
# NAVAL POSTGRADUATE SCHOOL

**MONTEREY, CALIFORNIA**

# THESIS

**APPLYING COOPERATIVE LOCALIZATION TO SWARM UAVS USING AN EXTENDED KALMAN FILTER**

by

Robert B. Davis

September 2014

Thesis Co-Advisors: Timothy H. Chung
Duane T. Davis

**Approved for public release; distribution is unlimited**

THIS PAGE INTENTIONALLY LEFT BLANK

# REPORT DOCUMENTATION PAGE

Form Approved OMB No. 0704–0188



| 1. AGENCY USE ONLY *(Leave Blank)* | 2. REPORT DATE | 3. REPORT TYPE AND DATES COVERED |
|---|---|---|
| | 09-26-2014 | Master's Thesis 01-06-2014 to 09-26-2014 |

**4. TITLE AND SUBTITLE**

APPLYING COOPERATIVE LOCALIZATION TO SWARM UAVS USING AN EXTENDED KALMAN FILTER

**5. FUNDING NUMBERS**

**6. AUTHOR(S)**

Robert B. Davis

**7. PERFORMING ORGANIZATION NAME(S) AND ADDRESS(ES)**

Naval Postgraduate School
Monterey, CA 93943

**8. PERFORMING ORGANIZATION REPORT NUMBER**

**9. SPONSORING / MONITORING AGENCY NAME(S) AND ADDRESS(ES)**

Department of the Navy

**10. SPONSORING / MONITORING AGENCY REPORT NUMBER**

**11. SUPPLEMENTARY NOTES**

The views expressed in this document are those of the author and do not reflect the official policy or position of the Department of Defense or the U.S. Government. IRB Protocol Number: N/A.

**12a. DISTRIBUTION / AVAILABILITY STATEMENT**

Approved for public release; distribution is unlimited

**12b. DISTRIBUTION CODE**

**13. ABSTRACT** *(maximum 200 words)*

Cooperative Localization (CL) is a process by which autonomous vehicles operating as a team estimate the position of one another to compensate for errors in the positioning sensors used by a single agent. By combining independent measurements originating from members of the team, a single estimate of increased accuracy will result. This approach has the potential to enhance the positional accuracy of an agent over use of a standard GPS, which would be essential for behaviors within a swarm requiring precision movements such as maintaining close formation. CL can also provide accurate positional information to the entire group when operating in an intermittent or denied GPS environment. In this thesis, a distributed CL algorithm is implemented on a swarm of Unmanned Aerial Vehicles (UAVs) using an Extended Kalman Filter. Using a technique created for ground robots, the equations are modified to adapt the algorithm to aerial vehicles, and then operation of the algorithm is demonstrated in a centralized system using AR Drones and the Robot Operating System. During tests, the positional accuracy of the UAV using CL improved over use of dead reckoning. However, the performance is not as expected based on the results noted from the referenced two-dimensional application of the algorithm. It is presumed that the sensors on-board the AR Drone are responsible. Since the platform is simply a low-cost solution to show proof-of-concept, it is concluded that the implementation of CL presented in this thesis is a suitable approach for enhancing positional accuracy of UAVs within a swarm.

| 14. SUBJECT TERMS | 15. NUMBER OF PAGES |
|---|---|
| cooperative localization, multi-robot coordination, swarm, autonomous aerial vehicles, Kalman filter | 107 |
| | 16. PRICE CODE |

| 17. SECURITY CLASSIFICATION OF REPORT | 18. SECURITY CLASSIFICATION OF THIS PAGE | 19. SECURITY CLASSIFICATION OF ABSTRACT | 20. LIMITATION OF ABSTRACT |
|---|---|---|---|
| Unclassified | Unclassified | Unclassified | UU |

NSN 7540-01-280-5500

Standard Form 298 (Rev. 2–89)
Prescribed by ANSI Std. 239–18

THIS PAGE INTENTIONALLY LEFT BLANK

**Approved for public release; distribution is unlimited**

**APPLYING COOPERATIVE LOCALIZATION TO SWARM UAVS USING AN EXTENDED KALMAN FILTER**

Robert B. Davis
Lieutenant Colonel, United States Marine Corps
B.S., Virginia Military Institute, 1997

Submitted in partial fulfillment of the
requirements for the degree of

**MASTER OF SCIENCE IN COMPUTER SCIENCE**

from the

**NAVAL POSTGRADUATE SCHOOL**
**September 2014**

Author: Robert B. Davis

Approved by: Dr. Timothy H. Chung
Thesis Co-Advisor

Dr. Duane T. Davis
Thesis Co-Advisor

Dr. Peter J. Denning
Chair, Department of Computer Science

THIS PAGE INTENTIONALLY LEFT BLANK

# ABSTRACT

Cooperative Localization (CL) is a process by which autonomous vehicles operating as a team estimate the position of one another to compensate for errors in the positioning sensors used by a single agent. By combining independent measurements originating from members of the team, a single estimate of increased accuracy will result. This approach has the potential to enhance the positional accuracy of an agent over use of a standard GPS, which would be essential for behaviors within a swarm requiring precision movements such as maintaining close formation. CL can also provide accurate positional information to the entire group when operating in an intermittent or denied GPS environment. In this thesis, a distributed CL algorithm is implemented on a swarm of Unmanned Aerial Vehicles (UAVs) using an Extended Kalman Filter. Using a technique created for ground robots, the equations are modified to adapt the algorithm to aerial vehicles, and then operation of the algorithm is demonstrated in a centralized system using AR Drones and the Robot Operating System. During tests, the positional accuracy of the UAV using CL improved over use of dead reckoning. However, the performance is not as expected based on the results noted from the referenced two-dimensional application of the algorithm. It is presumed that the sensors on-board the AR Drone are responsible. Since the platform is simply a low-cost solution to show proof-of-concept, it is concluded that the implementation of CL presented in this thesis is a suitable approach for enhancing positional accuracy of UAVs within a swarm.

THIS PAGE INTENTIONALLY LEFT BLANK

# Table of Contents

# List of Figures

# List of Tables

THIS PAGE INTENTIONALLY LEFT BLANK

# List of Acronyms and Abbreviations

| | |
|---|---|
| ARSENL | Advanced Robotic Systems Engineering Lab |
| BSD | Berkeley Software Distribution |
| CL | Cooperative Localization |
| DGPS | Differential Global Positioning System |
| DOD | Department of Defense |
| DOP | Dilution of Precision |
| DR | Dead Reckoning |
| EGI | Embedded Global Positioning System/Inertial Navigation System |
| EKF | Extended Kalman Filter |
| GPS | Global Positioning System |
| GUI | Graphical User Interface |
| Hz | hertz |
| IMU | Inertial Measurement Unit |
| INS | Inertial Navigation System |
| IP | Internet Protocol |
| IR | Infrared |
| LM | Levenberg-Marquardt |
| m | meters |
| m/s | meters per second |
| mm/s | millimeters per second |

| | |
|---|---|
| MAP | Maximum A Posteriori |
| MCL | Monte Carlo Localization |
| MEMS | Micro-Machined Electromechanical System |
| MLE | Maximum Likelihood Estimation |
| PID | proportional-integral-derivative controller |
| PPS | Precise Positioning Service |
| ROS | Robot Operating System |
| RPMG | Relative Position Measurement Graph |
| RTK | Real Time Kinematic |
| sec | seconds |
| SDK | Software Development Kit |
| SLAM | Simultaneous Localization and Mapping |
| SPS | Standard Positioning Service |
| TERCOM | Terrain Contour Mapping |
| UAV | Unmanned Aerial Vehicle |

# Acknowledgments

I would like to thank Co-Advisor Tim Chung for allowing me to be a part of and to contribute to the Advanced Robotic Systems Engineering Lab (ARSENL). His guidance and vision allowed me to think big while also staying grounded enough to prioritize my research and decompose my work into manageable parts. He reminded me time and again to slow down and ensure the basics were working before trying to tackle the more advanced topics. He also assured me that I would overcome the challenges when they appeared overwhelming. I would also like to thank my other co-advisor, Duane Davis, for taking the time to help me through the process. His patience while I stumbled through rookie mistakes, and sometimes taking the longer road to understanding during his robotics course was deeply appreciated. I also want to thank Mike Clement for accommodating my many requests for his technical knowledge and support. Even when busy with his own projects, he would gladly dedicate time to answer my questions or help debug my work in the Robot Operating System.

Many thanks also to my fellow classmates in ARSENL: LT Brenton Campbell, LT Nicole Ramos, and LT Blake Wanier. The hours spent getting the AR Drones to fly were frustrating at times but ultimately rewarding when our hard work paid off.

Most of all, I'd like to acknowledge the patience, devotion and unwavering commitment of my wife, Nikki. While I pursued a degree on the other side of the continent, she endured the life of a single parent while continuing to develop and excel at her own profession.

THIS PAGE INTENTIONALLY LEFT BLANK

# CHAPTER 1:
# Introduction

## 1.1 Motivation

### 1.1.1 The Expansion of Unmanned Aerial Vehicle Mission Sets

Unmanned Aerial Vehicles (UAVs) have become an essential tool for the execution of military plans and securing our national defense. This fact is evident by the 40 fold increase in the Department of Defense (DOD) unmanned aircraft inventory from 2002 to 2010 [1]. Similar to the expansion of military manned aircraft mission types from World War I to World War II, UAV mission types are also expanding. With thousands of successfully completed combat- and noncombat-related missions supporting intelligence, command and control, targeting, and limited weapons delivery, these systems have gained the confidence of military and government leaders. Development of UAVs has been accelerated by new technologies in miniaturization, the relatively low cost of manufacturing, and the development and distribution of open-source control software. Advances in navigation and communications technologies have also played a role in allowing operations of UAVs to be more practical by significantly increasing their operational range [1]. This increased confidence in UAVs along with recent matured technologies has fostered efforts aimed at increasing their capabilities and expanding their current mission set.

Unmanned systems are not limited to the United States. Other international governments are also actively developing and deploying these systems, including potential adversaries such as Iran and China. Doctrinal discussions in light of rapid innovation and development have focused on the feasibility and utility of "swarm tactics," where small distributed units converge rapidly on particular targets [2]. Specifically, research has been conducted to combat Harpy UAV swarm attacks, where an overwhelming number of UAVs attack a high value target by saturating air defense systems [3]. One way to counter such swarm attacks is to intercept the enemy swarm with our own swarm of UAVs.

### 1.1.2 Swarming

Swarming is a collective behavior or motion exhibited by a large number of self-propelled entities [4]. For example, an ant colony as a whole appears to be highly organized, yet each ant seems to have its own agenda [5]. Benefits of this collective behavior include completion of complex tasks, redundancy against the failure of a single entity, a decrease in time to complete certain tasks, and an increase in efficiency through cooperation. Scientists have found that coordination through social species arises through interactions among individuals and not from a hierarchical level. These simple interactions together can solve difficult problems such as finding the shortest route to a food source. This type of behavior is often a basic principle behind self-organizing systems and emergence where "organization" is related to an increase in the structure or order of behavior, and emergence is where properties, behavior, or patterns with the system as a whole dynamically arise from the novel interactions between individual parts of the system [6]. Over the years, there has been significant interest in this natural behavior, which has been dubbed "swarm intelligence" [5]. The works of [7], [8] and [9], just to name a few, have dedicated their efforts to understanding and modeling this behavior so that it can be applied to many other diverse tasks and complex problems. For example, the foraging of ants has led to a novel method for rerouting network traffic in busy telecommunications systems, and the division of labor among honeybees could help streamline assembly lines in factories [5].

The term "swarming" is also used by military historians to describe a battlefield tactic that involves decentralized, pulsed attacks. As stated by Parunak, the link between the two uses of the word is not coincidental. "Insect self-organization is robust, adaptive, and persistent, as anyone can attest who has tried to keep ants out of the kitchen or defeat a termite infestation, and military commanders would love to be able to inflict the frustration, discomfort, and demoralization that a swarm of bees can visit on their victims." [10]. As discussed in [11], the swarming tactic is a structured and coordinated way to attack an enemy from all directions by a sustainable pulse of force from stand-off and close-in positions. Swarming depends on a centralized strategy that is designed around a number of small, dispersed, and networked maneuver units that are allowed to operate in a decentralized manner. Swarming depends heavily on the operation of integrated information systems that can distribute specific targeting information as well as other information to update battlespace awareness. Examples of swarming are found throughout history, but it is now able to emerge as a doc-

trine for UAVs because advances in information technology have given us the ability to provide the required interconnection between the units [11].

For actions against the enemy in an objective area, the swarming tactic can be conceptually broken into four stages: locate the target, converge on it, attack, and disperse. Agents within the swarm must be capable of sustaining multiple waves of attack, quickly merging with other agents on a target, then redispersing and recombining for a new wave [2]. During the converging stage, agents within a swarm must come together to form a whole. However, some form of control is necessary to ensure agents do not collide while continuing to steer toward the target. The most basic form of control has been modeled and is based on the behaviors of flocking birds. First simulated by Craig Reynolds, the Boids model contains the following rules for each agent within the flock: avoid crowding neighbors (separation), steer toward the average position of neighbors (long-range cohesion), and steer toward the average heading of neighbors (alignment goal vector) [7]. Although effective in the simplest of terms regarding the convergence of agents within a swarm, more structure may be required in a formation control scheme to be effective in the interception of another swarm.

## 1.2 Problem Statement

A Congressional Budget Office study in 2011 compared the costs of DOD plans and the capabilities those plans might provide for reconnaissance and light attack missions with the costs and capabilities of some alternate options [12]. Not surprisingly, services within the DOD showed that systems with higher capability come at an increased cost. Although UAVs are usually less expensive than manned aircraft, many of the advanced systems and sensors carried by some UAVs to perform with increased capabilities are very expensive and cannot be viewed as expendable. Also, high levels of reliability to ensure safety of people on the ground and in other aircraft are required. This means additional and often expensive systems such as sense-and-avoid cameras, transponders, and other improvements in airworthiness and reliability are added to UAVs. Finally, support systems, such as ground stations, add costs that are not associated with manned aircraft [12, p. 31]. Therefore, successfully applying swarm tactics to UAVs involves development of as many expendable systems as possible to be cost effective. In the design and development of such systems, the

requirement to "do more with little" by creating small, simple, lightweight, and low-cost UAVs is a primary concern and major focus of effort.

### 1.2.1 Localization in Swarm UAVs

The problem of localization, also referred to as position estimation, is central in mobile robotics [13], and the need for accurate positional information is crucial to not only reduce the probability of collision between agents in a swarm but also to conduct its mission. No matter how many agents make up the swarm, the primary factor that determines the uncertainty of the position estimates of an agent is the accuracy of its proprioceptive sensors and orientation estimates [14]. To increase the positional accuracy of a UAV, incorporating more precise sensors such as a Differential Global Positioning System (DGPS) or an Embedded Global Positioning System/Inertial Navigation System (EGI) can be used. These systems as well as other techniques to improve single UAV navigation and reduce their uncertainty are further discussed in Sections 2.1 and 2.2. Because DGPS and EGI sensors are typically costly and heavy, another solution may be more applicable to swarm UAVs. Mahboubi et al. demonstrate the use of a camera-based system for measuring the relative position of one aircraft to another in [15]. Because high quality camera systems have become lighter and less expensive, they are ideal for UAVs over such systems as DGPS, which require both receivers to be moving over the ground, are very heavy, and typically cost much more than other systems [15]. Using a camera-based sensor to determine the relative position of another aircraft in the swarm as well as applying some form of cooperative localization may attain the same amount of positional precision as a DGPS or EGI at a more reasonable cost.

### 1.2.2 Cooperative Localization

Cooperative Localization (CL) is a term used to describe the technique where agents within a team estimate the position of one another. The relative position estimates acquired by an agent can be used to compensate for errors in odometry or other pose and positioning sensors by combining measurements [16]. By properly combining a number of independent measurements originating from the different members of the team, a single estimate of increased accuracy and reduced uncertainty results. The better estimate of the position and orientation of a landmark can significantly improve the outcome of the localization process, and thus, the entire group of robots can benefit from the collaboration [17]. There are

numerous algorithms and techniques that have been developed to conduct multi-agent CL to include use of probabilistic and statistical estimation. These approaches are discussed further in Section 2.3.1.

Applying CL to swarm UAVs would allow an agent with no or degraded Global Positioning System (GPS) solution to determine positional information necessary to carry out missions within the objective area as well as improve the location estimates of all agents. Based on preliminary research in this field, there is no evidence of CL being implemented in swarm UAVs to improve positional accuracy and no implementation or operationally realistic demonstration in aerial domains. The demonstration of CL by [13] and [18] in ground robots provides a firm base from which to move forward; however, other approaches may be more applicable for implementation in swarm UAVs. Since the goal of swarm aircraft is to be smaller, less expensive, and easier to procure than their larger counterparts, agents will nominally have limited computational power, communications bandwidth, and payload space. Therefore, techniques used to improve positional accuracy must account for these limitations. Use of Kalman filters is common in air vehicle applications; however, direct implementation of the Kalman filtering algorithm is not efficient [19]. Many matrix multiplications and inversions involved in the algorithm make it computationally complex [19, p. 61]. It is important to note that computational resources are continuing to get smaller and more powerful, allowing realization of more sophisticated approaches for conducting CL.

### 1.2.3 Research Questions

Research questions explored in this thesis include:

1. Can a CL scheme be implemented on a swarm of UAVs to improve not only the positional accuracy of a single agent but also the localization of the entire swarm?
2. Specifically, can modifications to the approach demonstrated in [20], which uses an Extended Kalman Filter (EKF), be made to conduct CL on a swarm of UAVs?
3. What are the challenges and considerations in successfully implementing CL in the domain of UAVs?

## 1.3 Benefits of the Study

This thesis demonstrates a means to improve overall localization of any land, sea, or undersea-based robots working on higher-level tasks that involve team coordination with one another. This improvement can be made with robots using noisy or degraded proprioceptive and exteroceptive sensors. This thesis also serves as another step in the overall assessment of core swarm capabilities and limitations as they evolve and become operationally relevant to DOD missions. Extensions of this work include the exploration of capabilities of distributed systems and advances in on-board autonomous capabilities for distributed agents.

## 1.4 Thesis Organization

The next chapter reviews existing single UAV navigation techniques while also highlighting some other localization techniques common in the field of robotics. Chapter 2 also introduces the concept of CL to include centralized and distributed approaches while also highlighting some of the previous work done in CL. In Chapter 3, an implementation of CL using an EKF in a two-dimensional (2-D) space of motion is presented, and further modified to allow for CL in UAVs by adding considerations for altitude (3-D). Chapter 4 outlines the method for demonstrating the modified CL algorithm by utilizing the Robot Operating System (ROS) and the Parrot AR Drone aerial robot. Results and analysis of the experiment are shown and discussed in Chapter 5. Finally, conclusions, recommendations, and future work are highlighted in Chapter 6.

# CHAPTER 2:
# Single and Cooperative Localization

## 2.1 Uncertainty in Single UAV Navigation Techniques

There are five common sources of error that lead to uncertainty in localization. They include environmental dynamics, random action effects, inaccurate models, sensor limitations, and approximate computation. This section is focused on highlighting these uncertainties and the trade-offs made among commonly used localization methods in UAVs. The goal is also to summarize many of the techniques available to localize a UAV while emphasizing the advantages and disadvantages of each. While researching each of the techniques, it quickly becomes apparent that no single localization method is applied in real-world autonomous systems that require a navigational solution. Rather, many methods must be combined or integrated to produce a robust and more accurate measurement. The cost of sensors, computational complexity, amount of storage, size and weight constraints, power consumption, dependency on external resources, knowledge of the operating environment, and desired accuracy are all trade-offs when considering the right localization method for the type of system being considered. It is also important to note that different localization techniques are used based on the type of problem they are trying to solve. Tracking or local techniques aim at compensating odometric errors that occur during navigation. They require an approximate location of the robot initially, and they typically cannot recover if they lose track of the system's position. There are also localization techniques that focus on global approaches. These methods can localize a system without prior knowledge about position and can recover when localization has been interrupted or positioning errors have been encountered. Since the focus of this thesis is on swarm UAVs, the objective is to use the methods that will minimize all the error from sources listed above while maximizing the accuracy for each agent within the swarm.

### 2.1.1 GPS and DGPS

The most common navigational aid is GPS. The geographical latitude and longitude as well as the elevation above sea level is calculated by determining the distances to at least three GPS satellites. Satellite position is known at all time by various observational and

orbital computational methods. When one distance is known, the user must be located on the surface of a sphere with the satellite at the center and with a radius equal to this distance. With two distances, the location must be on a circle that represents the intersection between the two spheres. With three distances known, two points are possible of which one will be far out in space and can be eliminated. Thus, the point on the surface has been determined. Over the years, GPS receivers have become cheaper, smaller, and can be found in everything from commercial aircraft to cell phones. However, most GPS receivers operate using the Standard Positioning Service (SPS) where it is only possible to reach a 15-meter accuracy in the best case [21]. Military users who have specially equipped receivers and the required cryptographic equipment and keys can use the Precise Positioning Service (PPS), where one meter accuracy or better can be achieved [21]. GPS, however, is prone to errors, and at times is unreliable due to a combination of noise, bias, and blunders. These errors mean the predictable accuracy of SPS is 100 meters (m) horizontal and 156 m vertical while PPS users observe a predictable accuracy of 22 m horizontal and 27.7 m vertical [22]. Sources of error in GPS include: measurements in signal arrival time, ephemeris and clock data, numerical calculations, atmospheric effects (i.e., temperature, pressure, humidity, and ionization), multi-path signals (range measurement), and natural interference. Control segment mistakes and receiver errors from software or hardware failures also lead to errors. The magnitude of errors depend on the geometric Dilution of Precision (DOP). A high DOP results when visible satellites are close, causing a small angular separation. Low DOP results when visible satellites are farther away, causing a higher angular separation and better positional accuracy [22]. Furthermore, since GPS signals at the receivers tend to be rather weak, signals can easily be blocked by buildings and other high obstacles, and are vulnerable to jamming. If signals are not blocked, these conditions can still produce local errors where signal reflection from structures can produce multipath errors and interference can be introduced into the signal from other stronger local radio signals.

Differential positioning aims to correct GPS bias errors at one location with measured bias errors at a known position. DGPS determines a localized pseudorange correction factor by computing the difference between positions indicated by the GPS satellites and known fixed positions. Positional accuracy can be further improved by utilizing the carrier phase from the L1 and L2 channels of the satellite signal and using carrier-phase differencing [23]. This approach requires the remote receiver to use dual channels to compare the reference signal

from the base station and the GPS satellite to determine the signal phase difference between the two. After they are compared, a range is determined by adding the phase difference to the total number of waves that occur between each satellite and the receiving antenna. The digital correction is broadcast locally over short range ground based transmitters. The most common DGPS is run by the United States Coast Guard and uses longwave radio to transmit the correction signals near major waterways and harbors. In [23], local differential services were reported to provide one to three foot accuracy while dual channel receivers can provide accuracy between three and twelve inches. Although DGPS has the potential to provide the accuracies needed to sustain close formation flight for swarm UAVs, flight of the aircraft would be limited to areas where ground stations are located. Also, typical DGPS systems equipped with dual-channel receivers presently range in price from $2,500 to $7,500 [23].

Real Time Kinematic (RTK) is another augmentation to GPS that uses a ground base station to correct for positional errors generated by the system. RTK uses measures of the phase in the satellite signal's carrier wave rather than the information content of the signal and relies on a single reference station to provide real-time corrections. RTK systems are the most precise of all GPS systems because they can provide accuracy within one to four centimeters and can achieve complete repeatability [23]. However, similar to DGPS, RTK requires a fixed ground station. Also, the cost of a full RTK system to include base station, rover, data logger, and software is often more than $40,000 [23].

### 2.1.2 Inertial Navigation

An Inertial Navigation System (INS) uses measurements provided by accelerometers and gyroscopes to track the position and orientation of an object relative to an initial starting point, orientation and velocity. Inertial Measurement Units (IMUs) use orthogonal rate-gyroscopes and accelerometers to measure angular velocity and linear acceleration in three axes. The resulting measurements can then be processed to track the position and orientation of a device using Dead Reckoning (DR). Unlike GPS, an INS is self-contained. It requires no external infrastructure, is available anywhere (e.g., inside buildings, close to large obstacles, and underwater), and is robust to jamming [24]. The gyroscopes and accelerometers in an INS can be extremely precise in measuring velocity and acceleration, and in [25] and [24], the most common gyroscope and accelerometer types are compared.

Even though measurements are precise in an INS, they are still prone to errors that originate from white noise, temperature, calibration errors, flicker noise, and bias. Error introduced in the computed orientation can cause the body-frame acceleration signals obtained from the INS to be projected incorrectly onto the global axis. This results in accelerations of the device being integrated in the wrong direction. Also, acceleration due to gravity can no longer be correctly removed, which results in an accumulation in position error when the signals are subsequently integrated. Since a rotation matrix obtained from an attitude algorithm is used to project the acceleration signals into global coordinates, errors in the angular velocity signals cause drift in the calculated position. Advances in the development of Micro-Machined Electromechanical System (MEMS) devices have made it possible to manufacture an INS that is small and lightweight. MEMS inertial sensors that include magnetometers to reduce drift in orientation can be used to construct inertial navigation systems, which suffer average drifts of around 5 m after 60 seconds (sec) of stationary operation. In addition, it is not currently possible to construct an INS which maintains sub-meter accuracy for more than 60 seconds using MEMS devices [26]. Drift can be reduced by fusing other sensors such as GPS and magnetometers. Drift can also be reduced by exploiting constraints that are known to apply to the movement of the IMU, such as known points in time at which the device must have a zero velocity. In [24], the INS/GPS cost as a function of inertial instrument technology and performance is shown. It is no surprise that more precision with use of ring laser or fiber-optic gyros comes at a higher cost; however, these systems will never be fully free from errors caused by drift and having a continuous or reliable GPS signal can not be guaranteed.

### 2.1.3 Terrain-Based Navigation

Another self-contained technique for UAVs to derive positional reference information is through terrain-based navigation. This form of localization can be broken into four methods and are described in detail in this section. The first, which has been used in the navigation of cruise missiles, compares terrain elevation models to measured ground elevation profiles. Next, sequences of on-board images are matched to geo-referenced images. The third method locks onto objects in the center of a camera image and tracks them in subsequent images. Image displacement is sensed in subsequent images and UAV motion between images is determined to incrementally update position and velocity. The final method uses optical flow, which is the measurement of image velocity. The sensors used to perform

terrain-based navigation can vary; however, focus in this thesis is on the potential use of video images for the last three methods. Compared to other sensors such as laser or radar, cameras are typically lighter, cheaper, easier to acquire, and consume less power. Color images also contain a large amount of information which can be used for various purposes such as feature or object identification. However, it is also important to note that cameras are sensitive to lighting conditions, affected by glare, and because they are limited to the visual spectrum, cannot visually represent a point of interest if it is obstructed by another object (e.g., a camera cannot see a target if it is obstructed by cloud cover).

