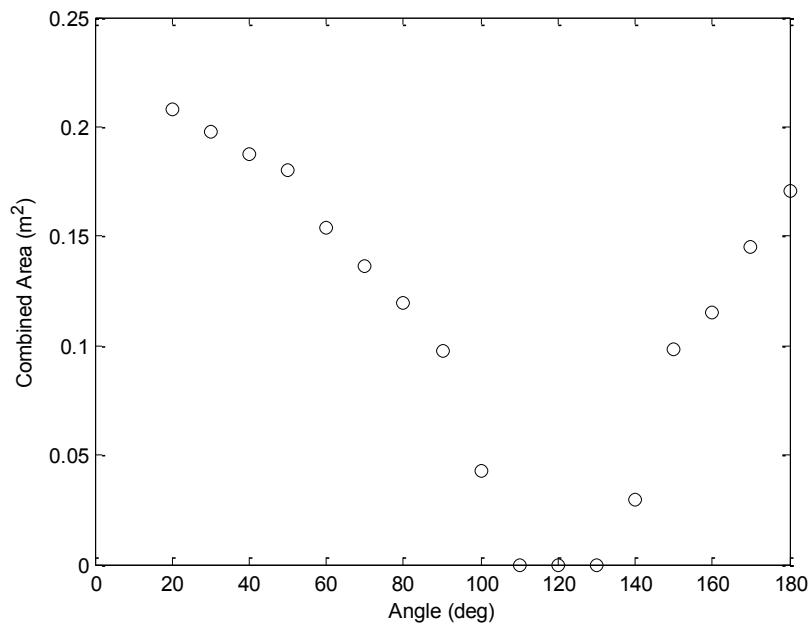
**Figure 6.15: Normalized results of the theoretical curve and physical experimental results with the Pioneer as the tracked object and three identical mobile tracking stations.**

These physical results initially showed a slower decrease in the area of the combined ellipse per change in angle of separation than predicted by theory, but resulted in a similar minimum value of 150 degrees. As in the two quadrotor case, the discrepancy between the theoretical minimum of 120 degrees and the experimental minimum of 150 degrees was believed to be due to the shape of the Pioneer. After reaching this minimum value, the area of the combined error ellipse increased more sharply than predicted by the theory. This is thought to be because of differences in the background of the test environment. Testing in an area with uniformly painted walls and uniformly distributed building structures is recommended for future work.

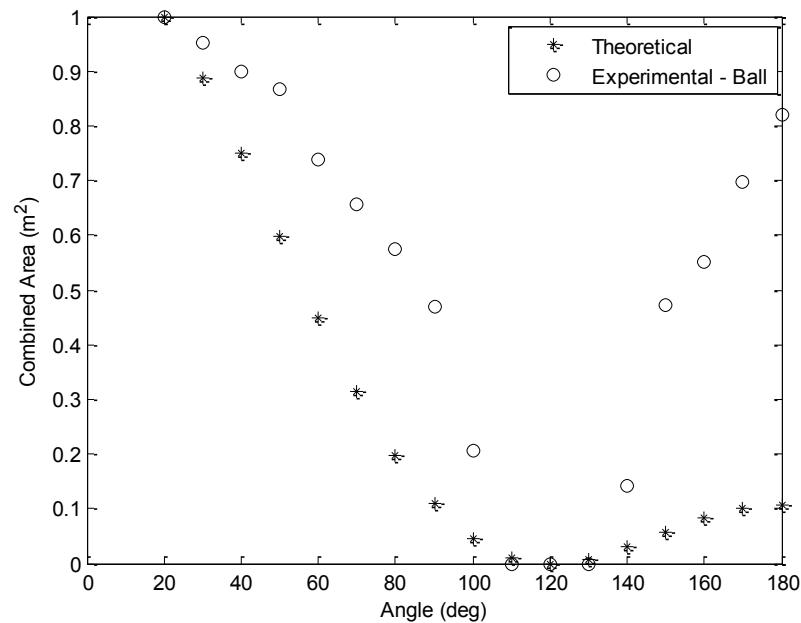
### *6.3.3. Ideal Object Position Estimate*

A final series of stationary experiments was performed using three AR.Drone 1.0 mobile tracking stations. In this series of tests, the Pioneer was replaced by a red ball (an ideal object) as the tracked object in order to remove the effect of the shape of the tracked object from the experimental results. The same series of tests were performed as in the previous section: the angle of separation was varied in 10 degree increments between 20 and 180 degrees and the area of the 60% confidence interval ellipse was found using the method described in Appendix G. Figure 6.16 shows the results of this series of tests and Figure 6.17 shows the normalized results on the same plot as the normalized results from Figure 3.13.

The results demonstrate a smooth decrease down to a minimum value of 120 degrees and then a sharper increase in the area of the combined error ellipse than predicted by the theory. However, the minimum found by this series of experiments was the same as the theoretical minimum of 120 degrees, unlike when the Pioneer was used as the tracked object. This confirms that most of the discrepancy in the minimum was due to the shape of the Pioneer.



**Figure 6.16:** Results using three AR.Drone 1.0s as the mobile tracking stations and an ideal object as the tracked object.



**Figure 6.17:** Normalized results of the theoretical curve and physical experimental results with an ideal object as the tracked object and three identical mobile tracking stations.

The application domain for this methodology assumed a maximum target speed of 0.315 m/s in the 19 by 12 m test area. Two solutions are presented here: one solution for the case with two mobile sensor systems and one solution for the case with three sensor systems. The case with two sensor systems resulted in a mean error of 0.68 m for the experimental results with the Pioneer. For the three sensor system case, the mean experimental error was even less at 0.6 m. This accuracy is sufficient to keep the object in view of the sensing systems, allowing for continued tracking of the object.

In general, the experimental results were found to match the theory within physical limitations. In the worst case scenario, two quadrotors with a separation angle of 180 degrees tracking a Pioneer, the mean total distance error was less than 1.5 m. This distance was less than the field of view of the quadrotors at a distance of 2.83 m from the tracked object, meaning that the Pioneer could still be correctly located even after such a large estimation error. The best case scenario for the experimental results, three quadrotors with a separation angle of 120 degrees tracking a ball, had a mean total distance error of 0.1 m which was very close in a test area measuring approximated 19 by 12 m. This allowed further experimentation to be performed with moving mobile tracking stations, as discussed in the next chapter.

# **Chapter 7**

---

## **7. Controlled Physical Experimental Results**

The experiments presented in this chapter are an extension of the experiments presented in the previous chapter. In this chapter, the mobile tracking stations move and are controlled to maintain the ideal formation while tracking a Pioneer using both two and three mobile tracking stations. In some experiments with two mobile tracking stations, the Pioneer is moving according to user-input joystick control. The results of each of these experiments are presented in this chapter.

### **7.1. Introduction**

The experiments in this chapter are organized into two main categories: tests with two mobile tracking stations and tests with three mobile tracking stations. Experiments were performed in both categories where the Pioneer was stationary and the mobile tracking stations moved under control of the cluster controller. The individual controllers used will be discussed in the next sections, but it is important to note that the controller performed computations and sent out commands at a rate of 8 Hz. The UWB tags used were 25 Hz tags, but, in practice, the more tags that were added to the system, the longer it took for the receivers to determine the positions of each of the tags. The UWB software, which was not accessible, does not report tag positions until it has received a reading for each tag. Thus, the number of tags had a nontrivial effect on the UWB system efficiency.

The AR.Drones were limited to a speed of 1 m/s (20% of their maximum speed), which resulted in a maximum change in position of 0.125 m each time step. The maximum radial error of the position estimate for the Pioneer observed during stationary testing at the ideal angle of

separation was 0.57 m. During this set of experiments, the Pioneer was limited to a speed of 0.14 m/s (20% of its maximum speed) and can move a maximum distance of 0.0175 m in a single time step. The control system had a pointing error of 5.7 degrees, as discussed in Chapter 3. At 2.83 m, this angular difference resulted in a distance error of 0.28 m. The AR.Drone 1.0's forward-facing onboard camera can see an area approximately 2.26 m wide at a distance of 2.83 m. This means that if the maximum errors all occur when the UWB system is operating below 8 Hz (missing a single controller time step), the quadrotor can still see the Pioneer since it will be 1.0 m from the center point. However, if the UWB system was operating more slowly than 4 Hz, missing two time steps for the controller, the Pioneer could be as much as 1.13 m from the center of the controller, just outside the camera's range. If the quadrotors cannot see the Pioneer, the assumption is made that the Pioneer is at its last known position. If the Pioneer is stationary, the quadrotors will be able to locate it again since the Pioneer will indeed be at its last known position. However, if the Pioneer is moving, this is unlikely. The longer the quadrotors cannot find the Pioneer, the more likely it is that they will not be able to find it again, especially if the UWB system continues to operate below 4 Hz and the Pioneer continues to move.

This is an issue because using three mobile tracking stations require the use of eight RFID tags, two for each mobile tracking station and two for the Pioneer. This number of tags often resulted in operating speeds below 4 Hz, enabling tracking when the tracked object was stationary, but not when it was moving. It is recommended that tracking with three objects be tested on a slower platform that can handle slower data rates or that a search algorithm is developed to recover from loss of the tracked object.

## 7.2. Two Quadrotor Results

The experiments presented in this section were performed with two AR.Drone 1.0s as the mobile tracking stations and a Pioneer as the tracked object. Two series of tests were performed: one with a stationary Pioneer to show that the control of the mobile tracking stations was sufficient to allow continuous tracking and one with a moving Pioneer. These experiments were performed with the controllers shown in Figure 7.1 and Figure 7.2. The cluster controller in the first figure is the same as the cluster controller presented in Section 4.6 implemented in Simulink. The green blocks represent the sensor input, the pink blocks represent the cluster configuration manager, the yellow blocks the PID controller, and the orange blocks the cluster control itself. The Pioneer controller presented in Figure 7.1 is used only in the experiments with the moving Pioneer. It is a separate system from the cluster controller and no data is shared between the two controllers. The position of the Pioneer is recorded for every test, whether it remains stationary or moves, so that the tracking accuracy can be quantified.

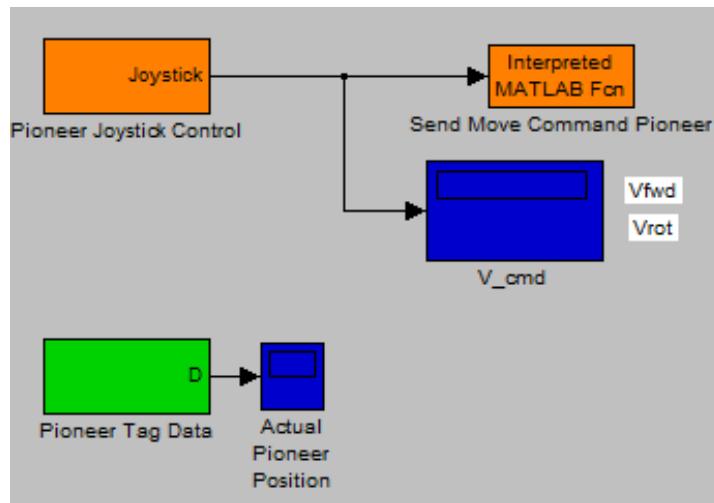


Figure 7.1: Pioneer user-input joystick control and position tracking.

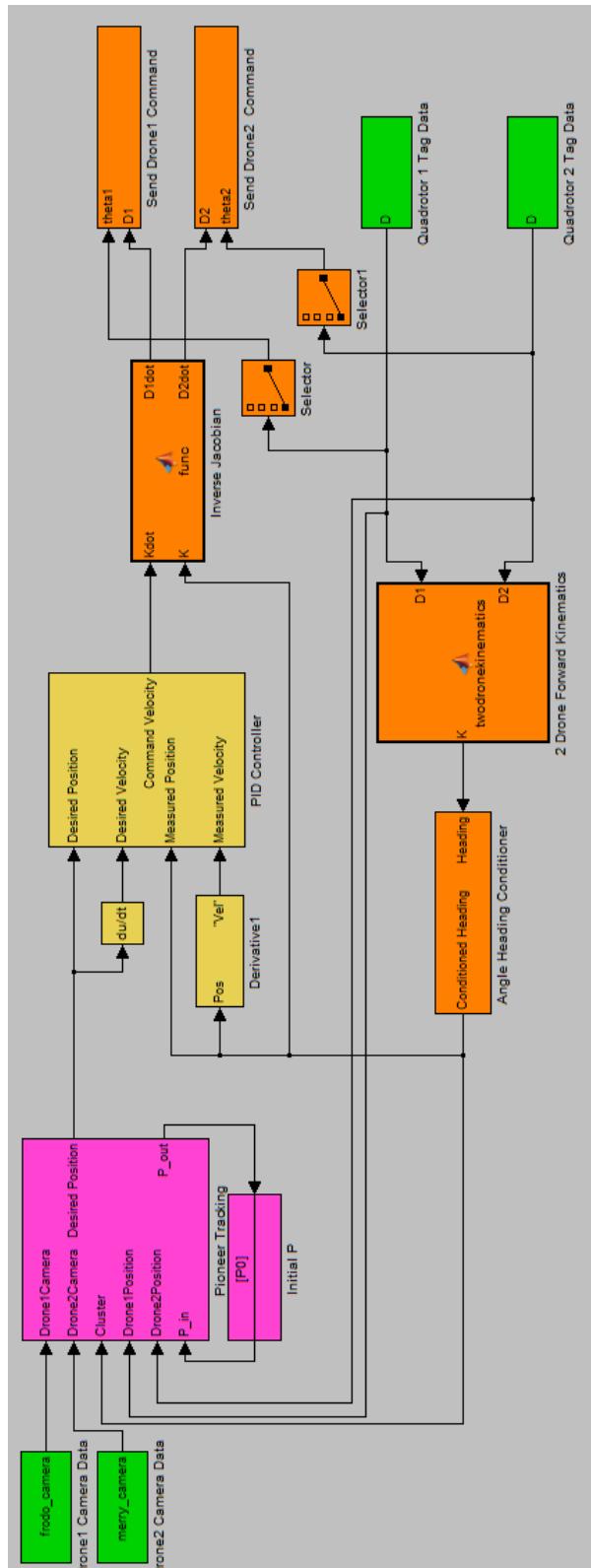
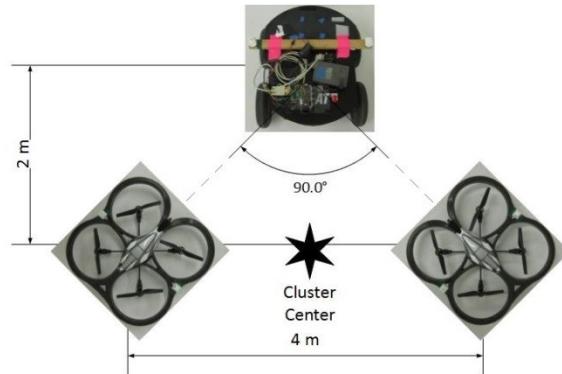


Figure 7.2: Cluster controller for two mobile tracking stations.

### 7.2.1. Stationary Pioneer

The first series of experiments presented here was performed using two mobile tracking stations with the ideal angle of separation of 90 degrees. In order to implement this angle of separation at the best viewing distance, the quadrotors were kept 4 m apart from each other and the cluster center was kept 2 m away from the Pioneer. This configuration, shown in Figure 7.3, was maintained throughout the experiment.



**Figure 7.3: Optimal two quadrotor configuration.**

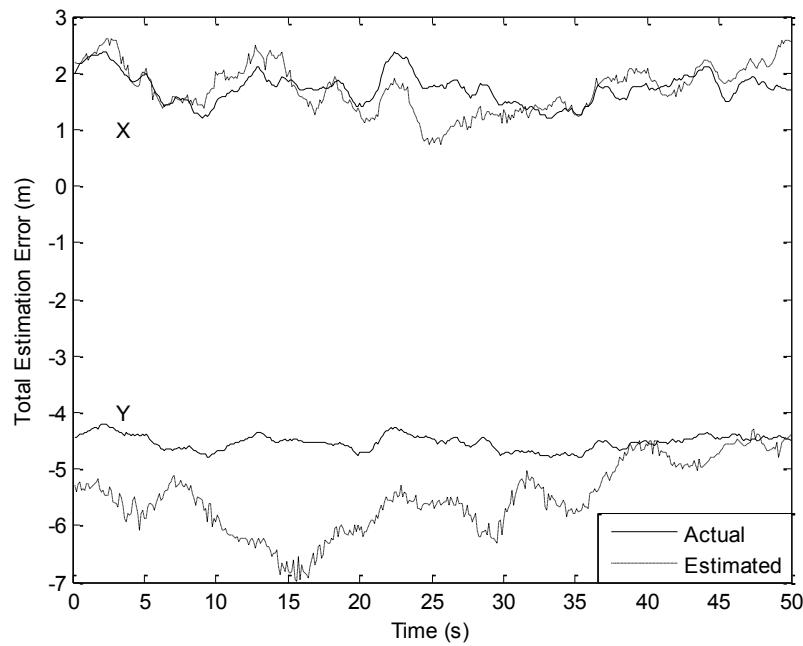
Each test is illustrated with a plot that shows the desired ( $x, y$ ) position of the cluster center compared to the actual ( $x, y$ ) position of the cluster center in the global coordinate frame. A table that summarizes the position estimation and cluster control parameters is also included. This table lists the minimum, maximum, mean, standard deviation, and the root mean square error of each parameter.

**Table 7.1: Location estimate summary for the first stationary Pioneer test with two mobile tracking stations.**

	X (m)	Y (m)	Total (m)
<b>Min Error</b>	0.00	0.00	0.09
<b>Max Error</b>	1.04	2.51	2.52
<b>Mean Error</b>	0.28	0.99	1.08
<b>Error Standard Deviation</b>	0.23	0.61	0.57
<b>Root Mean Squared Error</b>	0.36	1.16	1.22

**Table 7.2: Control variable summary for the first stationary Pioneer test with two mobile tracking stations.**

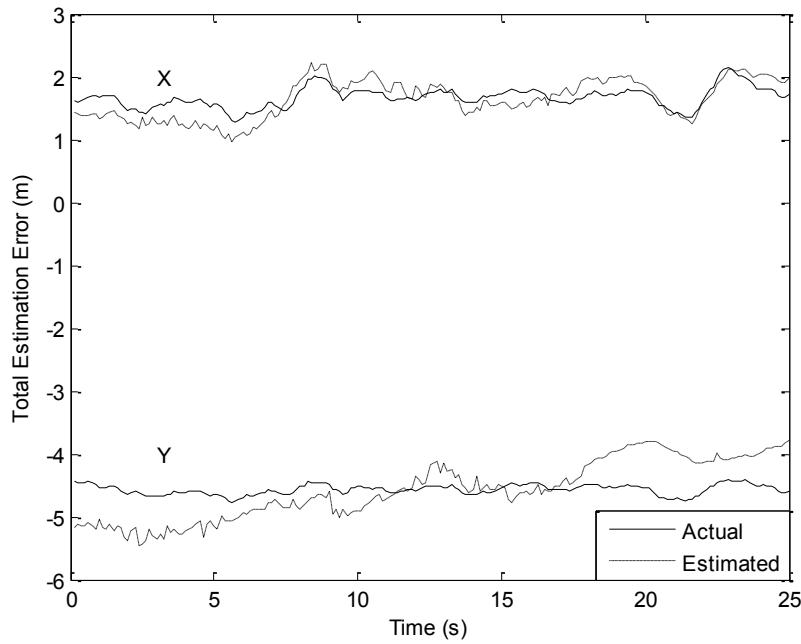
	X (m)	Y (m)	Z (m)	$\alpha$ (deg)	$\beta$ (deg)	$\varphi_1$ (deg)	$\varphi_2$ (deg)	P (m)
<b>Min Error</b>	0.00	0.00	0.01	0.01	0.01	0.06	0.00	0.00
<b>Max Error</b>	0.29	0.36	0.60	22.46	7.43	133.1 5	44.93	2.50
<b>Mean Error</b>	0.06	0.08	0.22	5.25	1.61	11.64	12.65	0.78
<b>Error Standard Deviation</b>	0.05	0.07	0.14	4.56	1.28	16.68	10.10	0.54
<b>Root Mean Squared Error</b>	0.08	0.11	0.26	6.95	2.05	20.33	16.18	0.95



**Figure 7.4: This plot shows the actual and estimate Pioneer positions throughout the first experiment with two mobile tracking stations and a stationary Pioneer.**

Figure 7.4 show the results for the first stationary Pioneer test with two mobile tracking stations while Table 7.1 and Table 7.2 provide a summary of these results. The location

estimation was accurate enough to allow the quadrotors to locate the Pioneer after a mean incorrect position estimate, but was still quite high. This is believed to be due to the very high  $\varphi_1$  maximum error which can occur because the yaw correction for the quadrotors is so fast that it is difficult to keep the quadrotors pointing at a single object. Limiting the yaw speed increased the time it took for the quadrotors to return to face the tracked object, so it was necessary to find a balance between the correction speed and overshooting the target.



