A Vision-Based Relative Navigation Approach for Autonomous Multirotor Aircraft

Robert C. Leishman
Brigham Young University - Provo

A Vision-Based Relative Navigation Approach

for Autonomous Multirotor Aircraft

Robert C. Leishman

A dissertation submitted to the faculty of
Brigham Young University
in partial fulfillment of the requirements for the degree of

Doctor of Philosophy

Department of Mechanical Engineering

Brigham Young University

April 2013

ABSTRACT

A Vision-Based Relative Navigation Approach
for Autonomous Multirotor Aircraft

Robert C. Leishman
Department of Mechanical Engineering, BYU
Doctor of Philosophy

Autonomous flight in unstructured, confined, and unknown GPS-denied environments is a challenging problem. Solutions could be tremendously beneficial for scenarios that require information about areas that are difficult to access and that present a great amount of risk. The goal of this research is to develop a new framework that enables improved solutions to this problem and to validate the approach with experiments using a hardware prototype.

In Chapter 2 we examine the consequences and practical aspects of using an improved dynamic model for multirotor state estimation, using only IMU measurements. The improved model correctly explains the measurements available from the accelerometers on a multirotor. We provide hardware results demonstrating the improved attitude, velocity and even position estimates that can be achieved through the use of this model.

We propose a new architecture to simplify some of the challenges that constrain GPS-denied aerial flight in Chapter 3. At the core, the approach combines visual graph-SLAM with a multiplicative extended Kalman filter (MEKF). More importantly, we depart from the common practice of estimating global states and instead keep the position and yaw states of the MEKF *relative* to the current node in the map. This relative navigation approach provides a tremendous benefit compared to maintaining estimates with respect to a single global coordinate frame. We discuss the architecture of this new system and provide important details for each component. We verify the approach with goal-directed autonomous flight-test results.

The MEKF is the basis of the new relative navigation approach and is detailed in Chapter 4. We derive the relative filter and show how the states must be augmented and marginalized each time a new node is declared. The relative estimation approach is verified using hardware flight test results accompanied by comparisons to motion capture truth. Additionally, flight results with estimates in the control loop are provided.

We believe that the relative, vision-based framework described in this work is an important step in furthering the capabilities of indoor aerial navigation in confined, unknown environments. Current approaches incur challenging problems by requiring globally referenced states. Utilizing a relative approach allows more flexibility as the critical, real-time processes of localization and control do not depend on computationally-demanding optimization and loop-closure processes.

Keywords: GPS-denied flight, relative navigation, sensor fusion, quadrotor dynamics, MEKF

ACKNOWLEDGMENTS

I feel privileged to have worked in the MAGICC Lab during the past four years. The lab is a great environment that fosters learning and discovery. I appreciate all the fruitful discussions and the patient instructions provided by my fellow “magiccians”; in particular John Macdonald, Peter Niedfeldt, Joe Nichols, Jeff Ferrin, and Caleb Chamberlain were very helpful over the years. Thanks for the fun times and for being patient sounding boards for my thoughts! I would also like to thank Dr. McLain for all of his help and patience. He has been a great advisor and mentor to me and has provided excellent guidance and wisdom. I especially appreciate all of his help with my writing.

I am extremely grateful to my wonderful and supportive wife, Lisa. She has been very helpful and patient with me during all of my schooling. I hope that one day I can be as supportive to her in following her dreams as she always has been for me. I am also appreciative of the love, patience, and support of my four boys: James, Michael, Joseph, and Adam. I cannot wait until I can start to work with them on some of the things I have learned in completing this degree.

I am indebted to the SMART Scholarship and the USAF Research Laboratory for providing me the opportunity to focus on my school work and support my family at the same time. I really have had the best scholarship in the world! In particular, I am grateful to Jesse Fanning for taking the gamble and hiring me without ever even meeting me, and to Tom Woodford for his support and patience. It cannot be easy to have an employee for four years that hardly ever shows up for work!

Finally, and most importantly, I am grateful to a loving Heavenly Father for providing me the talents and opportunities that I have been given. I am extremely grateful for the quiet whisperings of His Spirit that have come to prompt me forward, enlighten my mind, and enlarge my understanding.

TABLE OF CONTENTS

LIST OF TABLES

LIST OF FIGURES

CHAPTER 1. INTRODUCTION

A system capable of completing autonomous flight in unstructured, confined, and unknown GPS-denied environments would provide a great deal of value for inspecting and surveying areas that are difficult to access and that present a great amount of risk. Autonomous capability would allow the vehicle to intelligently explore an environment and avoid obstacles, permitting personnel to concentrate on the information returned by the vehicle rather than on maneuvering manually. It would enhance search and rescue operations in damaged buildings after disasters or in underground areas. It would also be valuable during reconnaissance operations in areas with limited or intermittent GPS information, like urban canyons. The goal of this research is to develop a new framework that enables improved solutions to this problem and validate the approach with experiments using a hardware prototype.

Currently, most autonomous aerial vehicles relay heavily on GPS information. GPS provides a unique 3D global position measurement for a vehicle. The measurements are commonly fused with inertial measurement unit (IMU) data to provide improved estimates of a vehicles' location, attitude, and velocity. As was mentioned above, however, there are areas and conditions where GPS is unavailable. Current navigation solutions are wholly inadequate in these scenarios. This fact has been identified and lamented by the Chief of Staff of the Air Force, General Schwartz, who said "It seems critical, to me, that the Joint Force should reduce its dependence on GPS-aided precision navigation..." [1]. This statement has been made more relevant by recent events where Iran has claimed to have downed two of the United States' advanced unmanned aerial systems (UAS) through GPS spoofing [2, 3].

Robust navigation solutions for aerial robotics are similarly important for civilian missions. Figures 1.1 and 1.2 present environments where visual information of the environment, conditions of survivors, hazards, and appropriate ingress and egress points would prove invaluable to search and rescue operators. Additionally, solutions to provide surveillance in urban canyon areas, like those shown in Figure 1.3, would also need to be robust to at least intermittent GPS availability.

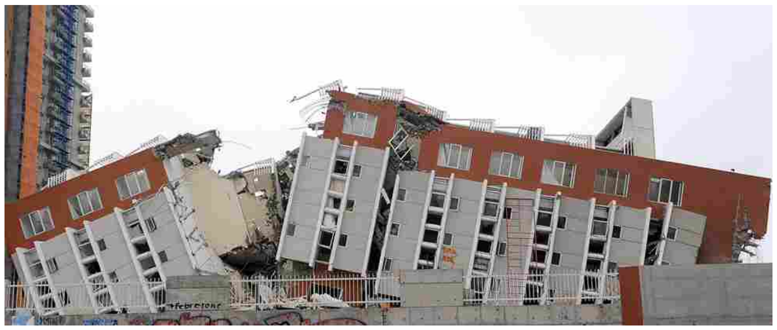

Figure 1.1: A building in Concepcion, Chile after the 2010 earthquake. The condition of the building would expose rescuers to a great amount of risk. The condition also offers a tremendous challenge in exploration. The walls could not be used as assumed structure for navigation (since they are no longer vertical), and the slanted floors would exclude a ground robot.

Figure 1.2: An underground environment which could possibly expose personnel to a great amount of risk. Similar to the building, there is also a tremendous challenge in autonomous exploration of this environment. No structure can be assumed and the low light precludes the use of vision.