Conceptually, the simplest of the terrain-based navigation methods is Terrain Contour Mapping (TERCOM) which is discussed in [27]. TERCOM compares a measured terrain profile to terrain profiles stored in the system computer and determines the geographic location of the measured profile by the best match. The system uses the assumption that a single geographic location on land is uniquely defined by the contours of the surrounding terrain. The measured terrain profile is determined by combining the altitude measurements of a radar and barometric altimeter along with the vertical accelerometer output of an IMU. The reference terrain profile consists of a digital representation of those elevation profiles over which the system is most likely to fly. The computer correlates the measured profile against each profile on the reference map. The reference point that matches best should be the one representing the terrain over which the missile flew. When a position error is noted, TERCOM sends a command to correct for drift error in the inertial system and the missile is given commands to correct its course. TERCOM depends on a reference map to be successful, which means it must know the environment for which it is operating. Also, TERCOM requires terrain that is sufficiently unique at each fix site and performs best where changes in elevation are large. Another technique in this category uses aerial images. In [28], a localization algorithm uses a multiple aerial image sequence to recover elevations from feature points in the images. These elevations are then compared with data from the digital elevation model (DEM) to estimate ground position.

Matching sequences of on-board images to geo-referenced images is essentially a more complex form of TERCOM. Here, instead is using altitude sensors and maps representing elevation, video images are being used. An image registration technique based on edge matching has been demonstrated in [29]. A Sobel edge detector is applied to both

a geo-referenced image and an image taken from the camera aboard a UAV. A matching algorithm then tries to find the position in the cropped reference image which gives the best match with the camera image. The position that results in the greatest number of overlapping pixels between the edges of the two images is taken as a matching result. The absolute position of the UAV can be calculated since the on-board image can be geo-referenced. Similar to TERCOM, matching images to geo-referenced images limits the operating environment and is bound to the stored reference data.

The next category of terrain-based navigational methods is the measurement of image displacement between multiple camera images, known as visual odometry. In [29] and [30], algorithms are described that update UAV position by locking onto a feature or features in a camera image and tracking them in subsequent images. Relying on the feature's ground location, the algorithms sense the feature location in the UAV frame. Along with attitude measurements taken from on-board angular sensors, the feature coordinates in the ground and UAV frames are used by the algorithms to compute the UAV's ground frame position.

Optical flow, as defined in [31], refers to the apparent movement of texture in a visual field resulting from motion. This information can be used to perceive relative depths of visible objects and self-motion such as speed and rotation. An example of optical flow is a bird flying low to the ground. When looking down, the bird would see objects moving faster through its visual field compared to those objects it observes while flying higher. In the forward direction, obstacles can be detected by detecting expansion or divergence of the obstacle. More rapid expansion of objects implies a closer proximity to the object. Barrows explains that the "focus of expansion" (FOE) from which optical flow originates can indicate the direction of heading. If the FOE is located inside a rapidly expanding object, collision with that object is imminent. However, if the FOE is located outside an expanding region, approach toward the object will be close but collision with the object would be avoided. The three stages of optical flow processing are as follows: prefiltering or smoothing to extract interest points and enhance the signal-to-noise ratio, the extraction of local correlation surfaces or space-time derivatives, and the integration of measurements to produce a 2-D flow field. A summary of the constraints and regularization of the estimation of optical flow can be found in [32]. A survey of differential, region-based matching,

energy-based, and phase-based techniques can be found in [33]. A more current evaluation and the latest benchmark for evaluating optical flow algorithms can be found in [34].

In [35], the feasibility of estimating the velocity of a ground robot using optical flow was studied. To estimate vehicle velocity, an optical flow algorithm is used to first estimate a flow field from an image pair. The resulting flow is then multiplied by the camera's frame rate to get an image velocity. The image velocity is then converted to vehicle velocity by multiplying the distance between the feature point and the camera center of projection and dividing by the camera focal length.

### 2.1.4 SLAM

Simultaneous Localization and Mapping (SLAM) is the process in which a UAV can sense and build a map of its environment while also using this map to update its position. An immediate advantage to SLAM is that no *a priori* information is needed about the map, the landmarks within the map, or the vehicle location within the map. This relieves the requirement for reference data or initial knowledge of position associated with other terrain-based navigation techniques listed above and allows the UAV to operate in unfamiliar areas. The SLAM process is discussed in [36], [37], and [38], though a brief review of its basic elements is provided herein. In SLAM, the vehicle starts its navigation using its dead reckoning sensor or vehicle model. As on-board sensors detect and extract features from the environment, an estimator augments the landmark locations to a map in some global reference frame and begins to estimate the vehicle and map states together with successive observations. The estimator, which is typically an EKF, keeps track of the estimate of the uncertainty in both the vehicles position and the landmarks it has seen in the environment. The ability to estimate both the vehicle location and the map is due to the statistical correlations which exist within the EKF between the vehicle and landmarks and between the landmarks themselves. When the vehicle moves, the uncertainty pertaining to the vehicle's new position is updated in the EKF. Landmarks are then extracted from the environment from the vehicle's new position. The vehicle then attempts to associate these landmarks to observations of landmarks it has seen previously. Re-observed landmarks are used to update the vehicle's position in the EKF and the map accuracy converges to a lower limit which is a function of the initial vehicle uncertainty when the first landmark was observed.

New landmarks which were not previously seen are added to the EKF as new observations and the process continues.

As stated in [37], [39] pioneered the application of SLAM in a large outdoor environment. The authors not only addressed computational issues of real-time operation, but also dealt with high-speed vehicle motion, nonflat terrain, and dynamic clutter. Their results were accompanied by RTK GPS ground truth which solidified the practical accuracy of the algorithm, which involved closing several large loops. SLAM applications exist in a wide variety of domains, but the work of [38] is most relevant to this research since the authors were able to formulate and implement a SLAM algorithm for UAVs. At a theoretical and conceptual level, SLAM is often considered a solved problem and the steps in SLAM can be implemented using a number of different algorithms. However, there are issues remaining in computation, convergences, and data association that remain the main focus of the SLAM research community [37].

## 2.2 Probabilistic and Statistical Localization Methods

As seen in SLAM, statistical techniques are used to provide an estimation of the posterior probability for the pose of the robot and for the parameters of the map. This leads to the discussion on a group of techniques that uses probabilistic and statistical approaches to aid in localization. The most common uses recursive Bayesian estimation, where an unknown probability density function is estimated recursively over time using a mathematical process model and incoming measurements of the environment. For simplicity, the following sections focus on the particle filter and the Kalman filter, since they tend to be the most commonly used approaches in autonomous systems.

### 2.2.1 Particle Filters

Particle filter localization, also known as Monte Carlo Localization (MCL), approximates the pose and orientation of a robot by using a Monte Carlo method. A set of samples (also called particles) are maintained where each particle represents a hypothesis of where the robot is located. When the robot moves, it shifts the particles based on its motion model to predict its new state. When the robot senses an object in the environment, each hypothesis is weighted according to its likelihood from the observation. The set is resampled according to their weights and the process continues with a posterior represented

by the entire particle set. As the robot continues to move and sense its environment, the particles converge towards the actual pose of the robot as long as the motion and sensor models are reasonable. Unlike other localization algorithms such as the Kalman filter and Markov localization, MCL does not rely on parametric distributions since the posteriors are represented by a random collection of weighted particles that approximate the desired distribution. As demonstrated by [40], MCL can accommodate arbitrary sensor characteristics, motion dynamics, and noise distributions. It can focus and adapt to computational resources by sampling in proportion to the posterior likelihood and control the number of samples on-line. Finally, MCL is relatively easy to implement, although potentially constrained by computational limitations. Because of the stochastic nature of the approximation there are a few pitfalls with MCL. If the number of samples is small, for instance, a robot might lose track of its position because the algorithm fails to generate a sample in the right location. Also, if the robot has been moved or navigation was interrupted, there may be no surviving samples for the robot to use near its new pose. A more thorough analysis of MCL and particle filtering can be found in [41], [42], and [43].

### 2.2.2 Kalman Filters

The Kalman filter is an algorithm that also operates recursively on measurements observed over time and produces a state estimate by minimizing the mean squared error. A Kalman filter represents belief using mixtures of Gaussians that enable the filter to pursue multiple, distinct hypotheses, each of which is represented by a separate Gaussian [40].

The filter is designed to estimate the state of a discrete-time controlled process that is governed by a linear stochastic difference equation; however, most processes to be estimated and the measurement relationships to the processes are typically nonlinear. To remedy this, a Kalman filter that linearizes about the current mean and covariance is referred to as an Extended Kalman Filter (EKF). Similar to a Taylor series, the estimation around the current estimate can be linearized using the partial derivatives of the process and measurement functions to compute estimates even if given a nonlinear relationship to the state variables. The distributions of the various random variables are no longer Gaussian after undergoing their respective nonlinear transformations so the EKF becomes more of an ad hoc state estimator that only approximates the optimality of Bayes' rule by linearization [44].

The Kalman filter provides efficient update rules that can be shown to be optimal; however, this is under the assumption that the uncertainty in the robot's position can be represented by a unimodal Gaussian distribution [45]. Sensor readings must also be assumed to map to Gaussian shaped distributions over the robot's position. Finally, localization approaches using Kalman filters typically require that the starting position of the robot be known [42].

As shown by [44], one cycle of estimation in a Kalman filter can be divided into a propagation and update step. Before showing the equations in each of these steps, the following equations define the state $\mathbf{x} \in \mathbb{R}^n$ of the discrete-time controlled process governed by a linear stochastic difference equation as follows

$$\mathbf{x}_{k+1} = A_k \mathbf{x}_k + B\mathbf{u}_k + \mathbf{w}_k \tag{2.1}$$

with a measurement $\mathbf{z} \in \mathbb{R}^m$ that is

$$\mathbf{z}_k = H_k \mathbf{x}_k + \mathbf{v}_k \tag{2.2}$$

where the $n \times n$ matrix $A_k$ relates the state at time step $k$ to the state at step $k+1$. The $n \times l$ matrix $B$ relates the control input $\mathbf{u} \in \mathbb{R}^l$ to the state $\mathbf{x}$, and the $m \times m$ matrix $H$ in the measurement equation relates the state to the measurement $\mathbf{z}_k$. The random variables $\mathbf{w}_k$ and $\mathbf{v}_k$ represent the process and measurement noise respectively, which are assumed to be independent of each other, white, and are drawn from normal probability distributions:

$$p(w) \sim N(0, Q) \tag{2.3}$$

$$p(v) \sim N(0, R). \tag{2.4}$$

In the propagation cycle, the state of the system is propagated to the next time step based on the assumptions about the evolution of the system equations, the measured control inputs, and the statistical description of the system noise. This relationship translates to the equations as follows:

$$\mathbf{x}_k^+ = A_k \mathbf{x}_k + B_k \mathbf{u}_k \tag{2.5}$$

$$P_k^+ = A_k P_k A_k^T + Q_k \tag{2.6}$$

where $\mathbf{x}_k^+$ and $P_k^+$ are the predicted mean and covariance of the state on the time step $k$ before seeing the measurement. In the update cycle, the measurement from the sensor is processed to update the propagated estimate calculated during the previous cycle. This translates to the equations as follows:

$$S_k = H_k P_k^+ H_k^T + R_k \tag{2.7}$$

$$K_k = P_k^+ H_k^T S_k^{-1} \tag{2.8}$$

$$\hat{\mathbf{x}}_k = \mathbf{x}_k^+ + K_k(\mathbf{z}_k - H_k \mathbf{x}_k^+) \tag{2.9}$$

$$\hat{P}_k = P_k^+ - K_k S_k K_k^T \tag{2.10}$$

and where the *a posteriori* state estimate $\hat{\mathbf{x}}_k$ is computed as a linear combination of an *a priori* estimate $\mathbf{x}_k^+$ and a weighted difference between an actual measurement $\mathbf{z}_k$ and a measurement prediction $H_k \mathbf{x}_k^+$. The difference, $\mathbf{z}_k - H_k \mathbf{x}_k^+$, is called the measurement innovation or the residual on time step $k$, which reflects the discrepancy between the predicted and actual measurement. The $n \times n$ matrix $S_k$ in Equation 2.7 is known as the measurement prediction covariance and the $n \times m$ matrix $K_k$ in Equation 2.8 is the Kalman gain or blending factor that minimizes the *a posteriori* error covariance. The justification for Equation 2.9 is rooted in the probability of the *a priori* estimate $\mathbf{x}_k^+$ conditioned on all prior measurements $\mathbf{z}_k$ (Bayes' rule). The *a posteriori* estimate error covariance, given by Equation 2.10, reflects the variance of the state distribution. In other words,

$$p(\mathbf{x}_k|\mathbf{z}_k) \sim N(\mathbb{E}[\mathbf{x}_k], \mathbb{E}[(\mathbf{x}_k - \hat{\mathbf{x}}_k), (\mathbf{x}_k - \hat{\mathbf{x}}_k)^T])$$

$$= N(\hat{\mathbf{x}}_k, P_k).$$

After the update step, the process is repeated where the *a posteriori* estimate is used to project or predict the next *a priori* estimate.

## 2.3 Overview of Cooperative Localization

There have been a number of studies conducted on the accuracy of position estimation for groups of mobile robots using some form of CL. Results of [13] and [18] established the analytical assessment of the positional accuracy of multi-robot teams by deriving the upper bounds on the positional uncertainty of a group of homogeneous robots. However,

the assumption of homogeneity and the requirement that every robot continuously measures the relative position of all other robots in the team is not applicable in a realistic scenario. Limitations in computational power and communications bandwidth along with inherent variability in sensors prohibit continuous transmission and classify the robot team as more heterogeneous. In [14], analytical expressions for assessing the positional accuracy of heterogeneous robot groups were determined as well as a study on the time evolution of the positional uncertainty with arbitrary topology of a Relative Position Measurement Graph (RPMG). Using CL, the exchange of positioning information benefits all robots since demonstrations have shown a smaller rate of uncertainty than the rate the single best agent would attain it it were localizing on its own [14]. In the same work, the authors were also able to show that the connectivity of the RPMG affected the constant part of the covariance matrix that describes the positional uncertainty of the group. A temporary reduction in the number of relative position measurements can only cause temporary loss of positioning accuracy. Moreover, if one robot has absolute position measurements, the localization uncertainty for all robots in the group is drastically reduced. Finally, they also concluded that no matter how many robots were in a group, the most important factors in determining an increase in position uncertainty was each robot's own proprioceptive sensors and orientation estimates.

Multi-robot systems can be classified as centralized or decentralized and distributed or non-distributed. The definitions of these terms are provided by [46] and [47]. It is important to note that a system's characterization as centralized or decentralized is independent of its characterization as distributed or non-distributed. For example, a distributed system may have each agent perform a part of a computation while requiring a centralized processor to combine the computations to determine the final result. In a decentralized architecture, redundancy exists by allowing all other agents to proceed with their computations even if an agent fails. A distributed system allows for improved efficiency by making use of the computational resources available from multiple agents. When a system is both decentralized and distributed, the computation for a task does not depend on a specific agent and can be divided among a number of agents.

One of the major challenges to CL is the computational complexity and the communications required between agents. Research in the field started with demonstrations of simple

concepts and as processes evolved, work in CL began to address the more complex issues. Early work in CL was done by [48] and [49]. Kurazume et al. were the first to introduce the idea of using robots as features in the environment. He as well as Rekleitis in [49] started by having one or more robots in a group move while others remained stationary. The static robots served as landmarks or acted as beacons as the moving robots tried to localize. The robots then switched roles which created a "leap-frogging" motion as the group moved towards its goal. The obvious drawback for this approach was the constraint on the motion of the robots. Also, computational and communication complexity was not addressed. Early variations of CL were essentially centralized systems. As computer processors began to get smaller, the ability to put multiple computers on multiple robots allowed researchers to begin examining how some of the computation for pose estimation could be distributed. Sanderson [50], Madhavan [51], and Fox [43] each had success in distributing estimate calculations but they were sub-optimal with respect to a centralized estimator. In [17] and [20], Roumeliotis and Bekey decomposed the EKF into a number of filters that performed the prediction step locally on each robot. They then showed how the propagation of the covariance matrix could be factored using singular value decomposition such that the factored terms could be computed by each robot using its own odometry data. For the measurement update, all robots in the group needed to communicate with each other so that the factored terms could be combined. Their approach was one of the most notable advances in CL because their results were comparable to the performance of a centralized estimator.

### 2.3.1 Probabilistic and Statistical Methods

The most common estimators in CL use a Kalman filter; however, there are other approaches that use other probabilistic and statistical methods such as particle filters and Maximum Likelihood Estimations (MLEs). In [43], Fox et al. use a particle filter where each robot maintains a probability distribution describing it own pose based on odometry and environmental sensing. This distribution is refined through the observation of the other robots in the group.

In [52], Howard uses a combination of MLE and numerical optimization. A set of pose estimates as well as a set of observations are collected for a particular robot at a particular time. There are two sets of observations where one is made by a motion detector and the other is made by a robot detector. Using numerical optimization, the set of pose estimates

that is likely to give rise to the combined set of observations is determined. The set of pose estimates are not used directly since the estimates are defined with respect to an arbitrary coordinate system whose relationship with the external environment is not defined. However, the robot uses the estimates to compute the pose of every other robot relative to itself, and uses this information to coordinate activity.

A distributed Maximum A Posteriori (MAP) estimator for CL is described in [53]. It is formulated as a nonlinear least-squares problem and solved iteratively using the Levenberg-Marquardt (LM) minimization algorithm. The LM algorithm is accomplished in parallel by all robots using a distributed conjugate gradient algorithm that provides an intermediate solution at every iteration. This, coupled with distributed marginalization of past robot poses, allows the robots to trade processing for accuracy when resources are scarce. Simulation results showed better performance in terms of accuracy while having lower computational requirements as compared to the EKF approaches, but a limitation of the distributed conjugate gradient algorithm is the synchronous communication that is required between the robots.

### 2.3.2 Cooperative Localization in Wireless Networks

Although most methods to CL originate in robotics, other disciplines also rely on precise localization. For example, CL methods are also used in wireless networks where automatic localization of sensors is important for their data to be meaningful and where location information can be useful for routing algorithms. In [47], [54], and [55], measurement-based statistical models such as angle-of-arrival, time-of-arrival, wideband and ultra-wideband measurements, and received-signal-strength are discussed as well as fundamental Bayesian and non-Bayesian CL algorithms. Methods to estimate location estimation precision, such as the Cramer-Rao bound, are also discussed to help select measurement technologies and evaluate localization algorithms.

### 2.3.3 Communications and Network Challenges

In a cooperative multi-agent system, reliable and effective communication is critical for enhancing the group's performance. A common assumption in research on multi-agent systems is that a fully connected network is continuously available and that it is always possible for information from one agent to reach another. This assumption, though, is

not always realistic in all environments. Occlusions can obstruct and limit communication range and limited bandwidth may not be able to support large exchanges of data between all agents at once. Sequencing of information can be a challenge, which leads to the out-of-sequence measurement or negative timestep problem. Missed measurements are also possible in a dynamic network. In those applications where more measurements are needed for better estimates, a missed measurement could cause sub-optimal performance. Another issue when performing CL in a dynamic network is the cyclic update problem where agents repeatedly use the same measurement. This can cause inconsistent or over-confident estimates. The focus of this thesis is the demonstration of CL in UAVs. Although overcoming these communications and network challenges offer many opportunities for future research, they are outside the scope this work. Ways to address these issues are further discussed in [46]. In his work, Leung not only summarizes remedies proposed by other researchers, but also introduces his centralized-equivalent approach. His work avoids both the out-of-sequence measurement and cyclic update problems in a dynamic network and allows robots to recover an estimate equivalent to that produced by a centralized estimator whenever possible.

THIS PAGE INTENTIONALLY LEFT BLANK

# CHAPTER 3:
# Algorithm Development

Roumeliotis and Bekey showed in [20] how to treat the Kalman filter equations of a centralized system so as to distribute the pose estimation process among $M$ Kalman filters, each of them operating on an agent within the group. They initially formulated the group localization problem in a centralized way and then described how it could be distributed among the agents in the group. Although the objective of this thesis is to demonstrate their distributed approach for 3-D motion where computations are being done by the agents, this chapter details an implementation of the distributed approach using a centralized node. Their equations and processes, detailed in [20], are applied to ground robots in 2-D using a unicycle motion model. The contribution of this thesis extends their work in generalizing equations where they presented special cases. This allows application of the equations to $M$ agents and where any agent $i$ interacts with any agent $j$. Another contribution includes incorporation of a simple aircraft motion model with lateral velocity control vice the unicycle model shown by Roumeliotis and Bekey. Finally, the modifications necessary to apply their CL algorithm in 3-D are derived and presented in this chapter. Before examining the 2-D implementation, the following general assumptions are made for CL using an EKF:

1. A group of $M$ independent agents move in an $N$-dimensional space. The motion of each agent is described by its own linear or nonlinear equations of motion;
2. Each agent carries proprioceptive and exteroceptive sensors to propagate and update its own position estimate;
3. Each agent also carries exteroceptive sensors that allow it to detect and identify other agents moving in its vicinity and measure their respective displacement (relative position and orientation);
4. All agents are equipped with communication devices that allow exchange of information within the group.

## 3.1 Two-Dimensional Approach

Before discussing the 3-D implementation relevant to aerial applications, this section starts by generalizing the 2-D ($N = 3$) equations presented in [20] where necessary. Extending

previous models, lateral velocity control to the motion model of each agent is also introduced. The propagation equations for the Kalman filter using the velocity measurements from the odometric or inertial sensors are described where the choice of notation is consistent with [20]. The state vector for each agent consists of the agent's pose with respect to a fixed reference frame $\mathbf{x}_i = [x_i \;\; y_i \;\; \theta_i]^T$ where $i$ = 1,..,$M$, $x_i$ and $y_i$ are the position of agent $i$ in Cartesian coordinates, and $\theta_i$ is the agent's heading. A diagram visualizing this convention as well as showing the respective linear and angular velocities of agent $i$ are shown in Figure 3.1. Note that lateral velocity ($V_i^y$) is used in the system dynamics model since quadrotors are being utilized. Therefore, the continuous time equations for the motion expressed in local coordinates are

$$L_{\dot{x}_i} = V_{im}^x, \quad L_{\dot{y}_i} = V_{im}^y, \quad \dot{\theta}_i = \omega_{im}, \quad i = 1,...,M \tag{3.1}$$

with

$$V_{im}^x = V_i^x + w_{V_i^x}, \quad V_{im}^y = V_i^y + w_{V_i^y}, \quad \omega_{im} = \omega_i + w_{\omega_i}, \quad i = 1,...,M \tag{3.2}$$

where $V_{im}^x$, $V_{im}^y$ and $\omega_{im}$ are the forward linear, lateral linear, and rotational velocity of agent $i$ as measured by the on-board sensors, $V_i^x$, $V_i^y$ and $\omega_i$ are the real values of these quantities, and $w_{V_i^x}$, $w_{V_i^y}$ and $w_{\omega_i}$ are zero-mean white Gaussian noise in the measured signals.

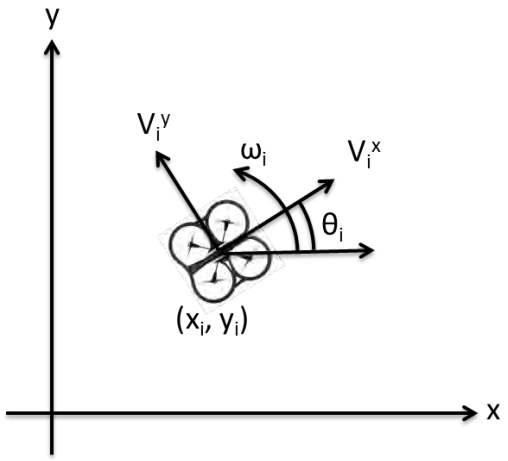



Figure 3.1: Visualization of fixed and relative reference frames for state representation of agent $i$ within a group of UAVs

### 3.1.1 Initial Propagation and Update

Based on Equation 3.1, the nonlinear system dynamics for propagating the state of agent $i$ expressed in global coordinates is

$$\begin{bmatrix} x_i \\ y_i \\ \theta_i \end{bmatrix}_{t_{k+1}} = \begin{bmatrix} x_i \\ y_i \\ \theta_i \end{bmatrix}_{t_k} + \begin{bmatrix} \delta t \cos\theta_i & -\delta t \sin\theta_i & 0 \\ \delta t \sin\theta_i & \delta t \cos\theta_i & 0 \\ 0 & 0 & \delta t \end{bmatrix} \begin{bmatrix} V_{im}^x \\ V_{im}^y \\ \omega_{im} \end{bmatrix}_{t_k} \tag{3.3}$$

or

$$\mathbf{x}_i^+ = \mathbf{x}_i(t_k) + B_i(t_k)\mathbf{u}_i(t_k) \tag{3.4}$$

for $i = 1,...,M$ where $\delta t = t_{k+1} - t_k$ is the discrete incremental time step, $\mathbf{x}_i$ is the state vector, $B_i \in \mathbb{R}^{3\times 3}$ is the effect matrix, and $\mathbf{u}_i$ is the input vector which includes noise in the measured linear and angular velocity. In the propagation step of the EKF, the nonlinear system as described in Equation 3.4 is used to compute the state propagation of each agent. To compute the propagation of the covariance, the system dynamics expressed above must be linearized. Equation 3.3 is re-written as

$$\begin{bmatrix} x_i \\ y_i \\ \theta_i \end{bmatrix}_{t_{k+1}} = \begin{bmatrix} x_i \\ y_i \\ \theta_i \end{bmatrix}_{t_k} + \begin{bmatrix} V_{im}^x \cos\theta_i - V_{im}^y \sin\theta_i \\ V_{im}^x \sin\theta_i + V_{im}^y \cos\theta_i \\ \omega_i \end{bmatrix} \delta t = \begin{bmatrix} f_1(x_i, y_i, \theta_i) \\ f_2(x_i, y_i, \theta_i) \\ f_3(x_i, y_i, \theta_i) \end{bmatrix} \tag{3.5}$$

and the Jacobian is computed, which results in

$$\begin{aligned} A_i &\triangleq \begin{bmatrix} \frac{\partial f_1}{\partial x_i} & \frac{\partial f_1}{\partial y_i} & \frac{\partial f_1}{\partial \theta_i} \\ \frac{\partial f_2}{\partial x_i} & \frac{\partial f_2}{\partial y_i} & \frac{\partial f_2}{\partial \theta_i} \\ \frac{\partial f_3}{\partial x_i} & \frac{\partial f_3}{\partial y_i} & \frac{\partial f_3}{\partial \theta_i} \end{bmatrix} \\ &= \begin{bmatrix} 1 & 0 & \left(-V_{im}^x \sin\theta_i - V_{im}^y \cos\theta_i\right)\delta t \\ 0 & 1 & \left(V_{im}^x \cos\theta_i - V_{im}^y \sin\theta_i\right)\delta t \\ 0 & 0 & 1 \end{bmatrix} \end{aligned} \tag{3.6}$$

where $A_i \in \mathbb{R}^{3\times 3}$ is the linearized system propagation matrix. The linearized discrete-time state propagation equation is now

$$\begin{aligned}\begin{bmatrix} x_i \\ y_i \\ \theta_i \end{bmatrix}_{t_{k+1}} &= \begin{bmatrix} 1 & 0 & \left(-V_{im}^x \sin\theta_i - V_{im}^y \cos\theta_i\right)\delta t \\ 0 & 1 & \left(V_{im}^x \cos\theta_i - V_{im}^y \sin\theta_i\right)\delta t \\ 0 & 0 & 1 \end{bmatrix} \begin{bmatrix} x_i \\ y_i \\ \theta_i \end{bmatrix}_{t_k} \\ &+ \begin{bmatrix} \delta t\cos\theta_i & -\delta t\sin\theta_i & 0 \\ \delta t\sin\theta_i & \delta t\cos\theta_i & 0 \\ 0 & 0 & \delta t \end{bmatrix} \begin{bmatrix} w_{V_{im}^x} \\ w_{V_{im}^y} \\ w_{\omega_i} \end{bmatrix}_{t_k}\end{aligned} \tag{3.7}$$

or

$$\mathbf{x}_i(t_{k+1}) = A_i(t_{k+1},t_k)\mathbf{x}_i(t_k) + G_i(t_k)\mathbf{w}_i(t_k) \tag{3.8}$$

where $G_i \in \mathbb{R}^{3\times 3}$ is the system noise input matrix, and $\mathbf{w}_i$ is a vector representing the system noise due to the errors in the linear and rotational velocity measurements of agent $i$. The system noise covariance $Q_i \in \mathbb{R}^{3\times 3}$ is given by :

$$Q_i(t_k) = G_i(t_k)\mathbb{E}\{\mathbf{w}_i(t_k)\mathbf{w}_i^T(t_k)\}G_i^T(t_k). \tag{3.9}$$