**Figure 7.5:** This plot shows the actual and estimate Pioneer positions throughout the second experiment with two mobile tracking stations and a stationary Pioneer.

**Table 7.3: Location estimation summary for second stationary Pioneer test with two mobile tracking stations.**

	X (m)	Y (m)	Total (m)
<b>Min Error</b>	0.00	0.01	0.04
<b>Max Error</b>	0.47	0.86	0.87
<b>Mean Error</b>	0.19	0.39	0.45
<b>Error Standard Deviation</b>	0.10	0.24	0.22
<b>Root Mean Squared Error</b>	0.21	0.46	0.50

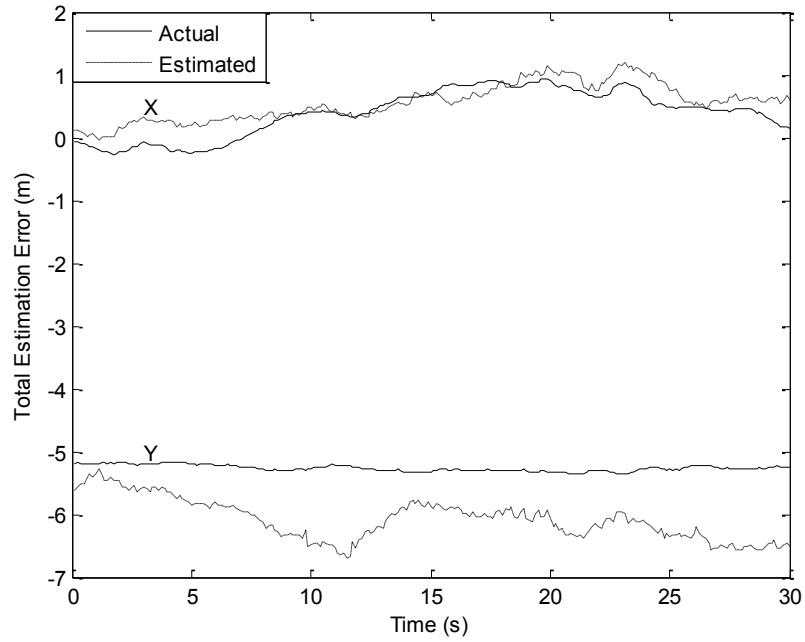
**Table 7.4: Control variable summary for second stationary Pioneer test with two mobile tracking stations.**

	X (m)	Y (m)	Z (m)	$\alpha$ (deg)	$\beta$ (deg)	$\varphi_1$ (deg)	$\varphi_2$ (deg)	P (m)
<b>Min Error</b>	0.00	0.00	0.14	0.56	0.03	0.05	0.36	0.00
<b>Max Error</b>	0.16	0.32	0.73	13.75	7.22	35.10	57.28	1.40
<b>Mean Error</b>	0.05	0.07	0.39	7.56	2.42	12.79	11.27	0.65
<b>Error Standard Deviation</b>	0.04	0.07	0.15	3.51	1.58	9.07	11.68	0.37
<b>Root Mean Squared Error</b>	0.06	0.10	0.42	8.33	2.89	15.66	16.21	0.75

The results of the second stationary test are shown in Figure 7.5 and the results are summarized in Table 7.3 and Table 7.4. Although the maximum errors for  $\phi_1$  and  $\phi_2$  were higher than desired, the mean error was acceptable. The high errors were seen because the yaw rate for the quadrotors was so fast that it was difficult to keep the quadrotors pointing at a single object. Limiting the yaw rate increased the time it took for the quadrotors to turn towards the tracked object, so it was necessary to find a balance between a fast response time and overshooting the target. Nonetheless, the quadrotor tracking performance was well inside the camera field of view, allowing the mobile tracking station to recover after an incorrect location estimate.

### 7.2.2. Moving Pioneer

Next, the same setup with two mobile tracking stations was used while the Pioneer moved independently from the cluster. In order to ensure that the Pioneer motion was not preprogrammed into the controller, the Pioneer was controlled by a user-input joystick control that did not share any data with the cluster controller itself. The cluster was maintained in the same ideal configuration as in the previous section since the ideal configuration depends only on the sensor systems, not the tracked object.



**Figure 7.6:** This plot shows the actual and estimate Pioneer positions throughout the first experiment with two mobile tracking stations and a moving Pioneer.

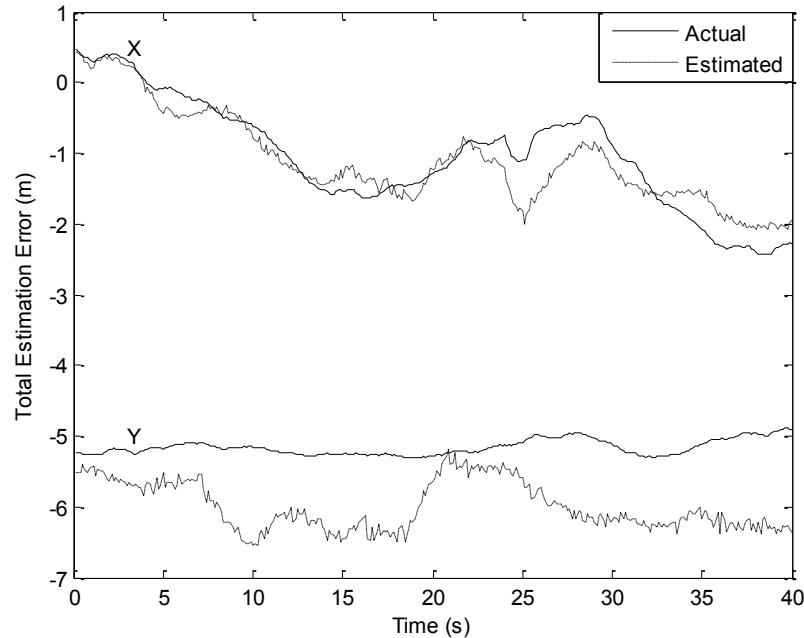
**Table 7.5:** Location estimation summary for the first moving Pioneer test with two mobile tracking stations.

	X (m)	Y (m)	Total (m)
<b>Min Error</b>	0.00	0.00	0.02
<b>Max Error</b>	0.50	0.96	0.96
<b>Mean Error</b>	0.21	0.36	0.46
<b>Error Standard Deviation</b>	0.14	0.25	0.21
<b>Root Mean Squared Error</b>	0.25	0.44	0.51

**Table 7.6:** Control variable summary for the first moving Pioneer test with two mobile tracking stations.

	X (m)	Y (m)	Z (m)	$\alpha$ (deg)	$\beta$ (deg)	$\varphi_1$ (deg)	$\varphi_2$ (deg)	P (m)
<b>Min Error</b>	0.00	0.00	0.00	0.01	0.03	0.02	0.02	0.00
<b>Max Error</b>	0.20	0.28	0.23	7.15	6.42	33.84	33.98	0.97
<b>Mean Error</b>	0.05	0.10	0.07	1.89	2.11	7.91	7.36	0.30
<b>Error Standard Deviation</b>	0.04	0.07	0.05	1.76	1.48	6.53	5.86	0.24
<b>Root Mean Squared Error</b>	0.07	0.12	0.09	2.58	2.58	10.25	9.41	0.38

The results of the first experiment with a moving Pioneer are shown in Figure 7.6 and summarized in Table 7.5 and Table 7.6. The control variables were well controlled in this test, resulting in a mean position error of 0.46 m, almost the same as the second stationary test. This is due to the level of control of the quadrotors' locations and headings. The greater accuracy in controlling the quadrotors results in greater position estimate accuracy.



**Figure 7.7:** This plot shows the actual and estimate Pioneer positions throughout the second experiment with two mobile tracking stations and a moving Pioneer.

**Table 7.7:** Location estimation summary for the second moving Pioneer test with two mobile tracking stations.

	X (m)	Y (m)	Total (m)
<b>Min Error</b>	0.00	0.00	0.01
<b>Max Error</b>	0.96	0.96	1.03
<b>Mean Error</b>	0.25	0.43	0.53
<b>Error Standard Deviation</b>	0.20	0.25	0.25
<b>Root Mean Squared Error</b>	0.32	0.50	0.59

**Table 7.8: Control variable summary for the second moving Pioneer test with two mobile tracking stations.**

	X (m)	Y (m)	Z (m)	$\alpha$ (deg)	$\beta$ (deg)	$\varphi_1$ (deg)	$\varphi_2$ (deg)	P (m)
<b>Min Error</b>	0.00	0.00	0.00	0.03	0.03	0.03	0.00	0.00
<b>Max Error</b>	0.19	0.35	0.98	13.73	4.68	25.98	41.95	1.18
<b>Mean Error</b>	0.04	0.12	0.11	5.15	1.45	8.11	11.60	0.44
<b>Error Standard Deviation</b>	0.03	0.07	0.14	3.83	1.06	6.18	9.61	0.29
<b>Root Mean Squared Error</b>	0.05	0.14	0.18	6.41	1.80	10.19	15.05	0.52

The results of the second moving Pioneer experiment can be seen in Table 7.7, Table 7.8 and Figure 7.7. At approximately 28 second, the Pioneer turns and the cluster turns to follow. Since the control variables in this test were well controlled, the mean position estimation error was 0.53 m, only 0.08 m greater than achieved in the second stationary Pioneer test. This level of accuracy resulted from the low mean error of the control variables and confirmed that the better the mobile tracking stations' positions and headings were controlled, the more accurate the position estimate of the tracked object.

### 7.3. Three Quadrotor Results

In this section, the experiments were performed using three AR.Drone 1.0s as the mobile tracking stations and a Pioneer as the tracked object. Only tests with a stationary Pioneer were performed in this configuration, as discussed in the introduction. This series of tests served to confirm that the controller is sufficient to allow for the tracking of a real object using controlled mobile tracking stations. The controller shown in Figure 7.8 was used to control the mobile tracking stations and is identical to the controller presented in Figure 7.2 with the addition of a third mobile tracking station. The position of the Pioneer was recorded using the model shown in Figure 7.1 even though the controller itself was not used. As in the two mobile tracking

station case, the position of the Pioneer was recorded in order to quantify the accuracy of the controller.

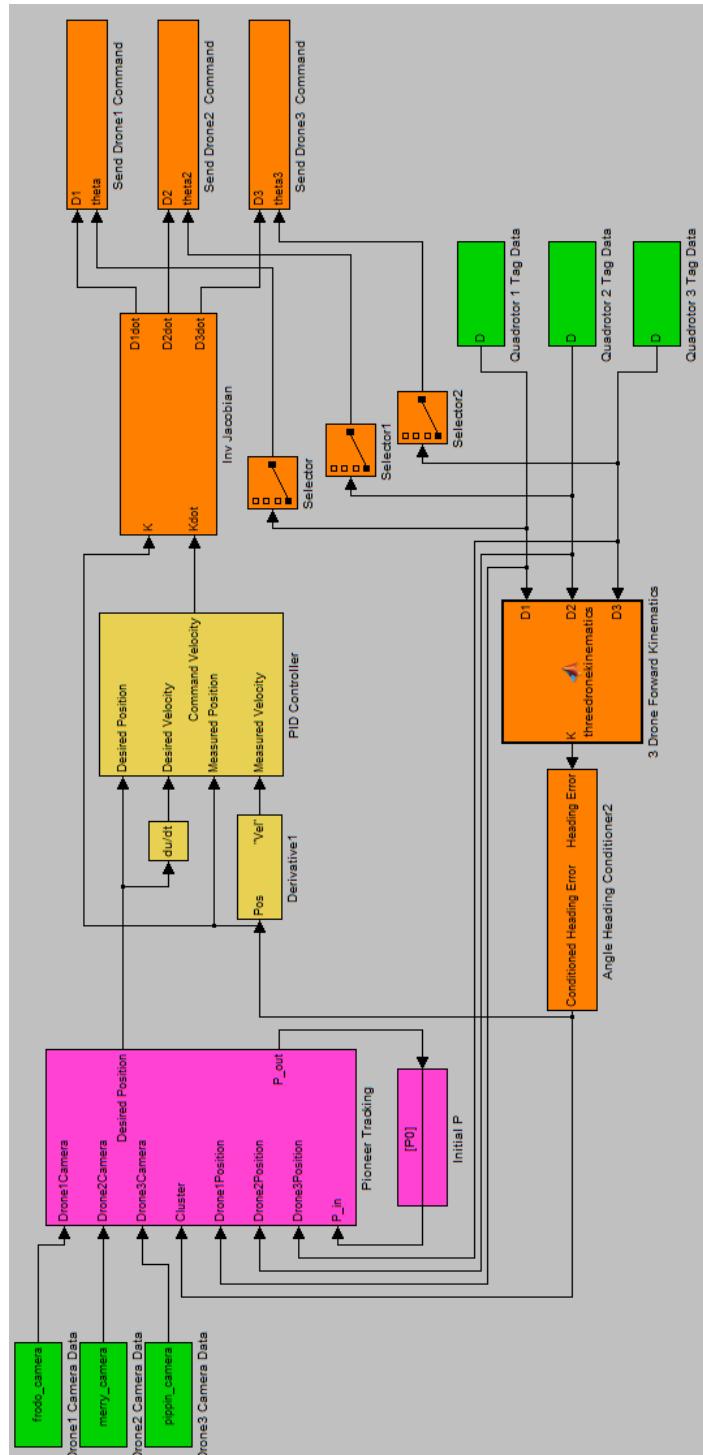
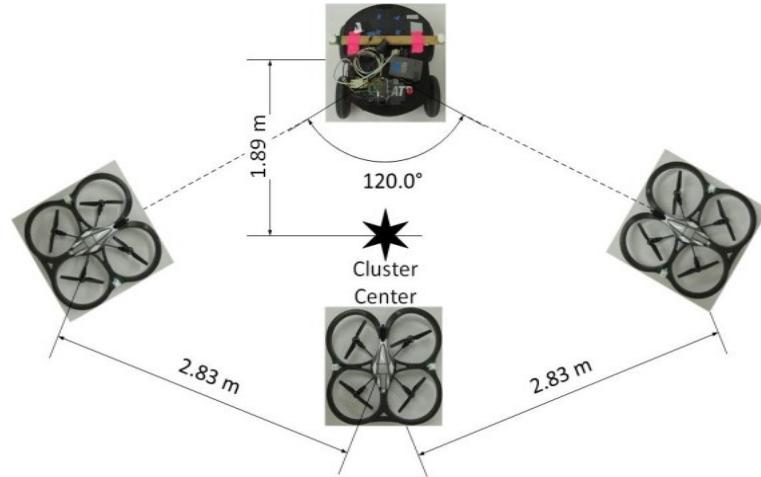
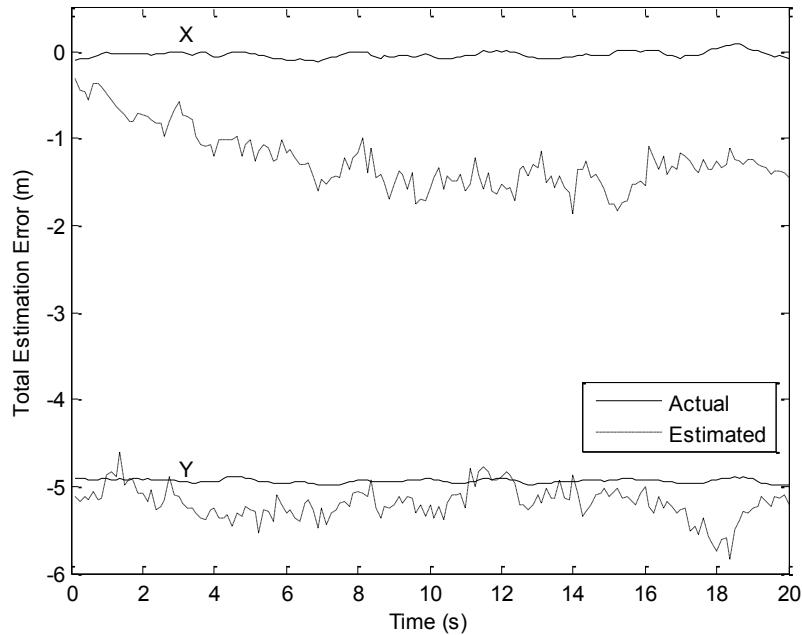


Figure 7.8: Three mobile tracking station cluster controller.

In the next series of tests, three mobile tracking stations were used to follow a Pioneer robot. The cluster was kept at the ideal angle of separation of 120 degrees in the configuration shown in Figure 7.9 throughout the test.



**Figure 7.9: Optimal three quadrotor configuration.**



**Figure 7.10: This plot shows the actual and estimated Pioneer positions throughout the first experiment with three mobile tracking stations and a stationary Pioneer.**

Table 7.9: First stationary Pioneer test with three mobile tracking stations location estimation summary.

	X (m)	Y (m)	Total (m)
<b>Min Error</b>	0.20	0.01	0.29
<b>Max Error</b>	1.83	0.93	1.83
<b>Mean Error</b>	1.21	0.27	1.25
<b>Error Standard Deviation</b>	0.34	0.16	0.33
<b>Root Mean Squared Error</b>	1.26	0.32	1.30

Table 7.10: First stationary Pioneer test with three mobile tracking stations distance control variable summary.

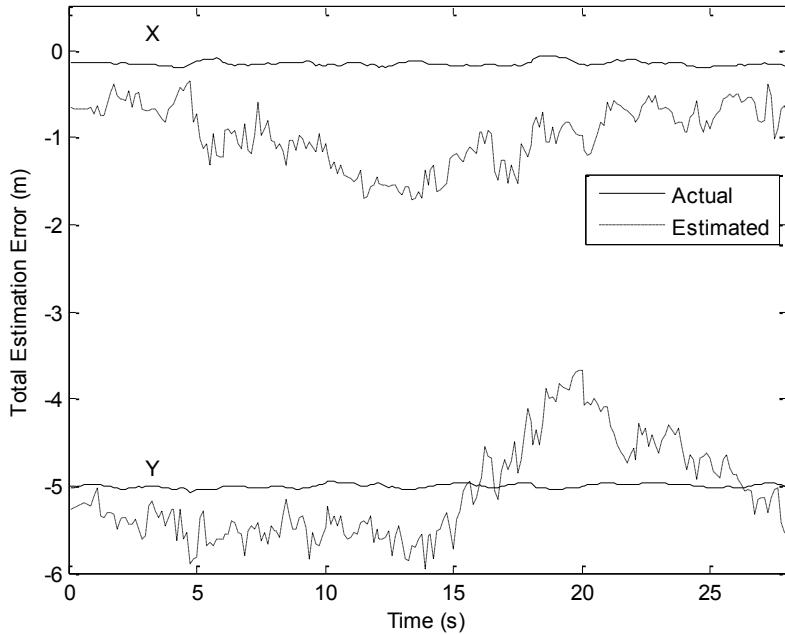
	X (m)	Y (m)	Z (m)	P (m)	Q (m)
<b>Min Error</b>	0.01	0.00	0.28	0.00	0.00
<b>Max Error</b>	0.85	0.47	0.78	0.86	0.89
<b>Mean Error</b>	0.47	0.16	0.54	0.37	0.44
<b>Error Standard Deviation</b>	0.18	0.11	0.13	0.30	0.27
<b>Root Mean Squared Error</b>	0.50	0.20	0.56	0.48	0.52

Table 7.11: First stationary Pioneer test with three mobile tracking stations angular control variable summary.