Figure 1.1 presents a particularly challenging environment for autonomous robotic navigation, and exploration. The slant of the floor, walls, and ceiling would preclude the use of any assumptions regarding the environment in a potential solution. A vehicle would also be required

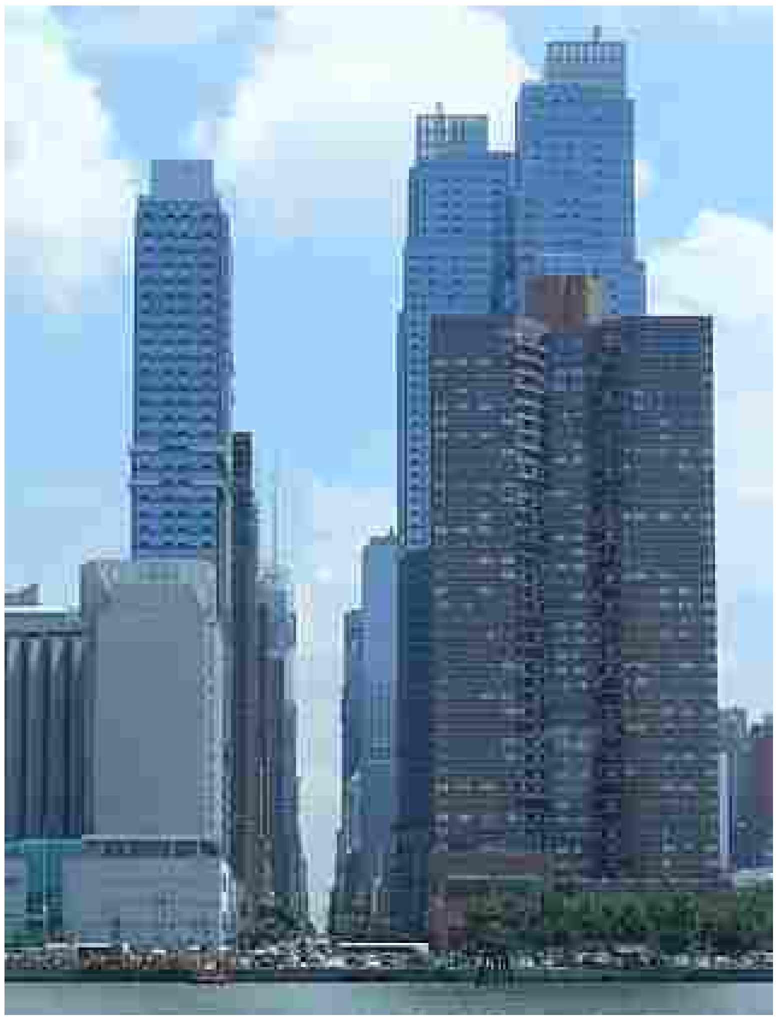

Figure 1.3: An example of an urban canyon. The tall buildings obstruct and create multi-path effects for GPS signals. Small UAS must be robust to degraded or intermittent GPS signals for flights in these areas. Photograph ©Ad Meskens, used with permission.

to plan paths in three dimensions (3D) to maneuver around obstacles that would be present inside. The slant of the floor would rule out the use of a ground robot as well. A highly maneuverable, autonomous aerial robot could provide a valid solution for investigating such an environment.

An underground environment, similar to that of Figure 1.2, would also be a challenge for autonomous vehicle navigation. There is not any structure in the environment that could be used to simplify a solution. Lighting issues would also present a challenge. Obstacles on the floor would severely restrict a ground robot implementation. Again, a maneuverable autonomous aerial vehicle would be a possible solution for investigating such an environment remotely.

Aerial surveillance of urban areas is highly desired by police agencies to reduce crime and to aid in providing additional perspectives during difficult situations. Upon changes to federal airspace restrictions, due in 2014, such missions may be more of a possibility. In urban canyon areas such as the one pictured in Figure 1.3, these aerial surveillance vehicles will be required to be robust to degradation or loss of GPS measurements. Additional sensors and algorithms will be required to replace GPS. Switching between GPS and onboard sensors will need to be dynamic and robust. There are not currently any solutions available that provide such capability.

Aerial flight without GPS information is a challenging problem as the robot must discover its own location using only onboard sensors and computational resources. The situation is further complicated by the need to control the vehicle's fast dynamics using the localized state estimates. Unlike ground robots, these vehicles cannot afford to pause in one place until complex algorithms converge and state estimates are sufficiently stable to continue. In this chapter we more fully introduce and discuss this problem. First, we present the aerial vehicle and the associated hardware that is used for this research in Section 1.1. Then we address the possibilities for precision navigation when GPS is unavailable in Section 1.2. The potential solutions are limited because of the assumption of an a priori unknown environment. In Section 1.3 we discuss the relevant research associated with vision-based GPS-denied navigation and then we finish this chapter with a brief overview of the contributions that this research brings to the problem in Section 1.4.

1.1 Multirotor Aircraft

Multirotor aircraft,[1] such as the quadrotor depicted in Figure 1.4 or the hexrotor in Figure 1.5, are ideal platforms for autonomous flight in unknown and complex environments. Their small size and high maneuverability are conducive to operating in confined spaces and avoiding obstacles. Equipped with appropriate sensors and algorithms, they could enable several applications currently infeasible for ground robots, such as the autonomous exploration of the environments in Figures 1.1, 1.2, or 1.3. They also provide a cheaper and mechanically-simpler alternative to helicopters.

[1]We use the term multirotor aircraft to refer to aircraft such as quadrotors or hexrotors. For simplicity, we will use the term *multirotor* when the number of rotors is not relevant.

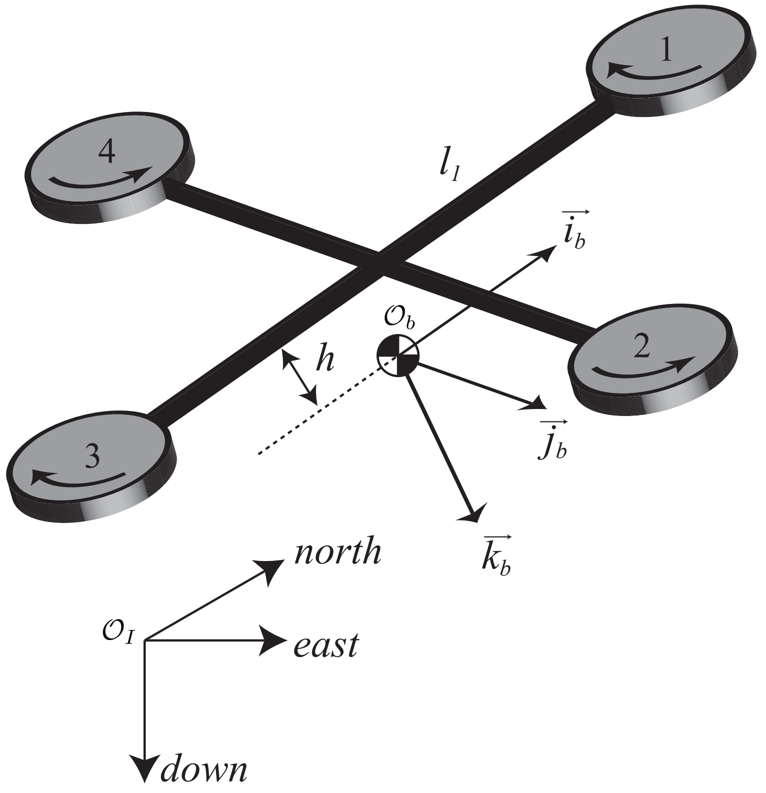

Figure 1.4: A schematic representation of the quadrotor showing coordinate frames and notation used in the research. The inertial coordinate frame has its arbitrary origin at $\mathscr{O}_I$ with right-handed axes oriented in north, east, and down (i.e. aligned with gravity) directions. The body-fixed reference frame has its origin, $\mathscr{O}_b$, at the quadrotor's center of mass, assumed here to be some distance h directly below the quadrotor's geometric center. The body frame $\vec{i}^b$ and $\vec{j}^b$ axes are parallel with the vectors from the geometric center to motors 1 and 2. The $\vec{k}^b$ axis is oriented to complete a right-handed coordinate system. The motors rotate in the directions shown.