The covariance propagation for each agent is therefore,

$$P_{ii}^+ = A_i(t_{k+1},t_k)P_{ii}(t_k)A_i^T(t_{k+1},t_k) + Q_i(t_k). \tag{3.10}$$

This covariance describes the uncertainty associated with the position of agent $i$. Under the assumption that the motion of each agent does not directly affect the motion of the others, the covariance propagation equation for the centralized system is

$$P^+ = \Phi(t_{k+1},t_k)P(t_k)\Phi^T(t_{k+1},t_k) + Q(t_k) \tag{3.11}$$

where the centralized system matrix $\Phi \in \mathbb{R}^{3M\times 3M}$ and system noise covariance matrix $Q \in \mathbb{R}^{3M\times 3M}$ contains the propagation matrix and noise covariance matrix of each agent,

respectively, on the diagonal as shown below

$$\Phi(t_{k+1},t_k) = \begin{bmatrix} A_1(t_{k+1},t_k) & 0_{3\times3} & 0_{3\times3} & \dots & 0_{3\times3} \\ 0_{3\times3} & A_2(t_{k+1},t_k) & 0_{3\times3} & \dots & 0_{3\times3} \\ 0_{3\times3} & 0_{3\times3} & A_3(t_{k+1},t_k) & \dots & 0_{3\times3} \\ \vdots & \vdots & \vdots & \ddots & \vdots \\ 0_{3\times3} & 0_{3\times3} & 0_{3\times3} & \dots & A_M(t_{k+1},t_k) \end{bmatrix}$$

$$Q(t_k) = \begin{bmatrix} Q_1(t_k) & 0_{3\times3} & 0_{3\times3} & \dots & 0_{3\times3} \\ 0_{3\times3} & Q_2(t_k) & 0_{3\times3} & \dots & 0_{3\times3} \\ 0_{3\times3} & 0_{3\times3} & Q_3(t_k) & \dots & 0_{3\times3} \\ \vdots & \vdots & \vdots & \ddots & \vdots \\ 0_{3\times3} & 0_{3\times3} & 0_{3\times3} & \dots & Q_M(t_k) \end{bmatrix}.$$

Initially, each agent may only know its own position in global coordinates and the uncertainty related to it. With no *a priori* shared knowledge among the agents, the covariance matrix $P$ for the centralized system is also diagonal, and each of the diagonal elements is the covariance for the state of each agent. Therefore,

$$P^+ = \begin{bmatrix} P_{11}^+ & 0_{3\times3} & 0_{3\times3} & \dots & 0_{3\times3} \\ 0_{3\times3} & P_{22}^+ & 0_{3\times3} & \dots & 0_{3\times3} \\ 0_{3\times3} & 0_{3\times3} & P_{33}^+ & \dots & 0_{3\times3} \\ \vdots & \vdots & \vdots & \ddots & \vdots \\ 0_{3\times3} & 0_{3\times3} & 0_{3\times3} & \dots & P_{MM}^+ \end{bmatrix}. \tag{3.12}$$

While no relative position information is exchanged between agents in the group, the covariance remains the same until the next propagation step.

Next, the update step considers the measurement between the agents. When an agent can sense and identify another, it measures the relative pose and orientation of the agent with respect to the frame of reference attached to the measuring agent. The measurement is used to update the pose estimate for the centralized system and the covariance of the estimate. In [20], Roumeliotis and Bekey use an example where agent 1 detects agent 2, and then

uses its exteroceptive sensing to measure the relative position and orientation of agent 2 with respect to the frame of reference attached to agent 1. Generalizing their example and representing the measurement when agent $i$ detects agent $j$ is given as:

$$\mathbf{z}_{ij}(t_k) = \begin{bmatrix} C^T(\theta_i)\left(\begin{bmatrix} x_j \\ y_j \end{bmatrix} - \begin{bmatrix} x_i \\ y_i \end{bmatrix}\right) \\ \theta_j - \theta_i \end{bmatrix} + \mathbf{n}_{ij} \tag{3.13}$$

where

$$C(\theta_i) = \begin{bmatrix} \cos(\theta_i) & -\sin(\theta_i) \\ \sin(\theta_i) & \cos(\theta_i) \end{bmatrix} \tag{3.14}$$

is a rotation matrix, and $\mathbf{n}_{ij}$ is a vector representing the measurement noise associated with the relative pose measurement between agents $i$ and $j$. The measurement noise is assumed to be zero-mean white Gaussian with covariance

$$R_{ij}(t_k) = \mathbb{E}\{\mathbf{n}_{ij}(t_k)\mathbf{n}_{ij}^T(t_k)\} \tag{3.15}$$

By linearizing Equation 3.13, the measurement error can be approximated by the following equation:

$$\tilde{\mathbf{z}}_{ij}(t_k) = H_{ij}(t_k)\tilde{\mathbf{x}}(t_k) + \mathbf{n}_{ij}(t_k) \tag{3.16}$$

where $H_{ij}$ is represented as a $1 \times M$ block matrix where each column contains an $N \times N$ sub-matrix. The $i$th column of the $1 \times M$ block matrix contains a $-\tilde{H}_{ij}$ sub-matrix that represents the linearized measurement with respect to agent $i$. The $j$th column of $H_{ij}$ contains the linearized measurement with respect to agent $j$ which happens to be an $N \times N$ identity sub-matrix $I_{N\times N}$. All other columns contain the $N \times N$ zero sub-matrix $0_{N\times N}$. This matrix is then multiplied by the transpose of the $N \times N$ transformation matrix $\Gamma_i$. In the example provided in [20], the $H_{12}$ matrix is therefore

$$H_{12} = \Gamma_1^T[-\tilde{H}_{12} \;\; I_{3\times 3} \;\; 0_{3\times 3}] \tag{3.17}$$

with

$$\Gamma_1 = \begin{bmatrix} \cos(\theta_1) & -\sin(\theta_1) & 0 \\ \sin(\theta_1) & \cos(\theta_1) & 0 \\ 0 & 0 & 1 \end{bmatrix} \tag{3.18}$$

and

$$\tilde{H}_{12} = \begin{bmatrix} 1 & 0 & -(y_2^+ - y_1^+) \\ 0 & 1 & x_2^+ - x_1^+ \\ 0 & 0 & 1 \end{bmatrix}. \tag{3.19}$$

To generalize the above example, $\tilde{H}$ when agent $i$ measures agent $j$ is therefore

$$\tilde{H}_{ij} = \begin{bmatrix} 1 & 0 & -(y_j^+ - y_i^+) \\ 0 & 1 & x_j^+ - x_i^+ \\ 0 & 0 & 1 \end{bmatrix}. \tag{3.20}$$

When applying $H_{ij}$ in the centralized system (as when calculating the covariance of the residual), it can be used as an $N \times (N \times M)$ matrix. By substituting from Equation 3.12 and Equation 3.17, the covariance of the residual is calculated as follows:

$$\begin{aligned} S_{ij}(t_k) &= H_{ij}(t_k) P^+ H_{ij}^T(t_k) + R_{ij}(t_k) \\ &= \Gamma_i^T \tilde{S}_{ij} \Gamma_i \end{aligned} \tag{3.21}$$

with

$$\tilde{S}_{ij} = \tilde{H}_{ij} P_{ii}^+ \tilde{H}_{ij}^T + P_{jj}^+ + \Gamma_i R_{ij}(t_k) \Gamma_i^T. \tag{3.22}$$

The residual covariance $\tilde{S}$ has dimensions $N \times N$, the same as if one were updating the pose estimate of only one agent instead of $M$ agents. In the latter case, the dimension of matrix $S$ would be $(N \times M) \times (N \times M)$. To complete the update step, the Kalman filter gain for this measurement is given by

$$K(t_k) = P^+ H_{ij}^T(t_k) S_{ij}^{-1}(t_k). \tag{3.23}$$

If $K(t_k)$ is a vector of length $M$ where each element represents the Kalman gain for each agent, then $K_i = -P_{ii}^+ \tilde{H}_{ij}^T \tilde{S}_{ij}^{-1} \Gamma_i$ and $K_j = P_{jj}^+ \tilde{S}_{ij}^{-1} \Gamma_i$. All other elements in $K(t_k)$ are 0 for

this measurement. In the example provided by [20],

$$K(t_k) = \begin{bmatrix} K_1 \\ K_2 \\ K_3 \end{bmatrix} = \begin{bmatrix} -P_{11}^{+}\tilde{H}_{12}^{T}\tilde{S}_{12}^{-1}\Gamma_1 \\ P_{22}^{+}\tilde{S}_{12}^{-1}\Gamma_1 \\ 0_{N\times N} \end{bmatrix}. \tag{3.24}$$

The pose estimate for the centralized system is given by

$$\hat{X} = X^{+} + K(t_k)\mathbf{r}_{ij}(t_k) \tag{3.25}$$

where $\mathbf{r}_{ij}$ is the residual of the relative pose measurement as shown in [20]. Only the agents involved in the measurement are updated. Therefore, let $\hat{X}$ be a vector containing the state of all agents in the group after the update step. Equation 3.25 is rewritten as

$$\hat{X} \triangleq \begin{bmatrix} \hat{\mathbf{x}}_1 \\ \hat{\mathbf{x}}_2 \\ \vdots \\ \hat{\mathbf{x}}_i \\ \vdots \\ \hat{\mathbf{x}}_j \\ \vdots \\ \hat{\mathbf{x}}_M \end{bmatrix} = \begin{bmatrix} \mathbf{x}_1^{+} \\ \mathbf{x}_2^{+} \\ \vdots \\ \left(I_{N\times N} - P_{ii}^{+}\tilde{H}_{ij}^{T}\tilde{S}_{ij}^{-1}\right)\mathbf{x}_i^{+} + P_{ii}^{+}\tilde{H}_{ij}^{T}\tilde{S}_{ij}^{-1}\left(\mathbf{x}_j^{+} - \Gamma_i\mathbf{z}_{ij}\right) \\ \vdots \\ \left(I_{N\times N} - P_{jj}^{+}\tilde{S}_{ij}^{-1}\right)\mathbf{x}_j^{+} + P_{jj}^{+}\tilde{S}_{ij}^{-1}\left(\mathbf{x}_i^{+} + \Gamma_i\mathbf{z}_{ij}\right) \\ \vdots \\ \mathbf{x}_M^{+} \end{bmatrix}. \tag{3.26}$$

Applying the example from [20] yields the special case:

$$\begin{bmatrix} \hat{\mathbf{x}}_1 \\ \hat{\mathbf{x}}_2 \\ \hat{\mathbf{x}}_3 \end{bmatrix} = \begin{bmatrix} \left(I_{3\times 3} - P_{11}^{+}\tilde{H}_{12}^{T}\tilde{S}_{12}^{-1}\right)\mathbf{x}_1^{+} + P_{11}^{+}\tilde{H}_{12}^{T}\tilde{S}_{12}^{-1}\left(\mathbf{x}_2^{+} - \Gamma_1\mathbf{z}_{12}\right) \\ \left(I_{3\times 3} - P_{22}^{+}\tilde{S}_{12}^{-1}\right)\mathbf{x}_2^{+} + P_{22}^{+}\tilde{S}_{12}^{-1}\left(\mathbf{x}_1^{+} + \Gamma_1\mathbf{z}_{12}\right) \\ \mathbf{x}_3^{+} \end{bmatrix}. \tag{3.27}$$

Finally, the covariance update for the centralized system is calculated as

$$\hat{P} = P^{+} - P^{+}H_{ij}^{T}(t_k)S_{ij}^{-1}(t_k)H_{ij}(t_k)P^{+}. \tag{3.28}$$

By substituting for $P^+$ from Equation 3.12 and for $H_{ij}$, only the covariances of the agents involved in the measurement as well as their cross correlation terms change. This means that

$$\begin{aligned}
\hat{P}_{ii} &= P_{ii}^+ - P_{ii}^+ \tilde{H}_{ij}^T \tilde{S}_{ij}^{-1} \tilde{H}_{ij} P_{ii}^+ \\
\hat{P}_{jj} &= P_{jj}^+ - P_{jj}^+ \tilde{S}_{ij}^{-1} P_{jj}^+ \\
\hat{P}_{ij} &= P_{ii}^+ \tilde{H}_{ij}^T \tilde{S}_{ij}^{-1} P_{jj}^+ \\
\hat{P}_{ji} &= P_{jj}^+ \tilde{S}_{ij}^{-1} \tilde{H}_{ij} P_{ii}^+
\end{aligned} \tag{3.29}$$

where $\hat{P}_{ij}$ corresponds to the $i$th row and $j$th column of the centralized covariance matrix $\hat{P}$. The remaining covariances remain unchanged and their cross covariances remain as 0. For the example in [20], this yields

$$P^+ = \begin{bmatrix} P_{11}^+ - P_{11}^+ \tilde{H}_{12}^T \tilde{S}_{12}^{-1} \tilde{H}_{12} P_{11}^+ & P_{11}^+ \tilde{H}_{12}^T \tilde{S}_{12}^{-1} P_{22}^+ & 0_{3\times 3} \\ P_{22}^+ \tilde{S}_{12}^{-1} \tilde{H}_{12} P_{11}^+ & P_{22}^+ - P_{22}^+ \tilde{S}_{12}^{-1} P_{22}^+ & 0_{3\times 3} \\ 0_{3\times 3} & 0_{3\times 3} & P_{33}^+ \end{bmatrix}. \tag{3.30}$$

### 3.1.2 The Introduction of Cross Correlation Terms

Note the above derivations of the covariance updates are only valid for the first iteration of the multi-agent CL approach. When the first update is complete, cross correlation elements are introduced in the covariance of the state estimate as shown in Equation 3.29. This matrix is now written as (c.f. Equation 3.12)

$$P^+ = \begin{bmatrix} P_{11}^+ & P_{12}^+ & \cdots & P_{1M}^+ \\ P_{21}^+ & P_{22}^+ & \cdots & P_{2M}^+ \\ \vdots & \vdots & \ddots & \vdots \\ P_{M1}^+ & P_{M2}^+ & \cdots & P_{MM}^+ \end{bmatrix}. \tag{3.31}$$

Each agent $i$ continues to move independently of the others and its motion is described by Equation 3.1. Since the measured quantities are local to agent $i$, the state propagation equations can be distributed among the agents and remain as described by Equation 3.3 and Equation 3.4. The measurement $\mathbf{z}$ from Equation 3.13 also remains unchanged. However,

the same is not true for the covariance. Substituting from Equation 3.31, the covariance propagation is given as

$$\begin{aligned} P^+ &= \Phi(t_{k+1},t_k)P(t_k)\Phi^T(t_{k+1},t_k) + Q_d(t_k) \\ &= \begin{bmatrix} A_1P_{11}^+A_1^T + Q_1 & A_1P_{12}^+A_2^T & \dots & A_1P_{1M}^+A_M^T \\ A_2P_{21}^+A_1^T & A_2P_{22}^+A_2^T + Q_2 & \dots & A_2P_{2M}^+A_M^T \\ \vdots & \vdots & \ddots & \vdots \\ A_MP_{M1}^+A_1^T & A_MP_{M2}^+A_2^T & \dots & A_MP_{MM}^+A_M^T + Q_M \end{bmatrix} \end{aligned} \tag{3.32}$$

where the calculation of each of the propagated diagonal sub-matrix elements of the centralized covariance matrix requires the processing of odometric or inertial measurements from the corresponding agent

$$P_{ii}^+ = A_i(t_{k+1},t_k)P_{ii}(t_k)A_i^T(t_{k+1},t_k) + Q_i. \tag{3.33}$$

The residual covariance matrix update will be

$$\begin{aligned} S_{ij}(t_k) &= H_{ij}(t_k)P^+H_{ij}^T(t_k) + R_{ij}(t_k) \\ &= \Gamma_i^T\tilde{S}_{ij}\Gamma_i \end{aligned} \tag{3.34}$$

with

$$\tilde{S}_{ij} = \tilde{H}_{ij}P_{ii}^+\tilde{H}_{ij}^T - P_{ji}^+\tilde{H}_{ij}^T - \tilde{H}_{ij}P_{ij}^+ + P_{jj} + \Gamma_iR_{ij}(t_k)\Gamma_i^T \tag{3.35}$$

where $R_{ij}(t_k)$ is the measurement noise covariance matrix associated with the relative pose measurement between agents $i$ and $j$ and is defined similarly to Equation 3.15. To calculate matrix $S_{ij}(t_k)$, only the covariance between the two agents is needed along with their cross correlation terms. These terms would be exchanged when the agents detect each other. The

Kalman gain for this update is now

$$\begin{aligned} K(t_k) &= P^+ H_{ij}^T(t_k) S_{ij}^{-1}(t_k) \\ &= \begin{bmatrix} \left(P_{1j}^+ - P_{1i}^+ \tilde{H}_{ij}^T\right) \tilde{S}_{ij}^{-1} \Gamma_i \\ \left(P_{2j}^+ - P_{2i}^+ \tilde{H}_{ij}^T\right) \tilde{S}_{ij}^{-1} \Gamma_i \\ \vdots \\ \left(P_{Mj}^+ - P_{Mi}^+ \tilde{H}_{ij}^T\right) \tilde{S}_{ij}^{-1} \Gamma_i \end{bmatrix} = \begin{bmatrix} K_1 \\ K_2 \\ \vdots \\ K_M \end{bmatrix} = \tilde{K} \Gamma_i. \end{aligned} \tag{3.36}$$

From the example given in [20], the Kalman gain when agent 2 meets agent 3 is

$$K(t_k) = \begin{bmatrix} \left(P_{13}^+ - P_{12}^+ \tilde{H}_{23}^T\right) \tilde{S}_{23}^{-1} \Gamma_2 \\ \left(P_{23}^+ - P_{22}^+ \tilde{H}_{23}^T\right) \tilde{S}_{23}^{-1} \Gamma_2 \\ \left(P_{33}^+ - P_{32}^+ \tilde{H}_{23}^T\right) \tilde{S}_{23}^{-1} \Gamma_2 \end{bmatrix} \tag{3.37}$$

or

$$\tilde{K} = \begin{bmatrix} \left(P_{13}^+ - P_{12}^+ \tilde{H}_{23}^T\right) \tilde{S}_{23}^{-1} \\ \left(P_{23}^+ - P_{22}^+ \tilde{H}_{23}^T\right) \tilde{S}_{23}^{-1} \\ \left(P_{33}^+ - P_{32}^+ \tilde{H}_{23}^T\right) \tilde{S}_{23}^{-1} \end{bmatrix}. \tag{3.38}$$

The pose estimate update is computed as

$$\begin{aligned} \hat{X} &= X^+ + K(t_k) \mathbf{r}_{ij}(t_k) \\ &= X^+ + \tilde{K} \left[ \Gamma_i \mathbf{z}_{ij}(t_k) - \left( \mathbf{x}_j^+ - \mathbf{x}_i^+ \right) \right] \end{aligned} \tag{3.39}$$

or

$$\hat{X} = \begin{bmatrix} \hat{\mathbf{x}}_1 \\ \hat{\mathbf{x}}_2 \\ \vdots \\ \hat{\mathbf{x}}_M \end{bmatrix} = \begin{bmatrix} \mathbf{x}_1^+ + \tilde{K}_1 \left[ \Gamma_i \mathbf{z}_{ij}(t_k) - \left( \mathbf{x}_j^+ - \mathbf{x}_i^+ \right) \right] \\ \mathbf{x}_2^+ + \tilde{K}_2 \left[ \Gamma_i \mathbf{z}_{ij}(t_k) - \left( \mathbf{x}_j^+ - \mathbf{x}_i^+ \right) \right] \\ \vdots \\ \mathbf{x}_M^+ + \tilde{K}_M \left[ \Gamma_i \mathbf{z}_{ij}(t_k) - \left( \mathbf{x}_j^+ - \mathbf{x}_i^+ \right) \right] \end{bmatrix}, \tag{3.40}$$

and the covariance update is calculated as before by applying Equation 3.28 which gives

$$\hat{P} = \begin{bmatrix} \hat{P}_{11} & \hat{P}_{12} & \dots & \hat{P}_{1M} \\ \hat{P}_{21} & \hat{P}_{22} & \dots & \hat{P}_{2M} \\ \vdots & \vdots & \ddots & \vdots \\ \hat{P}_{M1} & \hat{P}_{M2} & \dots & \hat{P}_{MM} \end{bmatrix} \tag{3.41}$$

where the first block row, second block row and $M$th row of the covariance matrix is

$$\begin{aligned}
\hat{P}_{11} &= P^+_{11} - \left[P^+_{1j} - P^+_{1i}\tilde{H}^T_{ij}\right]\tilde{S}^{-1}_{ij}\left[P^+_{j1} - \tilde{H}^T_{ij}P^+_{i1}\right] \\
\hat{P}_{12} &= P^+_{12} - \left[P^+_{1j} - P^+_{1i}\tilde{H}^T_{ij}\right]\tilde{S}^{-1}_{ij}\left[P^+_{j2} - \tilde{H}^T_{ij}P^+_{i2}\right] \\
&\vdots \\
\hat{P}_{1M} &= P^+_{1M} - \left[P^+_{1j} - P^+_{1i}\tilde{H}^T_{ij}\right]\tilde{S}^{-1}_{ij}\left[P^+_{jM} - \tilde{H}^T_{ij}P^+_{iM}\right]
\end{aligned}$$

$$\begin{aligned}
\hat{P}_{21} &= P^+_{21} - \left[P^+_{2j} - P^+_{2i}\tilde{H}^T_{ij}\right]\tilde{S}^{-1}_{ij}\left[P^+_{j1} - \tilde{H}^T_{ij}P^+_{i1}\right] \\
\hat{P}_{22} &= P^+_{22} - \left[P^+_{2j} - P^+_{2i}\tilde{H}^T_{ij}\right]\tilde{S}^{-1}_{ij}\left[P^+_{j2} - \tilde{H}^T_{ij}P^+_{i2}\right] \\
&\vdots \\
\hat{P}_{2M} &= P^+_{2M} - \left[P^+_{2j} - P^+_{2i}\tilde{H}^T_{ij}\right]\tilde{S}^{-1}_{ij}\left[P^+_{jM} - \tilde{H}^T_{ij}P^+_{iM}\right]
\end{aligned} \tag{3.42}$$

$$\begin{aligned}
\hat{P}_{M1} &= P^+_{M1} - \left[P^+_{Mj} - P^+_{Mi}\tilde{H}^T_{ij}\right]\tilde{S}^{-1}_{ij}\left[P^+_{j1} - \tilde{H}^T_{ij}P^+_{i1}\right] \\
\hat{P}_{M2} &= P^+_{M2} - \left[P^+_{Mj} - P^+_{Mi}\tilde{H}^T_{ij}\right]\tilde{S}^{-1}_{ij}\left[P^+_{j2} - \tilde{H}^T_{ij}P^+_{i2}\right] \\
&\vdots \\
\hat{P}_{MM} &= P^+_{MM} - \left[P^+_{Mj} - P^+_{Mi}\tilde{H}^T_{ij}\right]\tilde{S}^{-1}_{ij}\left[P^+_{jM} - \tilde{H}^T_{ij}P^+_{iM}\right].
\end{aligned}$$

## 3.2 Applying the Third Dimension

When adding another dimension, namely altitude $z$, the dynamical system first needs to be modified with appropriate inputs. The state $\mathbf{x}_i$ as well as the input $\mathbf{u}_i$ for agent $i$ is now

$$\mathbf{x}_i = \begin{bmatrix} x_i \\ y_i \\ z_i \\ \theta_i \end{bmatrix}, \qquad \mathbf{u}_i = \begin{bmatrix} V_{im}^x \\ V_{im}^y \\ V_{im}^z \\ \omega_{im} \end{bmatrix}, \tag{3.43}$$

where $z_i$ is the altitude of agent $i$, $V_{im}^z$ is the measured linear velocity in the $z$ axis defined similar to $V_{im}^x$ and $V_{im}^y$ in Equation 3.2. The heading and angular velocity about the $z$ axis remains defined as $\theta_i$ and $\omega_{im}$ as in the 2-D equations. Measured angles and rates to pitch and roll could also be applied, but these added degrees of freedom add unnecessary complexity to this nonlinear model of an aerial robot. Therefore, the nonlinear system dynamics are now

$$\begin{bmatrix} x_i \\ y_i \\ z_i \\ \theta_i \end{bmatrix}_{t_{k+1}} = \begin{bmatrix} x_i \\ y_i \\ z_i \\ \theta_i \end{bmatrix}_{t_k} + \begin{bmatrix} V_{im}^x \cos\theta_i - V_{im}^y \sin\theta_i \\ V_{im}^x \sin\theta_i + V_{im}^y \cos\theta_i \\ V_{im}^z \\ \omega_i \end{bmatrix} \delta t = \begin{bmatrix} f_1(x_i, y_i, z_i, \theta_i) \\ f_2(x_i, y_i, z_i, \theta_i) \\ f_3(x_i, y_i, z_i, \theta_i) \\ f_4(x_i, y_i, z_i, \theta_i) \end{bmatrix}. \tag{3.44}$$