	$\alpha$ (deg)	$\beta$ (deg)	$\gamma$ (deg)	$\varphi_1$ (deg)	$\varphi_2$ (deg)	$\varphi_3$ (deg)	$\zeta$ (deg)
<b>Min Error</b>	0.34	0.25	0.03	0.39	0.99	0.03	0.02
<b>Max Error</b>	26.19	13.16	10.21	52.68	149.65	43.07	18.59
<b>Mean Error</b>	12.87	4.94	3.01	21.98	28.25	19.15	6.99
<b>Error Standard Deviation</b>	7.57	2.80	2.84	13.93	37.65	12.73	5.55
<b>Root Mean Squared Error</b>	14.92	5.67	4.13	26.00	46.97	22.98	8.92

The results of the first experiment with three mobile tracking stations are presented in Figure 7.10 and summarized in Table 7.9 through Table 7.11. The angles  $\varphi_1$ ,  $\varphi_2$ , and  $\varphi_3$  have high mean errors and  $\varphi_2$  has an especially high maximum error due to outlier data. None the less, the

tracking error was 1.25 m which was close enough to continue tracking under most conditions. This tracking error was mainly due to the poor control of the individual robot headings. These results are connected, as discussed in the last section, because the tracking accuracy depends on the control accuracy. If the controller is not functioning well, then the tracking algorithm will not perform well either. This idea is explored further in the next two experiments.



**Figure 7.11:** This plot shows the actual and estimated Pioneer positions throughout the second experiment with three mobile tracking stations and a stationary Pioneer.

**Table 7.12:** Second stationary Pioneer test with three mobile tracking stations location estimation summary.

	X (m)	Y (m)	Total (m)
<b>Min Error</b>	0.20	0.02	0.32
<b>Max Error</b>	1.58	1.35	1.71
<b>Mean Error</b>	0.82	0.48	0.98
<b>Error Standard Deviation</b>	0.35	0.27	0.36
<b>Root Mean Squared Error</b>	0.89	0.55	1.04

**Table 7.13: Second stationary Pioneer test with three mobile tracking stations distance control variable summary.**

	X (m)	Y (m)	Z (m)	P (m)	Q (m)
<b>Min Error</b>	0.01	0.00	0.18	0.00	0.01
<b>Max Error</b>	1.11	0.71	0.85	1.35	2.31
<b>Mean Error</b>	0.54	0.22	0.48	0.47	1.00
<b>Error Standard Deviation</b>	0.22	0.15	0.19	0.37	0.71
<b>Root Mean Squared Error</b>	0.58	0.26	0.51	0.60	1.23

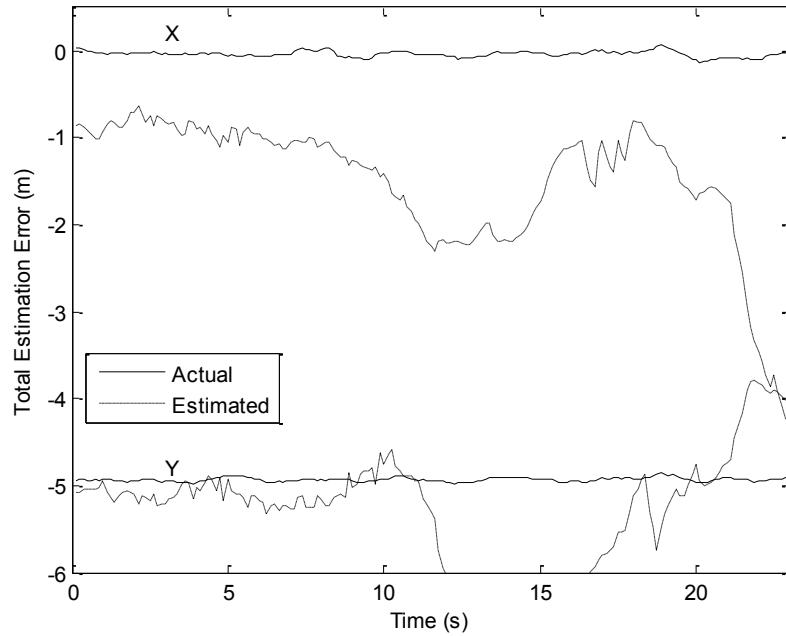
**Table 7.14: Second stationary Pioneer test with three mobile tracking stations angular control variable summary.**

	$\alpha$ (deg)	$\beta$ (deg)	$\gamma$ (deg)	$\varphi_1$ (deg)	$\varphi_2$ (deg)	$\varphi_3$ (deg)	$\zeta$ (deg)
<b>Min Error</b>	0.05	0.00	0.01	0.31	0.07	0.04	0.23
<b>Max Error</b>	34.92	52.13	9.59	51.06	148.6 9	56.96	53.58
<b>Mean Error</b>	14.70	6.00	2.18	20.43	37.25	23.89	11.95
<b>Error Standard Deviation</b>	10.64	7.47	1.88	12.27	47.64	15.34	12.83
<b>Root Mean Squared Error</b>	18.14	9.57	2.88	23.82	60.39	28.38	17.51

The results of this experiment are summarized in Table 7.12 through Table 7.14 and shown in Figure 7.11. In this experiment, the headings  $\varphi_1$ ,  $\varphi_2$ , and  $\varphi_3$  had high mean errors and  $\varphi_2$  had an especially high maximum error due to outlier data. Nonetheless, the mean tracking error of 0.98 m was still accurate enough to keep the Pioneer in the mobile sensor systems' fields of view, allowing for continued tracking.

The next experiment takes a closer look at the effect that the control variables have on the tracking accuracy. In this experiment, the mobile tracking stations initially track the stationary

Pioneer. At approximately 20 seconds, the alpha angle changes sign, resulting in a major loss of control of the cluster. When this occurs, the tracked object is immediately lost. While tracking of the object may have been resumed once control was again established, the mobile tracking stations drifted out of the UWB coverage area before control was able to be reasserted.



**Figure 7.12:** This plot shows the actual and estimated Pioneer positions throughout the third experiment with three mobile tracking stations and a stationary Pioneer.

**Table 7.15:** Stationary Pioneer exploration test with three mobile tracking stations location estimation summary.

	X (m)	Y (m)	Total (m)
<b>Min Error</b>	0.62	0.00	0.68
<b>Max Error</b>	4.31	1.86	4.40
<b>Mean Error</b>	1.46	0.55	1.62
<b>Error Standard Deviation</b>	0.74	0.56	0.83
<b>Root Mean Squared Error</b>	1.64	0.78	1.82

Table 7.16: Stationary Pioneer exploration test with three mobile tracking stations distance control variables.

	X (m)	Y (m)	Z (m)	P (m)	Q (m)
<b>Min Error</b>	0.03	0.00	0.06	0.00	0.00
<b>Max Error</b>	1.79	0.97	0.82	2.42	2.64
<b>Mean Error</b>	0.57	0.23	0.48	0.70	0.97
<b>Error Standard Deviation</b>	0.37	0.21	0.16	0.69	0.93
<b>Root Mean Squared Error</b>	0.68	0.31	0.51	0.98	1.35

Table 7.17: Stationary Pioneer exploration test with three mobile tracking stations angular control variables.

	$\alpha$ (deg)	$\beta$ (deg)	$\gamma$ (deg)	$\varphi_1$ (deg)	$\varphi_2$ (deg)	$\varphi_3$ (deg)	$\zeta$ (deg)
<b>Min Error</b>	0.00	0.04	0.02	0.22	0.02	0.07	0.12
<b>Max Error</b>	89.35	16.22	179.8 6	101.8 7	148.6 7	81.99	115.7 2
<b>Mean Error</b>	26.42	4.50	13.54	33.41	42.36	25.28	35.06
<b>Error Standard Deviation</b>	27.37	3.68	37.24	29.93	45.53	22.38	27.16
<b>Root Mean Squared Error</b>	37.99	5.81	39.53	44.80	62.09	33.72	44.31

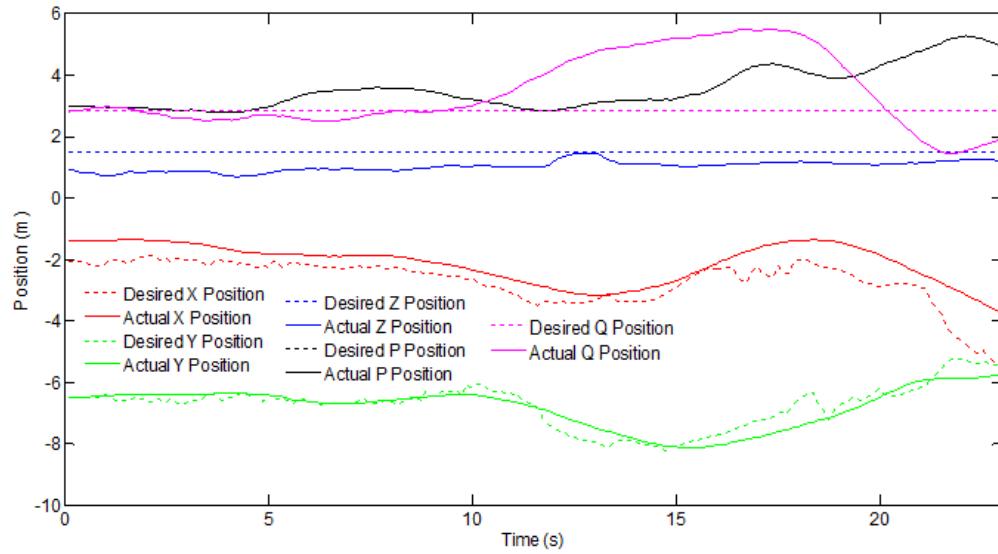
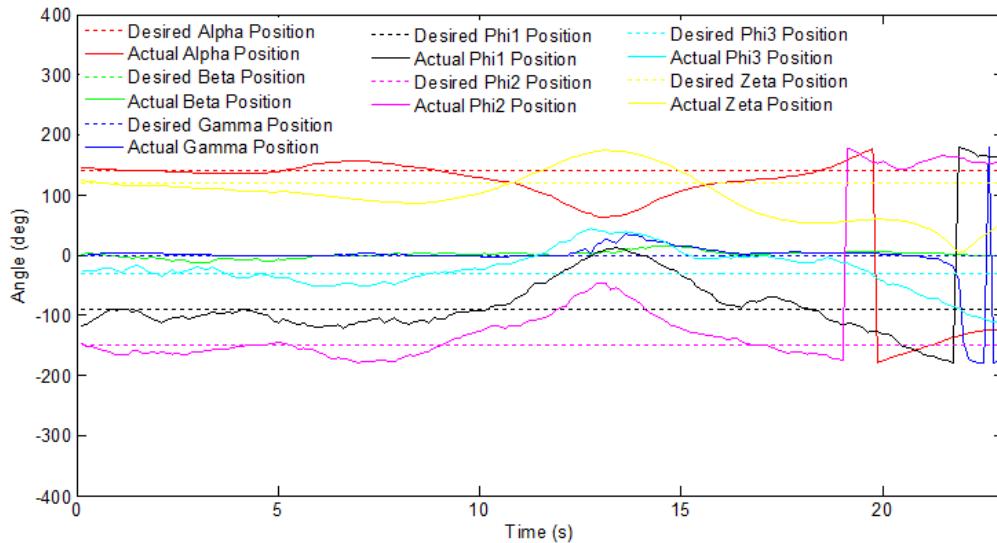


Figure 7.13: Actual and desired distance control variables for three mobile tracking station exploration test.



**Figure 7.14: Actual and desired angular control variables for three mobile tracking station exploration test.**

Figure 7.12 through Figure 7.14 show the results of a tracking experiment in which control has failed. These results are summarized in Table 7.15 through Table 7.17 and illustrate the importance of the  $\alpha$  parameter to the tracking accuracy. At about 11 seconds, the  $\alpha$  error increases. Figure 7.14 shows every other angle error increasing as well in response. The first plot also shows an increase in error at the same time. The  $\alpha$  error begins to get smaller at about 13 seconds, returning to a manageable level at about 16 seconds. Again, the error for the other angles and the tracking errors show the same trend at this time. When the  $\alpha$  angle changes sign at about 20 seconds, the other angles exhibit even higher errors in response, resulting in a very high tracking error and the eventual loss of the stationary Pioneer. This poor tracking accuracy can be seen in the mean tracking error, which is a very high 1.62 m in this test.

These experiments showed that tracking of an object is possible with the controller developed for three mobile tracking stations. As expected, the degree of control of the cluster variables has an impact on the tracking accuracy; the more consistent the cluster variables, the greater the tracking accuracy. While all the cluster variables are important to the tracking

accuracy, these experiments showed that the mobile tracking station headings and  $\alpha$  have particularly large impacts. This is because the mobile tracking stations cannot track an object that they cannot see and the tracking station headings and  $\alpha$  have a large impact on what the mobile tracking stations can see. The angle optimization utilized in this approach helped to minimize the time that the mobile tracking stations could not see the tracked object, improving the method's tracking accuracy.

# **Chapter 8**

---

## **8. Simulations with Optimization-in-the-Loop**

To verify the adaptability of the optimization methodology, simulations were performed with the position optimization integrated into the control loop. These tests were performed in simulation so that sensor failures of various types could be easily reproduced at specific times. Such an inclusion allowed the cluster to react to changing sensor parameters during run time by changing the ideal geometry to match the current conditions. While, under nominal operating conditions, the sensor properties would not change, ambient conditions may change and affect the sensor performance. A sensor could also become damaged during operation, causing a dramatic and abrupt change in its performance. In this series of simulations, the sensor range was not fixed at 2.83 m and was allowed to vary within a 1.3 m range. This range was chosen to match the constraints of the physical testbed.

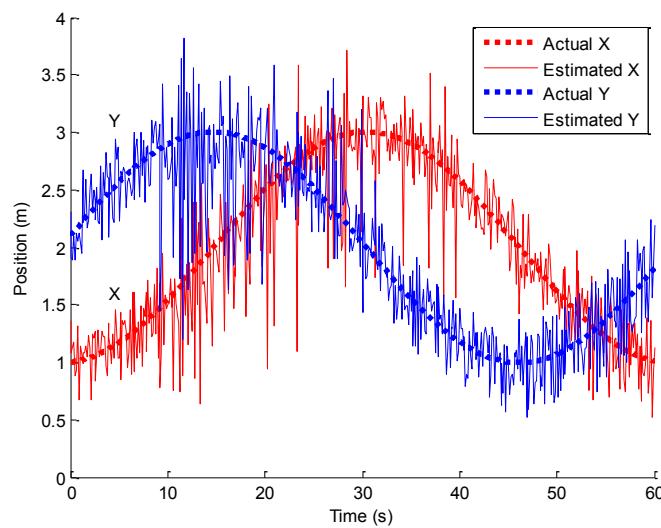
In these simulations, the controller was modified to include the geometrical optimization process as shown in Figure 3.17. This was done by using the positions of the quadrotors and the tracked object as inputs into the optimization process along with the sensor properties. The outputs were the ideal sensor radii and headings which were then used to calculate the cluster parameters. These simulations were also matched to the physical system by adding noise to the sensor system measurements and the reported robot locations. The noise was scaled to match the maximum values observed on the physical system, resulting in sensor measurement noise of  $(D, y_{dist}) + (\pm 1, \pm 1)$  m and location noise of  $(x, y, z) + (\pm 0.35, \pm 0.32, \pm 0.85)$  m. These values were used throughout the simulations.

## 8.1. Two Quadrotor Simulations

The first simulation performed used two identical mobile tracking stations to track a single moving object. Both tracking stations' sensor properties matched the AR.Drone 1.0's actual camera properties at the start of the simulation with a mean radial error of 0.4 m and a mean angular error of 5.7 degrees. Simulating gradual sensor degradation, the mean radial error was increased at a rate of 0.008 m/s and the mean angular error was increased by 0.46 deg/s. The simulation was run for 60 seconds in order to determine whether the optimization-in-the-loop was able to accommodate the changing conditions. The tracking accuracy and controllability results are summarized in Table 8.2 and shown in Figure 8.1.

**Table 8.1: Location estimation summary for two mobile tracking stations and slow sensor degradation.**

	X (m)	Y (m)	Total (m)
<b>Min Error</b>	0.00	0.00	0.02
<b>Max Error</b>	1.67	1.75	1.92
<b>Mean Error</b>	0.24	0.26	0.38
<b>Error Standard Deviation</b>	0.24	0.24	0.31
<b>Root Mean Squared Error</b>	0.34	0.35	0.49



**Figure 8.1: X estimation error for the two quadrotor simulation with slowly degrading identical sensors.**

**Table 8.1: Location estimation summary for two mobile tracking stations and slow sensor degradation.**

	X (m)	Y (m)	Total (m)
<b>Min Error</b>	0.00	0.00	0.02
<b>Max Error</b>	1.67	1.75	1.92
<b>Mean Error</b>	0.24	0.26	0.38
<b>Error Standard Deviation</b>	0.24	0.24	0.31
<b>Root Mean Squared Error</b>	0.34	0.35	0.49

**Table 8.2: Control variable summary for two mobile tracking stations and slow sensor degradation.**

	X (m)	Y (m)	Z (m)	$\alpha$ (deg)	$\beta$ (deg)	$\varphi_1$ (deg)	$\varphi_2$ (deg)	P (m)
<b>Min Error</b>	0.00	0.00	0.00	0.00	0.00	0.01	0.02	0.00
<b>Max Error</b>	1.47	1.45	0.77	7.69	19.63	24.53	26.53	0.60
<b>Mean Error</b>	0.38	0.38	0.21	2.27	5.98	6.43	7.06	0.18
<b>Error Standard Deviation</b>	0.27	0.26	0.00	1.62	4.39	4.60	5.10	0.12
<b>Root Mean Squared Error</b>	0.47	0.46	0.27	2.79	7.42	7.90	8.71	0.22

In this test, all of the control variables were controlled well; the  $\beta$ ,  $\varphi_1$ , and  $\varphi_2$  angles had higher maximum errors, but the mean error was quite low. This level of control resulted in an excellent tracking ability. The mean tracking radial error of 0.38 m is easily within the range of the quadrotor's on-board camera, demonstrating that including an optimization-in-the-loop allows the controller to cope with slow sensor degradation while continuing to track a moving object.

The second simulation featured two mobile tracking stations tracking a moving object. Initially, both sensor systems' properties matched those of the actual AR.Drone 1.0 camera: a mean radial error of 0.4 m and a mean angular error of 5.7 degrees. At 20 seconds, both sensor systems experienced an instantaneous degradation which resulted in a mean radial error of 0.8

m and a mean angular error of 5.7 degrees for Sensor 1 and a mean radial error of 0.4 m and an angular error of 10.1 degrees for Sensor 2. These new sensor properties resulted in a change in the ideal configuration from 90 degrees to 180 degrees, as shown in Figure 8.2. The results of this simulation are summarized in Table 8.3 and

Table 8.4 and shown in Figure 8.3.

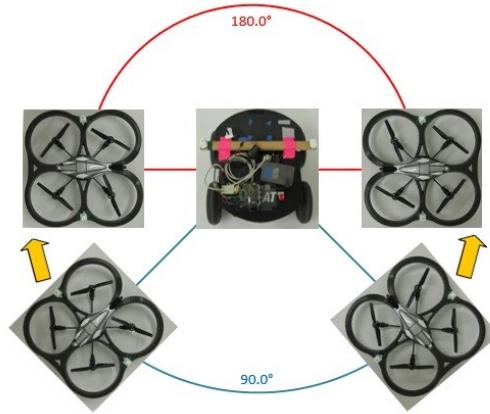


Figure 8.2: At 20 seconds, the ideal configuration changes from 90 degrees (blue) to 180 degrees (red).