A multirotor's small size and maneuverability also create problems for making them autonomous. The same fast dynamics that make them maneuverable require accurate and frequently updated position, orientation, and velocity state estimates to enable autonomous control. Additionally, constraints on the energy and payload capacity of a multirotor limit the available sensors and processing capability. These problems are especially relevant in unknown and complex 3D environments that demand information-rich sensor data and the associated processing-intensive algorithms.

We have developed a prototype vehicle for navigation in unstructured, confined, and unknown GPS-denied environments. It is relatively large and supports a considerable amount of computation to enable development. We have in mind a smaller, more compact version for an eventual final implementation.

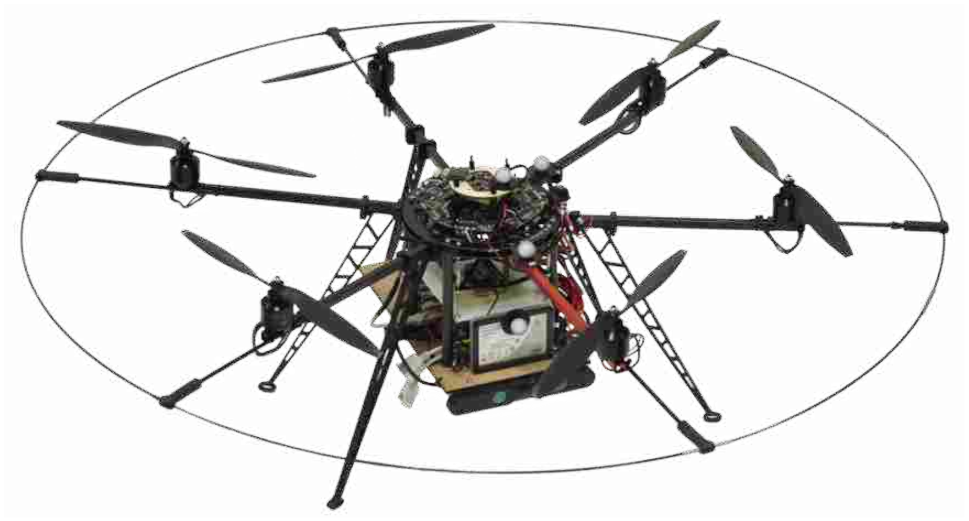

Figure 1.5: The Mikrokopter hexacopter prototype that we use in experiments discussed in this research. The only sensors we utilize are an altimeter, IMU, and front-facing RGB-D camera. All computation is completed onboard the vehicle and it communicates with a ground station computer using 802.11n Wi-Fi. The groundstation only provides goal locations to the platform.

We utilize a Mikrokopter hexacopter vehicle, shown in Figure 1.5 with the hardware specified in Table 1.1. The computer runs Ubuntu 12.04 Linux and all the applications are implemented in C++ and connected using the Robot Operating System (ROS) [4]. All the processing is performed onboard. Imagery from the RGB-D camera is available as fast as 30 Hz, but we routinely down-sample it to 15 Hz, as this provides sufficient information for accurate sensor fusion estimates. The IMU and altimeter provide measurements at 100 Hz and 40 Hz respectively. During the experiments we present from this research, the CPU usage averages about 40%. From these measurements, we believe that there is sufficient capacity for the onboard computer to run more computationally complex algorithms.

We employ a motion capture camera system from Motion Analysis, as shown in Figure 1.6, to provide truth data during experiments. This system provides 6DoF pose information at up to 200 Hz with sub-degree and sub-millimeter accuracy within a volume of about 14 ft by 10 ft by 6 ft. We used a filtered numerical derivative of the position data as velocity truth. The motion capture system has proved invaluable for testing and trouble-shooting algorithms, particularly the

Table 1.1: Hardware Details

Component	Description
Vehicle	Mikrokopter Hexacopter XL
Autopilot	Flight-Ctrl V2.1 ME
Sonar Altimeter	LV-MaxSonar®-EZ3
RGB-D Camera	ASUS Xtion Pro Live
IMU	MicroStrain® 3DM-GX3®-15
Motherboard	Global American EPI-QM67 EPIC
Processor	Intel® Core i7-2710QE
Harddrive	Intel® SSD 320 120GB

estimation and control. We were also able to autonomously fly the hexrotor using the truth data during interim steps, which aided development tremendously.

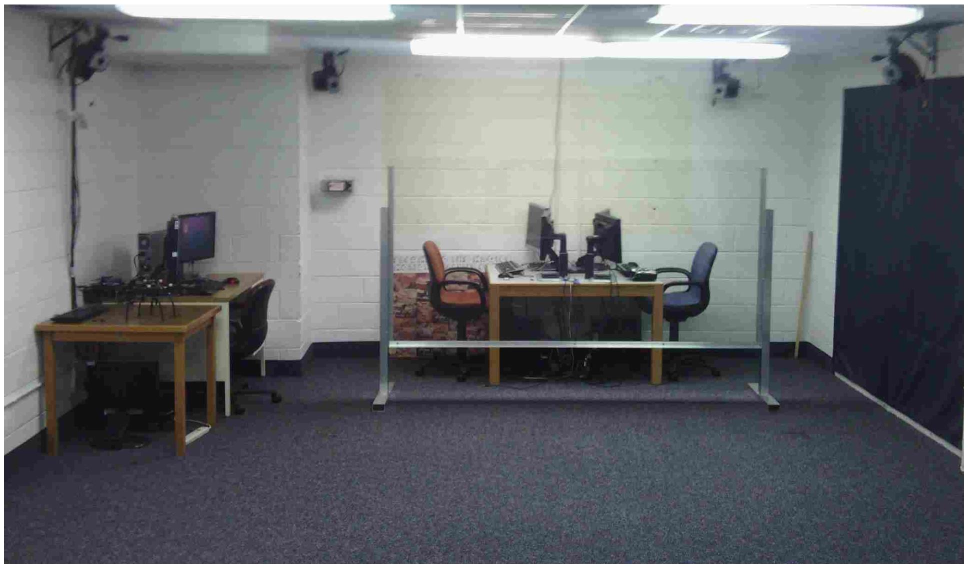

Figure 1.6: Pictured is the motion capture environment from Motion Analysis where most of the testing was conducted for this research. The system provides 6DoF pose information at 200 Hz with sub-degree and sub-millimeter accuracy.

1.2 Navigation without GPS

Precision navigation without GPS is a challenging problem. It is further complicated when we assume no prior knowledge of the environment, which is often the case for buildings damaged by natural disasters, in caves or underground, or in areas of conflict. To provide the level of accuracy needed for robust navigation, potential solutions typically require an exteroceptive sensor with a motion estimation or landmark recognition capability combined with an IMU. Solutions have employed monocular cameras [5, 6], stereo camera pairs [7, 8], planar-laser scanners [9, 10], 3D laser scanners [11], sonar [12], radar [13], and even Wi-Fi [14–16]. We will concentrate on vision-based solutions for our aerial implementation due to a camera sensor's low cost, low power requirements and light weight. Furthermore, machine vision approaches are more flexible in that they require fewer assumptions regarding the nature of the environment than do other sensors with comparable weight and power requirements.