To determine the linearized propagation matrix $A$, the Jacobian of the system dynamics is computed, which results in

$$\begin{aligned} A_i &\triangleq \begin{bmatrix} \frac{\partial f_1}{\partial x_i} & \frac{\partial f_1}{\partial y_i} & \frac{\partial f_1}{\partial z_i} & \frac{\partial f_1}{\partial \theta_i} \\ \frac{\partial f_2}{\partial x_i} & \frac{\partial f_2}{\partial y_i} & \frac{\partial f_2}{\partial z_i} & \frac{\partial f_2}{\partial \theta_i} \\ \frac{\partial f_3}{\partial x_i} & \frac{\partial f_3}{\partial y_i} & \frac{\partial f_3}{\partial z_i} & \frac{\partial f_3}{\partial \theta_i} \\ \frac{\partial f_4}{\partial x_i} & \frac{\partial f_4}{\partial y_i} & \frac{\partial f_4}{\partial z_i} & \frac{\partial f_4}{\partial \theta_i} \end{bmatrix} \\ &= \begin{bmatrix} 1 & 0 & 0 & \left(-V_{im}^x \sin\theta_i - V_{im}^y \cos\theta_i\right)\delta t \\ 0 & 1 & 0 & \left(V_{im}^x \cos\theta_i - V_{im}^y \sin\theta_i\right)\delta t \\ 0 & 0 & 1 & 0 \\ 0 & 0 & 0 & 1 \end{bmatrix}. \end{aligned} \tag{3.45}$$

The system noise input matrix $G_i$ from Equation 3.8 as well as the effect matrix $B_i$ from Equation 3.4 is now

$$G_i = B_i = \begin{bmatrix} \delta t \cos\theta_i & -\delta t \sin\theta_i & 0 & 0 \\ \delta t \sin\theta_i & \delta t \cos\theta_i & 0 & 0 \\ 0 & 0 & \delta t & 0 \\ 0 & 0 & 0 & \delta t \end{bmatrix}, \tag{3.46}$$

and the system noise covariance $Q_i$ is determined as in Equation 3.9, with the system noise $\mathbf{w}_i$ now $\begin{bmatrix} w_{V_{im}^x} & w_{V_{im}^y} & w_{V_{im}^z} & w_{\omega_i} \end{bmatrix}^T$. The propagation of the state and covariance remains the same as in Equation 3.4 and Equation 3.10, respectively, only now the covariance $P_{ii}$ is a $4 \times 4$ matrix for each agent $i$. The measurement for the update step incorporates the $z$ axis as follows

$$\mathbf{z}_{ij}(t_k) = \begin{bmatrix} C^T(\theta_i) \left( \begin{bmatrix} x_j \\ y_j \end{bmatrix} - \begin{bmatrix} x_i \\ y_i \end{bmatrix} \right) \\ z_j - z_i \\ \theta_j - \theta_i \end{bmatrix} + \mathbf{n}_{ij}, \tag{3.47}$$

where $C(\theta_i)$ remains the same as Equation 3.14. To determine $H_{ij}$, Equation 3.47 has to be linearized. Expanding the measurement yields

$$\mathbf{z}_{ij}(t_k) = \begin{bmatrix} (x_j - x_i)\cos\theta_i + (y_j - y_i)\sin\theta_i \\ -(x_j - x_i)\sin\theta_i + (y_j - y_i)\cos\theta_i \\ z_j - z_i \\ \theta_j - \theta_i \end{bmatrix} + \mathbf{n}_{ij}. \tag{3.48}$$

Since the distributed approach to CL is being used, the only concern is linearizing $\mathbf{z}_{ij}(t_k)$ with respect to the local agent $i$. Therefore, $\tilde{H}_{ij}$ needs to be found which incorporates the $z$ axis and satisfies the relationships found in the 2-D form of Equation 3.16 through

Equation 3.18. Taking the Jacobian of Equation 3.48 with respect to agent $i$ gives

$$\begin{aligned} H_{ij} &\triangleq \begin{bmatrix} \frac{\partial f_1}{\partial x_i} & \frac{\partial f_1}{\partial y_i} & \frac{\partial f_1}{\partial z_i} & \frac{\partial f_1}{\partial \theta_i} \\ \frac{\partial f_2}{\partial x_i} & \frac{\partial f_2}{\partial y_i} & \frac{\partial f_2}{\partial z_i} & \frac{\partial f_2}{\partial \theta_i} \\ \frac{\partial f_3}{\partial x_i} & \frac{\partial f_3}{\partial y_i} & \frac{\partial f_3}{\partial z_i} & \frac{\partial f_3}{\partial \theta_i} \\ \frac{\partial f_4}{\partial x_i} & \frac{\partial f_4}{\partial y_i} & \frac{\partial f_4}{\partial z_i} & \frac{\partial f_4}{\partial \theta_i} \end{bmatrix} \\ &= \begin{bmatrix} -\cos\theta_i & -\sin\theta_i & 0 & -(x_j - x_i)\sin\theta_i + (y_j - y_i)\cos\theta_i \\ \sin\theta_i & -\cos\theta_i & 0 & -(x_j - x_i)\cos\theta_i - (y_j - y_i)\sin\theta_i \\ 0 & 0 & -1 & 0 \\ 0 & 0 & 0 & -1 \end{bmatrix}. \end{aligned} \tag{3.49}$$

From the 2-D example in [20], multiplying $\Gamma_1^T$ from Equation 3.18 by $-\tilde{H}_{12}$ in Equation 3.19 gives

$$\begin{bmatrix} -\cos\theta_1 & -\sin\theta_1 & -(x_2 - x_1)\sin\theta_1 + (y_2 - y_1)\cos\theta_1 \\ \sin\theta_1 & -\cos\theta_1 & -(x_2 - x_1)\cos\theta_1 - (y_2 - y_1)\sin\theta_1 \\ 0 & 0 & -1 \end{bmatrix} \tag{3.50}$$

which matches the 2-D form of Equation 3.49 with respect to agent $i$, given by:

$$H_{12} \triangleq \begin{bmatrix} -\cos\theta_i & -\sin\theta_i & -(x_2 - x_1)\sin\theta_1 + (y_2 - y_1)\cos\theta_1 \\ \sin\theta_i & -\cos\theta_i & -(x_2 - x_1)\cos\theta_1 - (y_2 - y_1)\sin\theta_1 \\ 0 & 0 & -1 \end{bmatrix}. \tag{3.51}$$

Since Equation 3.50 and Equation 3.51 are equal, this relationship can be used to determine $\tilde{H}_{ij}$ that incorporates the $z$ axis. Define the new $\Gamma_i$ as

$$\Gamma_i = \begin{bmatrix} \cos(\theta_i) & -\sin(\theta_i) & 0 & 0 \\ \sin(\theta_i) & \cos(\theta_i) & 0 & 0 \\ 0 & 0 & 1 & 0 \\ 0 & 0 & 0 & 1 \end{bmatrix}. \tag{3.52}$$

To satisfy the relationship where $\Gamma_i^T(-\tilde{H}_{ij}) = H_{ij}$ with respect to agent $i$, $\tilde{H}_{ij}$ is therefore,

$$\tilde{H}_{ij} = \begin{bmatrix} 1 & 0 & 0 & -(y_j - y_i) \\ 0 & 1 & 0 & (x_j - x_i) \\ 0 & 0 & 1 & 0 \\ 0 & 0 & 0 & 1 \end{bmatrix}. \tag{3.53}$$

Equation 3.21 through Equation 3.42 can now be used to conduct CL with altitude incorporated.

# CHAPTER 4:
# Methodology

## 4.1 Resources

### 4.1.1 Research Platform

The implementation of CL presented in this thesis starts by selecting an appropriate UAV that can represent an agent within a swarm while being relatively easy to program, control, and conduct flight experiments. Another consideration includes compatibility where a common architecture can be used for communication and computation among different platforms. One last desire is to have a vehicle that does not have the flight restrictions that are placed on the larger UAVs. With these requirements in mind, the Parrot AR Drone was selected for this thesis. Shown in Figure 4.1, the AR Drone is a good compromise between the inexpensive toy helicopters, which are hard to fly and have limited capabilities, and the extremely expensive remote controlled helicopters built by enthusiasts and hobbyists. The quadrotor can be controlled by a smartphone or tablet, is intuitive to fly, can be flown indoors, and features two built-in cameras. The quadrotor relies on an Android or iOS device connected to a Wi-Fi hotspot; however, the `ardrone_autonomy` package for ROS is used to provide the ability for controlling the quadrotor through a laptop computer. ROS and the drivers necessary to control the AR Drone through ROS are discussed in more detail in Section 4.1.3. More details about the AR Drone and the AR Drone Software Development Kit (SDK) can be found in [56]. Technical specifications of the AR Drone and more details about its navigation and sensor suite can be found in [57] and [58].

### 4.1.2 Vicon

The Vicon motion capture system is used in this thesis to gather truth data and to serve as the upper-bound for measuring localization performance. Vicon is an Infrared (IR) marker tracking system that offers 3-D millimeter resolution of object displacement. It offers a high order of positional and angular precision by delivering positional data in all six degrees-of-freedom with low latency. The system used for this project consists of 10 cameras outfitted with IR optical filters and an array of IR LEDs. Two of these cameras are shown

Figure 4.1: Parrot AR Drone 1.0

in Figure 4.2. A set of five reflective markers placed in a unique pattern on the hull of the AR Drone (example shown in Figure 4.3) reflect the IR radiation emitted by the LEDs. All other light is filtered so the system only recognizes the markers. Software is then used to construct the 3-D representation of the markers from the images taken by all cameras. The software can then uniquely identify each AR Drone by the constellation of markers on its hull and precisely localize it when the quadrotor is within field of view of the cameras. There are a number of different software programs for Vicon but for this project, Vicon Tracker 1.3 is utilized. A screen shot of the Graphical User Interface (GUI) showing three AR Drones being identified and tracked is shown in Figure 4.4. More details about Vicon and its technical specifications are found in [59].

### 4.1.3 Robot Operating System

ROS is a flexible framework for developing and executing robot software. It is a collection of tools, libraries, and conventions that simplify the creation of robot behavior across a wide variety of platforms. One of its primary goals is to provide an open-ended collaboration framework for those in the robotics research community and it was developed in the open using the permissive Berkeley Software Distribution (BSD) open-source license. One of the core components of ROS that is relied on heavily in this project is its communication infrastructure. A node in ROS is simply any process that performs a computation. Nodes are combined together into a graph and communicate with one another using unidirectional, streaming topics. Topics have anonymous publish and subscribe semantics, which means nodes are not aware of who they are communicating with. Instead, nodes that are interested

Figure 4.2: Two of the ten Vicon motion capture cameras used to capture true position and orientation of the AR Drones used in this project.

in data subscribe to the relevant topic and nodes that generate data publish to the relevant topic. ROS also provides recording and playback of messages, request/response remote procedure calls, and a distributed parameter system. With this simple construct, ROS allows for the visualizing and recording of flight parameters, integration of other supporting tools for the experiment (i.e., Vicon), and for the ability to control multiple AR Drones on one centralized system. To be able to interface with multiple AR Drones and the Vicon system, the `ardrone_autonomy` and `vicon_bridge` packages provided for ROS are used. The `ardrone_autonomy` package not only provides the interface between ROS and the AR Drone, but also allows for more functionality and direct control of features that are suppressed by the stock applications provided for the quadrotor. The `vicon_bridge` package allows pose information in all six degrees of freedom for each agent to be passed into ROS from Vicon as a topic. More details about these packages can be found in [60] and [61], respectively. More information about ROS can be found in [62].

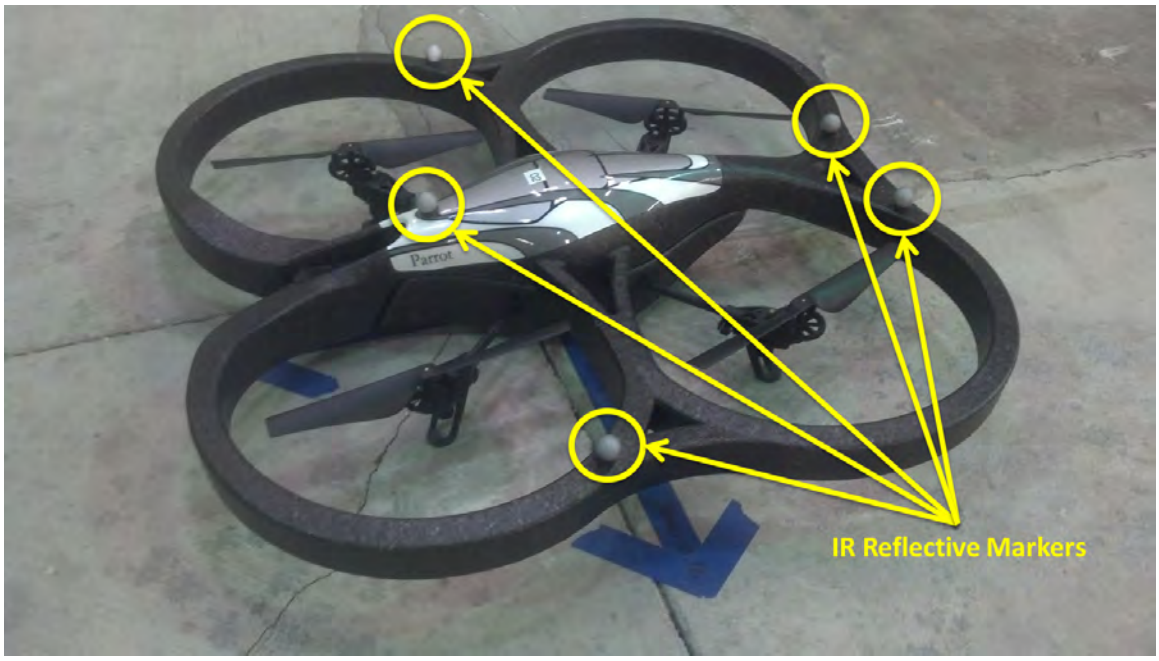



Figure 4.3: IR reflective markers placed on the hull of the AR Drone are used by the Vicon system for identifying and accurately tracking the quadrotor.

## 4.2 The Experimental Environment

### 4.2.1 Executing the CL Algorithm

The 2-D distributed CL algorithm for a unicycle model as shown by [20] was first coded and tested in Python 3.3.2. The `numpy` and `matplotlib` packages for Python were used to complete the matrix operations as well as visualize the results of the pose estimation. Then, the lateral velocity control inputs ($V_{im}^{y}$) were added to the dynamics model and appropriate changes to the affected vectors and matrices were made. Once the 2-D algorithm was functioning properly with no known semantic errors, the 3-D modifications from Section 3.2 were made using the applicable dynamics model and tested in the simulated environment. Figure 4.5 shows an example of the simulation results where three agents are conducting CL using the 2-D unicycle model. Each agent is assigned a constant forward linear velocity with no lateral movement or heading change. Agent 1, shown in blue, is at the top of the plot and is assigned to move at a heading of 45 degrees where 0 degrees is along the positive $x$ axis. Agent 2, shown in red, is heading at 0 degrees and Agent 3, shown in green, is heading at -45 degrees. The plot marked by an $\times$ is the track of the agent's dynamic

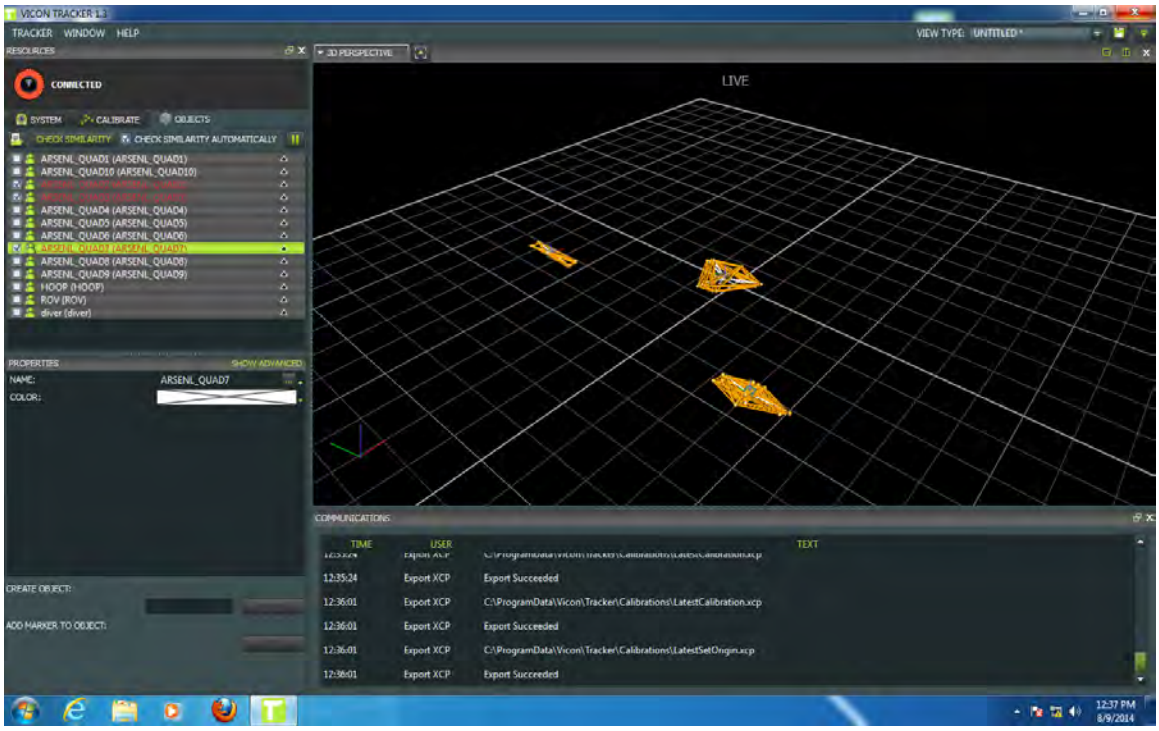



Figure 4.4: Screen capture of the Vicon Tracker 1.3 software displaying real time identification and tracking of the quadrotors used in this project.

model if there was no noise in the velocity measurement. The plot marked by a thin line is the agent's actual movement with noise introduced in the velocity input. Finally, the dotted line is the pose estimation of the agent using the CL algorithm where one measurement is made in each time step. The relative pose measurement is continuous and rotates between Agent 2 measuring the pose of Agent 3, and Agent 3 measuring Agent 1.

### 4.2.2 Pose Estimation and Control of the AR Drone

To examine the performance of the CL algorithm, another method of pose estimation is required for comparison. Because the AR Drone has no native means to estimate its position, a pose estimator using DR was created in ROS. A node called `est_state_imu` was written for each agent in the swarm. When the node is run, it takes the initial pose of the agent by receiving data from Vicon. At each time step, the node takes the linear and angular velocity measurements for each axis from the agent's on-board IMU. To get all required velocity measurements, the node has to subscribe to both the `ardrone/imu` and `ardrone/navdata` topics, which are included in the `ardrone_autonomy` ROS pack-

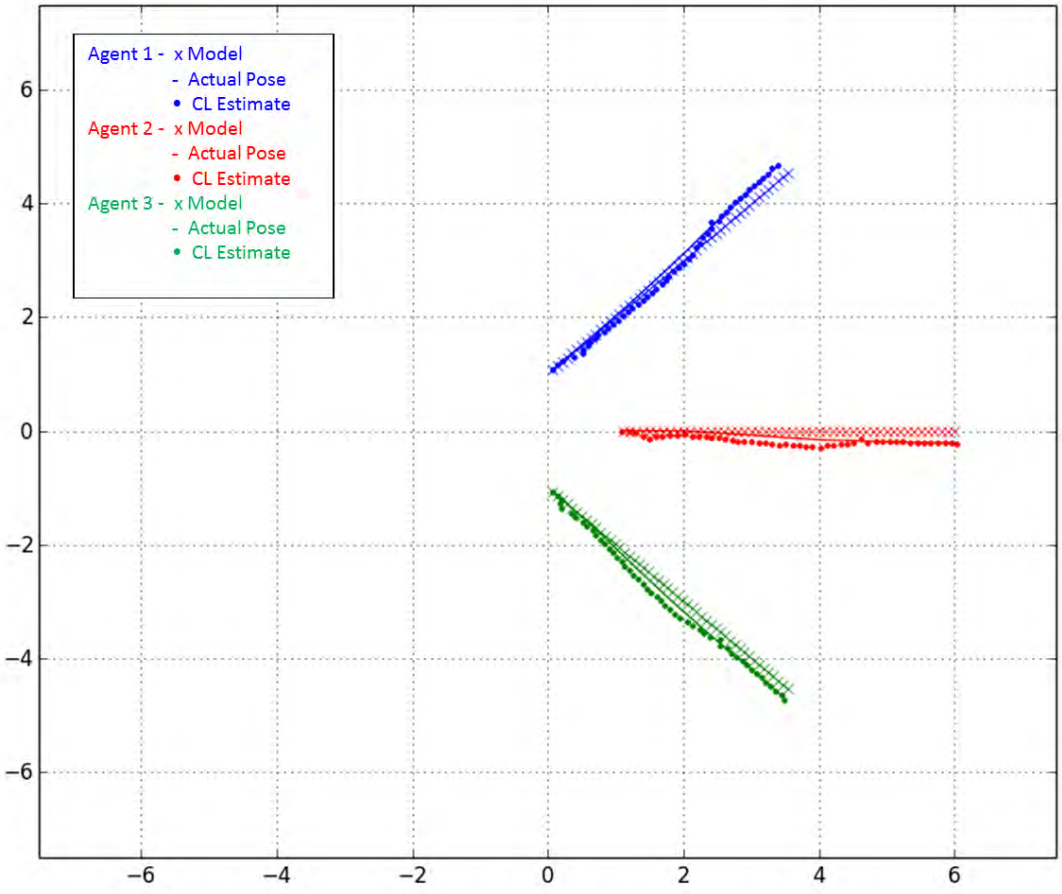



Figure 4.5: Three agents conducting CL in Python using the 2-D unicycle model found in [20]. Each agent is assigned a constant forward linear velocity with no heading change. Relative pose measurements are continuous where Agent 2 measures the pose of Agent 3, and Agent 3 measures Agent 1 on the subsequent time step.

age. At each time step, the `est_state_imu` node multiplies the measured velocity by the time step and adds the resulting displacement to the previous pose estimate. The resulting pose information including $x$, $y$, $z$, pitch, roll, and yaw in Euler angles is published via a topic called `estPose/imu`. In the same message, the linear and angular velocities from the quadrotor's IMU are echoed to consolidate the information which was extracted from two separate topics. This helps streamline and synchronize measurements.

To ensure the measured velocity from the IMU on the AR Drone was accurate, a comparison test with Vicon was completed. A single AR Drone was commanded by a simple controller to fly forward in its $x$ axis at a constant velocity and altitude for a designated period of time. No commands to turn about the $z$ axis or fly laterally were given. Vicon was used to track and record actual displacement of the quadrotor during its flight. Using the displacements and knowing the time between pose measurements made by Vicon (approximately 0.01 seconds), the actual velocity of the quadrotor was calculated and plotted for this single degree-of-freedom. Velocities reported by the AR Drone's IMU were also recorded and plotted. This process was repeated for commanded linear velocities of 0.1, 0.2, 0.5, 0.8. and 1.0. Rearward velocities (i.e., negative commanded velocity in the $x$ axis) at the settings specified previously were also recorded. When data for the forward and rearward velocity were collected, the test was repeated for the quadrotor's lateral velocity and for angular velocity about the $z$ axis. The same quadrotor was used for all tests and its configuration was kept constant. Calculations did not account for any test day conditions such as drift or any aerodynamic cross-coupling which in the context of this project were not significant to the results. The observed data showed that the velocities measured and reported by the AR Drone's IMU were comparable and for the most part within 0.3 meters per second (m/s) to the actual velocities calculated by the reported Vicon positions. The standard deviation was also estimated when the constant velocities were observed. All standard deviations from all velocity tests where then averaged to determine the value used for the standard deviation in the velocity measurements for the CL algorithm. Figure 4.6 and Figure 4.7 show two of the plots from the collected data. Each plot shows the linear velocity in the $x$ axis as calculated from the Vicon data and the velocity reported by the AR Drone IMU. The figures show a 0.1 and 1.0 commanded linear velocity respectively. Note that noise in the Vicon measurement required a low pass filter for smoothing the calculated velocity.

From [60], it was noted that linear command velocities are in the range from -1.0 to 1.0. These settings do not represent the actual velocity being commanded from the AR Drone, unlike the commanded angular velocities, and is significantly lower than what was actually observed in the above tests. For building controllers that can accurately maintain a constant velocity in the AR Drone, it was necessary to map the commanded velocity to the actual velocity observed at that setting. Using the forward linear velocity data collected from

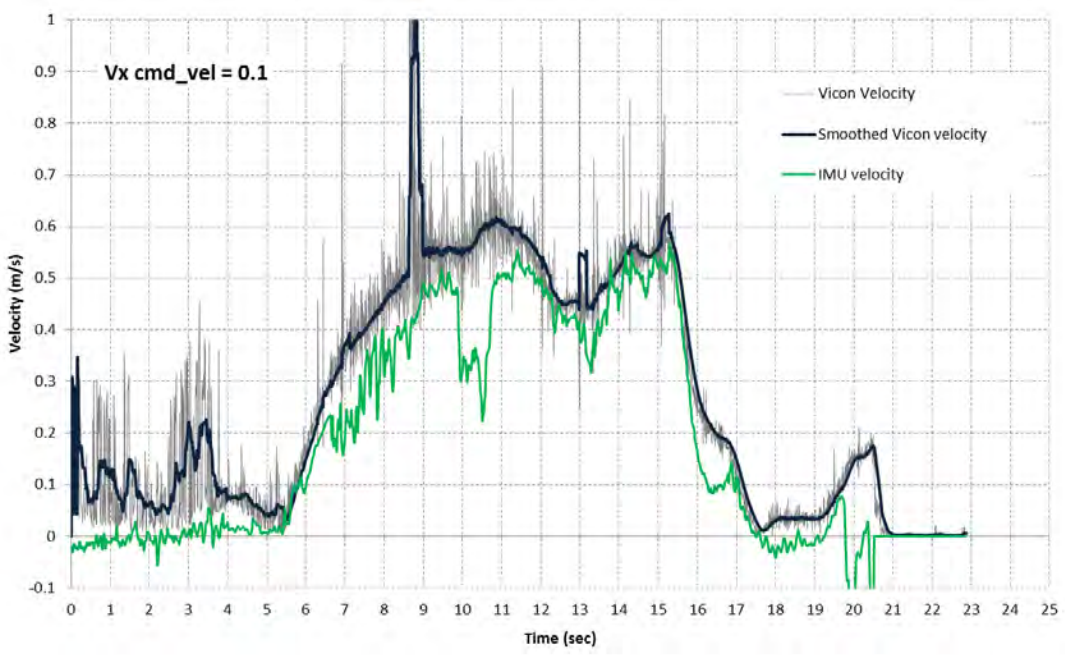



Figure 4.6: The forward linear velocity measured by Vicon and the forward velocity reported by the AR Drone IMU when a constant commanded velocity of 0.1 is sent to the quadrotor.

the above tests, Figure 4.8 shows the command velocity setting versus the sustained maximum velocity observed. A curve fit was applied to the points and the resulting function to represent the curve was estimated as

$$V_d = V_{max}(1 - e^{-\gamma(V_{cmd})}) \tag{4.1}$$

where $V_d$ is the desired velocity, $V_{max}$ is the maximum observed velocity of the AR Drone (1.4 m/s), $V_{cmd}$ is the command velocity sent to the quadrotor, and $\gamma$ is a constant set to 5.5. Therefore to command a specific velocity, one solves for $V_{cmd}$ to get:

$$V_{cmd} = \frac{\ln\left(-(\frac{V_d}{V_{max}}) - 1\right)}{-\gamma} \tag{4.2}$$

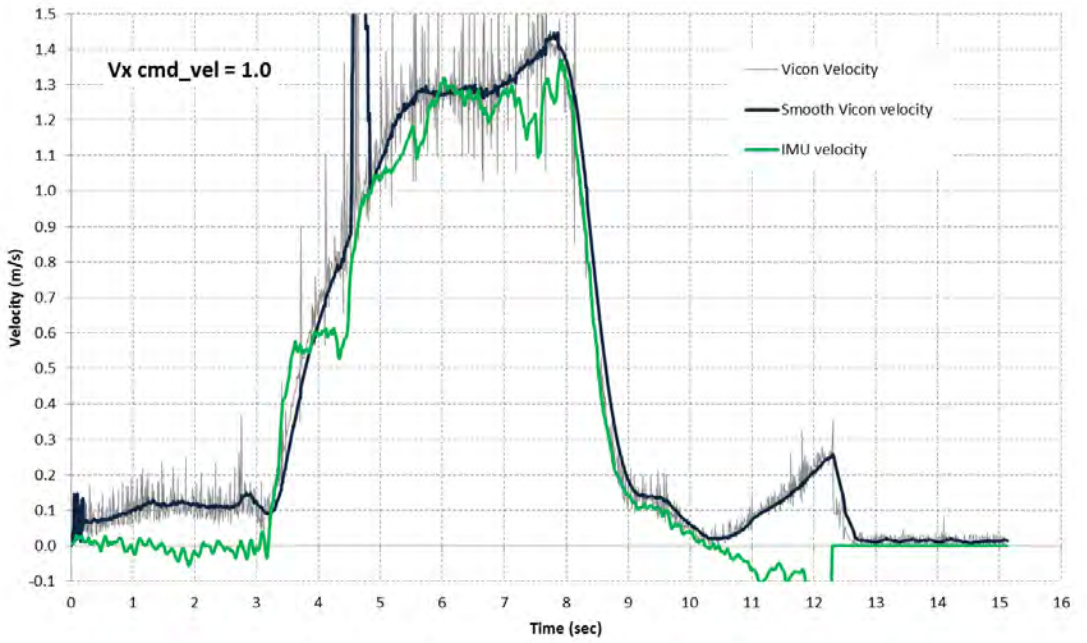



Figure 4.7: The forward linear velocity measured by Vicon and the forward velocity reported by the AR Drone IMU when a constant commanded velocity of 1.0 is sent to the quadrotor.