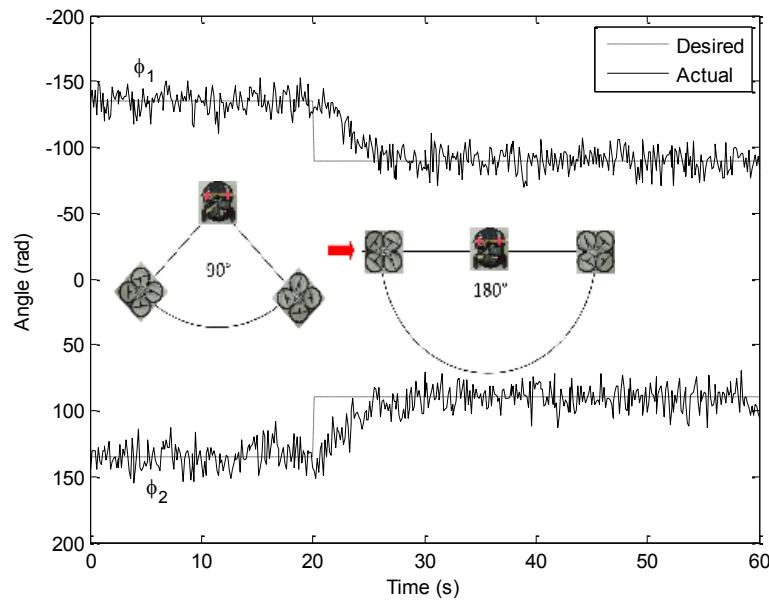


Figure 8.3: X estimation error for the two quadrotor experiment with abruptly changing sensor properties.

**Table 8.3: Location estimation for two mobile tracking stations and an abrupt change in sensor properties.**

	X (m)	Y (m)	Total (m)
<b>Min Error</b>	0.00	0.00	0.01
<b>Max Error</b>	1.92	1.66	2.06
<b>Mean Error</b>	0.19	0.19	0.30
<b>Error Standard Deviation</b>	0.17	0.27	0.30
<b>Root Mean Squared Error</b>	0.26	0.33	0.42

**Table 8.4: Control variable summary for two mobile tracking stations and an abrupt change in sensor properties.**

	X (m)	Y (m)	Z (m)	$\alpha$ (deg)	$\beta$ (deg)	$\phi_1$ (deg)	$\phi_2$ (deg)	P (m)
<b>Min Error</b>	0.00	0.00	0.00	0.00	0.04	0.02	0.01	0.00
<b>Max Error</b>	1.55	2.21	0.76	7.68	19.50	54.99	61.77	1.78
<b>Mean Error</b>	0.29	0.36	0.21	1.88	4.93	8.60	9.01	0.27
<b>Error Standard Deviation</b>	0.21	0.34	0.00	1.39	3.71	8.61	8.86	0.30
<b>Root Mean Squared Error</b>	0.36	0.50	0.26	2.34	6.17	12.17	12.64	0.40

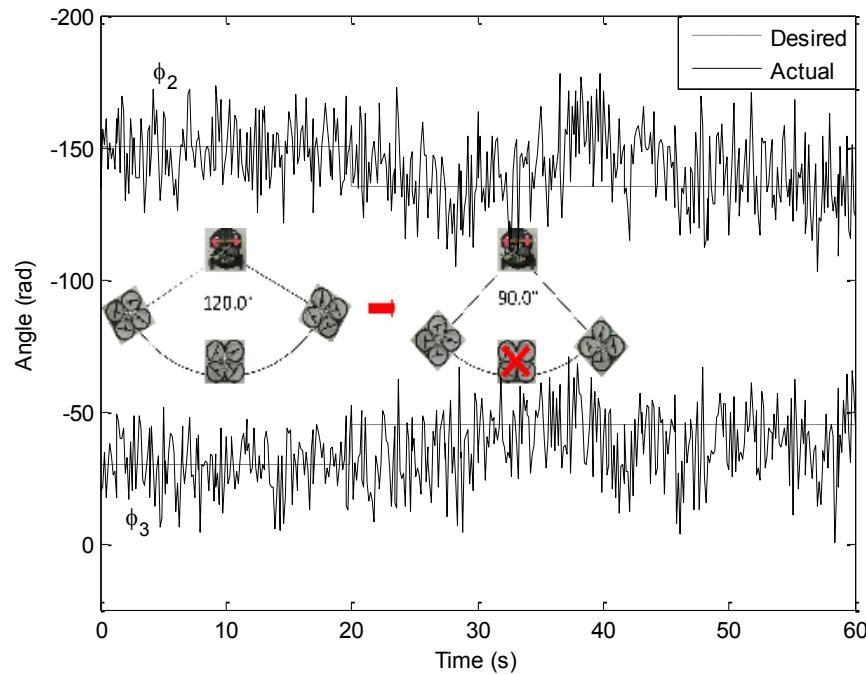
The mean errors for the control variables were low, meaning that the cluster remained in the desired configuration throughout the simulation. However, the maximum errors for  $\phi_1$  and  $\phi_2$  were high in this simulation. This was due to a step change in the corresponding desired values at 20 seconds. The value of both of these variables settled to the new desired values in about 5 seconds, an acceptably quick response time. The high level of control exhibited by this simulation, despite the abrupt change in the desired values of some control variables at 20 seconds, resulted in an excellent average radial error of 0.30 m.

The results of these simulations illustrate that the optimization-in-the-loop can respond to both large and small changes in the sensor properties and compute a new corresponding ideal angle of separation. The controller itself has sufficient robustness to cope with these changes

quickly enough that the quality of the tracking remains high. This proves that the method is efficacious and ready for further testing.

## 8.2. Three Quadrotor Simulations

A third simulation was performed that used three mobile tracking stations to track a stationary object. The three tracking stations began with identical sensor properties matching the actual AR.Drone 1.0 camera properties: a mean radial error of 0.4 m and a mean angular error of 5.7 degrees. At 20 seconds, Sensor 1 experienced a sensor failure that was simulated by setting the sensor errors to the very high values of 2 m for the mean radial error and 57.3 degrees for the mean angular error. The results are shown in Figure 8.4 and summarized in Table 8.5 through Table 8.7.



**Figure 8.4:** X estimation error for the three quadrotor experiment with an abrupt sensor failure at 20 seconds.

**Table 8.5: Location estimation summary for simulation with three mobile tracking stations and an abrupt failure of sensor 1 at 20 seconds.**

	X (m)	Y (m)	Total (m)
<b>Min Error</b>	0.00	0.00	0.01
<b>Max Error</b>	0.67	0.81	0.84
<b>Mean Error</b>	0.19	0.19	0.30
<b>Error Standard Deviation</b>	0.15	0.14	0.16
<b>Root Mean Squared Error</b>	0.24	0.24	0.34

**Table 8.6: Distance control variable summary for simulation with three mobile tracking stations and an abrupt failure of sensor 1 at 20 seconds.**

	X (m)	Y (m)	Z (m)	P (m)	Q (m)
<b>Min Error</b>	0.00	0.00	0.00	0.00	0.00
<b>Max Error</b>	0.51	0.71	0.76	1.00	1.04
<b>Mean Error</b>	0.12	0.14	0.20	0.34	0.36
<b>Error Standard Deviation</b>	0.09	0.12	0.00	0.22	0.24
<b>Root Mean Squared Error</b>	0.15	0.18	0.25	0.41	0.43

**Table 8.7: Angular control variable summary for simulation with three mobile tracking stations and an abrupt failure of sensor 1 at 20 seconds.**

	$\alpha$ (deg)	$\beta$ (deg)	$\gamma$ (deg)	$\varphi_1$ (deg)	$\varphi_2$ (deg)	$\varphi_3$ (deg)	$\zeta$ (deg)
<b>Min Error</b>	0.03	0.01	0.00	0.04	0.00	0.07	0.06
<b>Max Error</b>	28.27	62.75	17.55	40.34	43.21	48.55	38.51
<b>Mean Error</b>	7.59	17.57	2.34	10.28	11.14	12.73	12.25
<b>Error Standard Deviation</b>	5.67	12.51	2.60	7.72	8.17	10.01	8.15
<b>Root Mean Squared Error</b>	9.47	21.56	3.49	12.86	13.81	16.18	14.71

The optimizer returned a new ideal angle of separation of 90 degrees after the sensor system failure, which was the same ideal configuration as in the case of two identical mobile

tracking stations. This is as expected since only two of the tracking stations are functioning after 20 seconds. Despite this large change in sensor input, tracking in this simulation demonstrated a low mean radial error of 0.30 m. The control variables were controlled fairly well, with the highest error seen in the  $\beta$  values. This is because  $\beta$  is dependent on the z control variable, which has a much higher measurement error in the UWB system than the x and y values. Nonetheless, this simulation demonstrated that this method can experience a sensor failure and not only continue to track an object, but continue to track it accurately.

The simulations presented in this chapter illustrate the robustness of an optimization-in-the-loop system. The optimizer can quickly respond to a change in sensor properties and the control system can quickly reposition the robots to the new desired positions. The simulations also demonstrate the effectiveness of this methodology to cope with slow sensor degradation, abrupt sensor degradation, and abrupt sensor failure. Tracking of the object was not significantly impacted by any of these failures, implying that an optimization-in-the-loop can be tested in the real world in future work.

However, it is important to note a difference between simulation and real world trials at this point. The optimization routine runs in about 0.14 second for two mobile tracking stations and 0.33 second for three mobile tracking stations when computed on a conventional Pentium-class workstation with a 2.10 GHz processor and 4.00 GB of RAM. This is slightly slower than the control rate of 0.125 second used in the real world experiments presented here. The control rate was set to prevent the AR.Drone 1.0s' hover command from activating and causing a loss of control of the robot. Thus, either different robots that do not require such a high control rate should be used or the optimization process will have to run in a slower loop. This is a relatively

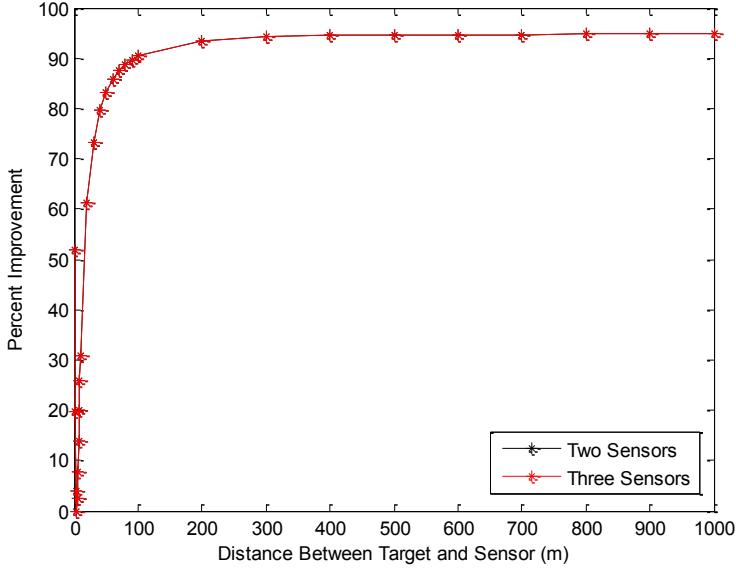
simple change, but it was not required in simulation since each time step in the control loop did not need to correspond to real time.

### 8.3. Exploration of the Target Location Estimate Improvement

In the work presented in Chapter 7, the optimization resulted in a 6% improvement in the target location estimate over the non-optimized worst-case scenario tested with identical sensors at the nominal fixed radius distance of 2.83 m. However, much greater improvements in the target location estimate were observed when the sensors had different properties, the radial distance between the sensors and the object was increased, or more sensors were used. This section examines these effects further by plotting the percent improvement of the target location estimate of two and three sensor systems under three sets of conditions: identical sensors, extreme sensor variations, and medium sensor variations. The fixed sensor radii values used in each case are shown in Table 8.8 below.

**Table 8.8: Radii examined for the three target estimate improvement cases.**

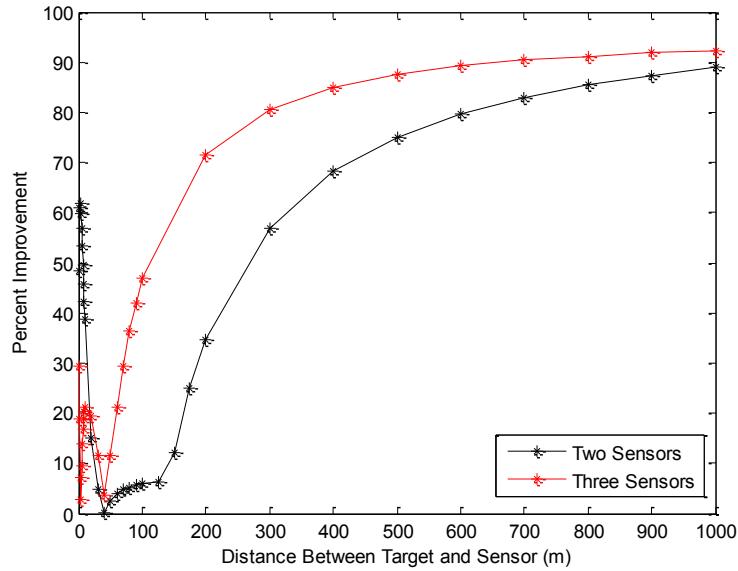
Radii	Case	Radii	Case	Radii	Case
1 m	All	30 m	All	200 m	All
2 m	All	40 m	All	300 m	All
3 m	All	50 m	All	400 m	All
4 m	All	60 m	All	500 m	All
5 m	All	70 m	All	600 m	All
6 m	All	80 m	All	700 m	All
7 m	All	90 m	All	800 m	All
8 m	All	100 m	All	900 m	All
9 m	All	125 m	Case 2 with Two Sensors Only	1000 m	All
10 m	All	150 m	Case 2 with Two Sensors Only		
20 m	All	175 m	Case 2 with Two Sensors Only		



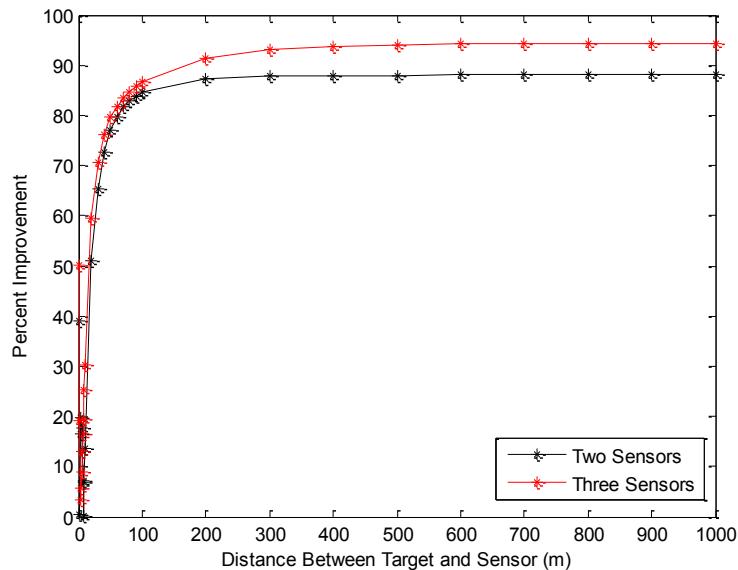
**Figure 8.5: Percent improvement of the optimal sensor configuration over the worst-case configuration of two and three identical sensors at the same fixed radius.**

In Case 1, shown in Figure 8.5, all sensors have identical properties with a mean radial error of 0.4 m and a mean angular error of 5.7 degrees for both the two and three sensor scenarios. The sensors in Case 2, shown in Figure 8.6, have extreme variations. For the two sensor scenario, the first sensor has a mean radial error of 0.4 m and a mean angular error of 57.3 degrees while the second sensor has a mean radial error of 4 m and a mean angular error of 5.7 degrees. The first sensor of the three sensor scenario has a mean radial error of 0.4 m and a mean angular error of 5.7 degrees, the second sensor has a mean radial error of 0.4 m and a mean angular error of 57.3 degrees, and the third sensor had a mean radial error of 4 m and a mean angular error of 5.7 degrees. Case 3, shown in Figure 8.7, features sensors with smaller variations in their properties. In the two sensor scenario, the first sensor has a mean radial error of 0.4 m and a mean angular error of 11.5 degrees while the second sensor has a mean radial error of 0.8 m and a mean angular error of 5.7 degrees. In the three sensor scenario, the first sensor has a mean radial error of 0.4 m and a mean angular error of 5.7 degrees, the second

sensor has a mean radial error of 0.4 m and a mean angular error of 11.5 degrees, while the third sensor has a mean radial error of 0.8 m and a mean angular error of 5.7 degrees.



**Figure 8.6:** Percent improvement of the optimal sensor configuration over the worst-case configuration of two and three sensors with extreme sensor properties at the same fixed radius.



**Figure 8.7:** Percent improvement of the optimal sensor configuration over the worst-case configuration of two and three sensors with medium differences in sensor properties at the same fixed radius.

From Figure 8.5, it is immediately apparent that the two scenarios in Case 1 result in identical curves. This is due to the fact that the optimal sensor configuration is independent of distance for the identical sensor case. However, distance does affect the optimal configuration when the sensors have different properties, as shown in Figure 8.6 and Figure 8.7. The further the sensor is from the target object, the less disparate valid sensor coverage areas appear, regardless of the difference in the sensor properties themselves. Therefore, the optimal configuration will change with distance as the apparent difference in the valid sensor coverage areas change. More variation is seen in the three sensor curves in Figure 8.6 and Figure 8.7 than in the two sensor curves because the number of possible sensor configurations increases with the number of sensors. This is reflected in smoother curves for the two sensor case and piecewise continuous curves for the three sensor case.

The studies presented in this section illustrate some of the factors that can affect the percent improvement in the target location estimate. The number of sensors, sensor properties, and distance of the target object from the sensor all had significant impacts on the percent improvement supplied by the optimization technique presented in this dissertation. These results indicate that improvements as high as 90% over the worst-case scenario can be achieved through optimization, indicating that further distances should be tested in the real world in future work to confirm these results.

# **Chapter 9**

---

## **9. Conclusions**

This dissertation responds to the need for a methodology to determine the sensor placement for a group of mobile tracking stations that will minimize the position estimation error of the target. While much work has been done in the literature to optimize the placement of sensors, little work has considered the interaction between multiple sensors. This work fills this gap by considering the sensor configuration as a whole rather than as the sum of its parts, providing a more comprehensive view of the system that is being optimized.

This work describes in this dissertation presents a novel, highly capable strategy for utilizing a multi-robot network to track a moving target. The configuration of mobile tracking stations was optimized in order to produce the target object position estimate that yielded the smallest estimation error, even when sensor performance varied. This resulted in a simple, robust system that accurately followed a moving object. The article also provided an overview of the cluster space description used to control groups of two and three mobile tracking stations as well as the corresponding control methodology used to maintain the ideal tracking configuration throughout the experiment. This allowed for the collection of the best tracked object position information using the novel method presented in Chapter 3. An overview of the method used to obtain an estimate of the tracked object's position was presented in Chapter 4 while simulation experimental results were presented in Chapters 6 through 8. These results demonstrated that the method was effective at tracking both a stationary and a moving object and can be applied to sensors with different or identical properties. It can also be applied whether the sensor properties remain constant over time, degrade, or even fail.

While the methodology presented here does not scale to thousands of robots, it works well for “small” groups of robots with as many as tens of robots. These small groups are easier to deploy and maintain than large numbers of robots, allowing them to be fielded in remote areas that are difficult for robot delivery, deployed by only a few people, or purchased by groups that cannot afford large groups of robots. This methodology makes the increased tracking accuracy of a multi-sensor system available for limited-resource systems without requiring large numbers of robots, creating a methodology that is accessible for more applications.

## 9.1. Contributions

In order to develop a rigorous mathematical formulation for optimizing sensor system locations for a cluster formation in a multi-sensor, single-object environment through the use of error ellipses, this dissertation presented:

- Two and three robot cluster space representations were developed to allow for robotic control. These cluster space representations were developed in conjunction with other students at Santa Clara University and have been used in additional work in the Robotics Systems Laboratory [55][56][57].
- Mathematical modeling of a group of  $n$  sensors and their resulting combined error covariance matrix.
- Closed-form optimization for a group of two mobile sensors to obtain the minimum area of the combined sensor error ellipse. This optimization was used in the physical experiments in this dissertation.
- Closed-form optimization for a group of three mobile sensors to obtain the minimum area of the combined sensor error ellipse, also used in the physical experiments in this dissertation.

- Development of a simple vision-processing algorithm that requires a very small amount of data transmission.
- Stationary simulations and physical experiments that verified the findings of the closed-form optimization for two and three mobile tracking stations. These results were presented in Chapter 6.
- Controlled physical experiments that demonstrated the ability of the tracking algorithm, with results presented in Chapter 7.
- Simulation with an optimization-in-the-loop that demonstrated the effectiveness of the optimization methodology to respond quickly to changing sensor properties. These results were presented in Chapter 8.