Regardless of the sensor suite employed for GPS-denied navigation, there are only two known general approaches for this particularly challenging scenario: dead reckoning and simultaneous localization and mapping (SLAM). We introduce and briefly discuss these two approaches below.

1.2.1 Dead Reckoning

Dead reckoning is the process of summing changes in pose or velocity provided by onboard sensors to estimate the trajectory of a vehicle. Dead reckoning has obviously been used for a variety of systems for quite some time, as high-quality, publicly-available GPS has only been around since about 1994. However, without extremely high-quality, expensive sensors, such as a high-quality inertial navigation system, precise global navigation in unknown environments using only dead reckoning is generally only viable for short periods of time. This is especially true with an aerial vehicle, where trajectories and motion are not as smooth as on a ground vehicle. If global information is not required, however, vehicles can navigate indefinitely with unbounded global drift in the estimates.

One method of conducting fairly reliable dead reckoning with vision information is visual odometry (VO) [17]. Good practical introductions to the topic can be found in [18, 19]. VO is the process of comparing images to find the relative six-degree-of-freedom (6DoF) change in pose between them. Valid solutions have been achieved using monocular [20–22], stereo [23–26], and RGB-D, pictured in Figure 1.7,[2] cameras [27–31]. VO has been shown to provide accurate estimates of vehicle motion over long trajectories on ground vehicles [17,32], particularly when the results are fused with information from a high-quality IMU [23]. We discuss our VO algorithm that uses a RGB-D camera in Chapter 3. This algorithm forms the base for our topological approach.

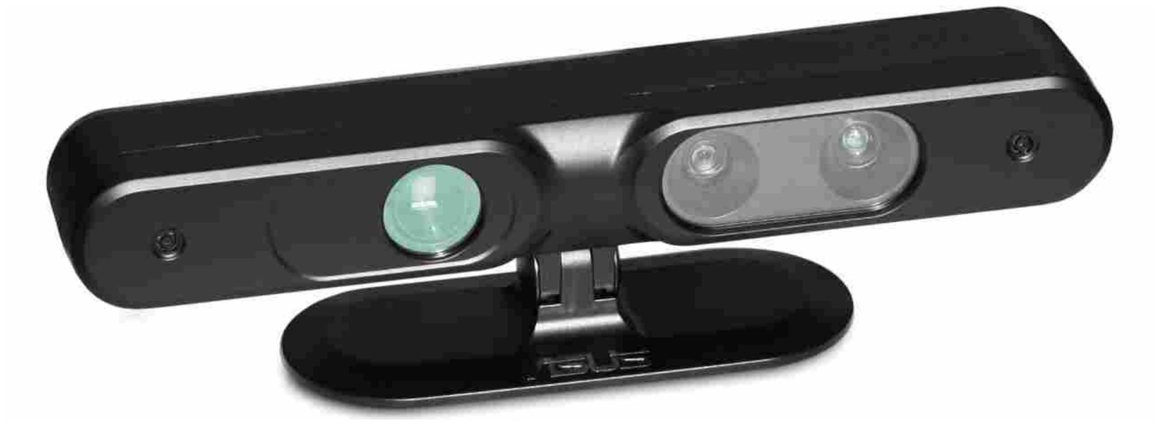

Figure 1.7: An RGB-D camera provided by ASUS. The infrared (IR) projector is on the left, the regular RGB camera in the middle, and the IR receiver is on the right. Dense stereo processing, which is completed on the device, is used on the projected IR points to find the depth in the scene.

1.2.2 Simultaneous Localization and Mapping

SLAM builds upon dead reckoning with an objective to provide globally consistent information for the vehicle trajectory and map of the environment. It has been termed the quintessential robotics problem [33], as it could provide a robot with the ability to navigate in an unknown environment without the aid of an external source, thereby enabling true autonomy.

SLAM is the process of building up a map of the environment using the onboard sensors and then localizing the vehicle within that map. The improvements to SLAM over dead reckoning are made through the use of *loop closures*. A loop closure is made when a location or landmark is

[2]A RGB-D camera contains a regular red, green, and blue (RGB) color camera and a range camera. The range camera consists of an infrared (IR) emitter and an IR-filtered camera. The emitter projects a dense IR pattern in the environment which is used by the filtered camera to perform dense stereo to extract depth information. The resulting RGB-D imagery provides depth information for each pixel of the RGB camera. The cameras were introduced for console game systems but have proven revolutionary for the robotics industry because of the quality 3D information provided at a very affordable cost.

recognized as having been previously added to the map. Loop closures are essential for constraining the drift in the map and keeping it consistent.

A particularly important method of loop closure is visual place recognition using images [34–37]. Each image is classified using visual words and then compared to the other images in the map. Loop closures can then be detected more robustly by matching similar images instead of using the potentially drifted vehicle path alone.

SLAM is well developed in the literature. Good introductions and overviews to the topic are found in [33, 38–41]. There remains, however, a great need for further research to develop systems that demonstrate the high reliability required for use in military and commercial products. Additionally, a large majority of the research that has been done was completed using ground vehicles, without thought of using the location estimates for feedback control of the vehicle position and attitude. Consequently, more work is especially needed to develop mature capability for 6DoF aerial vehicles that utilize the localization estimates for feedback control.

There are many different variations on SLAM, however almost all of them can be placed into one of three categories or paradigms: extended Kalman filter (EKF) SLAM, particle filter SLAM, and graph-based SLAM. There are pros and cons to each of the three methods. A brief description and references to some of the relevant literature are presented.

EKF SLAM

EKF SLAM uses an EKF to keep track of the position and orientation (pose) of the robot and the locations of all the observed landmarks in a single state vector and covariance matrix [13]. Consequently, only a sparse feature-based global map of the environment is developed. This method was the first to provide a solution to the SLAM problem and has since been employed in extensive research [6, 20, 42–47]. It has been shown that the state estimates of the EKF become inconsistent over time [48, 49]. By inconsistent, we mean that the error in the EKF estimate is no longer zero-mean with a covariance within the bounds of the estimated covariance. Ways of postponing this inconsistency, however, have also been shown [50]. A disadvantage to this algorithm is the computational growth and complexity that result as the state vector and covariance matrix are appended by new landmark information. The state can grow extremely large and computation requirements for updating the covariance matrix become too demanding for real-time processing. The use of sub-maps provides a solution to this problem [24, 51].

Particle Filter SLAM

The second paradigm, particle filter SLAM, uses particle filters to represent the positions and landmarks in the map. Each particle can be thought of as a single estimate of the true state of the robot and the landmarks. A population of many particles is then used to estimate the distribution of the truth [40]. This method is particularly useful when multi-modal distributions are present or in a lost-robot scenario. Probably the best-known form of this paradigm is FastSLAM [9, 39, 52]. The computation time is reduced by using Rao-Blackwell particles [53, 54]. As shown in [50], the FastSLAM algorithm degenerates with time and becomes overconfident, regardless of the number of particles used.