### 4.2.3 Controlling Multiple Quadrotors with a Single ROS Master

Additional processes are needed to develop a centralized architecture for control of multiple AR Drones. This centralized control eases data collection and analysis, safety of flight, communication, and management of the numerous ROS nodes required for the experiment. First, code was developed that scans WiFi for AR Drones attempting to communicate with other wireless devices when they power-up. When a quadrotor is found, it is assigned a unique Internet Protocol (IP) address based on its network id and brought into a centralized network. This network contains the single ROS master as well as the network interface with Vicon. Another program written in Python called `SwarmLoad.py` scans for active AR Drones by pinging all IP addresses assigned to the quadrotors by the scanner. When it receives a response from a quadrotor, the program creates a ROS namespace for the AR Drone, and launches the driver on the network using its assigned IP address. These drivers serve as the controlling interface between the AR Drone and ROS. Since the same driver is launched for each active AR Drone, the ROS namespace prevents naming conflicts and ensures each resulting topic has a unique name. Finally, the `SwarmLoad.py` program

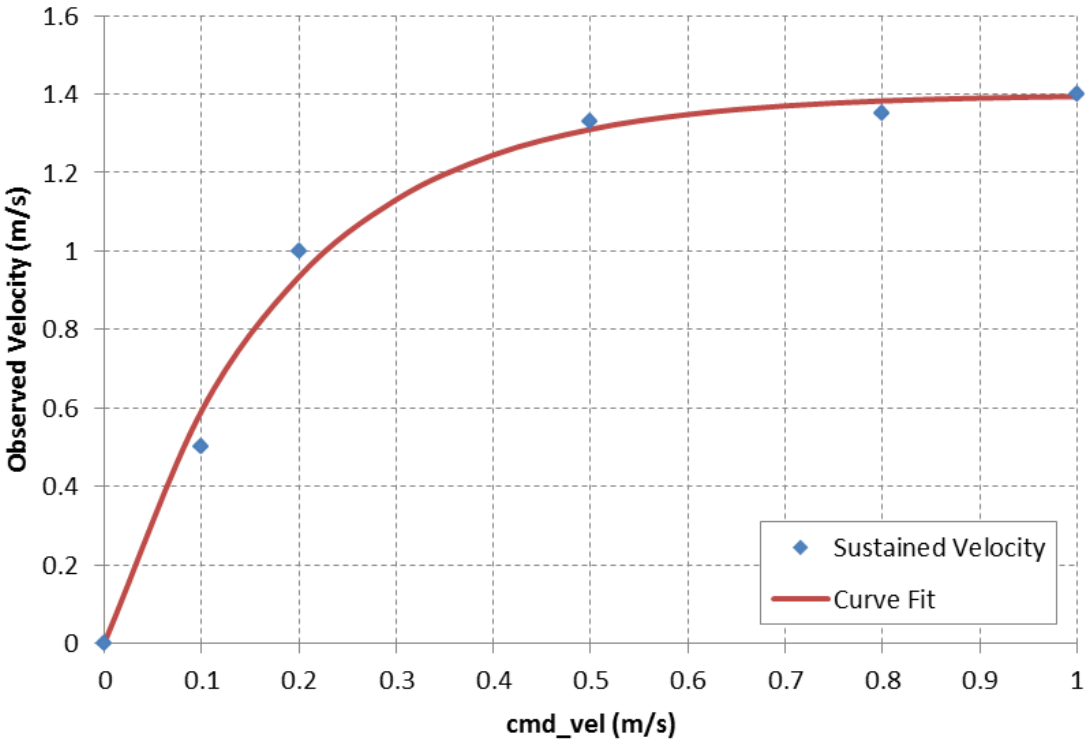



Figure 4.8: Constant commanded forward velocity setting from the `ardrone_autonomy` ROS package versus the observed maximum sustained forward velocity. The fitted curve represents the mapping between the command velocity and the constant maximum velocity achieved for that setting by the AR Drone.

maintains a list of quadrotors that are actively on the network and publishes this list via the `DroneList` topic. The list of AR Drones used in the project along with their ROS namespace, wireless id, and unique network address are shown in Table 4.1. To send basic commands to all quadrotors on the active list another Python program called `SwarmGUI.py` is used. This program provides a GUI for sending take-off, land, emergency, and reset commands to selected quadrotors or all quadrotors on the `DroneList`. The GUI also provides a display for the current state and remaining battery life of each AR Drone.

Higher level behavior such as path following, loiter, obstacle avoidance, and formation control can now be written in separate ROS nodes and run. There is one controller for each quadrotor. The node sends its agent velocity commands in all six degrees-of-freedom via the `cmd_vel` ROS topic. For this experiment, a simple waypoint controller was built

| Side Number | ROS namespace | Wireless ID | Assigned IP Address |
|---|---|---|---|
| 01 | Quad1 | ardrone_197467 | 192.168.0.101 |
| 02 | Quad2 | ardrone_198504 | 192.168.0.102 |
| 03 | Quad3 | ardrone_258674 | 192.168.0.103 |
| 04 | Quad4 | ardrone_256959 | 192.168.0.104 |
| 05 | Quad5 | ardrone_266111 | 192.168.0.105 |
| 06 | Quad6 | ardrone_266678 | 192.168.0.106 |
| 07 | Quad7 | ardrone_198307 | 192.168.0.107 |
| 08 | Quad8 | ardrone_150853 | 192.168.0.108 |

Table 4.1: Identifiers for Each AR Drone in the Swarm

that uses a basic proportional-integral-derivative controller (PID). Agents receive pose data through either the `est_state_imu` node via the `estPose/imu` topic or from Vicon via the `vicon/ARSENL_QUADm/ARSENL_QUADm` topic where `m` is the quadrotor's id number. Forward velocity, track error, altitude, and heading error are determined based on the agent's current pose and the goal waypoint. A gain is then applied to each of the errors and a resulting velocity is sent to the agent to correct for the error. When an agent reaches the goal waypoint, another waypoint is assigned. This process is repeated until the agent is commanded to do otherwise by the `SwarmGUI`. Four waypoints were created that, when connected, form a three by three meter square where the center of the square is the origin of the Vicon reference area. Each quadrotor is assigned to fly separately to each of the four waypoints and then repeat the cycle. Since there is no collision avoidance built-in to the controller or in any other node, each AR Drone is initially assigned a different starting waypoint. Upon reaching their assigned waypoint, the next point is assigned in a counter-clockwise fashion. More details about the `ardrone_scanner.py`, `SwarmLoad.py`, `SwarmGUI.py`, and waypoint controller programs can be found in the source code which is located at `http://faculty.nps.edu/thchung` under `Software Resources`.

### 4.2.4 The Relative Pose Measurement Graph

To measure relative pose among the agents within the swarm, another node in ROS called `est_relDist` is created. Although ultimately, the sensors that are native to the AR Drone (i.e., the video cameras) would be used to help identify and measure the relative pose of agents in the swarm, Vicon is used in this experiment. It provides a more precise means to measure the position of each agent and allows better control of the noise in these mea-

surements when evaluating the performance of the CL implementation. The `est_relDist` node first determines which AR Drones are actively on the network by subscribing to the `DroneList`. An object is created for each agent which then subscribes to the pose information for that agent from Vicon. The difference in $x$, $y$, $z$, pitch, roll, and heading ($\theta$) expressed in local coordinates are taken between each agent and can be represented by an RPMG. The RPMG can be shown by an $M \times M$ block matrix as follows:

$$
\begin{array}{c}
 \\
\text{Agent 1} \\
\text{Agent 2} \\
\text{Agent 3} \\
\vdots \\
\text{Agent M}
\end{array}
\begin{array}{c}
\begin{array}{cccccc}
\text{Agent 1} & \text{Agent 2} & \text{Agent 3} & \dots & \text{Agent M}
\end{array} \\
\begin{bmatrix}
\vec{0}_{6\times 1} & \mathbf{x}_2 - \mathbf{x}_1 & \mathbf{x}_3 - \mathbf{x}_1 & \dots & \mathbf{x}_M - \mathbf{x}_1 \\
\mathbf{x}_1 - \mathbf{x}_2 & \vec{0}_{6\times 1} & \mathbf{x}_3 - \mathbf{x}_2 & \dots & \mathbf{x}_M - \mathbf{x}_2 \\
\mathbf{x}_1 - \mathbf{x}_3 & \mathbf{x}_2 - \mathbf{x}_3 & \vec{0}_{6\times 1} & \dots & \mathbf{x}_M - \mathbf{x}_3 \\
\vdots & \vdots & \vdots & \ddots & \vdots \\
\mathbf{x}_1 - \mathbf{x}_M & \mathbf{x}_2 - \mathbf{x}_M & \mathbf{x}_3 - \mathbf{x}_M & \dots & \vec{0}_{6\times 1}
\end{bmatrix}
\end{array}
$$

where $\mathbf{x} \in \mathbb{R}^6$ is the vector representing the pose of each agent as defined in Section 3.1 with pitch and roll angles added, and $M$ is the number of agents in the swarm. Although pitch and roll angles are not used in the dynamics model for this experiment, it was written into this node for future use in other projects as necessary. The rows and columns of the matrix represent each of the agents while their intersection is the difference vector between the two agents' poses. The diagonal is the zero vector. From Equation 3.47, the $x$ and $y$ differences were multiplied by the rotation matrix $C^T(\theta_i)$. Because the measurement of $\theta_i$ for each agent is in this node, the multiplication with the rotation matrix for each relative measurement is completed before publishing the RPMG to a ROS topic called `RPMG`. The frequency in which all measurements are made can be varied but is set to 50 hertz (Hz) in this experiment. Although all measurements are made in one cycle, the user can determine if all measurements or random ones selected by the program will be published. Noise may also be added to the measurements and can be adjusted by the user before the program is run.

### 4.2.5 Implementing CL in ROS

Similar to Section 4.2.1, the EKF implementation of CL was coded in Python and then modified to operate in the ROS environment. The program is written to accept multiple

AR Drones by subscribing to the `DroneList` topic and then creating an object for each agent in the list. The object contains the agent's state as well as all applicable matrices used in the EKF. The program can estimate agent pose in the 2-D and 3-D plane based on user input. Initial pose information for each agent is received from the respective Vicon or `estPose/imu` topic which is also set by the user prior to launching the node. Each agent's velocity measurements are updated by subscribing to the applicable`estPose/imu` topic and relative pose measurements are received by subscribing to the `RPMG` topic. When the estimate step of the EKF cycle is complete, the pose estimate of each agent is published via their own `estPose/cl` topic. Although pose messages in ROS normally contain rotations in quaternions, the rotations in Euler angles are published and the fourth-dimension is used to publish the trace of the covariance matrix for the applicable agent. The frequency of a single cycle is set to 10 Hz but can be varied by the user before running the program. A summary of the experimental set-up in ROS is shown in Figure 4.9. Associated code can be found at `http://faculty.nps.edu/thchung` under `Software Resources`.

## 4.3 Experimental Method

The lab environment for the experiment is established, and the CL algorithm is ready to be implemented and its performance observed. Experiments conducted in [20] observed the performance of the CL algorithm when agents have and did not have access to absolute positioning information (e.g., using GPS or a map of the environment). Their goal was to demonstrate how the CL algorithm could process and distribute relative and absolute positioning information across a group of agents. They also conducted experiments where agents had intermittent relative pose measurements to show that even sparse measurements could effectively reduce the rate of increase in the position uncertainty over having no measurements at all. For this experiment, the CL algorithm for 3-D is first implemented using various measures of RPMG connectedness to validate the expected rate of position uncertainty as the connectedness of the RPMG grows. Tests begin where no relative measurements are made between agents. Then, the same flight data is used to execute the CL algorithm were only one agent measures another. Finally, CL with a strongly connected and fully connected RPMG is demonstrated. To demonstrate the consistency of performance, the algorithm is applied to multiple flights using a fully connected RPMG.

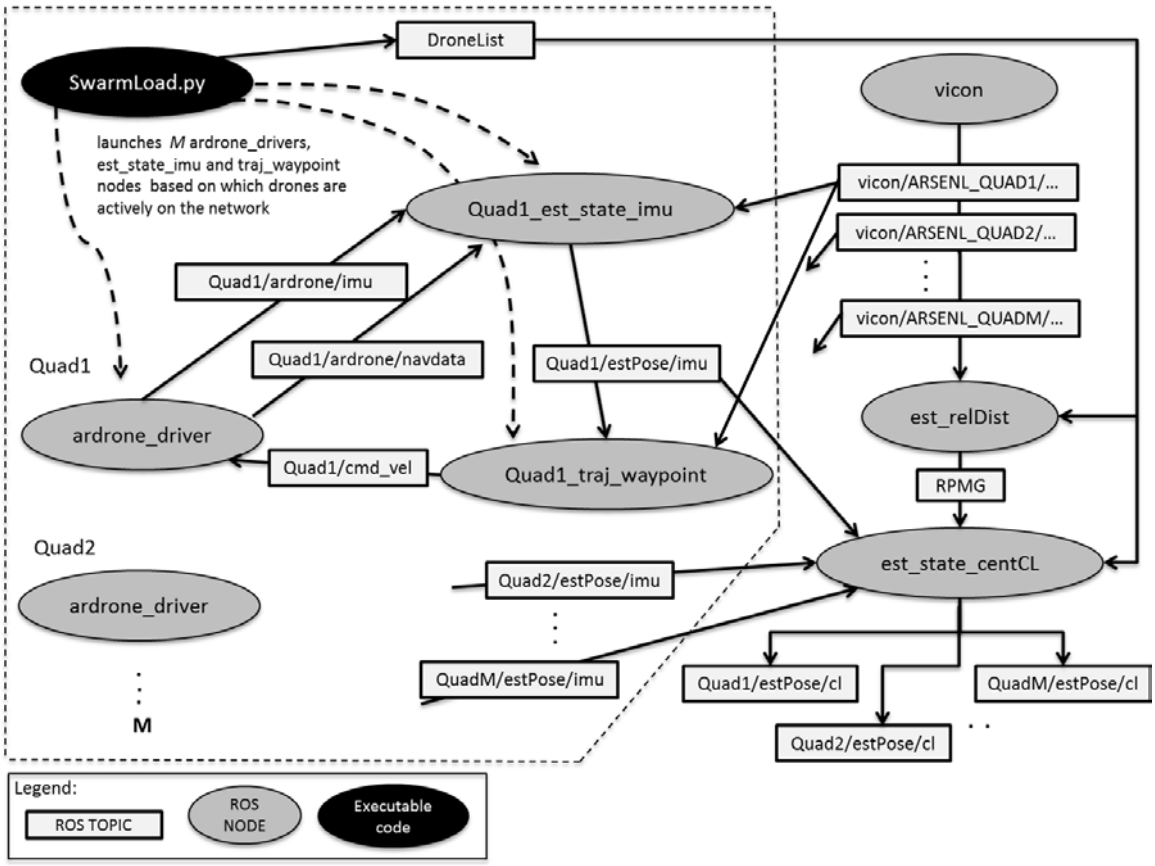



Figure 4.9: AR Drone Cooperative Localization ROS Architecture

For the experiments, all programs and nodes as described in the above sections are run with Quad2, Quad3, and Quad7. The `est_state_centCL` node is run prior to commanding takeoff and continues to provide pose estimates while the quadrotors fly to their designated waypoints at a commanded airspeed of 0.2 m/s and an altitude of 1.5 m. To prevent trajectories from being cluttered during data plotting, each quad flies to each waypoint once before being commanded to land. The algorithm receives the initial pose of all agents via Vicon and publishes its estimates at a rate of 10 Hz.

To represent the system noise $\mathbf{w}_i$ and initialize the noise covariance matrix $Q_i(t_k)$, Equation 3.9 is revised as

$$Q_i(t_k) = G_i(t_k) D_i G_i^T(t_k) \tag{4.3}$$

where $D_i$ is a $4 \times 4$ diagonal matrix in the 3-D model as follows

$$D_i = \begin{bmatrix} \sigma^2_{V_i^x} & 0 & 0 & 0 \\ 0 & \sigma^2_{V_i^y} & 0 & 0 \\ 0 & 0 & \sigma^2_{V_i^z} & 0 \\ 0 & 0 & 0 & \sigma^2_{\omega_i} \end{bmatrix} \tag{4.4}$$

and $\sigma_{V_i^x}$, $\sigma_{V_i^y}$, $\sigma_{V_i^z}$, and $\sigma_{\omega_i}$ are the standard deviation of error in the respective linear and rotational velocity of agent $i$. For the initialization of measurement noise $R_{ij}(t_k)$, Equation 3.15 in the 3-D model is rewritten as

$$R_{ij}(t_k) = \sigma^2_{sensor} I_{N \times N} \tag{4.5}$$

where $\sigma_{sensor}$ is the standard deviation of error for the relative position measurement of agent $i$ in each axis. Relative pose measurements between all agents are computed making the directed RPMG fully connected but can be adjusted by the `est_state_centCL` node. There is no noise added to the measurements and the resulting RPMG is published at 50 Hz.

A total of five flights are executed using the same three quadrotors starting from the same location. All associated topics are recorded and converted to a comma-separated value file format for later analysis using MatLab. Specifically, the Vicon trajectory as well as the DR and CL position estimates of each quadrotor are plotted on a 3-D line plot in MatLab. Performance of the CL algorithm is compared against the DR pose estimate by calculating the absolute error of both estimates where pose data from Vicon serves as the truth. The difference between the estimate and actual position of each axis is taken and squared. These values are then added and the square root of the resulting sum is made. This results in the following equation,

$$err = \sqrt{(x_a - x_i)^2 + (y_a - y_i)^2 + (z_a - z_i)^2} \tag{4.6}$$

where $err$ is the absolute error for each time step, $x_i$, $y_i$, $z_i$ is the position estimate in each axis, and $x_a$, $y_a$, $z_a$ is the position reported by Vicon. The trace of the covariance for each agent, which represents the uncertainty in the Kalman filter's estimate, is also plotted to

monitor performance. Successful implementation of CL is defined as the absolute error and error rate remaining less than the error of the DR pose estimate for the respective agent. Quantitative results using these metrics are provided in the next chapter.

# CHAPTER 5:
# Results and Analysis

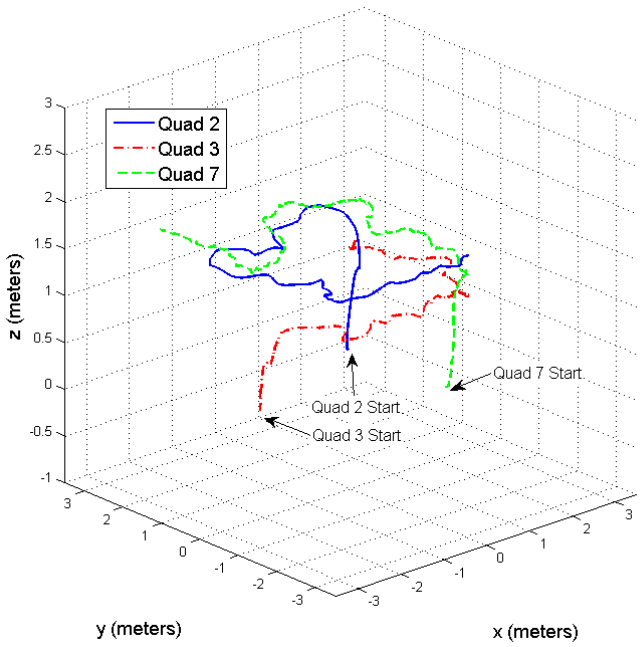
The CL algorithm for 3-D was implemented and tested for three AR Drones starting from different locations and flying within the same area as described in Section 4.3. Relative pose measurements were taken using information from Vicon. All $\sigma$ values of matrix $D$ from Equation 4.4 were set to 0.045, and all $\sigma_{sensor}$ values of $R_{ij}(t_k)$ from Equation 4.5 were set to $10^{-9}$ m. This small value ensures there is very little noise associated with the position measurement. Finally, the covariance matrix $P_{ii}$ for each agent was initialized to 0.2 along the diagonal. The CL algorithm was run just prior to takeoff and test data was recorded for approximately 60 sec. An example of the Vicon trajectory for each AR Drone is shown in Figure 5.1 from a 3-D, birds-eye (i.e., X-Y view), and eye-level view of the flight area.

## 5.1 Varied Connectedness of the RPMG

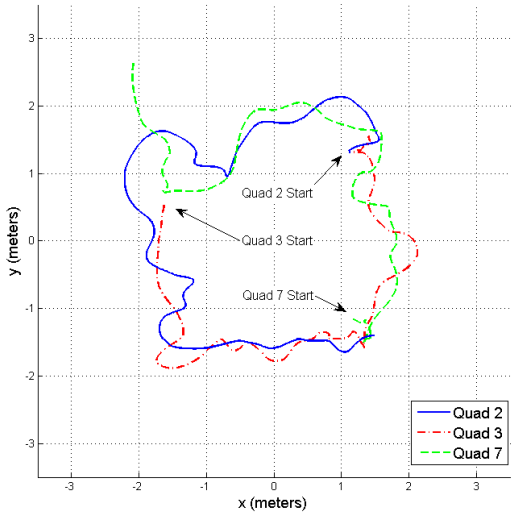
### 5.1.1 No Communication between the Agents

The performance of the CL algorithm is first examined when no relative pose measurements are made between agents. Since there is no exchange of information, each agent dead reckons its pose estimate independently using the output of the propagation step as the pose estimate. The covariance remains the same until the next propagation step. Figure 5.2 shows the Vicon trajectory of each agent as well as the CL and DR estimates from an X-Y view. Figure 5.3 shows the resulting error, and Figure 5.4 plots the trace of the covariance for each agent. In Figure 5.4, the uncertainty for all three quadrotors increases over time except for Quad 2, which peaks at approximately 3 m then begins to decrease about halfway through the flight. It was expected that the uncertainty in all three quadrotors would be unbounded and increase over time; however, Quad 2 does not react this way. This observation is not fully understood, and requires more analysis in future work. The examination should start by looking at the covariance at each time step and determine if negative values are introduced by elements in the propagation matrix $A$ or the noise covariance $Q$. When the matrix causing this decrease is determined, examine each of the variables that make up the individual elements of the matrix. From Figure 5.2, the CL pose estimate

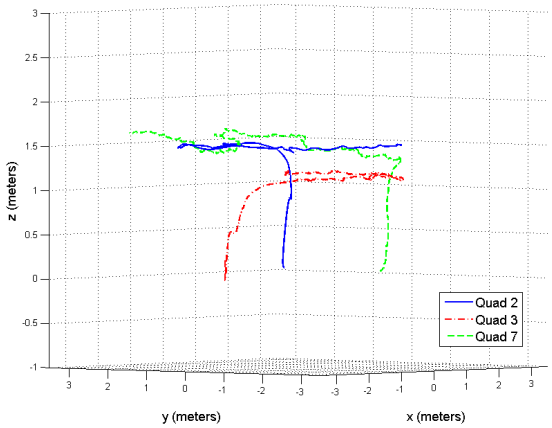
for all three quadrotors is similar in shape and in the general location of the DR estimate. This is consistent since the CL algorithm is only using the velocity measurements of the quadrotor's IMU to generate its pose estimate.




(a) 3-D View

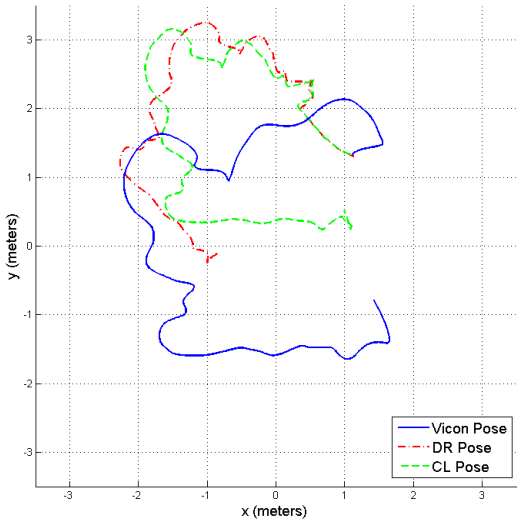



(b) X-Y View

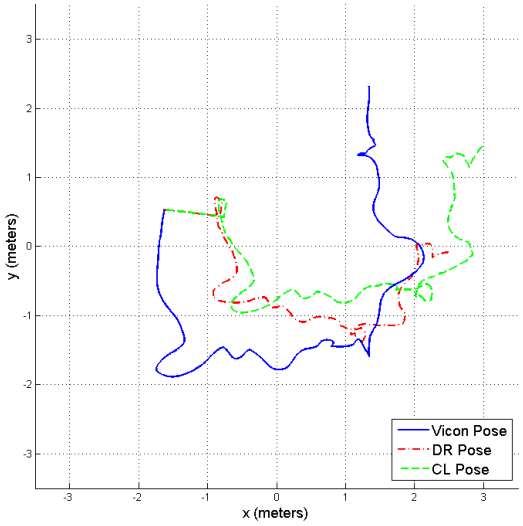



(c) Eye-level View

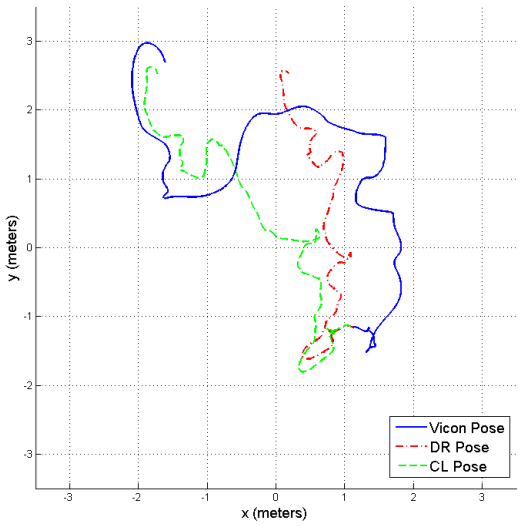
Figure 5.1: Position plot of all three quadrotors as recorded from Vicon




(a) Quad 2




(b) Quad 3




(c) Quad 7

Figure 5.2: Position plot of all quadrotors conducting CL where no relative pose measurements are made between agents. The plot includes the trajectory as recorded by Vicon as well as the CL and DR pose estimates for all agents in an X-Y View.