This dissertation is based upon previous work by various researchers in the fields of robotics, vision processing, and optimization. It extends this body of knowledge to include the optimization of a group of sensors as a whole to minimize the resulting tracking error. It is believed to be the first time that a rigorous mathematical framework has been applied to a multi-sensor optimization system and tested with physical experiments. Further, the optimization can be applied to a wider variety of conditions than current methods such as that presented in [11] which requires a Kalman filter and [43] which fails to find an optimal solution for the fixed radius case. In the work presented in this dissertation, the optimization led to a 6% improvement in the target location estimate over the non-optimized worst-case scenario tested with identical sensors at the nominal fixed radius distance of 2.83 m and even more significant improvements of over 90% at larger radial distances.

## 9.2. Future Work

The work presented here suggests many areas for future work. Although the closed-form expressions extend to  $n$  robots relatively easily, the mathematics relating the optimization output and the cluster variables does not. Future work is suggested to develop this mathematical relationship for more than three robots. Future work is also recommended to combine the effect of position uncertainty in the tracking system with position uncertainty of the tracking stations themselves in the analysis. Currently, the model assumes that the tracking stations know their exact position. Since knowledge of a robot's exact position is difficult, if not impossible, in the field, this error should be considered when optimizing the sensor configuration as it affects the obtained position estimate of the tracked object.

The tracking process itself was tested with both a Pioneer 3-AT and a uniform object as the tracked object. Use of the uniform object helped to determine whether discrepancies between the theory and physical results were due to the shape of the Pioneer. During this process, it was also discovered that different lighting conditions and background colors also had an effect on the tracking results. It is recommended that further work be performed to evaluate the extent of these additional noise sources.

Additionally, testing could be performed on different platforms with different sensors. This extension could include another homogenous sensor mix, a heterogeneous sensor mix on identical platforms, or a heterogeneous sensor mix on different platforms. Testing could also be extended to include changing sensor properties or sensor failure in physical experiments as well as extended to include farther distances between the sensors and the target object.

This extended testing suggests the inclusion of a dual-rate controller where the controller works at one rate and the optimization-in-the-loop works at a slower rate, allowing the cluster

to adapt to changing conditions. The properties of the sensors themselves would also need to be calculated in real time in order to be fed into the optimization-in-the-loop. A sliding average that resets when a large change is detected is suggested for this purpose, but further analysis is required to determine the most efficient method.

Finally, future work is recommended to compensate for the dynamics of the system. The sensor systems would compute the optimal sensor system configuration at a future time step and then move into an intercept course that would result in this optimal configuration at the desired time step. To the best of the author's knowledge, this has not been explored on a physical testbed before. This research will help extend the tracking methodology to additional applications.

## References

- [1] C. A. Kitts and I. Mas (2009, April). Cluster Space Specification and Control of Mobile Multirobot Systems. *IEEE/ASME Transactions on Mechatronics* [Online]. 14(2), pp. 207-218. Available:  
[http://ieeexplore.ieee.org/xpl/articleDetails.jsp?arnumber=4814792&sortType%3Dasc\\_p\\_Sequence%26filter%3DAND%28p\\_IS\\_Number%3A4814791%29](http://ieeexplore.ieee.org/xpl/articleDetails.jsp?arnumber=4814792&sortType%3Dasc_p_Sequence%26filter%3DAND%28p_IS_Number%3A4814791%29)
- [2] C. A. Kitts and M. Egerstedt (2008, March). Design, Control, and Application of Real-World Multirobot Systems [From the Guest Editors]. *IEEE Robotics & Automation Magazine* [Online]. 15(1), p. 8. Available:  
<http://ieeexplore.ieee.org/stamp/stamp.jsp?arnumber=4476318>
- [3] P. Corke, C. Detweiler, M. Dunbabin, M. Hamilton, D. Rus, and I. Vasilescu, "Experiments with Underwater Robot Localization and Tracking," in *IEEE ICRA*, Roma, Italy, 2007, pp. 4556-4561.
- [4] K. Chakrabarty, S. Iyengar, H. Qi, and E. Cho. (2002, December). Grid Coverage for Surveillance and Target Location in Distributed Sensor Networks. *Computers, IEEE Transactions on* [Online]. 51(12), pp. 1448-1453. Available:  
<http://ieeexplore.ieee.org/xpl/articleDetails.jsp?tp=&arnumber=1146711&url=http%3A%2F%2Fieeexplore.ieee.org%2Fiel5%2F12%2F25838%2F01146711.pdf%3Farnumber%3D1146711>
- [5] L. Shang, K. Zhao, Z. Cai, D. Gao, and M. Hu. (2014, July). An Energy-Efficient Collaborative Target Tracking Framework in Distributed Wireless Sensor Networks. *International Journal of Distributed Sensor Networks* [Online]. Available:  
<http://www.hindawi.com/journals/ijdsn/2014/396109/>

- [6] Y. Zhang, D. Wei, W. Fu, and B. Yang. (2014, June). Target Positioning with GDOP Assisted Nodes Selection Algorithm in Wireless Sensor Networks. *International Journal of Distributed Sensor Networks* [Online]. Available: <http://www.hindawi.com/journals/ijdsn/2014/404812/>
- [7] J. Lin, W. Xiao, F. Lewis, and L. Xie. (2008, October). Energy-Efficient Distributed Adaptive Multisensor Scheduling for Target Tracking in Wireless Sensor Networks. *Instrumentation and Measurement, IEEE Transactions on* [Online]. 58(6), pp. 1886-1896. Available: [http://ieeexplore.ieee.org/xpl/login.jsp?tp=&arnumber=4652563&url=http%3A%2F%2Fieeexplore.ieee.org%2Fxppls%2Fabs\\_all.jsp%3Farnumber%3D4652563](http://ieeexplore.ieee.org/xpl/login.jsp?tp=&arnumber=4652563&url=http%3A%2F%2Fieeexplore.ieee.org%2Fxppls%2Fabs_all.jsp%3Farnumber%3D4652563)
- [8] J. Curcio, J. Leonard, J. Vaganay, A. Patrikalakis, A. Bahr, D. Battle, H. Schmidt, and M. Grund, “Experiments in Moving Baseline Navigation Using Autonomous Surface Craft,” in *OCEANS*, Washington, D. C., 2005, pp. 730-735.
- [9] A. Bahr, J. Leonard, and M. Fallon. (2009, June). Cooperative Localization for Autonomous Underwater Vehicles. *The International Journal of Robotics Research* [Online]. 28(6), pp. 714-728. Available: <http://dspace.mit.edu/handle/1721.1/58207>
- [10] S. Martínez and F. Bullo. (2006, April). Optimal Sensor Placement and Motion Coordination for Target Tracking. *Automatica* [Online]. 42(4), pp. 661-668. Available: <http://www.sciencedirect.com/science/article/pii/S000510980600015X>
- [11] A. Bahr, J. Leonard, and A. Martinoli, “Dynamic Positioning of Beacon Vehicles for Cooperative Underwater Navigation,” in *IEEE IROS*, Vilamoura, Algarve, 2012, pp. 3760-3767.
- [12] R. Zivan, H. Yedidsion, S. Okamoto, R. Clinton, and K. Sycara. (2014, April). Distributed Constraint Optimization for Teams of Mobile Sensing Agents. *Autonomous Agents and*

- Multi-Agent Systems* [Online]. Available:  
<http://link.springer.com/article/10.1007%2Fs10458-014-9255-3>
- [13] A. Marjovi and L. Marques. (2013, October). Optimal Spatial Formation of Swarm Robotic Gas Sensors in Odor Plume Finding. *Autonomous Robots* [Online]. 35(2-3), pp. 93-109. Available: <http://link.springer.com/article/10.1007%2Fs10514-013-9336-1>
- [14] J. Spletzer and C. Taylor. (2003, January). Dynamic Sensor Planning and Control for Optimally Tracking Targets. *The International Journal of Robotics Research* [Online]. 22(1), pp. 7-20. Available: <http://ijr.sagepub.com/content/22/1/7.refs>
- [15] J. Cashbaugh and C. Kitts, (2015, June). Optimizing Sensor Locations in a Multisensor Single-Object Tracking System. *International Journal of Distributed Sensor Networks* [Online]. Available: <http://www.hindawi.com/journals/ijdsn/2015/741491/>
- [16] R. Beetem, "Improved Tracking Estimates with Multirobot-Based Tracking Networks," M.S. thesis, Dept. Mech Eng, Santa Clara University, Santa Clara, CA, 2012.
- [17] S. Dayanidhi, "Target Tracking Using Mobile Robotic Stations," M.S. thesis, Dept. Mech Eng, Santa Clara University, Santa Clara, CA, 2013.
- [18] T.J. Leising, "Six Degree of Freedom Cluster Space Control Simulation for Spacecraft Formations," M.S. thesis, Dept. Mech Eng, Santa Clara University, Santa Clara, CA, 2012.
- [19] Y. Mohan and S. Ponnambalam, "An Extensive Review of Research in Swarm Robotics," in World Congress on Nature & Biologically Inspired Computing, Coimbatore, India, December 9-11, 2009.
- [20] A. Marjovi and L. Marques, "Swarm Robotic Plume Tracking for Intermittent and Time-Variant Odor Dispersion," in European Conference on Mobile Robots, Barcelona, Spain, September 25-27, 2013.

- [21] A. Atyabi and D. Powers, "The Use of Area Extended Particle Swarm Optimization (AEPSO) in Swarm Robotics," in 11<sup>th</sup> International Conference on Control, Automation, Robotics and Vision, Singapore, December 7-10, 2010.
- [22] M. Soorki, H. Talebi, and S. Nikravesh, "A Robust Dynamic Leader-Follower Formation Control with Active Obstacle Avoidance," in IEEE International Conference on Systems, Man, and Cybernetics, Anchorage, AK, October 9-12, 2011.
- [23] D. Mercado, R. Castro, and R. Lozano, "Quadrotors Flight Formation Control Using a Leader-Follower Approach," in European Control Conference, Zürich, Switzerland, July 17-19, 2013.
- [24] T. Dierks, B. Brenner, and S. Jagannathan, "Near Optimal Control of Mobile Robot Formations," in IEEE Symposium on Adaptive Dynamic Programming and Reinforcement Learning, Paris, France, April 11-15, 2011.
- [25] J. Guo, Z. Lin, M. Cao, and G. Yan, "Adaptive Leader-Follower Formation Control for Autonomous Mobile Robots," in American Control Conference, Baltimore, Maryland, June 30 – July 2, 2010.
- [26] I. Mas and C. Kitts. (2014, May). Dynamic Control of Mobile Multirobot Systems: The Cluster Space Formulation. *IEEE Access* [Online]. 2, pp. 558-570. Available: <http://ieeexplore.ieee.org/stamp/stamp.jsp?arnumber=6818372>
- [27] Mathworks. (2006). atan2. *Mathworks Documentation*. [Online] Available: <https://www.mathworks.com/help/matlab/ref/atan2.html>
- [28] J. Craig, "Jacobians: velocities and static forces," in *Introduction to Robotics: Mechanics and Control*. 3<sup>rd</sup> ed. Upper Saddle River, NJ: Pearson Prentice Hall, 2005, ch. 5, sec. 8, pp. 151-153.

- [29] C. Zempel, J. Cashbaugh, A. Mahacek, and A. Sherban. "Forward and Inverse Position and Velocity Kinematics for a Cluster of Three Aerial Drones," Robotics Systems Laboratory Document, 2014.
- [30] G. Vanderplaats, "Genetic Search," in *Numerical Optimization Techniques for Engineering Design*. 4<sup>th</sup> ed. Colorado Springs, CO: Vanderplaats Research and Development, Inc., 2005, ch. 6, sec. 3, pp. 195-198.
- [31] G. Vanderplaats, "Particle Swarm," in *Numerical Optimization Techniques for Engineering Design*. 4<sup>th</sup> ed. Colorado Springs, CO: Vanderplaats Research and Development, Inc., 2005, ch. 6, sec. 4, pp. 198-200.
- [32] R. Eberhart and Y. Shi, "Particle Swarm Optimization: Developments, Applications and Resources," in *2001 Congress on Evolutionary Computation*, Seoul, South Korea, 2001, pp. 81-86.
- [33] R. Hooke and T. A. Jeeves. (1961, April). "Direct Search" Solution of Numerical and Statistical Problems. *Journal of the ACM* [Online]. 8(2), pp. 212-229. Available: <http://dl.acm.org/citation.cfm?id=321069>
- [34] G. Buzzi-Ferraris and F. Manenti, "Heuristic Methods," in *Nonlinear Systems and Optimization for the Chemical Engineer*. 1<sup>st</sup> ed. Weinheim. Germany: Wiley-VCH, 2014, ch. 3, sec. 2, pp. 86-97.
- [35] J. Devore, "Expected Values, Covariance, and Correlation," in *Probability and Statistics for Engineering and the Sciences*. 6<sup>th</sup> ed. Toronto, Canada: Brooks/Cole – Thomson Learning, 2004, ch. 5, sec. 2, pp. 219-225.
- [36] P. Schubert and M. Kirchner (2014, January). Ellipse Area Calculations and Their Applicability in Posturography. *Gait and Posture* [Online]. 39(1), pp. 518-522. Available: <http://www.sciencedirect.com.libproxy.scu.edu/science/article/pii/S0966636213005961>

- [37] M. H. DeGroot and M. J. Schervish, "Tables," in *Probability and Statistics*, 3<sup>rd</sup> ed. Boston: Addison-Wesley, 2002, pp. 774-775.
- [38] J. E. Davis (2007, August). Combining Error Ellipses. Harvard. Cambridge, MA. [Online]. Available: [cxc.harvard.edu/csc/memos/files/Davis\\_ellipse.pdf](http://cxc.harvard.edu/csc/memos/files/Davis_ellipse.pdf)
- [39] S. Leon, *Linear Algebra with Applications*. Upper Saddle River: Pearson, 2010.
- [40] S. Dayanidhi, R. Beetem, and C. Kitts, "Initial Experiments in Multirobot-Based Tracking Networks," in *ASME-ISPS/JSME-IIP*, Santa Clara, CA, 2012.
- [41] M. H. DeGroot and M. J. Schervish, "Variance," in *Probability and Statistics*, 3<sup>rd</sup> ed. Boston: Addison-Wesley, 2002, pp. 197-202.
- [42] G. Tonel. (2007, May). Unconstrained optimization using Hooke & Jeeves. *Matlab Central*. [Online] Available: <http://www.mathworks.com/matlabcentral/fileexchange/15070-unconstrained-optimization-using-hooke---jeeves>
- [43] T. H. Chung, V. Gupta, J. W. Burdick, R. M. Murray, "On a Decentralized Active Sensing Strategy using Mobile Sensor Platforms in a Network," in IEEE CDC, Atlantis, Bahamas, December 14-17, 2004.
- [44] T. Mukai and M. Ishikawa (1996, June). An Active Sensing Method Using Estimate Errors for Multisensor Fusion Systems. *Industrial Electronics, IEEE Transactions On* [Online]. 43(3), pp. 380-386. Available: [http://ieeexplore.ieee.org/xpls/abs\\_all.jsp?arnumber=499810&tag=1](http://ieeexplore.ieee.org/xpls/abs_all.jsp?arnumber=499810&tag=1)
- [45] M. Gupta, L. Behera, V. Subramanian, M. Jamshidi (2014, May). A Robust Visual Human Detection Approach With UKF-Based Motion Tracking for a Mobile Robot. *IEEE Systems Journal* [Online]. PP(99), pp. 1-13. Available: <http://ieeexplore.ieee.org/xpl/articleDetails.jsp?arnumber=6810181>

- [46] G. Rodríguez-Canosa, S. Thomas, J. del Cerro, A. Barrientos, and B. MacDonald (2012, April). A Real-Time Method to Detect and Track Moving Objects (DATMO) from Unmanned Aerial Vehicles (UAVs) Using a Single Camera. *Remote Sensing* [Online]. 4(4), pp. 1090-1111. Available: <http://www.mdpi.com/2072-4292/4/4/1090>
- [47] A. Giusti, J. Nagi, L. Gambardella, G. Di Caro, "Cooperative Sensing and Recognition by a Swarm of Mobile Robots," in IEEE/RSJ International Conference on Intelligent Robots and Systems, Vilamoura, Algarve, Portugal, October 7-12, 2012.
- [48] T. de Wolf, private communication, March 2014.
- [49] "Studio Diip: Projects," [Online]. Available: <http://www.studiodiip.com/projects>. [Accessed 2 July 2015].
- [50] "AR.Drone Specifications," [Online]. Available: <http://diydrones.com/profiles/blogs/parrot-ardrones-specs-arm9>. [Accessed 6 July 2013].
- [51] S. Piskorski, N. Brulez, and P. Eline, 2011, *AR.Drone Developer Guide*, Parrot S. A..
- [52] W. H. Ittelson (1951, April). Size as a Cue to Distance: Radial Motion. *The American Journal of Psychology* [Online]. 64(2), pp. 188-202. Available: <http://www.jstor.org/stable/1418666>
- [53] G. Welch and G. Bishop, 2006, "An Introduction to the Kalman Filter," UNC-Chapel Hill, TR 95-041.
- [54] J. Cashbaugh, A. Mahacek, C. Kitts, C. Zempel, and A. Sherban, "Quadrotor Testbed Development and Preliminary Results," in IEEE Aerospace, Big Sky, Montana, March 7-14, 2015.
- [55] A. Sherban, "Cluster Space Control for Heterogeneous Degrees of Freedom Robots," M.S. thesis, Dept. Mech Eng, Santa Clara University, Santa Clara, CA, 2014.

- [56] C. Zempel, "Cluster Space Control of Three Aerial Drones," M.S. thesis, Dept. Mech Eng, Santa Clara University, Santa Clara, CA, 2014.
- [57] A. Mahacek, "Autonomous Cluster Control of Two Robots in Three Dimensional Space," M.S. thesis, Dept. Mech Eng, Santa Clara University, Santa Clara, CA, 2014.
- [58] J. R. R. Tolkien, *The Lord of the Rings*. London: Harper Collins, 1955.
- [59] Adept Technology, Inc., "MobilePioneer 3-AT," 2011. [Online]. Available: <http://www.google.com/url?sa=t&rct=j&q=&esrc=s&source=web&cd=2&ved=0CDkQFjAB&url=http%3A%2F%2Fwww.mobilerobots.com%2FLibraries%2FDownloads%2FPioneer3AT-P3AT-RevA.sflb.ashx&ei=afZnUq-1Osi-2gWihYHoCA&usg=AFQjCNG4YJfNJRJFUXqJNZFYqHrzsp9N2A&sig2=CyYYf6CwKyHUILQQjVtyAg&bvm=bv.55123115,d.b2I>
- [60] *Sapphire Dart Digital Active Real-Time Tracking (Model H651/651A Indoor/ Outdoor Users Guide Revision 2.1*, Multispectral Solutions, Inc., Lincolnshire, IL, 2006.
- [61] "Data Turbine," [Online]. Available: <http://www.dataturbine.org/>. [Accessed 6 July 2013].
- [62] P. Mahacek, C. Kitts, and I. Mas. (2012, Feb.). Dynamic Guarding of Marine Assets Through Cluster Control of Automated Surface Vessel Fleets. *IEEE/ASME Transactions on Mechatronics* [Online]. 17 (1), 65-75. Available: [http://ieeexplore.ieee.org/xpl/login.jsp?tp=&arnumber=6097059&url=http%3A%2F%2Fieeexplore.ieee.org%2Fxpls%2Fabs\\_all.jsp%3Farnumber%3D6097059](http://ieeexplore.ieee.org/xpl/login.jsp?tp=&arnumber=6097059&url=http%3A%2F%2Fieeexplore.ieee.org%2Fxpls%2Fabs_all.jsp%3Farnumber%3D6097059)
- [63] "javadrone: AR.Drone Java API," [Online]. Available: <http://code.google.com/p/javadrone/>. [Accessed 15 June 2012].

# Appendix

---

## Appendix A

This appendix provides the full two robot forward and inverse Jacobians described in Chapter 2.