Graph SLAM

Graph SLAM is one of the newer and most popular forms of SLAM [7,8,21,34,41,55–60]. Graph theory is employed to represent locations, and sometimes landmarks, topologically as nodes in a graph map and the relative transformations between the nodes as edges. Typically, some form of nonlinear bundle adjustment optimization [61–63] is used to adjust the nodes in the map and make them globally consistent. This method was originally developed as a batch process, meaning that all the processing was completed on saved data from a robot trajectory. There have since been many improvements to enable its real-time implementation.

We are interested in the implementations of graph SLAM that use vision. In particular, the methods outlined in [57, 59, 64] offer benefits that we desire to incorporate into our approach. A *keyframe* method for VO is used as a basis for the system. Rather than comparing each current image to the previous image, each is compared to a reference image, called a keyframe. Once the overlap between the current image and the keyframe becomes too small and the accuracy of the pose estimates is reduced, a new keyframe is declared and the image comparison process continues with respect to the new keyframe. The keyframes form the nodes in the graph map. Instead of landmarks, the images themselves are saved to provide a high-quality representation of the environment at each node. This kind of graph is referred to as a pose-only graph, or pose graph. Place recognition can then be used to match spatially consecutive keyframes in the map. An additional benefit is the ability to marginalize out keyframes from the map to make it more sparse for reducing computational demands when global optimization is employed.

1.3 Literature Review

It this section we review the relevant literature that applies to the problem we have laid out: vision-based autonomous flight in unstructured, confined, and unknown GPS-denied environments. As we have restricted the scope of our problem to vision-based solutions, we omit much discussion of work which utilizes planar laser scanners as the main exteroceptive sensor. Many researchers have introduced solutions using these scanners [65–67], but too many limiting assumptions must be made about the environment to be useful within the constraints we have set. However, there are a couple implementations that will be addressed.

1.3.1 Select Laser-based Aerial GPS-Denied Navigation Solutions

Researchers with the Robust Robotics Group at MIT were the first to provide a working solution to this problem [68, 69]. The authors provide an update to their original work in [70]. The motivation for the work was the International Aerial Robotics Competition,[3] which they won in 2009. They employed a planar laser scanner with a fast, 40 Hz, scan-matching algorithm which they characterized as the key technology allowing their vehicle to fly. The fast scan matching was required because they numerically differentiated the position measurements to provide velocity estimates for the vehicle. The authors present results for goal-directed and autonomous exploration flight tests. An important point to make regarding their approach is that feedback from a 2D particle filter SLAM algorithm is a required input for the EKF used for sensor fusion, to reduce the drift from the laser odometry. This is necessary as they represent the navigation metrically with respect to a single global frame. The SLAM computation was completed on a ground station that communicated to the vehicle via Wi-Fi. Additionally, these authors note that doubly-integrating

[3]http://iarc.angel-strike.com

accelerometer measurements to obtain relative position causes very fast drift in estimates. In Chapter 2 we provide a method to improve upon this approach for use with multirotor aircraft.

A more recent implementation, also using a laser and a 20 Hz scan-matching algorithm, is the work described in [71, 72]. Impressively, all computation is completed on board the vehicle. Many algorithm modifications were required to enable the complete onboard system. Results are provided that demonstrate both indoor and outdoor goal-directed and autonomous exploratory flights over multiple floors. However, the authors propose a complex system architecture that uses multiple sensor-fusion EKFs and requires feedback into the sensor fusion from visual-recognition loop closure and graph-based SLAM algorithms. Similarly to [70], this complicated structure is necessary for estimates defined with respect to a single global reference frame. The authors augment the traditional quadrotor dynamic model, described in Chapter 2, with terms to account for unmodeled aerodynamic effects. We propose the use of an enhanced dynamic model in Chapter 2 that provides additional insight into these unmodeled terms.

1.3.2 Vision-based Aerial GPS-Denied Navigation

As we mentioned previously, planar laser scanner-based implementations require strict assumptions regarding the nature of the environment. 6DoF motion estimation using vision is more desirable due to a camera sensor's low cost, low power requirements, and light weight. Additionally, high-quality solutions can be obtained using fewer assumptions about the environment.

Some of the earliest examples of vision-based estimation for quadrotor vehicles are [73–76]. Among the first to use vision-based estimates in the control loop was [77]. A few others utilize vision-based estimates in the control loop, but must use other aids, such as off-board processing [25, 78], simulated vision using motion capture data [79, 80], or artificial markers [78, 81, 82] to enable their approaches.

Huang et al. [29] detail results from combining the quadrotor framework in [69], discussed above, with the RGB-D VO and mapping work from [83]. They present results for 3D maps in small environments with estimates in the control loop. Their approach, however, requires feedback into the estimation from the visual graph-SLAM algorithm due to the use of globally referenced states. They are unable to complete the SLAM processes of loop closure and global optimization on board. Interestingly, the VO estimates are used to update the vehicle velocities rather than positions. It seems this is necessary to avoid requiring additional states to handle the relative nature of the visual measurements. The SLAM measurements provide the only position updates.

Researchers with the Autonomous Systems Lab at ETH Zurich have worked over the past several years to develop an autonomous vision-based quadrotor [22, 77, 78, 80, 84, 85]. Their approach uses measurements from a pressure sensor and an IMU at 1000 Hz, and a down-pointed camera using a modified version of parallel tracking and mapping (PTAM) [21], computed at a rate between 12 to 20 Hz. PTAM is a graph-SLAM approach and has been shown to work well and provide globally-consistent results in small environments. It retains a sparse map of strong 3D features of the environment. The authors modify the original PTAM algorithm to keep only a fixed number of keyframes to avoid the convergence delays that occurred in the original system [77]. This modification results in a rolling window mapping and localization approach. The measurements are fused using a multiplicative extended Kalman filter (MEKF) [86] to provide globally consistent state estimates. Their system also provides calibration refinement for the transformation between the body-fixed frame and the camera. In one of the most recent papers [22], they add 40 Hz measurements from an optical flow algorithm to the PTAM. The optical flow is necessary to maintain stability of the vehicle when the global navigation fails and needs to be reinitialized. The authors provide results demonstrating the accuracy of the state estimates compared to truth and results for autonomous hover and waypoint-directed flights for small indoor and outdoor environments. As the camera points downward, however, they are unable to do motion planning with obstacle avoidance.

Tomic et al. [26] introduce a quadrotor designed for urban search and rescue missions which utilizes navigation based on either stereo VO or laser scan matching, a combination which provides robustness. The authors organize the system into low-level and high-level components and adopt a novel distributed computation approach. The relative position measurements from the VO or scan matching are fused with IMU measurements using an indirect filter based on stochastic cloning [86, 87]. The approach does not maintain a metric map but it does keep a locally topological one containing known landmarks for small environments. Relative measurements to these landmarks are sent to the sensor fusion for drift reduction of the position states. The authors define and demonstrate a high-level controller that provides some cognitive behavior such as recognizing, tracking, and flying through a window; such behavior is facilitated by the topological map. The authors report qualitative autonomous flight results of the vehicle moving from indoor to outdoor environments. They discuss the difficulties in dealing with relative measurements from the VO and jumps that occur in position estimates with the recognition of landmarks. As the system utilizes constraints set by the IMAV competition[4] for map initialization and landmark recognition, the system is not configured for use in general a priori unknown environments.