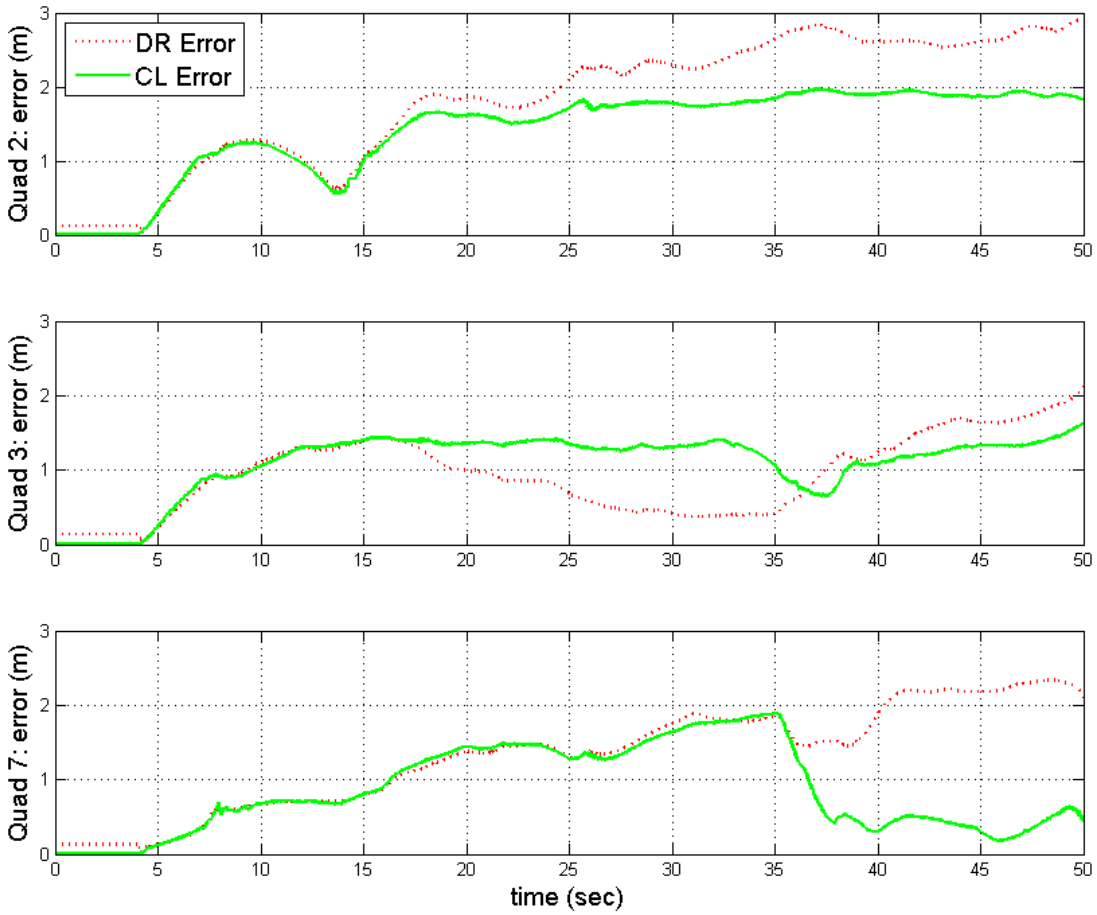



Figure 5.3: CL and DR pose errors of all three agents using CL with no communication between the agents.

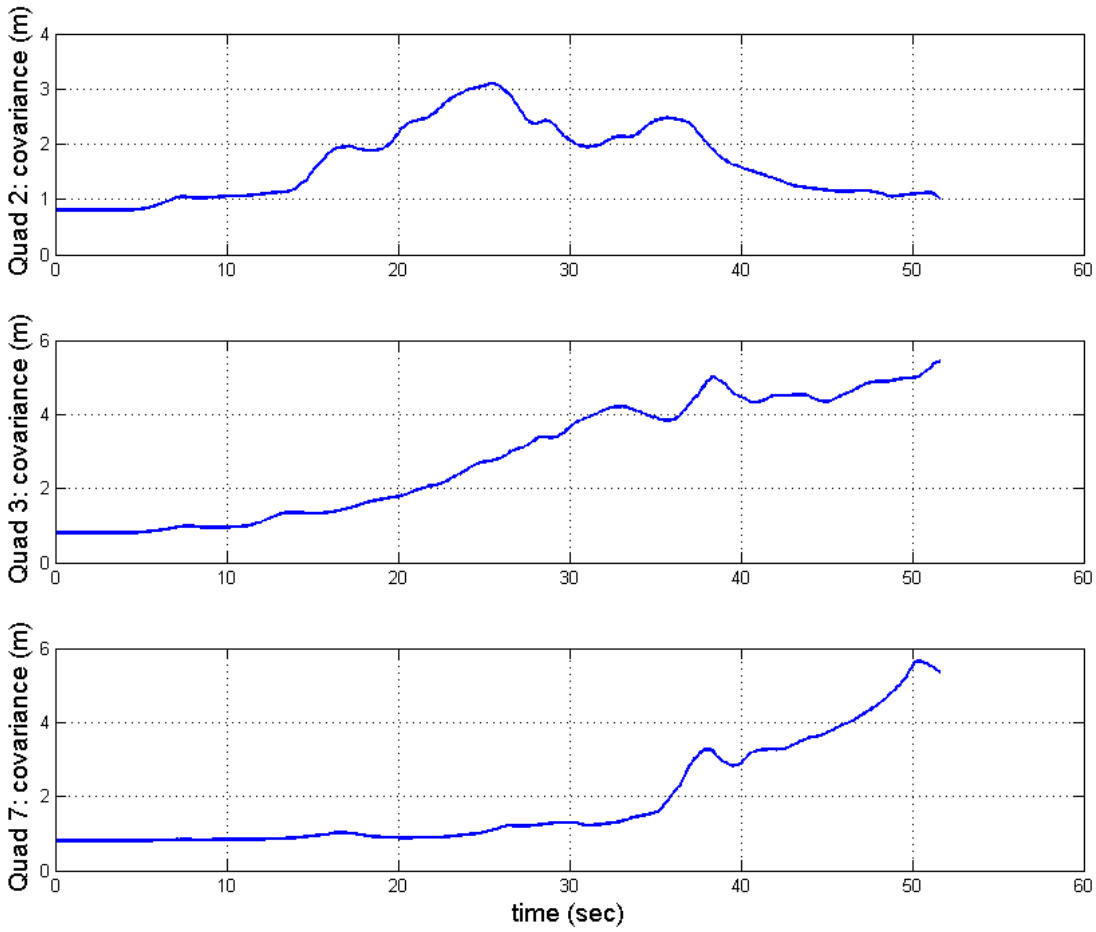



Figure 5.4: Covariance trace of all three agents using CL with no communication between the agents. The covariance trace represents the uncertainty in the Kalman filter's pose estimate.

### 5.1.2 Relative Pose Measurements by a Single Agent

For this case, Quad 2 takes continuous relative pose measurements of Quad 3 while Quad 7 receives no information. Figure 5.5 shows the resulting error, and Figure 5.6 plots the trace of the covariance for each agent. After takeoff, Quad 2 CL pose error increases slightly over the DR estimate, however, it remains between 1.0 and 1.5 m while the DR estimate continues to grow and peak at approximately 3 m. This is an improvement over the pose estimates when there was no communication between agents. Quad 3 also benefits from the communication early in the flight but the CL pose estimate error grows approximately

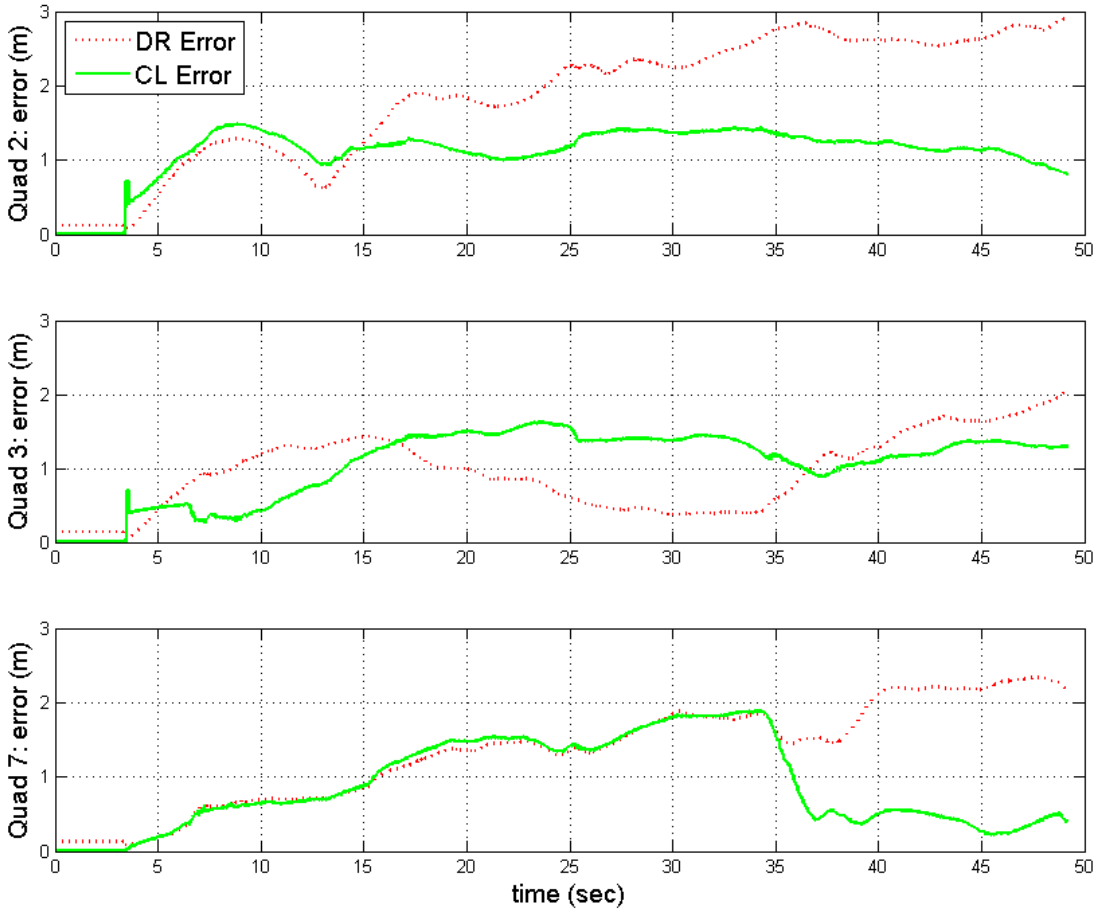



Figure 5.5: CL and DR pose errors plotted using CL when Quad 2 only measures relative pose of Quad 3 continuously.

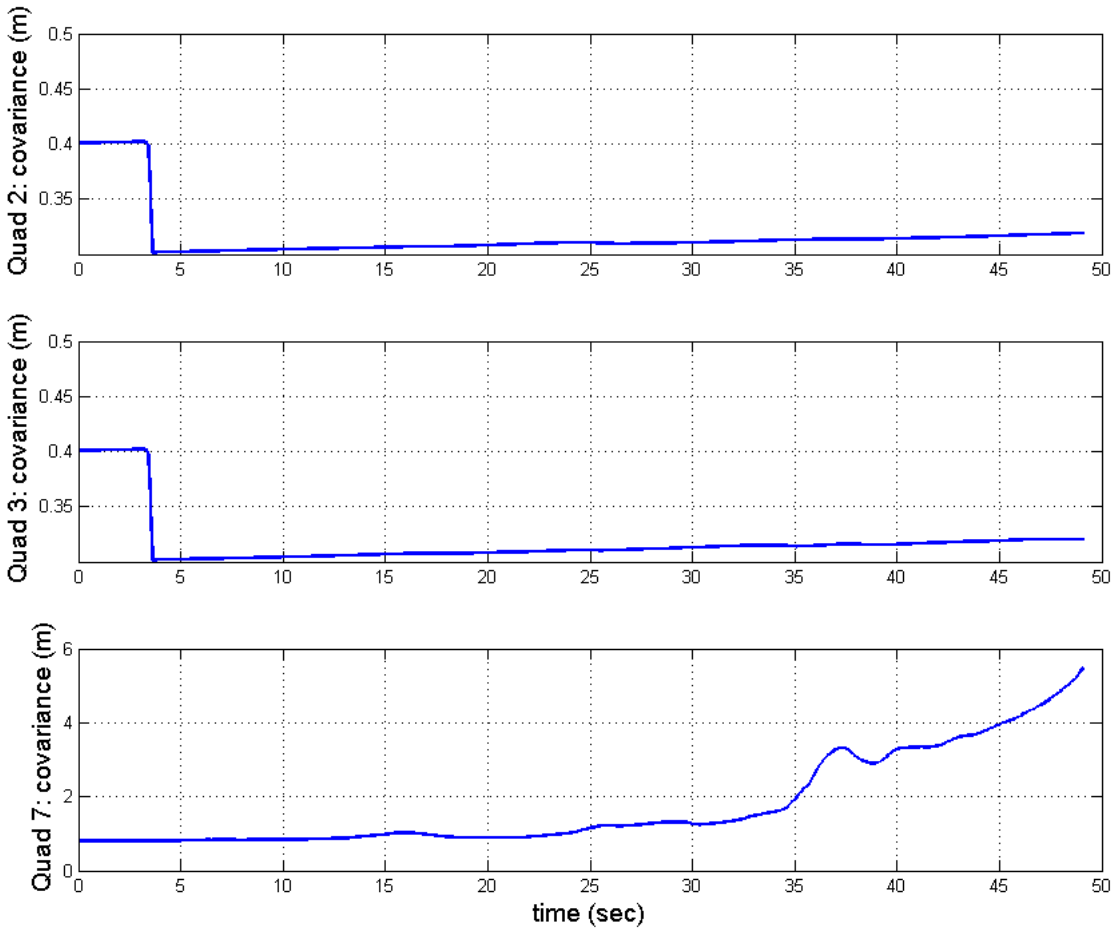



Figure 5.6: Covariance of all agents when Quad 2 only measures relative pose of Quad 3 continuously.

the same as the DR estimate error 17 sec after the start of the flight. When the agents start CL, the covariance trace of Quad 2 and Quad 3 drop from approximately 1.0 m in Figure 5.4 to 0.4 m in Figure 5.6. Both covariances then drop to 0.3 m when the UAVs takeoff and then climb steadily at a rate of approximately 0.4 millimeters per second (mm/s) for the rest of the flight. It was noted in [20], that when agents are moving slowly or standing still while continuously measuring their relative pose, there is almost no new information regarding pose uncertainty. These repeated measurements trigger updates that only suppress the uncertainty associated with the relative pose measurement and not with the agents' pose estimates. Therefore, when Quad 2 and Quad 3 takeoff, the measurements are no longer re-

peated and the calculation of uncertainty improves. It is presumed that the slight but steady increase in uncertainty after takeoff results from the Quads having no access to absolute positioning. The agents improve their position tracking accuracy by measuring their relative positions, but they are not able to bound the overall uncertainty since no one has access to absolute positioning [20]. Quad 7 performance remains unchanged which is expected since it is not receiving any information being shared between Quad 2 and Quad 3.

### 5.1.3 Agents Communicate over a Strongly Connected RPMG

For this test, Agent 2 takes continuous relative pose measurements of Agent 3, Agent 3 measures Agent 7, and Agent 7 measures Agent 2 to form a strongly connected graph as shown in Figure 5.7.

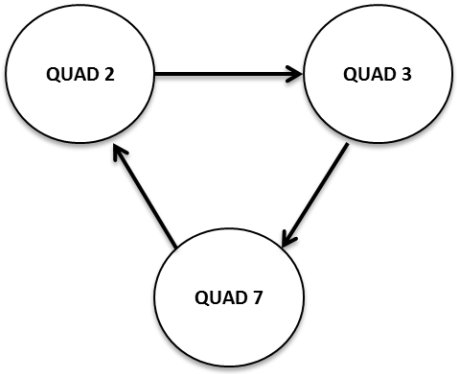



Figure 5.7: Strongly Connected Graph Between Agents

Figure 5.8 shows the resulting error, and Figure 5.9 plots the trace of the covariance for each agent. The covariance for all agents is 0.127 m after the first update and rises at a rate of 0.2 mm/s to 0.138 m before the flight ends. The error in the CL pose estimate for each agent is reduced as compared to the errors shown in Figure 5.5. For Quad 2, the average DR error is 1.89 m while the average error for the CL estimate is 0.57 m. The average DR and CL error for Quad 3 is 1.13 m and 0.67 m respectively, while the average DR and CL estimate error for Quad 7 is 1.39 m and 0.66 m.

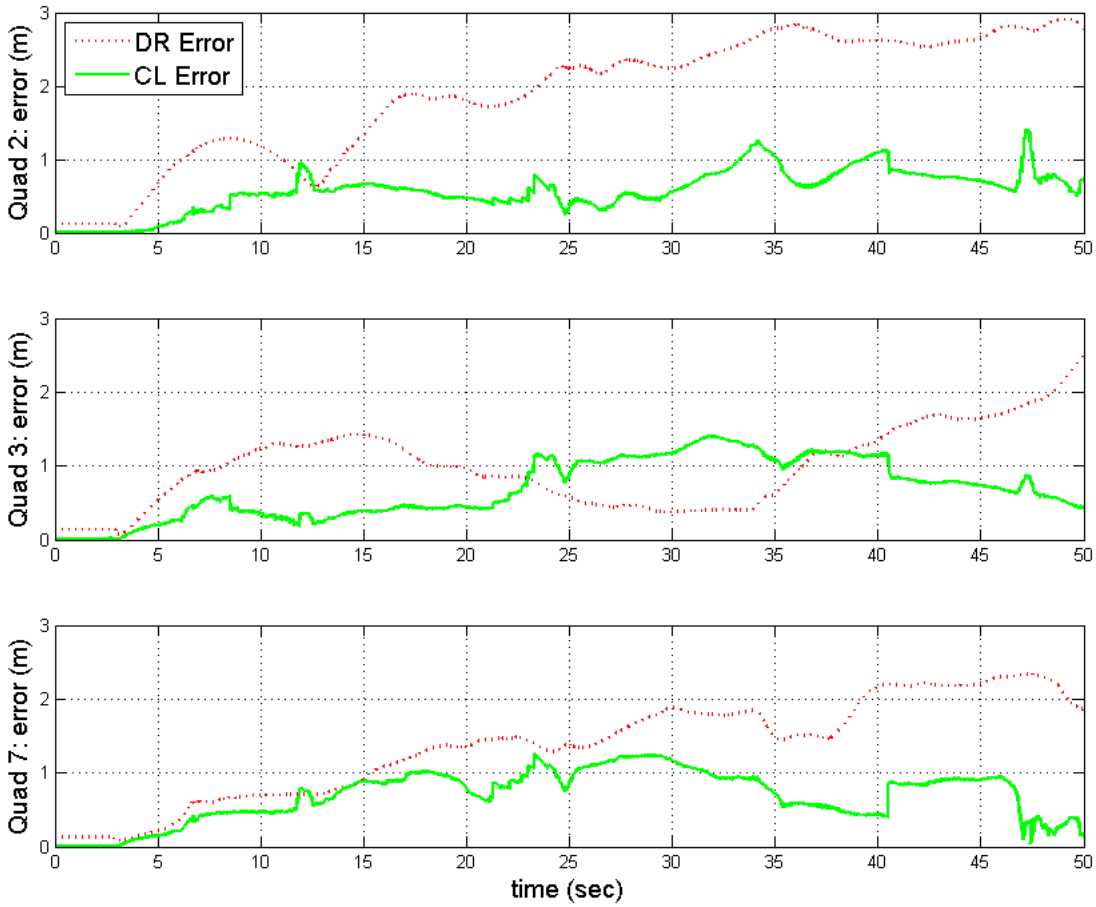



Figure 5.8: CL and DR pose errors plotted using CL with a strongly connected RPMG and continuous relative pose measurements.

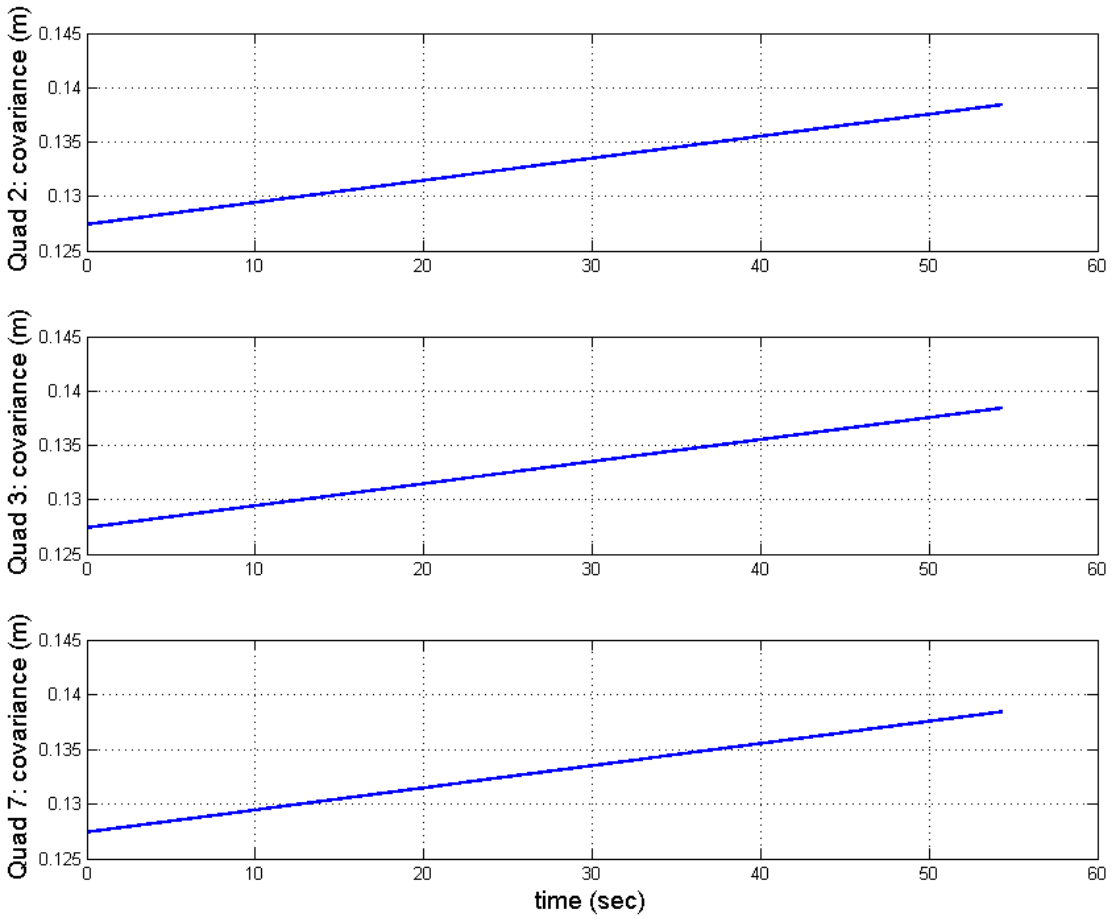



Figure 5.9: Covariance of all agents conducting CL with a strongly connected RPMG and continuous relative pose measurements.

### 5.1.4 Agents Communicate over a Fully Connected RPMG

Here, each agent continuously measures the other agents in the group to form a fully connected graph as shown in Figure 5.10. Figures 5.11, 5.12, and 5.13 are plots of the Vicon trajectory as well as the CL and DR estimates for Quad 2, 3, and 7 respectively. Figure 5.14 shows the resulting error, and Figure 5.15 plots the trace of the covariance for each agent.