### A.1. Two Robot Forward Jacobian

$$J = \begin{bmatrix} \frac{1}{2} & 0 & 0 & 0 & \frac{1}{2} & 0 & 0 & 0 \\ 0 & \frac{1}{2} & 0 & 0 & 0 & \frac{1}{2} & 0 & 0 \\ 0 & 0 & \frac{1}{2} & 0 & 0 & 0 & \frac{1}{2} & 0 \\ \frac{-y_1 + y_2}{(x_1 - x_2)^2 + (y_1 - y_2)^2} & \frac{x_1 - x_2}{(x_1 - x_2)^2 + (y_1 - y_2)^2} & 0 & 0 & \frac{y_1 - y_2}{(x_1 - x_2)^2 + (y_1 - y_2)^2} & \frac{-x_1 + x_2}{(x_1 - x_2)^2 + (y_1 - y_2)^2} & 0 & 0 \\ \frac{(-x_1 + x_2)(z_1 - z_2)}{A} & \frac{(-y_1 + y_2)(z_1 - z_2)}{A} & C & 0 & \frac{(x_1 - x_2)(z_1 - z_2)}{A} & \frac{(y_1 - y_2)(z_1 - z_2)}{A} & -B & 0 \\ \frac{y_1 - y_2}{(x_1 - x_2)^2 + (y_1 - y_2)^2} & \frac{-x_1 + x_2}{(x_1 - x_2)^2 + (y_1 - y_2)^2} & 0 & 1 & \frac{-y_1 + y_2}{(x_1 - x_2)^2 + (y_1 - y_2)^2} & \frac{x_1 - x_2}{(x_1 - x_2)^2 + (y_1 - y_2)^2} & 0 & 0 \\ \frac{y_1 - y_2}{(x_1 - x_2)^2 + (y_1 - y_2)^2} & \frac{-x_1 + x_2}{(x_1 - x_2)^2 + (y_1 - y_2)^2} & 0 & 0 & \frac{-y_1 + y_2}{(x_1 - x_2)^2 + (y_1 - y_2)^2} & \frac{x_1 - x_2}{(x_1 - x_2)^2 + (y_1 - y_2)^2} & 0 & 1 \\ \frac{x_1 - x_2}{C} & \frac{y_1 - y_2}{C} & \frac{z_1 - z_2}{C} & 0 & \frac{-x_1 + x_2}{C} & \frac{-y_1 + y_2}{C} & \frac{-z_1 + z_2}{C} & 0 \end{bmatrix}$$

$$A = \sqrt{(x_1 - x_2)^2 + (y_1 - y_2)^2 * ((x_1 - x_2)^2 + (y_1 - y_2)^2 + (z_1 - z_2)^2)}$$

$$B = \frac{\sqrt{(x_1 - x_2)^2 + (y_1 - y_2)^2}}{(x_1 - x_2)^2 + (y_1 - y_2)^2 + (z_1 - z_2)^2}$$

$$C = \sqrt{(x_1 - x_2)^2 + (y_1 - y_2)^2 + (z_1 - z_2)^2}$$

### A.2. Two Robot Inverse Jacobian

$$J^{-1} = \begin{bmatrix} 1 & 0 & 0 & -\frac{1}{2}p\cos(\alpha)\cos(\beta) & \frac{1}{2}p\sin(\alpha)\sin(\beta) & 0 & 0 & -\frac{1}{2}\sin(\alpha)\cos(\beta) \\ 0 & 1 & 0 & -\frac{1}{2}p\sin(\alpha)\cos(\beta) & -\frac{1}{2}p\cos(\alpha)\sin(\beta) & 0 & 0 & \frac{1}{2}\cos(\alpha)\cos(\beta) \\ 0 & 0 & 1 & 0 & \frac{1}{2}p\cos(\beta) & 0 & 0 & \frac{1}{2}\sin(\beta) \\ 0 & 0 & 0 & 1 & 0 & 1 & 0 & 0 \\ 1 & 0 & 0 & \frac{1}{2}p\cos(\alpha)\cos(\beta) & -\frac{1}{2}p\sin(\alpha)\sin(\beta) & 0 & 0 & \frac{1}{2}\sin(\alpha)\cos(\beta) \\ 0 & 1 & 0 & \frac{1}{2}p\sin(\alpha)\cos(\beta) & \frac{1}{2}p\cos(\alpha)\sin(\beta) & 0 & 0 & -\frac{1}{2}\cos(\alpha)\cos(\beta) \\ 0 & 0 & 1 & 0 & -\frac{1}{2}p\cos(\beta) & 0 & 0 & -\frac{1}{2}\sin(\beta) \\ 0 & 0 & 0 & 1 & 0 & 0 & 1 & 0 \end{bmatrix}$$

## Appendix B

This appendix presents the Matlab script used in the optimization block of the Simulink controllers. This script is a modified version of the Hooke and Jeeves method downloaded on the Math Works File Exchange [42]. The modifications made consist of bounding the allowable radii and penalizing objective function results that violate these bounds.

```
function
[xopt,Opt,Nav]=hkjeevesmodified(Sx,guess,ip,Lb,Ub,problem,tol,mxit,stp,
amp,red,varargin)
%   Unconstrained optimization using Hooke & Jeeves.
%
%
[xopt,Opt,Nav]=hkjeeves(Sx,guess,ip,Lb,Ub,problem,tol,mxit,stp,amp,red,
par1, par2,...)
%
%   Sx           : objective function
%   guess        : initial point
%   par          : parameters needed to function
%   ip           : (0): no plot (default), (>0) plot figure ip with
%   pause, (<0) plot figure ip
%   Lb, Ub : lower and upper bound vectors to plot (default =
guess*(1+/-2))
%   problem : (-1): minimum (default), (1): maximum
%   tol         : tolerance (default = 1e-4)
%   mxit        : maximum number of stages (default = 50*(1+4*~(ip>0)))
%   stp         : stepsize vector for the independent variables
(default = max(0.01*abs(guess+~guess),0.1))
%   amp         : stepsize enlargement factor (1,oo) (default = 1.5)
%   red         : stepsize reduction factor (0,1) (default = 0.5)
%   xopt        : optimal point
%   Opt         : optimal value of Sx
%   Nav         : number of objective function evaluations

%
% Copyright (c) 2001 by LASIM-DEQUI-UFRGS
% $Revision: 1.0 $ $Date: 2001/07/05 21:10:15 $
% Argimiro R. Secchi (arge@enq.ufrgs.br)
%
%
Modified by Giovani Tonel (giotonel@enq.ufrgs.br) - April 2007
direction = - 1 --> reverse
            1 --> direct
idx: variables index vector with enlarged stepsize
top: end of idx list
bottom: beggin of idx list
next: index of enlarged variable

if nargin < 2,
    error('hkjeeves requires two input arguments ''Sx,guess''' );
end
```

```

if nargin < 3 | isempty(ip),
    ip=0;
end
if nargin < 4 | isempty(Lb),
    Lb=-guess-~guess;
end
if nargin < 5 | isempty(Ub),
    Ub=2*guess+~guess;
end
if nargin < 6 | isempty(problem),
    problem=-1;
end
if nargin < 7 | isempty(tol),
    tol=1e-4;
end
if nargin < 8 | isempty(mxit),
    mxit=50*(1+4*~(ip>0));
end
if nargin < 9 | isempty(stp),
    stp=max(0.01*abs(guess+~guess),0.1);
else
    stp=abs(stp(:));
end
if nargin < 10 | isempty(amp) | amp <= 1,
    amp=1.5;
end
if nargin < 11 | isempty(red) | red <= 0 | red >= 1,
    red=0.5;
end

% guess=guess(:);

yo= feval(Sx,guess)*problem;
% Assigning an out of bounds penalty
if (guess(1) < 1.7) | (guess(1) > 4) | (guess(5) < 1.7) | (guess(5) > 4)
    yo = yo*100;
end
n=size(guess,1);

x=guess;
xopt=guess;
Opt=yo;
it=0;
Nav=1;
top=0;
bottom=0;
idx=zeros(n+1,1);
idx(bottom+1)=top;

while it < mxit
    next=bottom;
    norma=0;
        % exploration
    for i=1:2:n,
        stp_i = stp(i);

```

```

for direction=[1 -1],
    x(i)=xopt(i)+stp_i*direction;
    y= feval(Sx,x, varargin{:} )*problem;
    % Assigning an out of bounds penalty
    if (i == 1) | (i == 5)
        if (x(i) < 1.7) | (x(i) > 4)
            y = y*10;
        end
    end
    Nav=Nav+1;

    if y > yo,      % success
        yo=y;
        if ip & n == 2,
            plot([x(1) xopt(1)], [x(2) xopt(2)], 'r');
            if ip > 0,
                disp('Pause: hit any key to continue...'); pause;
            end
        end
    end

    xopt(i)=x(i);
    idx(next+1)=i;
    next=i;
    break;
end
end

x(i)=xopt(i);
norma=norma+stp_i*stp_i/(x(i)*x(i)+abs(x(i))<tol));
end

it=it+1;

if sqrt(norma) < tol & abs(yo-Opt) < tol*(0.1+abs(Opt)),
    break;
end
% progression
if next==bottom,
    stp=stp*red;
else
    good=1;
    idx(next+1)=top;

    while good,
        Opt=yo;

        next=idx(bottom+1);
        while next ~= top,
            x(next)=x(next)+amp*(x(next)-guess(next));
            guess(next)=xopt(next);
            next=idx(next+1);
        end

        if idx(bottom+1) ~= top,

```

```

y= feval(Sx, x, varargin{:} )*problem;
% Assigning an out of bounds penalty
if (x(1) < 1.7) | (x(1) > 4) | (x(5) < 1.7) | (x(5) > 4)
    y = y*10;
end

Nav=Nav+1;

if y > yo,
    yo=y;
    good=1;

if ip & n == 2,
    plot([x(1) xopt(1)], [x(2) xopt(2)], 'r');
    if ip > 0,
        disp('Pause: hit any key to continue...'); pause;
    end
else
    good=0;
end

next=idx(bottom+1);
while next ~= top,
    if good,
        xopt(next)=x(next);
    else
        x(next)=xopt(next);
    end
    next=idx(next+1);
end
end
end

Opt=yo*problem;

if it == mxit,
    disp('Warning Hkjeeves: reached maximum number of stages! ');
elseif ip & n == 2,
    h2=plot(xopt(1),xopt(2),'r*');
    legend([h1,h2], 'start point', 'optimum');
end

disp('Optimization by hkjeeves terminated!')

```

## Appendix C

This appendix provides the raw data, plots, and  $R^2$  values for the curve fitting described in Chapter 4.

### C.1. Raw Data

The following figure shows the two orientations of the Pioneer used as the tracked object. These two orientations were chosen because they represent the largest and smallest surface area of the Pioneer that the quadrotor was likely to see during normal operations. At each distance, a minimum of 200 data points were recorded. The mean number of white pixels seen by the controller were recorded and can be seen in Table 0.1.



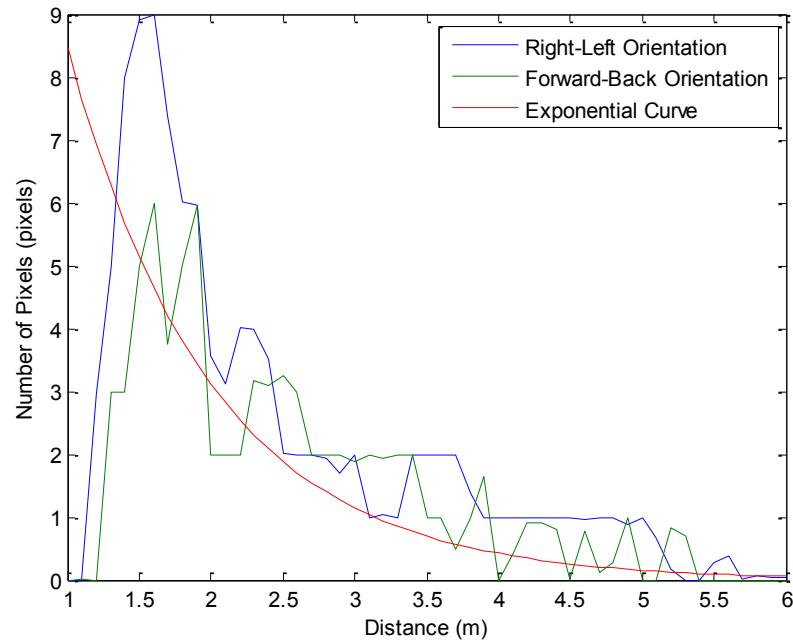
**Figure 0.1: The Pioneer on the left has a forward-back orientation while the Pioneer on the right has a right-left orientation.**

**Table 0.1:** Raw data of the size of the Pioneer as “seen” by the quadrotor at various distances.

Distance (m)	Right-Left (number of pixels)	Forward-Back (number of pixels)	Distance (m)	Right-Left (number of pixels)	Forward-Back (number of pixels)
1.0	0	0	3.6	1.9981	1
1.1	0	0.0056	3.7	1.9952	0.4982
1.2	2.9704	0	3.8	1.375	1
1.3	5	3	3.9	1	1.6466
1.4	8	3	4.0	1	0
1.5	8.9112	4.9927	4.1	1	0.4366
1.6	8.9786	6	4.2	1	0.9116
1.7	7.3807	3.7558	4.3	1	0.9172
1.8	6.0028	5.0187	4.4	1	0.8012
1.9	5.9695	5.9669	4.5	1	0.0222
2.0	3.5773	2	4.6	0.9482	0.7742
2.1	3.1101	2	4.7	1	0.1197
2.2	4.0127	2	4.8	1	0.2841
2.3	4	3.1797	4.9	0.8742	1
2.4	3.5066	3.1004	5.0	0.9874	0
2.5	2.0021	3.2431	5.1	0.6613	0
2.6	2	3	5.2	0.1588	0.8274
2.7	2	2	5.3	0	0.6962
2.8	1.9356	2	5.4	0	0
2.9	1.6939	2	5.5	0.2687	0
3.0	2	1.8918	5.6	0.3876	0
3.1	1	2	5.7	0.0176	0
3.2	1.0443	1.9472	5.8	0.0728	0
3.3	1	1.9967	5.9	0.0448	0
3.4	2	2	6.0	0.0397	0
3.5	2	1			

## C.2. Plots

The raw data and the exponential curve used to approximate this data are shown in Figure 0.2. Distances of 1.7 m to 3.0 m were chosen for both sets of data since the quadrotor could easily “see” the Pioneer in both orientations at these distances. The curve’s  $R^2$  value was measured in this range for both the right-left orientation and the forward-back orientation. The results are shown in Table 0.2 below. These results confirm the exponential curve’s validity for this data range. The exponential equation was then solved for distance and used to determine  $D$  in the experiments and simulations discussed in this dissertation.



**Figure 0.2: Raw data and exponential curve.**

**Table 0.2:  $R^2$  values for the exponential curve fit.**

Orientation	$R^2$ Value
Right-Left	0.96
Forward-Back	0.90

## Appendix D

This appendix provides the shared testbed description written by Jasmine Cashbaugh, Anne Mahacek, Alicia Sherban, and Christian Zempel. It provides additional detail on the testbed described in Chapter 5.

### D.1. System Overview

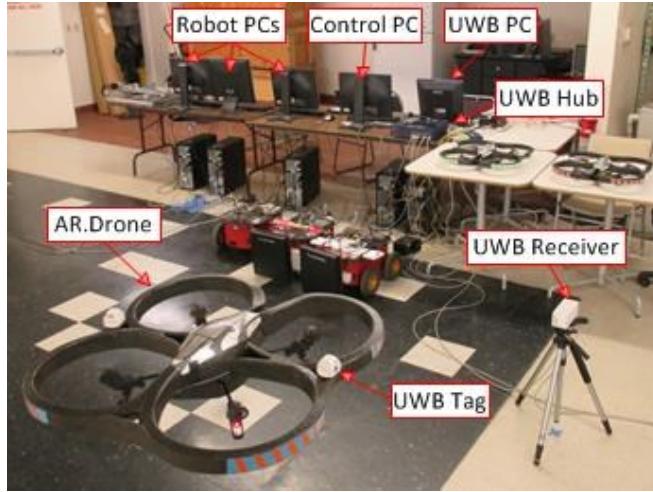
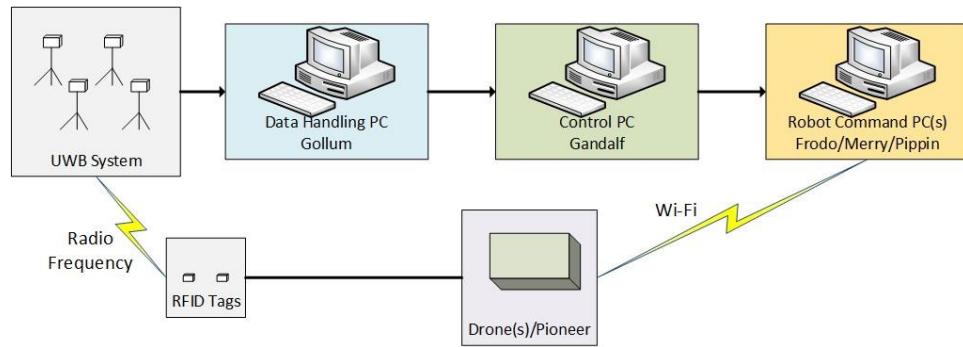


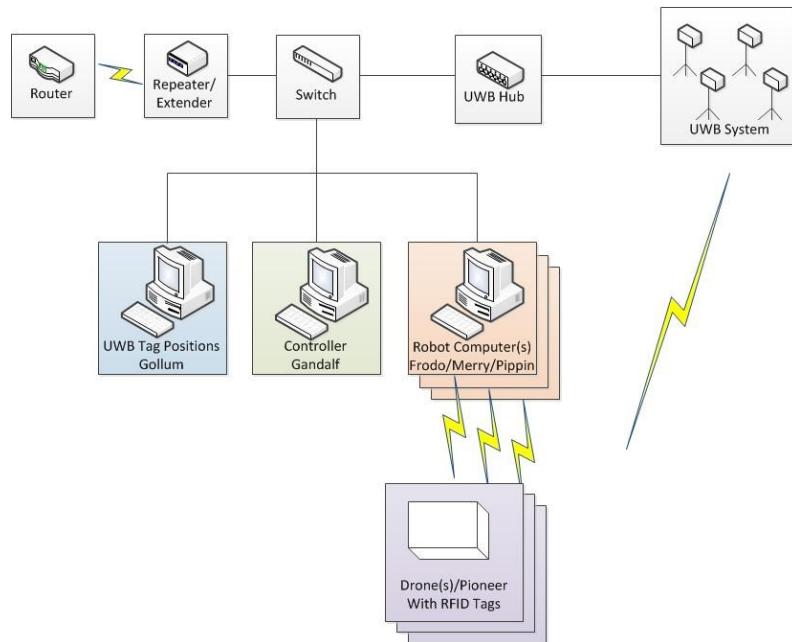
Figure 0.1: Physical hardware layout.

This testbed was designed to apply multi-robot cluster control techniques for aerial vehicles and land rovers. The testbed consists of the robots, an Ultra Wideband (UWB) system that can track the position of Radio Frequency Identification (RFID) tags attached to individual robots, a data receiving and parsing computer, a controller computer, individual robot command computers, WiFi connections, and a modem connection. These components can be described in the following four categories: robots, sensing system, networking, and data handling. To limit the number of environmental factors that might affect the test results, the aerial robots were tested in an indoor environment where their positions could be measured accurately at all times. Additionally, the robots were controlled using Matlab and Simulink since legacy controllers were developed in the Robotic Systems Laboratory (RSL) using this software [1]. A more detailed

explanation of these components can be found in the following sections. A physical hardware layout can be seen in Figure 0.1 accompanied by the testbed flowchart and hardware layout diagram in Figure 0.2 and Figure 0.3, respectively.



**Figure 0.2: Testbed flowchart.**



**Figure 0.3: Hardware layout diagram.**

## D.2. AR.Drone

Quadrotors are classed as an aerial vehicle driven by four rotors. As these rotors vary in speed, the quadrotor is able to move translationally and rotationally depending on the rotor

configuration. The AR.Drone is a hobby class quadrotor developed by Parrot to be controlled via Apple and Android operating systems. The drones have a total of four independently controlled degrees of freedom (DOF): three translational and one rotational. Drone movement is constrained translationally along the local  $x$ ,  $y$ , and  $z$  axes and rotationally about the  $z$  axis, or yaw; rotation about the  $x$  axis, or roll, and the  $y$  axis, or pitch, are coupled with the translation of  $y$  and  $x$ , respectively.