Work by members of the Computer Vision and Geometry Lab at ETH Zurich [30, 88] has culminated in a quadrotor capable of vision-based autonomous flight and exploration. The approach uses stereo VO from forward-looking cameras and optical flow from a downward looking camera and sonar altimeter. The objective of the approach is to provide a 3D textured global map of the environment in real time for operator use and for autonomous exploration and path planning onboard the quadrotor. Most of the computation is completed onboard, including the textured 3D map. Graph SLAM in the form of global optimization and loop closure are required for global states and these must be computed offboard. The fusion of position information derived from the VO and optical flow algorithms is accomplished using only a low-pass filter and attitude information is provided directly from the IMU. The authors present results for exploration and mapping in unknown indoor and outdoor environments and also localization within a known map. They emphasize that the optical flow of the downward-pointing camera is essential for the system to function. We note that the improved model we explain and utilize in Chapter 2 can reduce the need for such an optical flow algorithm.

In almost all the work cited above [22, 26, 29, 70, 71, 77, 78, 80, 84, 85, 88, 89] and in many other published papers about multirotors, a subtle mistake is made when relating gravity to the accelerometer measurements. In Chapter 2, we clarify this issue and provide an improved method for finding quality attitude and velocity estimates of the vehicle using only IMU measurements. These improved estimates, in turn, can then contribute to a lower dependence on exteroceptive sensor or GPS position measurements for a system. As we demonstrate, correctly modeling the accelerometer measurements has a significant impact on position, velocity, and attitude estimates.

[4]http://www.imav2011.org

1.3.3 Relative Navigation Background

As a general rule in unmanned systems research, robots utilize a metric representation of the environment with respect to a global coordinate frame. By this, we mean that a map is employed which tracks the global coordinates of elements in the environment, like the textured 3D maps produced in [88], and navigation is always referenced to a single global frame. This principle is emphasized by the literature discussed previously, where all the approaches but one [26] incorporate this ideology, even though a majority of the approaches employ graph SLAM methods with an underlying topological framework. There are many problems with using a globally metric approach. They are particularly noticeable and challenging when sensor fusion estimates must be used in controlling a vehicle with fast dynamics. The biggest issue is the requirement of feedback from computationally expensive optimization algorithms into the sensor fusion algorithm(s), causing the optimization to be in the critical path for real-time estimation and control. This arrangement incorporates non-trivial delays, which if unaccounted for, can cause instability and crashes [71, 78]. Very often, additional terms are required by the estimation to handle the relative measurements provided by the vision processing [**?**, 26, 86]. Finally, additional logic is necessary to handle large jumps in position information after states are updated when loop closure constraints are added and optimization is performed.

An alternative is to use a more flexible, topological representation of the world; an approach that uses the relative relationships between locations and landmarks, and de-emphasizes the use of a single global frame of reference. This is more aligned with how a human perceives and navigates in the world [90]. In a topological framework, places are represented as nodes, and edges represent relative transformations connecting the nodes. Metric information can be build upon the topological framework as needed for path planning and navigation [58]. This concept has been studied for quite awhile [8, 58, 91–93], but has not been widely adopted in the SLAM community. We note that this topological idea is approximated by many of the graph-SLAM approaches mentioned earlier. However the main objective of those approaches is to globally optimize the produced topological map to reference a single global coordinate frame.

A key enabler for a consistent topological framework is visual place recognition [34]. The ability to recognize locations using images, without any prior metric information, is critical to establishing the appropriate constraints to eliminate drift and allow a robot to maneuver through the world based on the relative relationships.

In [32], the authors contend that a purely relative, topological representation of both locations and landmarks is both necessary and sufficient for long-term autonomous navigation in typical human environments. They present a method for relative bundle adjustment [62] that provides optimization in relative space and that achieves constant-time complexity. This approach seems robust, however we believe that there are times when globally consistent information is important and helpful. This is especially true with a UAS that often must report back information to remote operators.

We detail in this research a new architecture for aerial vehicles that employs a relative navigation approach that is more flexible than a globally metric one. We accomplish this by separating the critical, real-time processes that are essential for navigation and control from any need of globally consistent information. The algorithms providing real-time estimation and control use a relative coordinate frame based on the keyframes from the VO. A lower priority back end provides globally consistent information for the topological map when desired. We detail the advantages and flexibility of this proposed system, along with details of the system architecture, in Chapter 3. We provide additional detail specifically for the relative estimation approach in Chapter 4.

1.4 Summary of Contributions

The contributions of this research were made while developing a platform for vision-based autonomous flight in unstructured, confined, and unknown GPS-denied environments. In developing the prototype, we created a new framework for GPS-denied aerial navigation, which offers many advantages and a greater flexibility compared to current systems. The highlights of this framework, which correspond to the contributions made by this research, are summarized by chapters below.

- Several new filtering concepts using a drag-force-enhanced model are discussed in Chapter 2.
  - A tutorial is provided for an improved understanding of the behavior of accelerometers on multirotor vehicles.

  - Results for improved attitude, velocity, and position estimates are shown that were generated using only the cheap MEMs-based IMU data available on a multirotor.
  - The capability to estimate the drag coefficient in the state is proved; which demonstrates the simplicity with which the improved estimation methods can be implemented.

- A new relative navigation approach for aerial vehicles is described in Chapter 3.
  - The approach combines graph methods with a multiplicative extended Kalman filter (MEKF) for relative mapping and localization.
  - A new architecture framework separating the essential real-time processes for flight from computationally expensive optimization and loop closure algorithms is developed.
  - The approach allows flexibility as changes to the map do not affect the relative localization and control.
  - The keyframes designated by the VO algorithm provide the basis for the relative coordinate frames .
  - Some general path planning and control algorithms are modified to work in a changing, relative frame of reference.

- Chapter 4 details a MEKF that provides state estimates relative to a changing local coordinate system.
  - A MEKF is derived that uses relative states and the improved dynamic model from Chapter 2.
  - The unique setup of the filter provides a way to easily incorporate the relative position measurements provided by the VO.
  - A method is shown that provides proper handling of the state and covariance when a change of coordinate systems is made.
  - The ability to estimate and refine IMU to camera calibration parameters in the state is demonstrated.

Each of the chapters in this work is based heavily on at least one publication. We often expand the concepts in these chapters compared to the content available in the publications as additional space is available. Chapter 2 is based on [94, 95]. Chapter 3 is an expanded version of the original content in [96, 97]. Finally, the MEKF of Chapter 4 is documented in [79, 98].

CHAPTER 2. QUADROTORS & ACCELEROMETERS STATE ESTIMATION WITH AN IMPROVED DYNAMIC MODEL

Quadrotors are ideal platforms for autonomous flight in unknown and complex environments. Their small size and maneuverability are conducive to operating in confined spaces and avoiding obstacles. Equipped with appropriate sensors and algorithms, quadrotors could enable several applications currently infeasible for ground robots.

A quadrotor's small size and maneuverability also create problems for making them autonomous. The same fast dynamics that make quadrotors maneuverable require accurate and frequently updated position, orientation, and velocity state estimates to enable autonomous control. Additionally, the amount of energy and payload available on a quadrotor limits the available sensors and processing capability. These problems are especially relevant in unknown and complex 3D environments that demand more processing-intensive algorithms and information-rich sensor data.

A few research groups have made noteworthy progress toward deploying fully autonomous quadrotors. The authors of [70] present their system while noting that one of the major challenges is estimating the position and velocity. To address this problem they develop a sophisticated laser scan-matching algorithm. They characterize that algorithm as the key technology that allows their vehicle to fly.

In [71] the authors present a quadrotor system also using a scanning laser rangefinder. They state that the fast vehicle dynamics require pose estimates with update rates of 20 Hz. Several of their design decisions are driven by the need to meet the system's computational limitations.