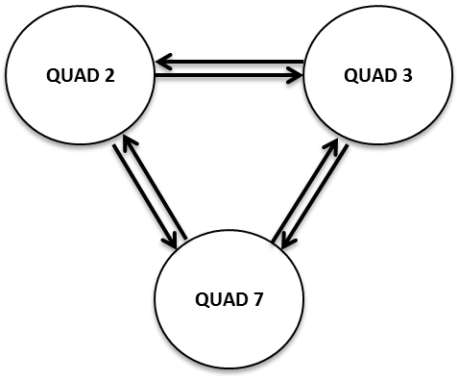
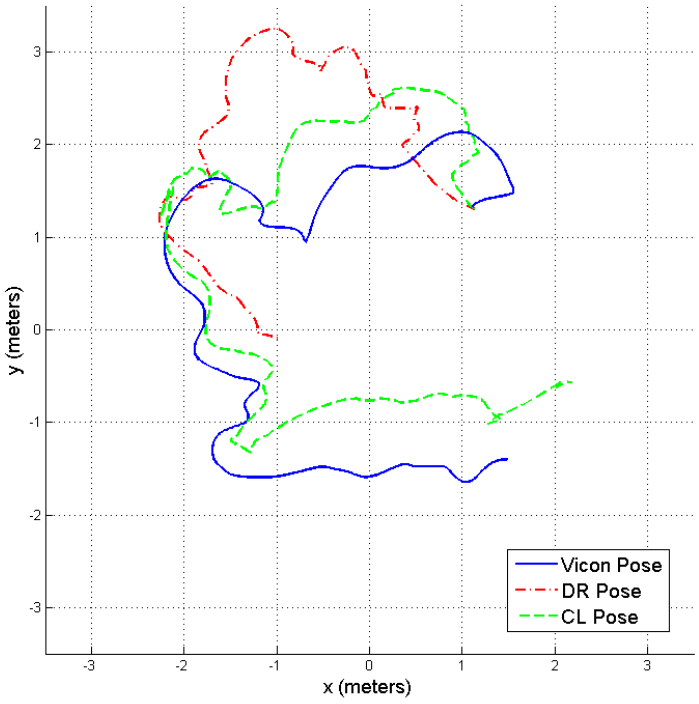
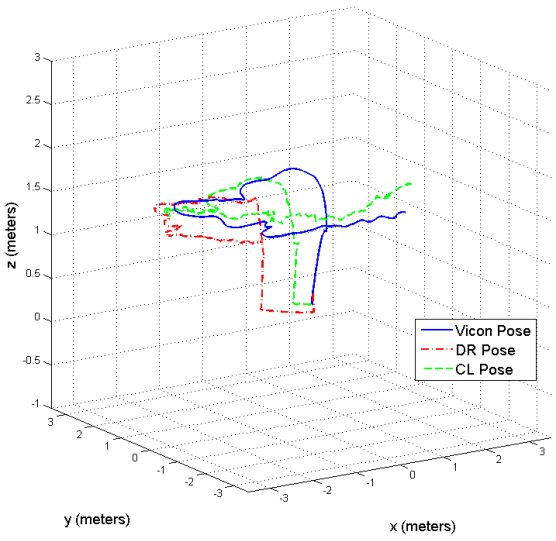



Figure 5.10: Fully Connected RPMG Between Agents

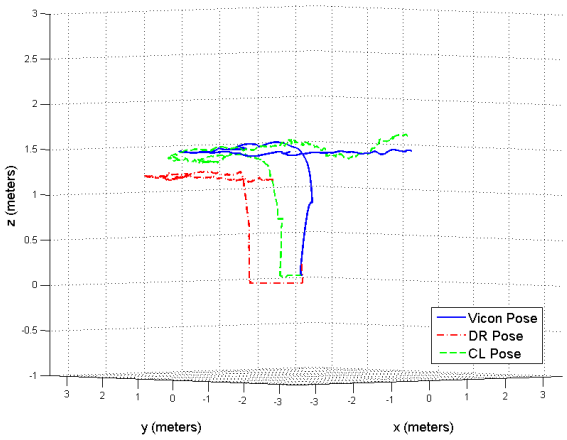
The covariance for all agents is 0.127 m after the first update and rises at a rate of 0.2 mm/s to 0.137 m before the flight ends. The error in the CL pose estimate for each agent is further reduced as compared to the errors shown in Figure 5.8. For Quad 2 and Quad 3, the average error for the CL estimate is 0.50 m. The average CL error for Quad 7 is 0.60 m. A summary of results from the various connections of the RPMG is shown in Table 5.1.




(a) X-Y View

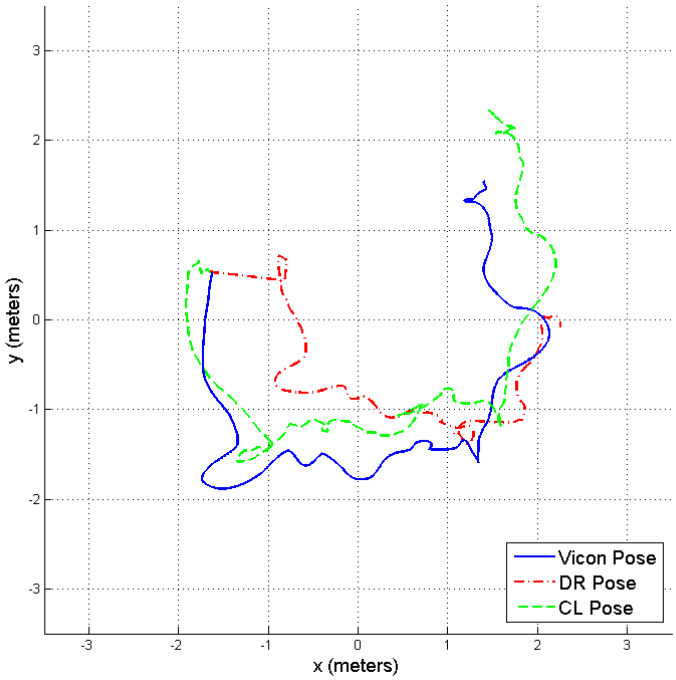



(b) 3-D View

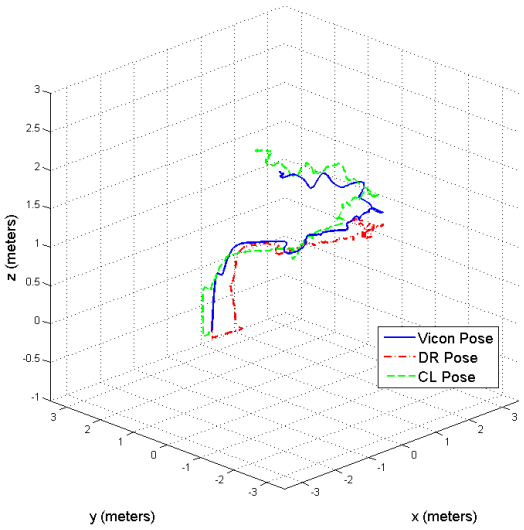



(c) Eye-level View

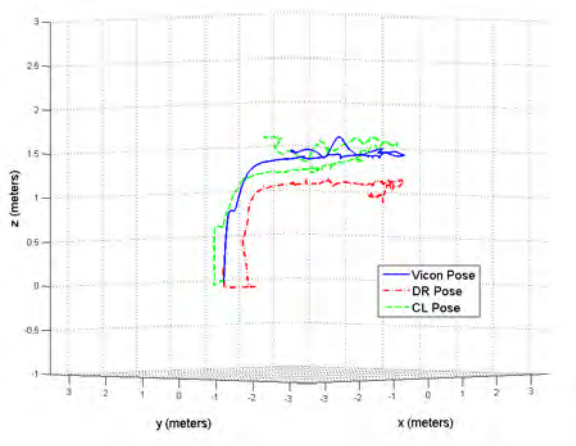
Figure 5.11: Position plot of Quad 2 conducting CL with a fully connected RPMG and continuous relative pose measurments between agents. The plot includes the trajectory as recorded by Vicon as well as the CL and DR pose estimates.




(a) X-Y View

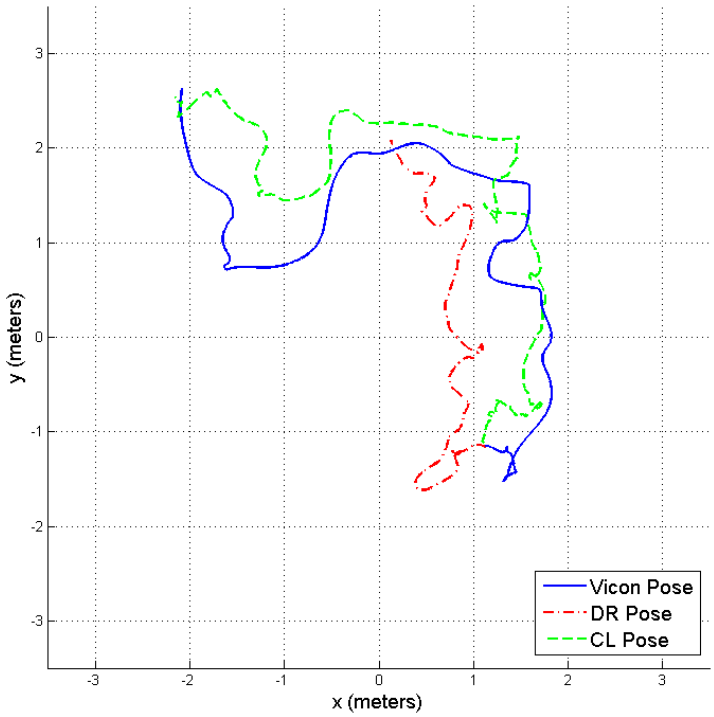



(b) 3-D View

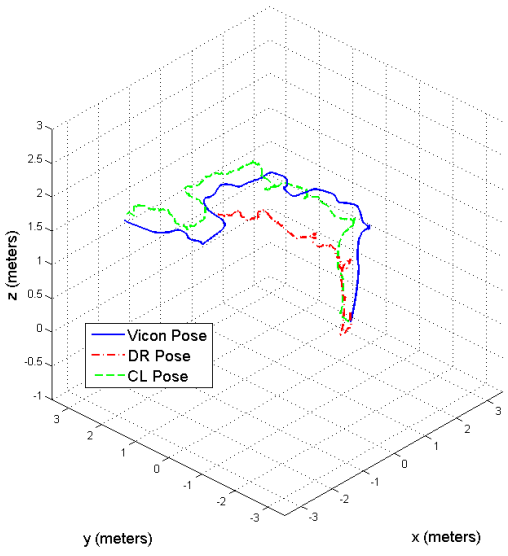



(c) Eye-level View

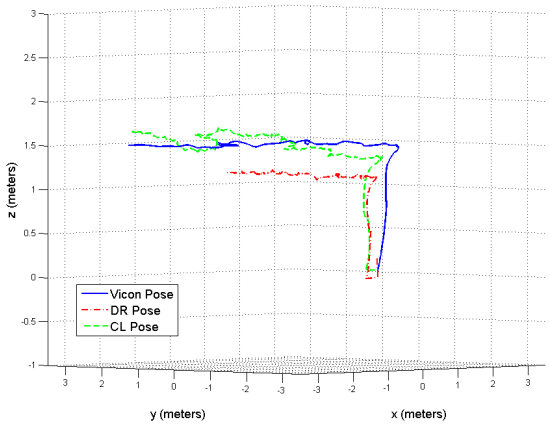
Figure 5.12: Position plot of Quad 3 conducting CL with a fully connected RPMG and continuous relative pose measurments between agents. The plot includes the trajectory as recorded by Vicon as well as the CL and DR pose estimates.




(a) X-Y View




(b) 3-D View




(c) Eye-level View

Figure 5.13: Position plot of Quad 7 conducting CL with a fully connected RPMG and continuous relative pose measurments between agents. The plot includes the trajectory as recorded by Vicon as well as the CL and DR pose estimates.

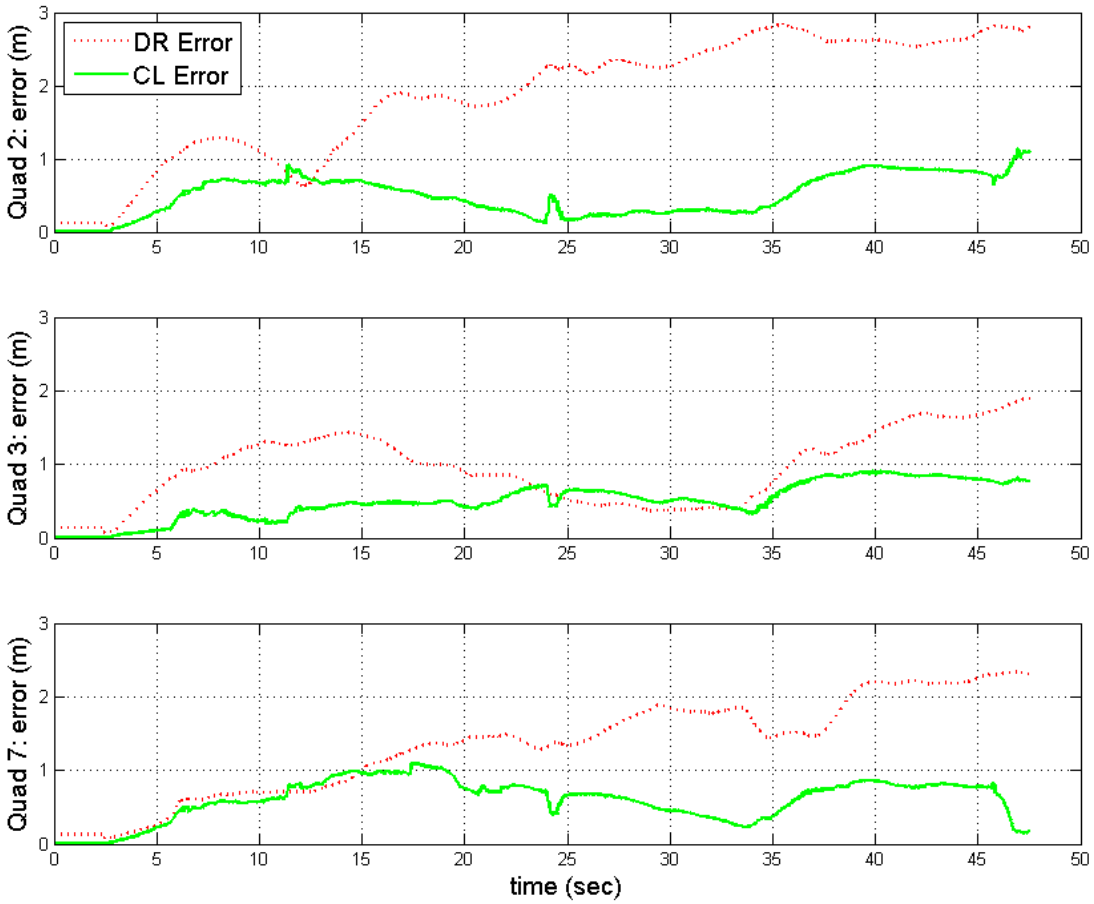



Figure 5.14: CL and DR pose errors plotted using CL with a fully connected RPMG and continuous relative pose measurements.

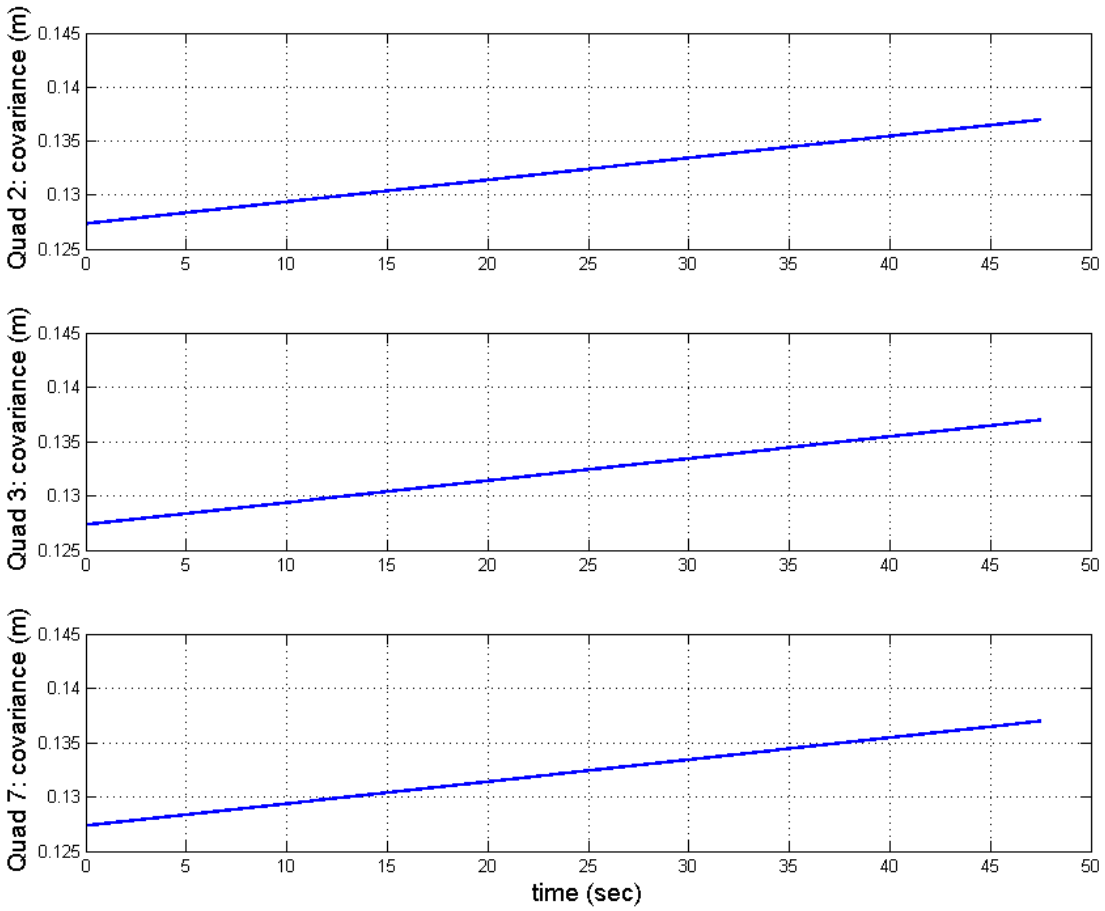



Figure 5.15: Covariance of all agents conducting CL with a fully connected RPMG and continuous relative pose measurements.

| Connectivity | Quad ID | Max Error (m) | Avg Error (m) | Min Uncertainty (m) | Avg Uncertainty Rate (mm/s) |
|---|---|---|---|---|---|
| Single Measurement | Quad 2 | 1.48 | 1.11 | 0.302 | 0.4 |
| | Quad 3 | 1.63 | 1.05 | 0.302 | 0.4 |
| | Quad 7 | 1.88 | 0.89 | 0.802 | 102.9 |
| Strong | Quad 2 | 1.40 | 0.57 | 0.127 | 0.2 |
| | Quad 3 | 1.40 | 0.67 | 0.127 | 0.2 |
| | Quad 7 | 1.26 | 0.66 | 0.127 | 0.2 |
| Full | Quad 2 | 1.14 | 0.5 | 0.127 | 0.2 |
| | Quad 3 | 0.91 | 0.5 | 0.127 | 0.2 |
| | Quad 7 | 1.10 | 0.6 | 0.127 | 0.2 |

Table 5.1: Localization performance summary of each agent conducting CL with varied connectedness of the RPMG. Note the average uncertainty rate is in mm/s due to the small resulting values.

## 5.2 Continuous Pose Measurements and a Fully Connected RPMG

For this test, consistency of algorithm performance is demonstrated by applying CL with a fully connected RPMG to multiple flights. The algorithm was run on four additional flights using the same AR Drones starting from the same location. The error and covariance plots for one of these flights are shown in Figures 5.16 and 5.17 respectively. Generally, the same average errors and uncertainty in position estimates were observed, however one flight showed a significant departure in performance. Figure 5.18 shows the Vicon trajectory of each agent as well as the CL and DR estimates at an X-Y view for this flight. The error and covariance plots are shown in Figures 5.19 and 5.20. When reviewing the recorded data, the measured velocities were either very noisy (as indicated by the erratic and unpredictable spikes in the velocity data) or ROS messages that contained the velocity data were either delayed or not being published to the `estPose/imu` node.

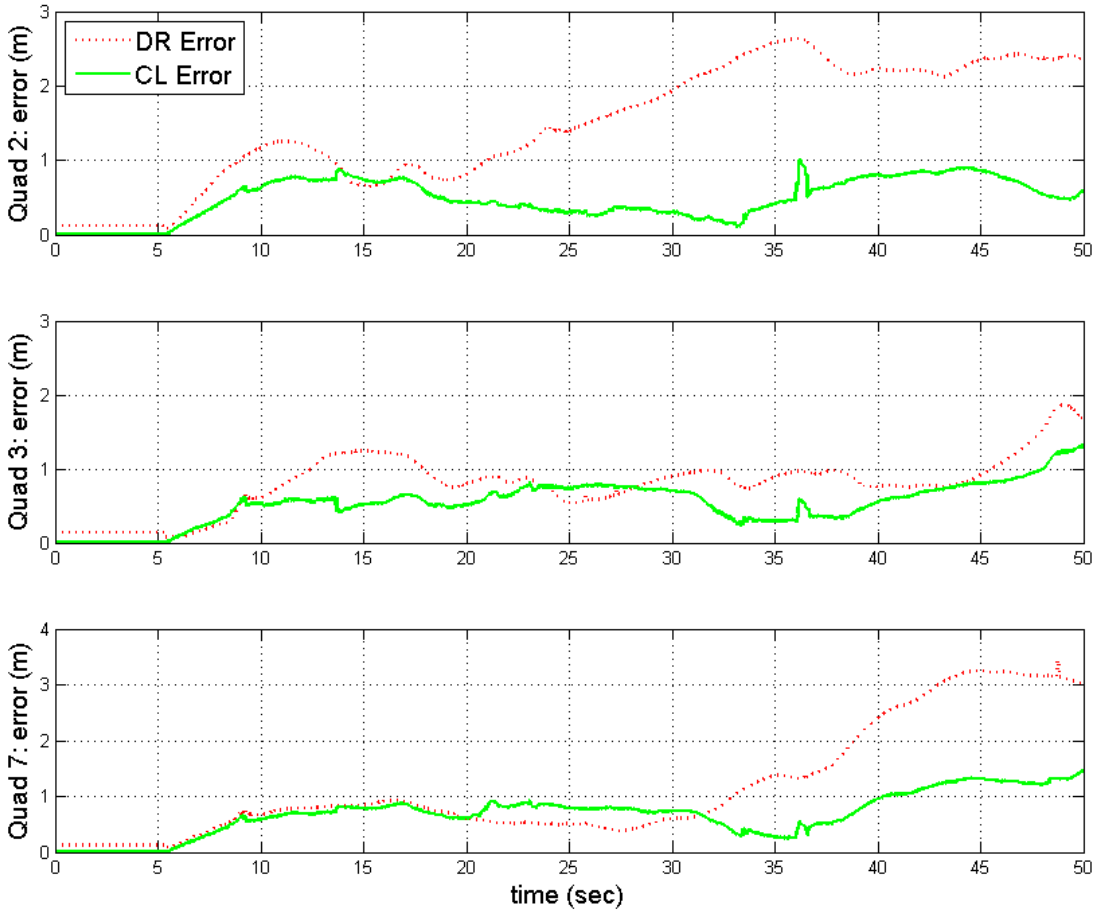



Figure 5.16: CL and DR pose errors of all three agents using CL with a fully connected RPMG and continuous relative pose measurements during the second flight.

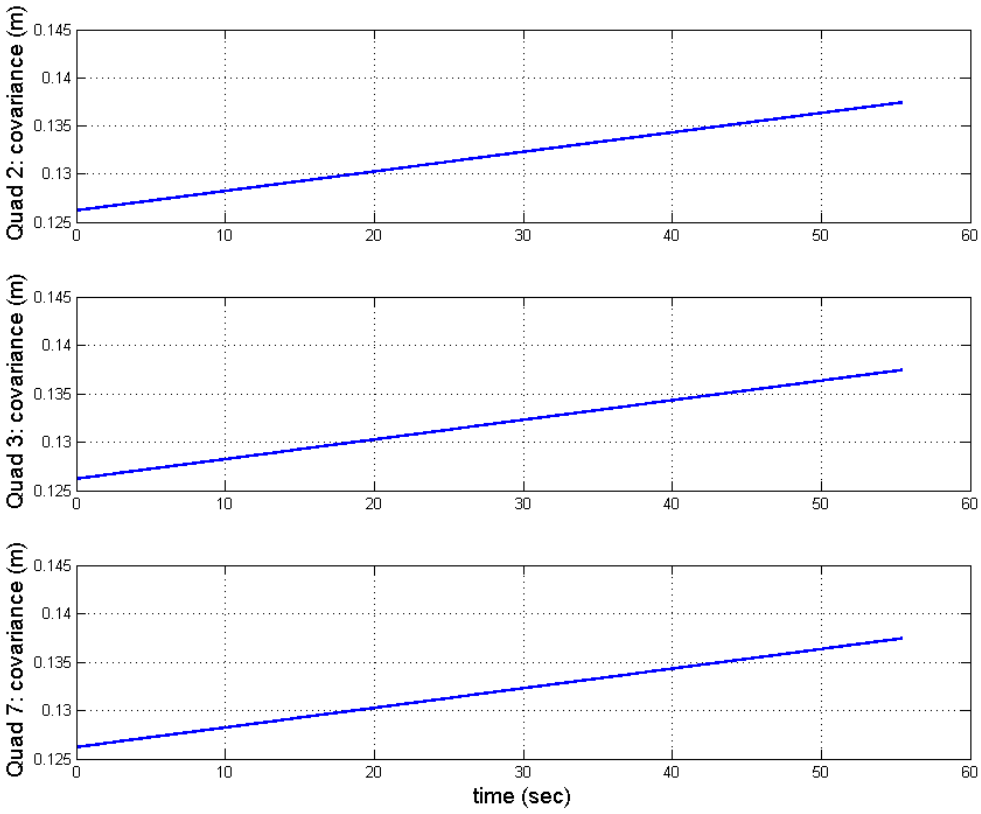
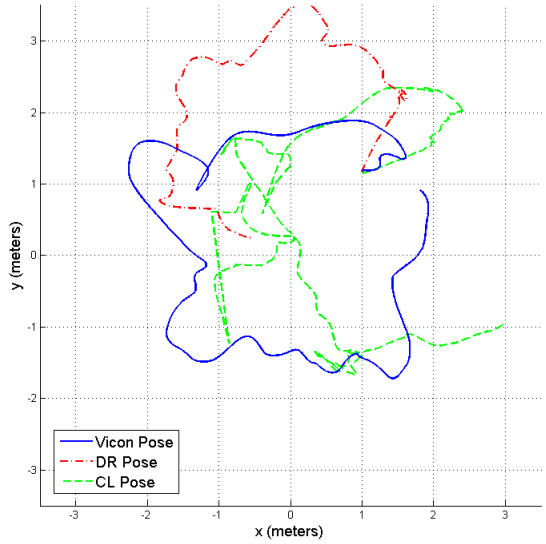
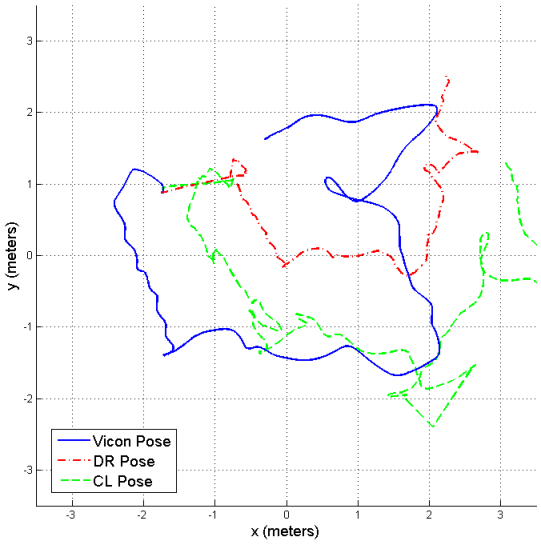


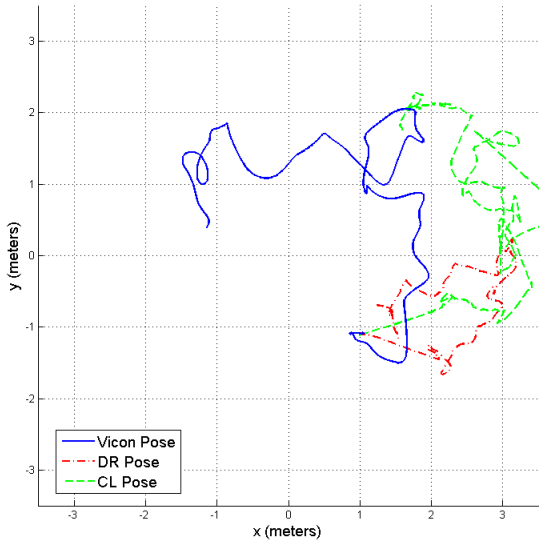

Figure 5.17: Agent covariance of all three agents using CL with a fully connected RPMG and continuous relative pose measurements during the second flight.