The AR.Drone hull measures 52.5 cm by 51.5 cm and weighs 400 g. The drone has a maximum running speed of 5 m/s with an approximate running time of 15 minutes. The AR.Drone includes a carbon tube structure and four brushless motors that run at 3,500 rpm. The drone additionally includes a 93 degree wide angle diagonal lens front camera, a 64 degree diagonal high speed vertical camera, an ultrasound altimeter, a three axis accelerometer, a two axis gyroscope, and a one axis yaw precision gyroscope. The drones are powered by rechargeable Lithium-Polymer batteries. The 3-cell battery has a capacity of 1000 mAh and 11.1 volts [2].

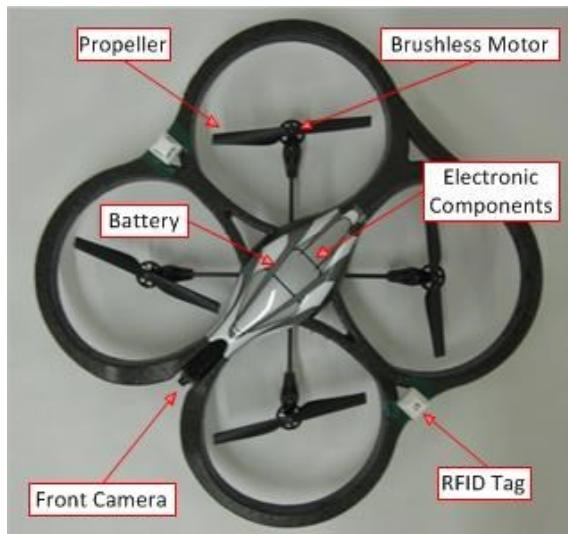


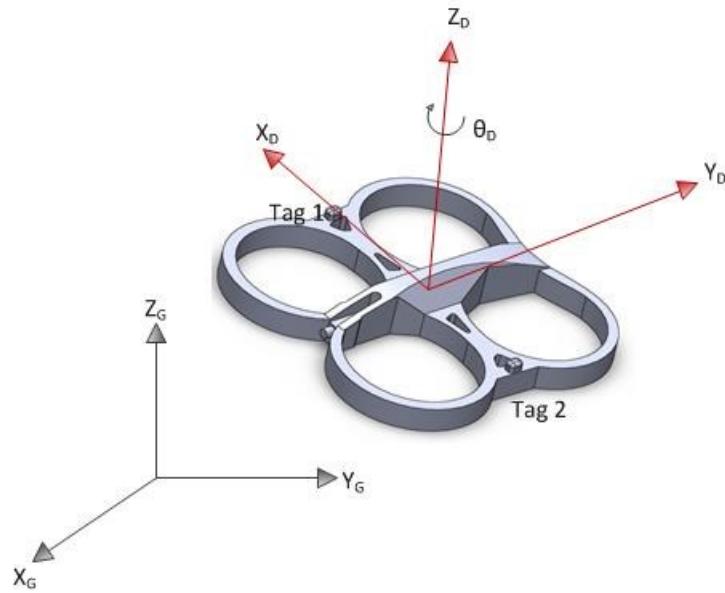
Figure 0.4: AR.Drone components.

The AR.Drone has an onboard control that converts commands, in the form of a vector shown in Eq. (0.1) below, to drive actuators on each of the four motors.

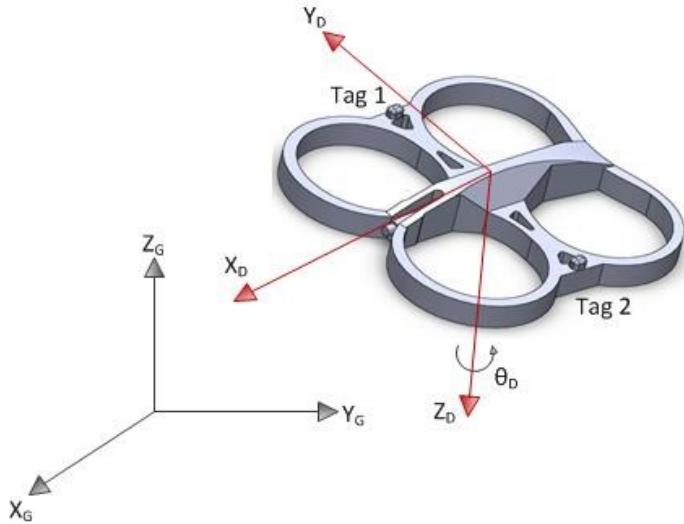
$$[x \text{ velocity} \quad y \text{ velocity} \quad z \text{ velocity} \quad \text{yaw rate}] \quad (0.1)$$

It is important to note that the command reference frame of the AR.Drone differs from the reference frame used for modeling the system, necessitating an additional rotation matrix to match these reference frames. The difference between these two frames is shown in Figure 0.5 and Figure 0.6 below while the rotation matrix is shown in Eq. (0.2).

$${}_{AR}^D R = \begin{bmatrix} 0 & -1 & 0 \\ 1 & 0 & 0 \\ 0 & 0 & -1 \end{bmatrix} \quad (0.2)$$



**Figure 0.5: AR.Drone coordinate frame.**



**Figure 0.6: Drone coordinate frame.**

The AR.Drone also has onboard hover control and stability control that is activated if commands are not continually sent to the drone. In the absence of a command, the hover control allows the drone to hover autonomously over a single location utilizing the downward facing camera; an emergency land is executed if a large enough roll or pitch is detected.

Each AR.Drone natively broadcasts its own wireless network so no additional software was needed on the drones themselves.

### D.3. Sensing System

#### D.3.1. Hardware

The Sapphire DART Ultra Wideband Tracking System was used in order to collect robot position data. The UWB Tracking System consists of a series of receivers placed around the perimeter of the testbed area at various heights and radio frequency identification tags placed on the robots. The RFID tags broadcast at 25 Hz while the receivers triangulate the position of the tags. Two RFID tags are secured to the robots in a known configuration; the individual tag positions are

then used to calculate the position and orientation of the robot in the  $x$ ,  $y$ , and  $z$  dimensions. The testbed consists of eleven receivers with three separate “daisy chain” connections. Figure 0.8, Table 0.1, and Table 0.2 show the placement of the receivers and a reference tag which allows the system to have a relative zero location within the workspace [3].



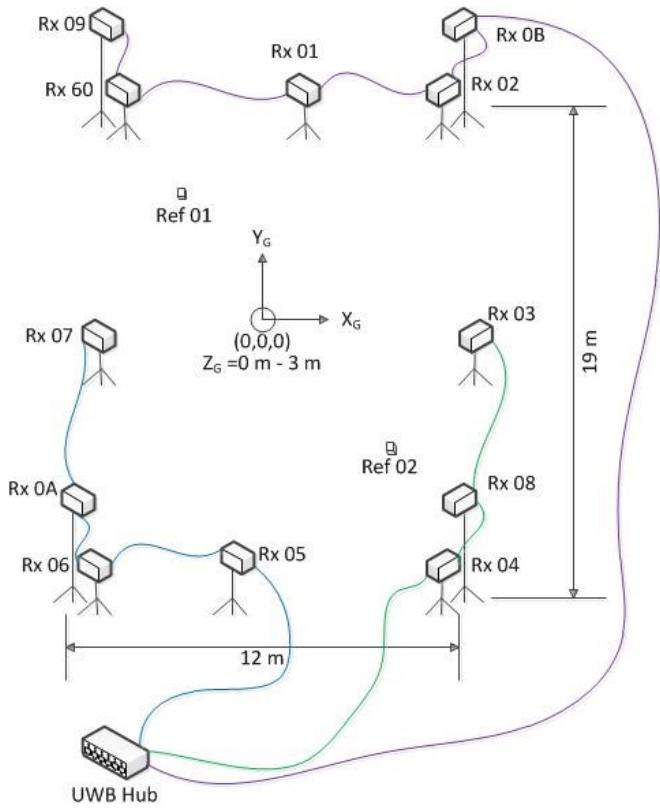
**Figure 0.7: UWB receiver and RFID tag.**

**Table 0.1: Locations of the UWB reference tags.**

Reference Tag	X Position (m)	Y Position (m)	Z Position (m)
Ref 01	4.04	-3.98	0.00
Ref 02	-4.03	4.14	0.00

**Table 0.2: Locations of the UWB receivers.**

Receiver Name	X Position (m)	Y Position (m)	Z Position (m)
Rx 01	-3.02	7.57	0.63
Rx 02	6.32	7.44	0.64
Rx 03	6.06	-2.51	0.64
Rx 04	5.83	-11.94	0.46
Rx 05	-0.80	-11.95	0.62
Rx 06	-7.00	-11.98	0.64
Rx 07	-7.44	-3.02	0.62
Rx 08	5.78	-12.01	4.39
Rx 09	-7.15	7.91	3.58
Rx 0A	-7.12	-12.08	4.22
Rx 0B	6.43	7.57	4.19
Rx 60	-7.19	7.82	0.46



**Figure 0.8: Layout of UWB system.**

### D.3.2. Software

The sensor system data is read using Java line commands in Matlab to read in the data available on the sensing system's socket (IP host and port). This data is then parsed into a vector to make it easier to use in the controller. The vector consists of time,  $x$ ,  $y$ , and  $z$  values for each RFID tag.

## D.4. Networking

### D.4.1. Hardware

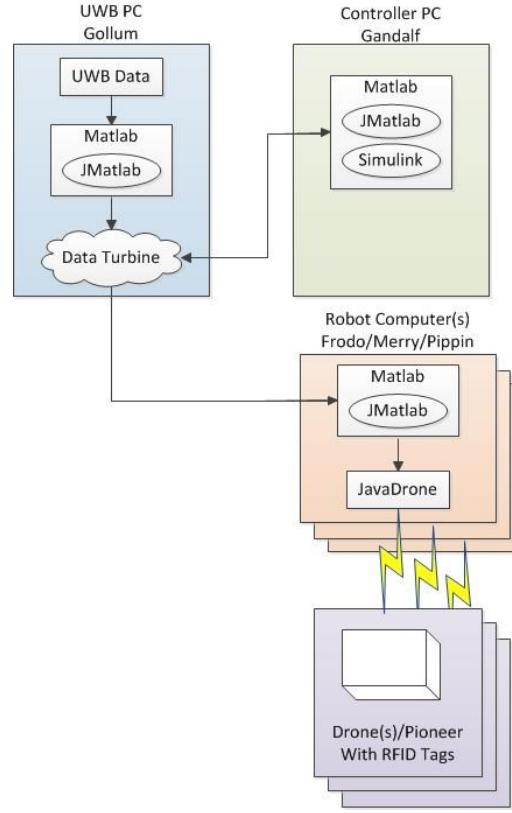
As seen in Figure 0.3, all computers and the UWB hub are connected via Ethernet cables to a Netgear ProSafe 8 port Gigabit Switch. A dual band range extender/repeater was also connected

via Ethernet to the switch to allow the computer to have internet access. The first computer, named Gollum, receives and parses data from the UWB hub and posts to a DataTurbine channel. The second computer, Gandalf, receives the parsed tag data, runs the Simulink controller, and posts individual drone commands to separate DataTurbine channels. There is one computer for each robot that receives its respective commands through a WiFi connection using a Netgear N300 Wireless USB Adapter.

#### *D.4.2. Software*

AR.Drones broadcast their own WiFi network at 192.168.1.x; therefore, all network devices needed to be set to a different address to avoid conflicting IP issues. The extender/repeater relays the DHCP request from the workstations to the router. Then the router distributes the IP address of 192.168.15.xxx to the workstations. Router subnet masks were set to match the AR.Drone subnet mask of 255.255.255.0.

DataTurbine was used to make various data accessible on all of the networked computers used in this testbed. DataTurbine software is designed for this purpose and allows the user to upload data to a data engine and download the data to any computer that can connect to the same data engine [4]. This software has been used extensively in SCU's RSL for connecting to multiple data sources using the same computer [5]. However, in this testbed, DataTurbine is used across a network of computers where a single instance of DataTurbine is run on Gollum and all data is sent to or received from this single instance. Figure 0.9 illustrates the software layout used in this testbed. Data from the sensor system is received on Gollum and imported into Matlab and then uploaded to DataTurbine and made available to the other computers in the network.



**Figure 0.9: Software layout.**

Gandalf acts as the control computer and runs Matlab and the Simulink cluster controller. Gandalf downloads the tag data from DataTurbine posted by Gollum, imports it into Simulink, and then uses this data to calculate velocity commands for each robot. Simulink outputs the robot command velocities and uploads them to DataTurbine, residing on Gollum. Each robot has its own command channel where its commands are uploaded by the control computer and downloaded by the computer dedicated to that robot.

The robot computers each use Matlab to download the robot command velocities and send them to the robot. The drone commands are sent using JavaDrone, an open source software package written in Java to control Parrot's AR.Drones using a graphical user interface (GUI) [6] [2]. To integrate Matlab and JavaDrone so that commands can be sent directly from Matlab to

JavaDrone, a modified version of jmatlab was used. Jmatlab is a Java software package developed by SCU's RSL to serve as a bridge between DataTurbine, written in Java, and Matlab. It enables users to implement Matlab functions without tying up the command line, allowing for multiple commands to be run at the same time [5]. Hooks were added to this version of jmatlab to allow the user to bypass the JavaDrone GUI and call the drone commands directly from Matlab.

## D.5. Characterization of Stationary Positioning

To verify the testbed, a simple stationary test was performed where a quadrotor with two tags attached to the hull was placed in the testbed at various locations while position data was collected and analyzed. The following table summarizes the UWB Tracking System error. Here, the actual value is the true position of the quadrotor, and the measured position is the average measured position plus or minus the measured standard deviation from this average. The x and y values are fairly accurate for a test area of approximately 12 m by 19 m. The z value is not as reliable, with errors over twice as high as for the x and y values. The errors for the theta values are also higher than the errors for the x and y values. This is expected because the theta values are calculated from the x and y values, thus compounding the errors.

**Table 0.3: UWB system error analysis.**

		X (m)	Y (m)	Z (m)
Two Quadrotors	<b>Mean Error</b>	0.05	0.07	0.39
	<b>Max Error</b>	0.35	0.32	0.85
	<b>Min Error</b>	0.00	0.00	0.03
	<b>Error Std</b>	0.06	0.08	0.11
	<b>RMS Error</b>	0.06	0.08	0.40
Three Quadrotors	<b>Mean Error</b>	0.07	0.07	0.35
	<b>Max Error</b>	0.39	0.28	1.27
	<b>Min Error</b>	0.00	0.00	0.00
	<b>Error Std</b>	0.08	0.07	0.11
	<b>RMS Error</b>	0.09	0.09	0.37

## References

- [1] C. Kitts and I. Mas, "Cluster Space Specification and Control of Mobile Multirobot Systems," *Mechatronics, IEEE/ASME Transactions on*, vol. 14, no. 2, pp. 207-218.
- [2] "AR.Drone Specifications," [Online]. Available: <http://diydrones.com/profiles/blogs/parrot-ardrones-specs-arm9>. [Accessed 6 July 2013].
- [3] Multispectral Solutions, Inc., *Sapphire Dart Digital Active Real-Time Tracking (Model H651/651A Indoor/ Outdoor) Users Guide Revision 2.1*, 2006.
- [4] "Data Turbine," [Online]. Available: <http://www.dataturbine.org/>. [Accessed 6 July 2013].
- [5] I. Mas, O. Petrovic and C. Kitts, ""Cluster Space Specification and Control of a 3-Robot Mobile System," *Proceedings - IEEE International Conference on Robotics and Automation*, 2008.
- [6] "javadrone: AR.Drone Java API," [Online]. Available: <http://code.google.com/p/javadrone/>. [Accessed 15 June 2012].

## Appendix E

This appendix provides an overview of the two drone simulation model used in this dissertation.

A block diagram of the entire simulation is shown in Figure 0.1. The foundation of this simulation is a two drone cluster controller, as discussed in Chapter 2. A tracking algorithm was then implemented in the large cyan block on the right of Figure 0.1. Details of this tracking algorithm can be seen in Figure 0.2. The tracking algorithm took the individual Pioneer location estimates from each quadrotor and combined them to obtain a single location estimate. This estimate was then input into a Kalman filter to smooth the estimate and eliminate any large jumps in the position estimate. The Pioneer was constrained to move at a maximum speed of 0.315 m/s so only small changes in position were physically possible. Next, the smoothed position estimate was used to determine where the cluster center should be positioned and a cluster position command was sent to the simulation. This cluster position command was input into a PID controller, shown in Figure 0.3, along with the current cluster position in order to determine the cluster velocity commands that were used to control the robots. The cluster controller then proceeded as discussed in Section 2.3. In this case, the current cluster position was simulated based on the cluster commands and typical actuator and measurement errors observed during physical testing.

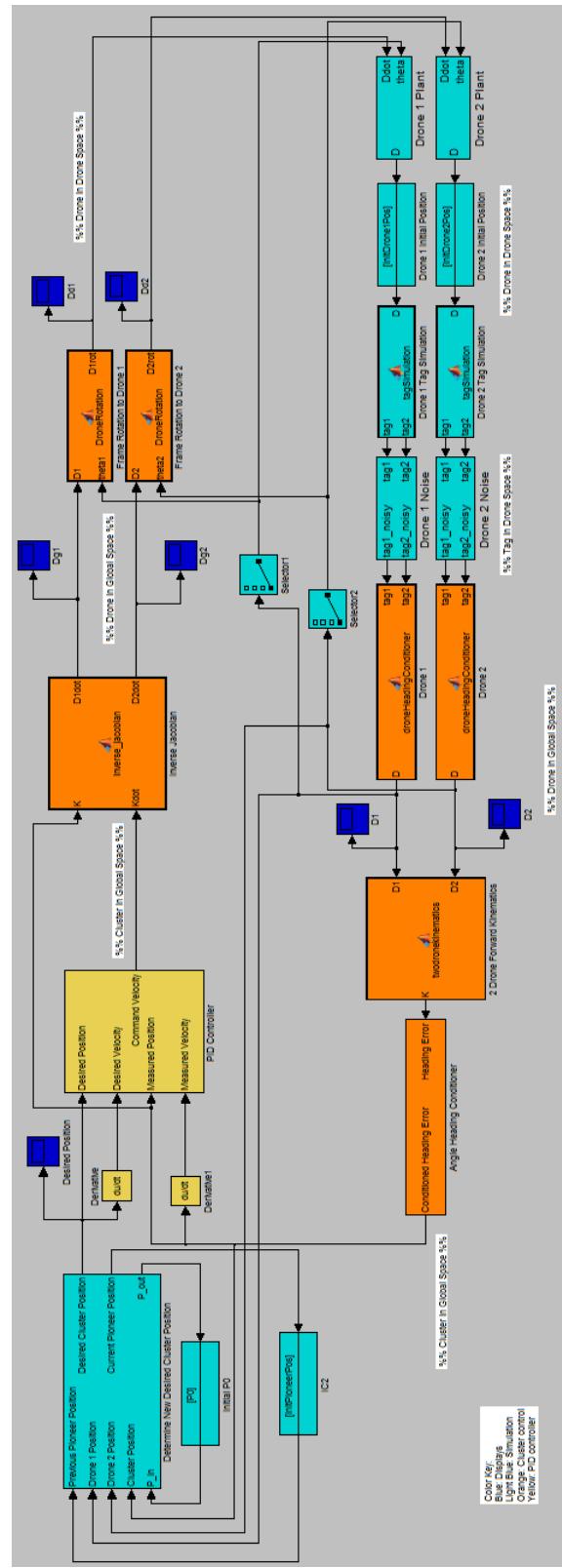


Figure 0.1: Block diagram of the two drone simulation.