The authors of [81] rely on the increased information available from cameras at the expense of increased computation. They emphasize that the estimated velocity is critical to damp the system. Their estimator relies on a simple constant-velocity motion model that raises the minimum update rate the filter requires from the vision processing.

It is clear that accurate and timely state estimates, of attitude and velocity in particular, are key ingredients to enable an autonomous quadrotor. In this chapter, we present results showing how velocity and attitude estimates can benefit from an improvement to the traditional quadrotor dynamic model.

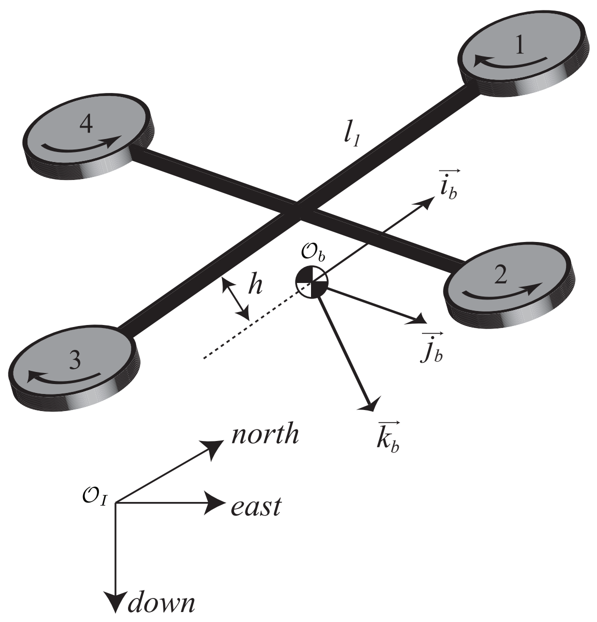

Figure 2.1: A schematic representation of the quadrotor showing coordinate frames and notation used in the chapter. The inertial coordinate frame has its arbitrary origin at $\mathscr{O}_I$ with right-handed axes oriented in north, east, and down (i.e. aligned with gravity) directions. The body-fixed reference frame has its origin, $\mathscr{O}_b$, at the quadrotor's center of mass, assumed here to be some distance h directly below the quadrotor's geometric center. The body frame $\vec{i}_b$ and $\vec{j}_b$ axes are parallel with the vectors from the geometric center to motors 1 and 2. The $\vec{k}_b$ axis is oriented to complete a right-handed coordinate system. The motors rotate in the directions shown.

An assumption of the widely-used (e.g. [25,65,67,77,99–102]) traditional quadrotor model is that the only significant forces acting on the vehicle are gravity and the thrust produced by the rotors (see Figure 2.1). This assumption leads to a dynamic model for the quadrotor's linear acceleration given by

$$\begin{bmatrix} \dot{u} \\ \dot{v} \\ \dot{w} \end{bmatrix} = \mathbf{R}_I^b \begin{bmatrix} 0 \\ 0 \\ g \end{bmatrix} - \begin{bmatrix} 0 \\ 0 \\ \frac{T}{m} \end{bmatrix}, \tag{2.1}$$

where $\dot{u}$, $\dot{v}$, and $\dot{w}$ are the components of acceleration along the body-fixed $\vec{i}_b$, $\vec{j}_b$, and $\vec{k}_b$ axes, as shown in Figure 2.1. $\mathbf{R}_I^b$ is the rotation matrix from the inertial to the body-fixed reference frame. The constants g and m represent the gravitational acceleration and the quadrotor's mass. The combined thrust of the rotors is T and is an input to the system.

This model is acceptable for some applications, such as designing a controller, because it captures the external forces with the most significant magnitudes. However, it leads to an interesting paradox that was articulated in [103]. The model implies accelerometers aligned with the $\vec{i}_b$ and $\vec{j}_b$ axes will always measure zero. Yet many successful quadrotor implementations using this model also use the accelerometer measurements to effectively improve estimates of the quadrotor's orientation.

An improved quadrotor model should explain how to more appropriately use the data from the accelerometers aligned with the $\vec{i}_b$ and $\vec{j}_b$ axes. Many papers acknowledge that some drag force must act on the vehicle's body, but this is reasonably dismissed as being small as it is proportional to the square of the vehicle's linear velocity. Other researchers include a drag force that is directly proportional to the quadrotor's linear velocity. For example, [104], [105], [106] and [107] identify similar terms. However, the authors' emphasis on control algorithms avoids any discussion of the physics that generate the drag or its effect on accelerometers or the state estimation process. A drag force proportional to linear velocity is included in the estimation approach of [70] based on the authors' observation that something must prevent the quadrotor from accelerating indefinitely. They too offer no physical explanation for the effect and instead rely on a motion capture system to estimate the proportionality constant.

Both [103] and [108] identify a term called *rotor drag* as the force acting in the body-fixed $\vec{i}_b$ and $\vec{j}_b$ axes. Reference [103] derives a dynamic model of the quadrotor based on concepts of fundamental blade-element theory, and they identify the rotor drag with the accelerometer measurements. They conclude with a discussion of two controllers based on their drag-force-enhanced model. However, the resulting hardware implementations are only assessed qualitatively by stating that the systems were much easier to fly than with the usual scheme.

In this chapter we discuss how accelerometer measurements in quadrotor flight lead to improved estimation performance. We confirm, using hardware and truth data, that the accelerometers directly measure the translational velocity, allowing more accurate estimates of the attitude and velocity of the vehicle than can be achieved with traditional methods. In Section 2.1 we lay the groundwork to explain why accelerometers in quadrotor flight measure the rotor drag. In many publications, like [81], [67], [66] for example, a subtle mistake is made when relating gravity to the accelerometer measurements on a quadrotor. We clarify this issue and show the agreement between the improved accelerometer model and actual measurements.

We also show how easy it is to use this new, drag-force-enhanced model. In Section 2.4 it is shown that the drag force constant can be estimated as a state in a filter driven only by IMU measurements, thus removing the need for experimental tuning as in [103] or an expensive motion capture system as in [70]. We present several filters designed to work with only IMU measurements in Section 2.3. We compare estimates to truth as well as estimates from more traditional approaches to quantify the benefit of the enhanced model in state estimation. In the results in Section 2.5 we show a twofold to threefold improvement in average attitude error compared to standard approaches.

We have used the drag-force-enhanced model that is introduced in this paper in other scenarios. Some preliminary analysis and a simplified application are presented in [94]. The benefits that the enhanced model provides to a filter that estimates position and yaw using exteroceptive sensor measurements is discussed in [109].

2.1 Accelerometer Tutorial

If $\mathbf{a} \in \mathbb{R}^3$ represents the acceleration of a vehicle, m is the mass, and $\mathbf{F}_T \in \mathbb{R}^3$ is the total external force acting on the vehicle, then Newton's second law states that

$$\mathbf{a} = \frac{1}{m}\mathbf{F}_T. \tag{2.2}$$

However, accelerometers do not measure the total acceleration **a**. Accelerometers measure the specific acceleration, meaning the difference between the acceleration of the vehicle and gravitational acceleration.