(a) Quad 2




(b) Quad 3




(c) Quad 7

Figure 5.18: Position plot of all quadrotors from the fourth flight of CL tests in an X-Y view. The plot includes the trajectory as recorded by Vicon as well as the CL and DR pose estimates with a fully connected RPMG.

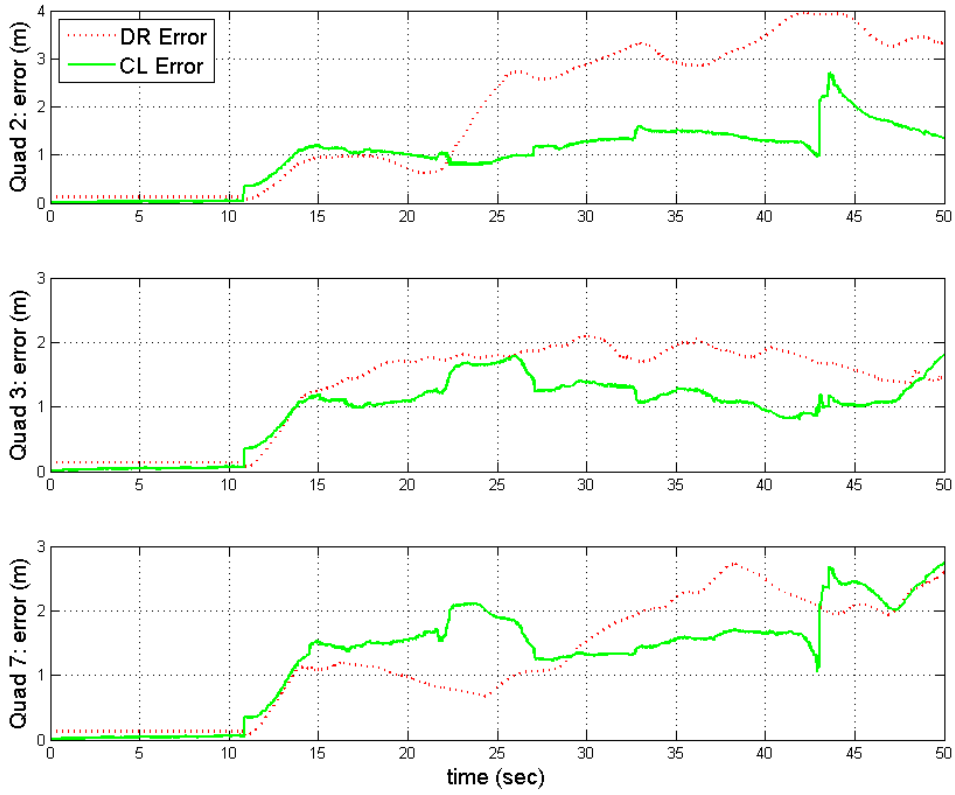



Figure 5.19: CL and DR pose errors of all three agents using CL with a fully connected RPMG and continuous relative pose measurements during the fourth flight.

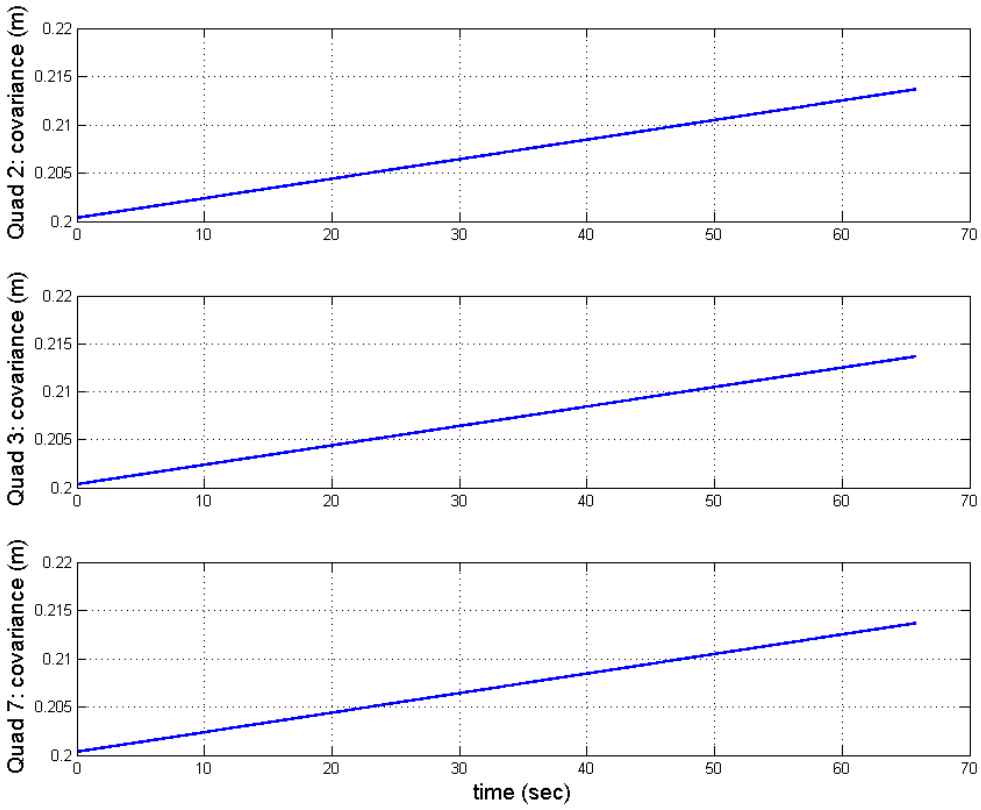



Figure 5.20: Agent covariance of all three agents using CL with a fully connected RPMG and continuous relative pose measurements during the fourth flight.

# CHAPTER 6:
# Conclusions and Recommendations

## 6.1 Conclusions

The purpose of this thesis is to explore the use of Cooperative Localization (CL) in a group of UAVs by modifying and implementing an Extended Kalman Filter (EKF) where the agents' on-board IMU provides velocity measurements for the propagation step, and relative position measurements from agents with the swarm are used for the update step. Using the distributed approach presented in [20], the special cases presented are first generalized as needed so that the equations can be applied to all agents within the group as well as all relative pose measurements between the agents. The motion model presented is also replaced with an aerial vehicle model which includes lateral velocity (for quadrotors), vertical velocity, and altitude. Upon modifying the EKF approach to CL, correct operation of the algorithm is demonstrated in a centralized system using AR Drones and ROS. The uncertainty in measurements react consistently with the connectedness of the RPMG and overall this approach provides a better estimate of pose over using dead reckoning. However, the performance of the algorithm does not provide the expected results after review of the performance data from [20]. The accuracy of the IMU on-board the AR Drone is presumed to be overestimated, and the characterization of noise in the velocity measurements may be inaccurate. It is also predicted that significant noise from the IMU as well as delayed or missing velocity messages from the quadrotors also contribute to a poor prediction of the agent state in the propagation step of the Kalman filter. The AR Drone is used as a low-cost solution to show proof-of-concept and it did aid in demonstrating correct operation of the proposed CL algorithm. Use of a better IMU may have provided better results.

## 6.2 Recommendations

The EKF approach to CL as described in this thesis is possible for use by swarm UAVs and is a recommended approach for enhancing positional accuracy of agents within a group. This approach not only has the potential to enhance the positional accuracy over use of a standard GPS but can also provide accurate positional information when encountering an intermittent or denied GPS environment. Further study into the application of this ap-

proach with use in formation control and in conjunction with other absolute positioning sensors such as GPS is recommended. Areas of focus should center on the challenges associated with computational resources on-board the UAV and communication between agents. The AR Drone is a great low-cost option to conduct swarm research; however, the platform's performance is limited. For experimentation with higher level autonomous features that require more computational power and higher precision from its navigational sensors, modification to the AR Drone or use of more capable platforms may be required. These modifications are possible and have been shown in [63]. Further examination into possible modifications to the AR Drone that further build upon its strengths for the research community is recommended.

## 6.3 Future Work

In this thesis, the distributed approach to CL using an EKF is implemented in a centralized way. It may be beneficial to code, test, and implement the centralized approach as described in [20]. This would help validate the approach made by Roumeliotis and Bekey, and ensure the performance observed in this implementation is not due to any errors in their processes. Future work would also involve exploring and re-evaluating the performance of the IMU on-board the AR Drone. Re-examining the accuracy of the IMU and determining a more precise model for noise in the sensor may help determine if the CL approach in this thesis can be "tuned" for optimal performance or if a better IMU is required.

In this thesis, the performance of the CL algorithm with continuous relative pose measurements is examined. To further observe the power of CL, it may be beneficial to run this implementation of CL when a random number of measurements are made between agents (frequency), when measurements are randomly spread over time (intermittency), and when more noise is added to the measurements (robustness). It is also proposed to introduce absolute positioning to verify the algorithm distributes the absolute position across the group. An extension of this test is to allow only one agent within a group to receive initial pose information when CL starts. A natural extension of this work is to add more agents to the swarm to ensure scalability and to examine the absolute communication requirements to conduct our CL algorithm. It would also benefit the future implementation of this algorithm by examining a means to identify and measure relative position between agents using on-board sensors (e.g., the video cameras) rather than relying on Vicon. Finally, the

AR Drone has limited to no on-board processing power for programming additional autonomous features and functions. In this project, CL is demonstrated in a centralized way. Future work should include re-creating Roumeliotis and Bekey's work by implementing their decentralized approach.

Results of this work have established a framework and associated infrastructure in which other CL algorithms may be implemented for the AR Drone or other quadrotors. A single node for ROS that contains the algorithm as well as any additional environmental measurements needed for its operation are the only requirements. The stage is therefore set for implementing and examining other CL techniques. When a suitable approach is determined, future work should also involve examining its use in autonomous formation control such that agents within a formation can maintain an optimal position for specific phases of flight.

THIS PAGE INTENTIONALLY LEFT BLANK

# List of References


[1] J. Gertler, "U.S. unmanned aerial systems," Congressional Research Service, Washington, DC, CRS Report No. R42136, Jan. 2012. [Online]. Available: http://www.fas.org/sgp/crs/natsec/R42136.pdf

[2] S. Edwards, *Swarming on the Battlefield: Past, Present, and Future*. Santa Monica, CA: National Defense Research Institute, 2000.

[3] B. Cobb, "Adaptive discrete event simulation for analysis of Harpy swarm attack," M.S. thesis, Dept. Operations Research, Naval Postgraduate School, Monterey, CA, 2011.

[4] O. O'Loan and M. Evans, "Alternating steady state in one-dimensional flocking," *Journal of Physics A: Mathematical and General*, vol. 32, no. 8, p. L99, 1999. [Online]. Available: http://iopscience.iop.org/0305-4470/32/8/002

[5] E. Bonabeau and G. Théraulaz, "Swarm smarts." *Scientific American*, vol. 282, no. 3, pp. 72–79, 2000.

[6] T. D. Wolf and T. Holvoet, "Emergence and self-organisation: a statement of similarities and differences," *Engineering Self-Organising Systems*, vol. 3464, pp. 1–15, 2004. [Online]. Available: https://trac.v2.nl/export/7711/rui/projects/UnleashCulture/UnleashCulture3/EmergenceandSelf-Organisation,similaritiesanddifferences.pdf

[7] C. Reynolds, "Flocks, herds, and schools: a distributed behavioral model," *Computer Graphics*, vol. 21, no. 4, pp. 25–34, 1987. [Online]. Available: http://www.macs.hw.ac.uk/~dwcorne/Teaching/CraigReynoldsFlocks,Herds,andSchoolsADistributedBehavioralModel.htm

[8] R. Mach and F. Schweitzer, "Multi-agent model of biological swarming," in *Advances in Artificial Life*. Springer, 2003, pp. 810–820.

[9] M. Dorigo and M. Birattari, "Swarm intelligence," *Scholarpedia*, vol. 2, no. 9, p. 1462, Sep. 2007. [Online]. Available: http://www.scholarpedia.org/article/Swarm_intelligence

[10] H. V. D. Parunak, "Making swarming happen," in *Proceedings of Swarming and Network-Enabled C4ISR*. Citeseer, 2003. [Online]. Available: http://citeseerx.ist.psu.edu/viewdoc/download?doi=10.1.1.116.1990&rep=rep1&type=pdf

[11] J. Arquilla and D. Ronfeldt, "Swarming and the future of conflict," National Defense Reserach Institute, 2000. [Online]. Available: http://oai.dtic.mil/oai/oai?verb=getRecord&metadataPrefix=html&identifier=ADA384989

[12] B. Kempinski and D. Arthur, "Policy options for unmanned aircraft systems," Congressional Budget Office, Washington, DC, CBO Pub. No. 4083, 2011.

[13] S. Roumeliotis and I. Rekleitis, "Analysis of multirobot localization uncertainty propagation," in *Proceedings 2003 IEEE/RSJ International Conference on Intelligent Robots and Systems (Cat. No.03CH37453)*, vol. 2., IEEE, 2003, pp. 1763–1770. [Online]. Available: http://ieeexplore.ieee.org/lpdocs/epic03/wrapper.htm?arnumber=1248899

[14] A. Mourikis and S. Roumeliotis, "Performance analysis of multirobot cooperative localization," *IEEE Transactions on Robotics*, vol. 22, no. 4, pp. 666–681, 2006. [Online]. Available: http://ieeexplore.ieee.org/xpls/abs_all.jsp?arnumber=1668252

[15] Z. Mahboubi, Z. Kolter, and T. Wang, "Camera based localization for autonomous UAV formation flight," in *Proceedings of the AIAA@ Infotech Conference*, 2011. [Online]. Available: http://arc.aiaa.org/doi/pdf/10.2514/6.2011-1658

[16] I. Rekleitis, G. Dudek, and E. Milios, "Multi-robot cooperative localization: a study of trade-offs between efficiency and accuracy," *IEEE/RSJ International Conference on Intelligent Robots and Systems*, vol. 3, pp. 2690–2695, 2002. [Online]. Available: http://ieeexplore.ieee.org/xpls/abs_all.jsp?arnumber=1041676

[17] S. Roumeliotis and G. Bekey, "Collective localization: a distributed Kalman filter approach to localization of groups of mobile robots," *IEEE International Conference on Robotics and Automation*, vol. 3, pp. 2958–2965, 2000. [Online]. Available: http://ieeexplore.ieee.org/xpls/abs_all.jsp?arnumber=846477

[18] S. Roumeliotis and I. Rekleitis, "Propagation of uncertainty in cooperative multirobot localization: analysis and experimental results," *Autonomous Robots*, pp. 41–54, 2004. [Online]. Available: http://link.springer.com/article/10.1023/B:AURO.0000032937.98087.91

[19] G. Chen and L. Guo, "The FPGA implementation of Kalman filter," in *Proceedings of the 5th WSEAS International Conference on Signal Processing, Computational Geometry and Artificial Vision*, 2005, pp. 61–65. [Online]. Available: http://web.cecs.pdx.edu/~mperkows/temp/KALMAN/KalmanFilter.pdf

[20] S. Roumeliotis and G. Bekey, "Distributed multirobot localization," *IEEE Transactions on Robotics and Automation*, vol. 18, no. 5, pp. 781–795, 2002. [Online]. Available: http://ieeexplore.ieee.org/xpls/abs_all.jsp?arnumber=1067998

[21] R. West. (1998). *What is GPS?* [Online]. Available: http://www.eso.org/public/outreach/eduoff/seaspace/docs/navigation/navgps/navgps-1.html

[22] P. Dana. (1999). *Global Positioning System Overview*. [Online]. Available: http://www.colorado.edu/geography/gcraft/notes/gps/gps_f.html

[23] S. C. Stephens and V. P. Rasmussen, "High-End DGPS and RTK systems," NASA and the existing Land Grant Cooperative Extension System, Utah State Univ., Logan, UT, Mar. 2010. [Online]. Available: http://extension.usu.edu/nasa/files/uploads/gtk-tuts/rtk_dgps.pdf

[24] M. Eren, S. Davison, G. Schmidt, and J. Niemela, "Basic Guide to Advanced Navigation," Reserach and Technology Organization, North Atlantic Treaty Organization, RTO Publication SET-054/RTG-30, Dec. 2003.

[25] N. Barbour, J. M. Elwell, and R. H. Setterlund, "Inertial instruments: where to now?" in *Proceedings of the AIAA Guidance and Control Conference*, 1992, pp. 566–574. [Online]. Available: http://arc.aiaa.org/doi/pdf/10.2514/6.1992-4414

[26] O. J. Woodman, "An introduction to inertial navigation," *University of Cambridge, Computer Laboratory, Tech. Rep. UCAMCL-TR-696*, vol. 14, p. 15, 2007. [Online]. Available: http://triplanets.com/sites/default/files/UCAM-CL-TR-696.pdf

[27] J. P. Golden, "Terrain contour matching (TERCOM): a cruise missle guidance aid," *SPIE 24th Annual Technical Symposium*, pp. 10–18, 1980.

[28] D. Sim and R. Park, "Localization based on the gradient information for DEM matching." *IAPR Workshop on Machine Vision Applications*, pp. 266–269, 1998. [Online]. Available: http://citeseerx.ist.psu.edu/viewdoc/download?doi=10.1.1.217.3375&rep=rep1&type=pdf

[29] G. Conte and P. Doherty, "An integrated UAV navigation system based on aerial image matching," *2008 IEEE Aerospace Conference*, pp. 1–10, Mar. 2008. [Online]. Available: http://ieeexplore.ieee.org/lpdocs/epic03/wrapper.htm?arnumber=4526556

[30] O. Amidi, "An autonomous vision-guided helicopter," Ph.D. dissertation, Dept. Electrical and Computer Engineering, Carnegie Mellon Univ., Pittsburgh, PA, 1996. [Online]. Available: http://citeseerx.ist.psu.edu/viewdoc/download?doi=10.1.1.13.2210&rep=rep1&type=pdf

[31] G. L. Barrows, L. C. Nw, J. S. Chahl, and M. V. Srinivasan, "Biomimetic visual sensing and flight control," in *Proc. Bristol UAV Conf*, 2002, pp. 159–168.

[32] F. Raudies, "Optic flow," *Scholarpedia*, vol. 8, no. 7, p. 30724, Jul. 2013. [Online]. Available: http://www.scholarpedia.org/article/Optic_flow

[33] J. Barron, D. Fleet, and S. Beauchemin, "Performance of optical flow techniques," *International Journal of Computer Vision*, vol. 12, no. 1, pp. 43–77, 1994. [Online]. Available: http://link.springer.com/article/10.1007/BF01420984

[34] S. Baker, D. Scharstein, J. P. Lewis, S. Roth, M. J. Black, and R. Szeliski, "A database and evaluation methodology for optical flow," *International Journal of Computer Vision*, vol. 92, no. 1, pp. 1–31, Nov. 2010. [Online]. Available: http://link.springer.com/10.1007/s11263-010-0390-2

[35] S. Chhaniyara, "Optical flow algorithm for velocity estimation of ground vehicles: a feasibility study," *International Journal on Smart Sensing and Intelligent Systems*, vol. 1, no. 1, pp. 246–268, 2008. [Online]. Available: http://www-ist.massey.ac.nz/s2is/Issues/v1/n1/papers/paper15.pdf

[36] S. Riisgaard and M. Blas, "SLAM for dummies," *A Tutorial Approach to Simultaneous Localization and Mapping*, pp. 1–127, 2003. [Online]. Available: http://burt.googlecode.com/svn-history/r106/trunk/1aslam_blas_repo.pdf

[37] H. Durrant-Whyte and T. Bailey, "Simultaneous localization and mapping: part I," *IEEE Robotics & Automation Magazine*, vol. 13, no. 2, pp. 99–110, 2006. [Online]. Available: http://ieeexplore.ieee.org/xpls/abs_all.jsp?arnumber=1638022

[38] J. Kim and S. Sukkarieh, "Autonomous airborne navigation in unknown terrain environments," *IEEE Transactions on Aerospace and Electronic Systems*, vol. 40, no. 3, pp. 1031–1045, Jul. 2004. [Online]. Available: http://ieeexplore.ieee.org/lpdocs/epic03/wrapper.htm?arnumber=1337472

[39] J. Guivant and E. Nebot, "Optimization of the simultaneous localization and map-building algorithm for real-time implementation," *IEEE Transactions on Robotics and Automation*, vol. 17, no. 3, pp. 242–257, Jun. 2001. [Online]. Available: http://ieeexplore.ieee.org/xpls/abs_all.jsp?arnumber=938382http://ieeexplore.ieee.org/lpdocs/epic03/wrapper.htm?arnumber=938382

[40] S. Thrun, D. Fox, W. Burgard, and F. Dellaert, "Robust monte carlo localization for mobile robots," *Artificial intelligence*, vol. 128, no. 1, pp. 99–141, 2001. [Online]. Available: http://www.sciencedirect.com/science/article/pii/S0004370201000698

[41] I. Rekleitis, "A particle filter tutorial for mobile robot localization," *Centre for Intelligent Machines, McGill University, Tech. Rep. TR-CIM-04-02*, 2004. [Online]. Available: ftp://202.191.56.69/vinhlt/Papers/MustReadPapers/ParticleFilterTutorial.pdf

[42] D. Fox, W. Burgard, F. Dellaert, and S. Thrun, "Monte carlo localization: efficient position estimation for mobile robots," *AAAI/IAAI*, vol. 1999, pp. 343–349, 1999. [Online]. Available: http://www.aaai.org/Papers/AAAI/1999/AAAI99-050.pdf

[43] D. Fox, W. Burgard, H. Kruppa, and S. Thrun, "A probabilistic approach to collaborative multi-robot localization," *Autonomous robots*, vol. 8, no. 3, pp. 325–344, 2000. [Online]. Available: http://link.springer.com/article/10.1023/A:1008937911390

[44] G. Welch and G. Bishop, "An introduction to the Kalman filter," pp. 1–16, 1995. [Online]. Available: http://clubs.ens-cachan.fr/krobot/old/data/positionnement/kalman.pdf

[45] P. S. Maybeck, J. J. D'Azzo, C. H. Houpis, H. B. Kepler, V. Rehg, C. T. Triscari Jr, M. R. W. Hitzelberger, D. G. Shankland, M. T. C. Harrington, W. A. Fischer *et al.*, "Stochastic models, estimating, and control," *New Directions in Effective Management*, vol. 8, pp. 12–14, 1982.

[46] K. Leung, "Cooperative localization and mapping in sparsely-communicating robot networks," Ph.D. dissertation, Graduate Dept. Aerospace Science and Engineering, Univ. of Toronto, Toronto, ON, Canada, 2012.

[47] N. Patwari and J. Ash, "Locating the nodes: cooperative localization in wireless sensor networks," *IEEE Signal Processing Magazine*, vol. 22, no. 4, pp. 54–69, 2005. [Online]. Available: http://ieeexplore.ieee.org/xpls/abs_all.jsp?arnumber=1458287

[48] R. Kurazume, S. Nagata, and S. Hirose, "Cooperative positioning with multiple robots," in *Proceedings, IEEE International Conference on Robotics and Automation*. [Online]. Available: http://ieeexplore.ieee.org/xpls/abs_all.jsp?arnumber=351315

[49] I. Rekleitis, G. Dudek, and E. Milios, "Multi-robot collaboration for robust exploration," *Annals of Mathematics and Artificial Intelligence*, vol. 31, no. 1-4, pp. 7–40, 2001. [Online]. Available: http://link.springer.com/article/10.1023/A:1016636024246

[50] A. Sanderson, "A distributed algorithm for cooperative navigation among multiple mobile robots," *Advanced Robotics*, 1997. [Online]. Available: http://www.tandfonline.com/doi/abs/10.1163/156855398X00235

[51] R. Madhavan, K. Fregene, and L. Parker, "Distributed cooperative outdoor multirobot localization and mapping," *Autonomous Robots*, vol. 17, no. 1, pp. 23–39, Jul. 2004. [Online]. Available:

http://link.springer.com/10.1023/B:AURO.0000032936.24187.41http://link.springer.com/article/10.1023/B:AURO.0000032936.24187.41

[52] A. Howard, M. Matarić, and G. Sukhatme, "Localization for mobile robot teams: a distributed MLE approach," *Experimental Robotics VIII*, pp. 146–155, 2003. [Online]. Available: http://link.springer.com/chapter/10.1007/3-540-36268-1_12

[53] E. Nerurkar, S. Roumeliotis, and a. Martinelli, "Distributed maximum a posteriori estimation for multi-robot cooperative localization," *2009 IEEE International Conference on Robotics and Automation*, pp. 1402–1409, May 2009. [Online]. Available: http://ieeexplore.ieee.org/lpdocs/epic03/wrapper.htm?arnumber=5152398

[54] I. Amundson and X. D. Koutsoukos, "A survey on localization for mobile wireless sensor networks," in *Mobile Entity Localization and Tracking in GPS-less Environnments*. Springer, 2009, pp. 235–254. [Online]. Available: http://link.springer.com/chapter/10.1007/978-3-642-04385-7_16

[55] H. Wymeersch, J. Lien, and M. Z. Win, "Cooperative localization in wireless networks," *Proceedings of the IEEE*, vol. 97, no. 2, pp. 427–450, Feb. 2009. [Online]. Available: http://ieeexplore.ieee.org/lpdocs/epic03/wrapper.htm?arnumber=4802193

[56] S. Piskorski, N. Brulez, P. Eline, and F. D'Haeyer, "AR. Drone Developer Guide," 2012.

[57] J. Lugo and A. Zeil, "Framework for autonomous onboard navigation with the ar. drone," in *International Conference on Unmanned Aircraft Systems (ICUAS)*. IEEE, 2013, pp. 575–583. [Online]. Available: http://ieeexplore.ieee.org/xpls/abs_all.jsp?arnumber=6564735

[58] P. Bristeau, F. Callou, D. Vissière, and N. Petit, "The navigation and control technology inside the ar. drone micro uav," in *18th IFAC world congress*, vol. 18, no. 1, 2011, pp. 1477–1484. [Online]. Available: http://cas.ensmp.fr/~petit/papers/ifac11/PJB.pdf

[59] *Vicon | Homepage*. [Online]. Available: http://www.vicon.com/

[60] M. Hamer. (2012). *AutonomyLab/ardrone_autonomy · GitHub*. [Online]. Available: https://github.com/AutonomyLab/ardrone_autonomy

[61] M. Achtelik. (2011). *ros-drivers/vicon_bridge at master · ethz-asl/ros-drivers · GitHub*. [Online]. Available: https://github.com/ethz-asl/ros-drivers/tree/master/vicon_bridge

[62] *ROS.org | Powering the world's robots*. [Online]. Available: http://www.ros.org/

[63] D. G. Kottas. (2013). *Autonomous (Camera & IMU based) localization for a Parrot AR.DRONE Quadrotor*. [Online]. Available: http://diydrones.com/profiles/blogs/autonomous-camera-imu-based-localization-for-a-ar-drone-quadrotor

THIS PAGE INTENTIONALLY LEFT BLANK

# Initial Distribution List

1. Defense Technical Information Center
   Ft. Belvoir, Virginia

2. Dudley Knox Library
   Naval Postgraduate School
   Monterey, California