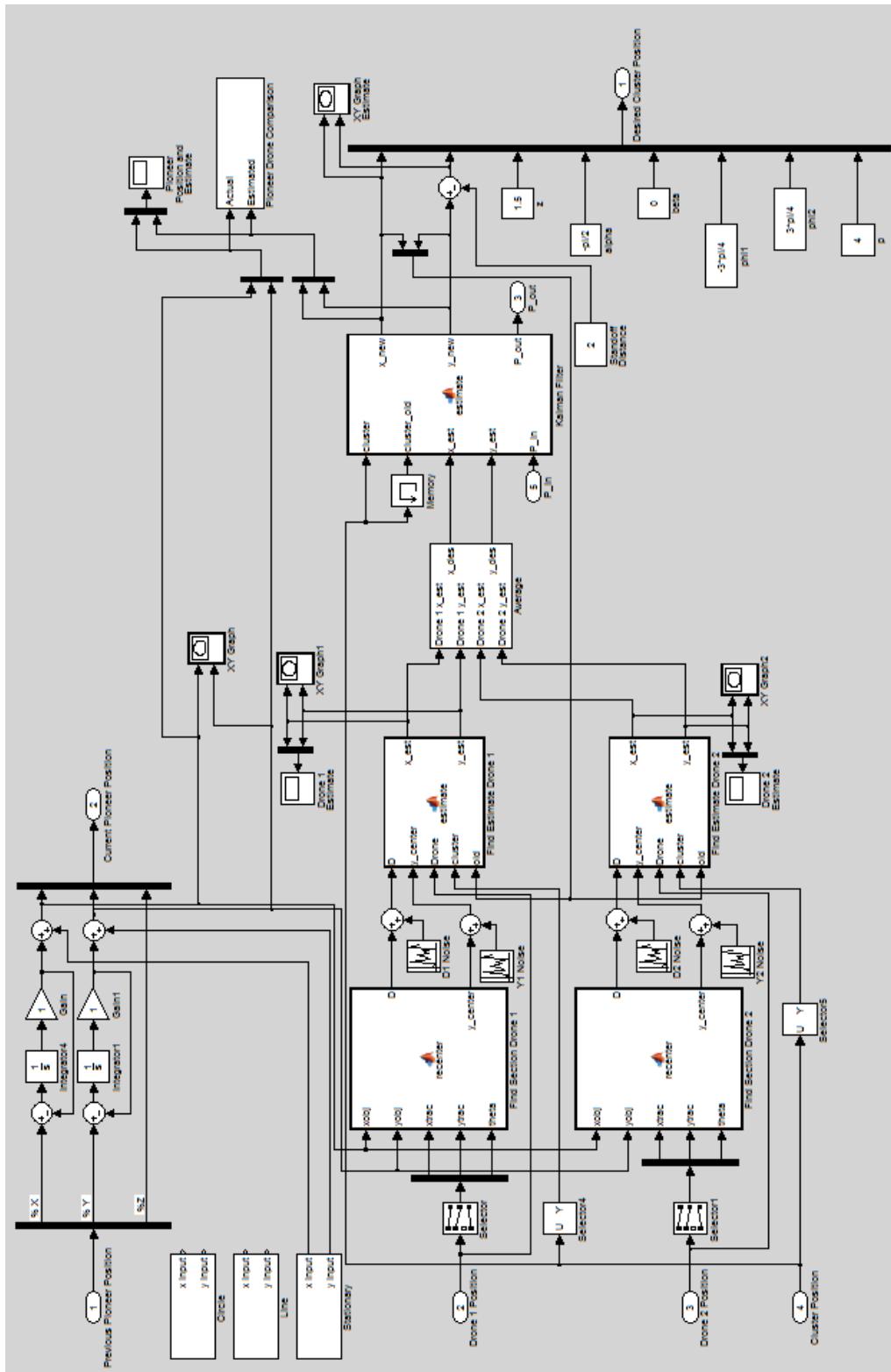


Figure 0.2: Block diagram of the tracking algorithm used in the two drone simulation.

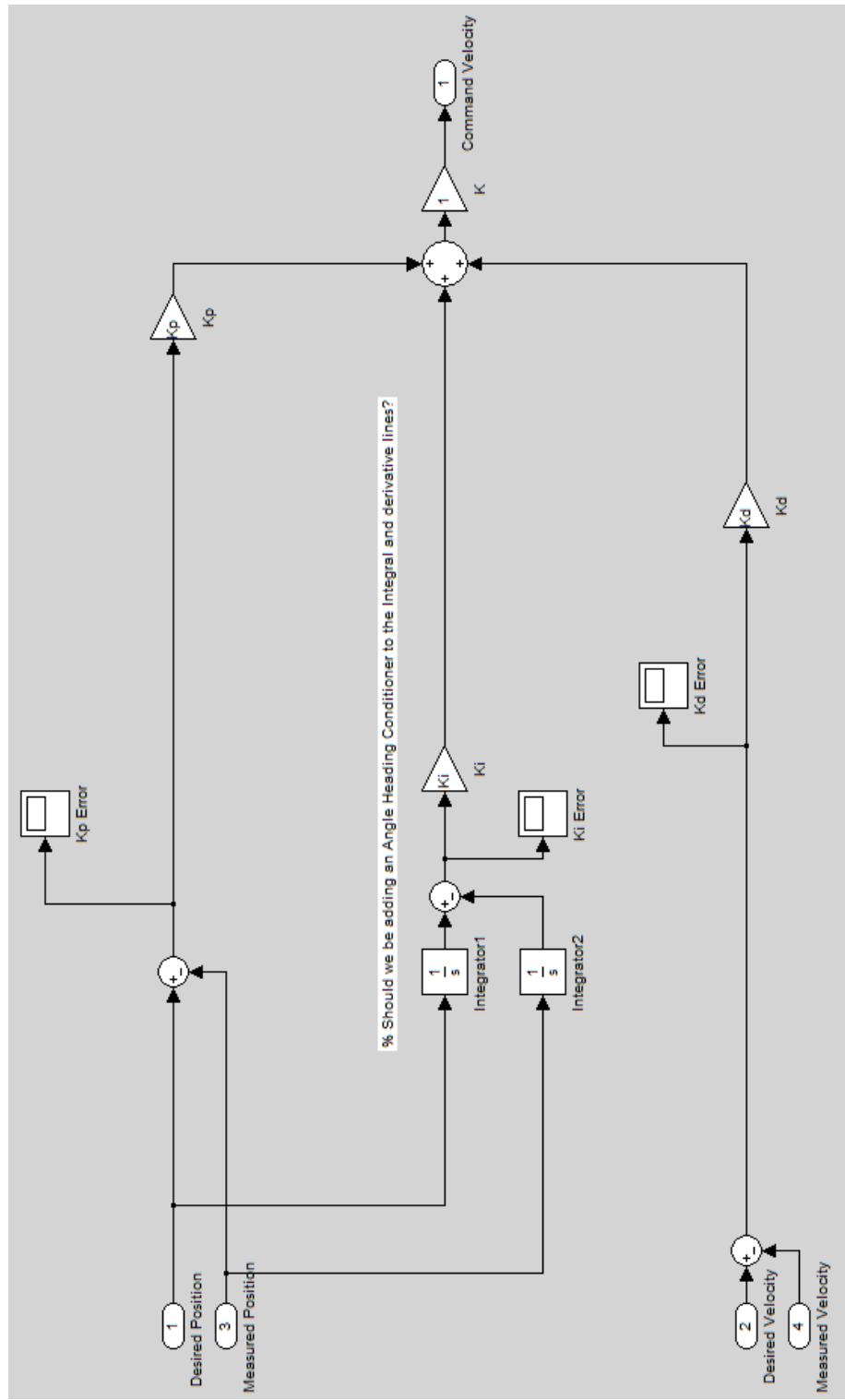
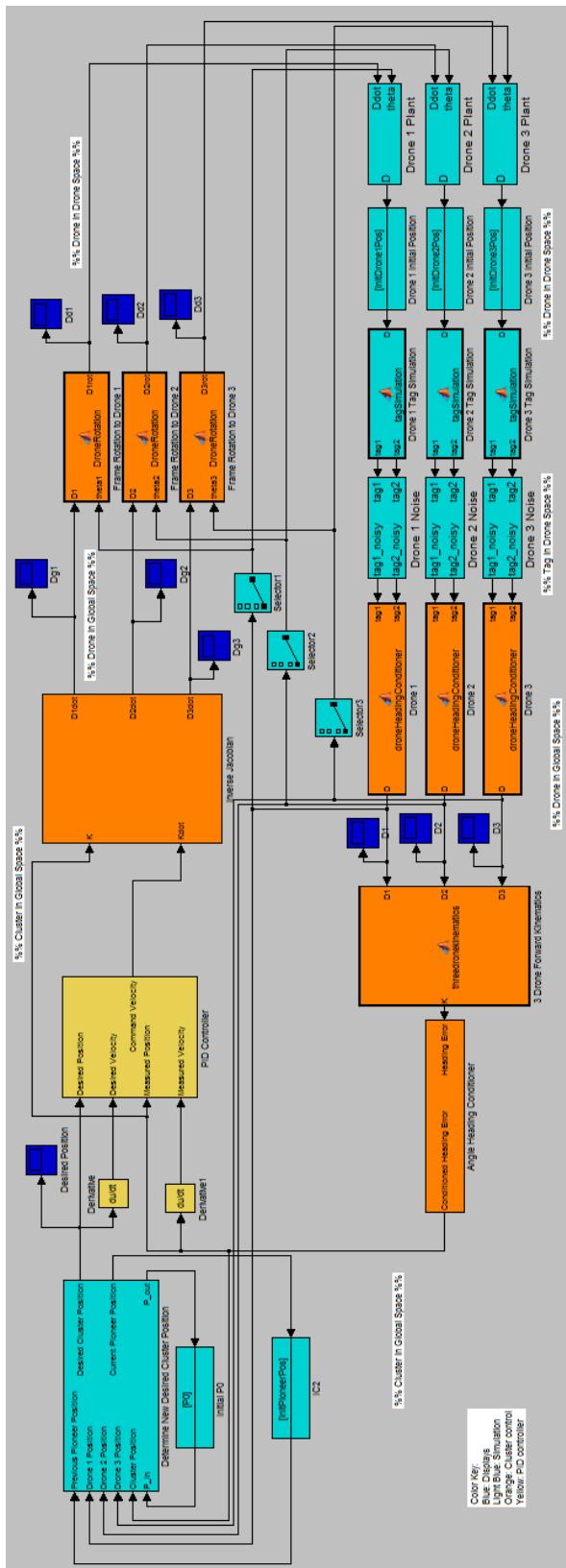


Figure 0.3: Block diagram of the PID controller used in the two drone simulation.

## Appendix F

This appendix provides an overview of the three drone simulation model used in this dissertation. A block diagram of the entire simulation is shown in Figure 0.1. The foundation of this simulation is a three drone cluster controller, discussed in Chapter 2. A tracking algorithm was then implemented in the large cyan block on the right of Figure 0.1, the details of which can be seen in Figure 0.2. The tracking algorithm took the individual Pioneer location estimates from each quadrotor and combined them to obtain a single location estimate. This estimate was then input into a Kalman filter to smooth the estimate and eliminate any large jumps in the position estimate. The Pioneer was constrained to move at a maximum speed of 0.315 m/s so only small changes in position were physically possible. Next, the smoothed position estimate was used to determine where the cluster center should be positioned and a cluster position command was sent to the simulation. This cluster position command was input into a PID controller, shown in Figure 0.3, along with the current cluster position in order to determine the cluster velocity commands that were used to control the robots. The cluster controller then proceeded as discussed in Section 2.3. In this case, the current cluster position was simulated based on the cluster commands and typical actuator and measurement errors observed during physical testing.



**Figure 0.1: Block diagram of the three drone simulation.**

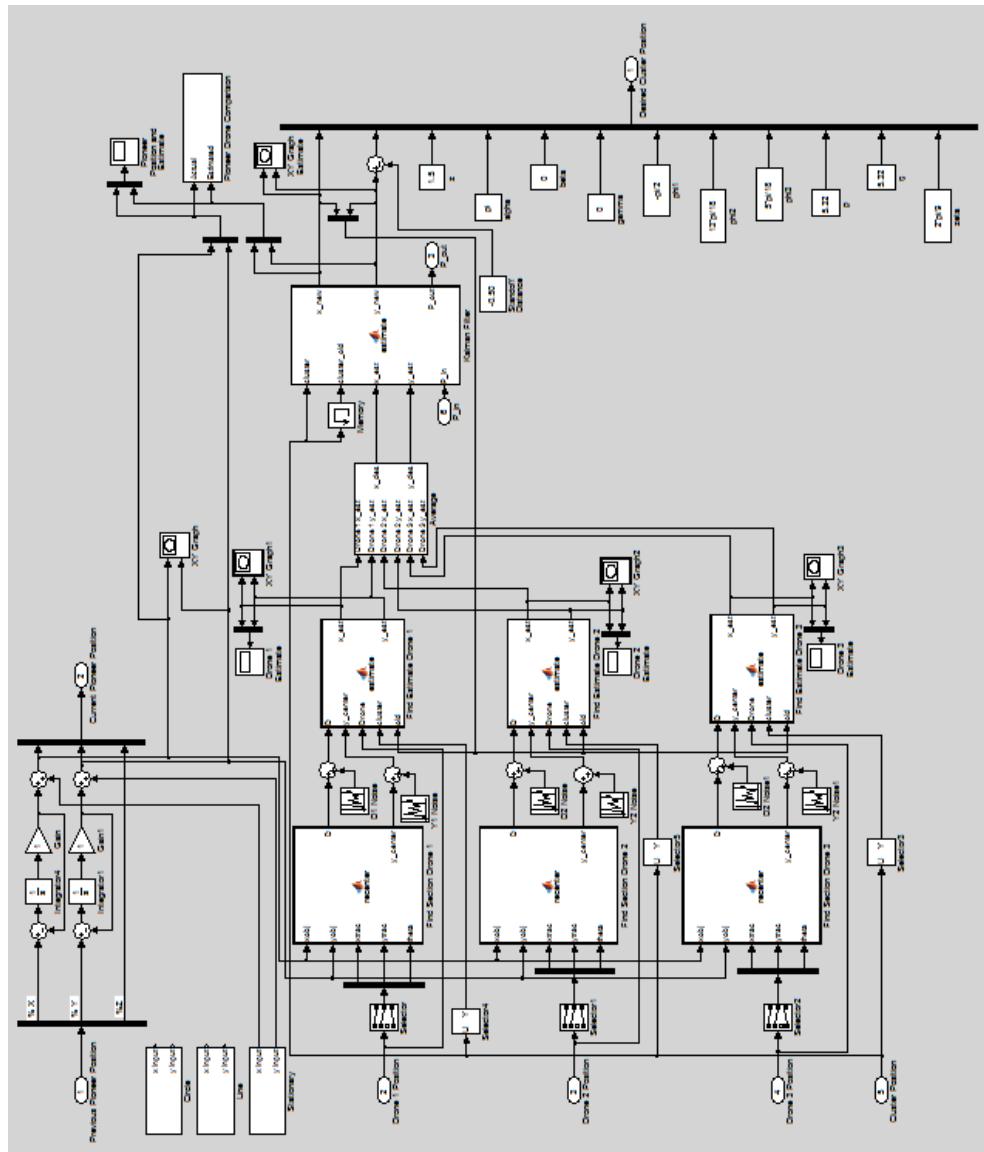


Figure 0.2: Block diagram of the tracking algorithm used in the three drone simulation.

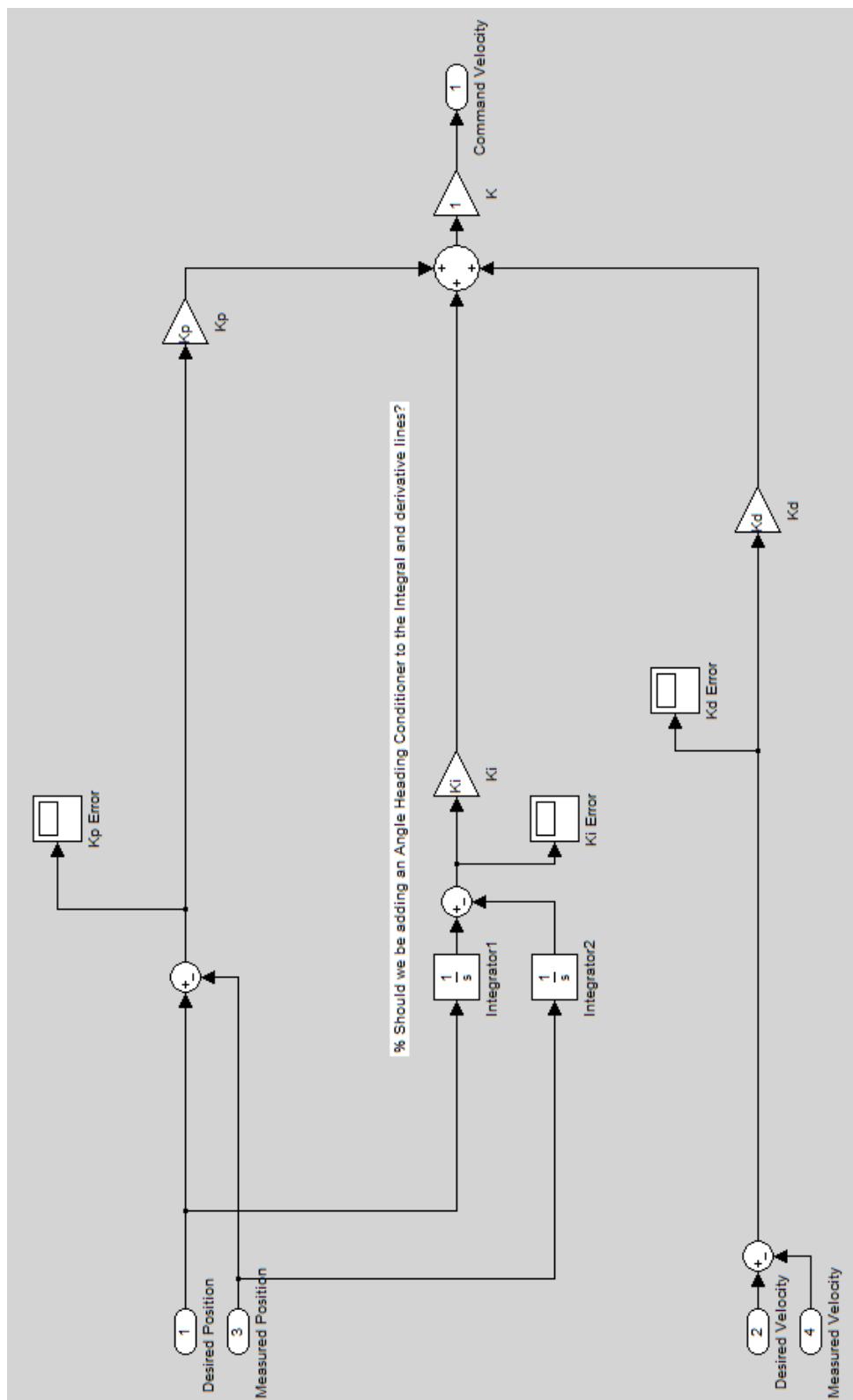


Figure 0.3: Block diagram of the PID controller used in the three drone simulation.

## Appendix G

This appendix provides the calculations used to determine the area of the 60% error covariance matrix.

### G.1. Calculating the Position Errors

The first step in calculating the error covariance matrix is to calculate the position errors. The position errors are the difference between the mobile tracking stations' estimate of the tracked object's position and the tracked object's actual position. The tracked object was assumed to be on the ground, so no estimate was made of its z position. The  $x$  and  $y$  position errors were found using Eq. (0.1) and (0.2), respectively, and were then used to calculate the total difference, as shown in Eq. (0.3).

$$x_{error} = x_{estimate} - x_{actual} \quad (0.1)$$

$$y_{error} = y_{estimate} - y_{actual} \quad (0.2)$$

$$diff_{total} = \sqrt{x_{error}^2 + y_{error}^2} \quad (0.3)$$

### G.2. Calculating the 60% Confidence Interval Error Ellipse Area

Next, the total difference calculated in Eq. (0.3) was entered into the Matlab script in Figure 0.1 to calculate the 60% confidence interval error ellipse area.

```
pd = fitdist(diff_total, 'Normal');
comb_cov = paramci(pd, 0.4);
eigenvalues_comb = eig(comb_cov);
comb_ellipse_axes = sqrt(eigenvalues_comb);
a = comb_ellipse_axes(1);
b = comb_ellipse_axes(2);
A = pi*a*b;
fprintf('\nThe area of the combined covariance ellipse is %1.4f m^2.\n', A);
```

Figure 0.1: Matlab script used to calculate the 60% confidence interval error ellipse area.