Figure 2.2 shows a simplified diagram of a one-axis accelerometer, where a proof-mass is attached by a flexure to the housing of the accelerometer. When the proof mass undergoes an acceleration that is different than the acceleration experienced by the housing, the proof mass

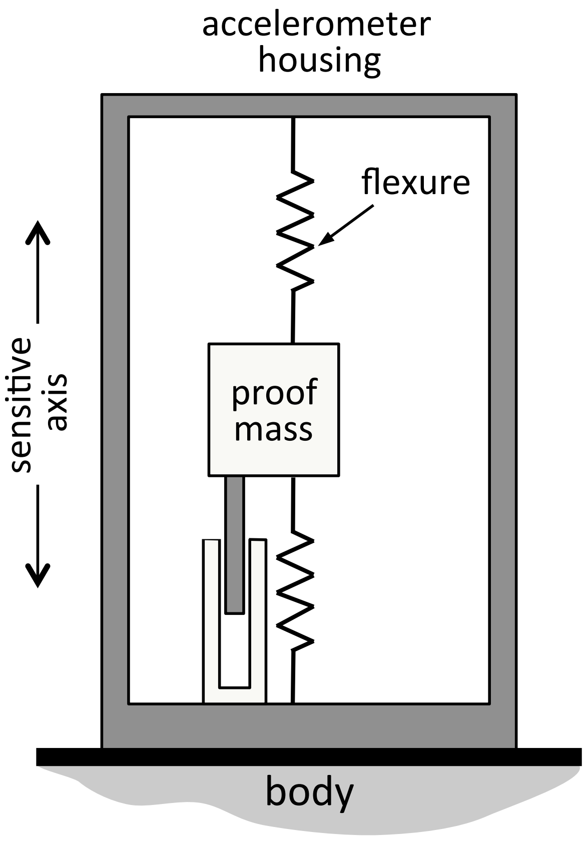

Figure 2.2: A simplified model of an accelerometer. When the proof mass undergoes an acceleration that is different than the acceleration experienced by the housing, the proof mass deflects and a non-zero measurement is produced.

deflects and a non-zero measurement is produced. With the accelerometer in Figure 2.2 on a horizontal surface, the normal force offsets the force due to the weight. The proof-mass, which does not experience the normal force, deflects under the influence of gravity, causing the accelerometer to measure an upward acceleration of 1 g. On the other hand, during free fall, gravity would accelerate both the housing and the proof-mass, resulting in a measurement of zero.

The output of a three-axis accelerometer mounted on a rigid body is then given by

$$\mathbf{a}_m = \frac{1}{m}\left(\mathbf{F}_T - \mathbf{F}_g\right), \tag{2.3}$$

where $\mathbf{a}_m \in \mathbb{R}^3$ is the measured acceleration and $\mathbf{F}_g \in \mathbb{R}^3$ is the force due to gravity. Equation (2.3) states that for an accelerometer to measure the effect of gravity only, all the external forces must sum to zero.

Throughout this work we assume that the axes of the accelerometer are aligned with the body-frame axes. We also assume that the accelerometer has been properly calibrated to remove misalignment errors and cross-axis sensitivity.

Below we detail two different methods for state estimation using accelerometers. Many researchers have shown that using one of these methods to fuse accelerometers with an exteroceptive sensor increases performance compared to using the exteroceptive sensor alone.

2.1.1 Accelerometer-based Attitude Estimation

From (2.3), when the sum of external forces is zero the accelerometers will measure

$$\mathbf{a}_m = -\frac{1}{m}\mathbf{F}_g. \tag{2.4}$$

Let $\mathbf{a}_m^b \overset{\triangle}{=} (a_{mi}, a_{mj}, a_{mk})^\top$, represent the acceleration measured in the body-frame axes. In the inertial frame the force of gravity is $\mathbf{F}_g = (0, 0, mg)^\top$. Expressing (2.4) in the body frame gives

$$\begin{bmatrix} a_{mi} \\ a_{mj} \\ a_{mk} \end{bmatrix} = \mathbf{R}_I^b \begin{bmatrix} 0 \\ 0 \\ -g \end{bmatrix}, \tag{2.5}$$

$$= \begin{bmatrix} g\sin\theta \\ -g\sin\phi\cos\theta \\ -g\cos\phi\cos\theta \end{bmatrix}, \tag{2.6}$$

where ϕ is the roll angle and θ is the pitch angle of the ground vehicle. The roll and pitch angles can therefore be estimated as

$$\hat{\phi}_{\text{accel}} = \tan^{-1}\left(\frac{a_{mj}}{a_{mk}}\right) \tag{2.7}$$

$$\hat{\theta}_{\text{accel}} = \sin^{-1}\left(\frac{a_{mi}}{g}\right). \tag{2.8}$$

This estimation method, expressed by (2.7) and (2.8), will be termed the *traditional attitude method*. We note here that the underlying assumption required for this method of zero external forces is rarely met during quadrotor flight.

2.1.2 Accelerometer-based Velocity Estimation

If the attitude is known, measured accelerations can be integrated to estimate velocity using

$$\dot{\mathbf{v}}^b = \mathbf{a}_m^b + \frac{1}{m}\mathbf{R}_I^b\mathbf{F}_g. \tag{2.9}$$

where $\mathbf{v}^b = (u, v, w)^\top$ is the velocity expressed in the body-fixed frame. It is critical to note here that this method of using accelerometers is sensitive to the underlying assumption of known attitude. The traditional attitude method should not be used to provide the necessary attitude estimates since it requires that $\dot{\mathbf{v}}^b = 0$. In the remainder of this chapter we will refer to this approach as the *integrated velocity method*.

2.1.3 Accelerometers on Quadrotors

Quadrotors obviously differ from ground vehicles because of the thrust required to keep them airborne. Yet many researchers using quadrotors treat them as ground vehicles with respect to accelerometer measurements. Quite often a variant of the traditional attitude method provides IMU-based estimates of attitude to a higher-level observer. This second observer then uses the integrated velocity method despite the fact that the methods' assumptions are contradictory.

As a simple example, when a quadrotor rests on an inclined surface, as shown in the left half of Figure 2.3, it experiences a normal force $\mathbf{F}_n$, a friction force $\mathbf{F}_f$ and the force due to gravity

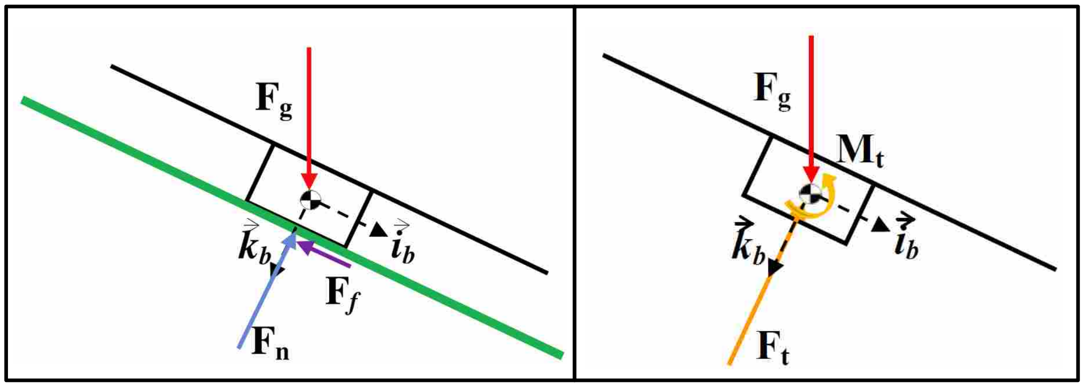

Figure 2.3: Free-body diagrams of a quadrotor in two scenarios. The left image shows the forces when the quadrotor is sitting on an inclined surface. The right image illustrates the forces according to the standard model when the quadrotor is at a constant attitude in the air and $\mathbf{M}_t$ is zero.

$\mathbf{F}_g$. The summation of forces in the body frame is

$$\mathbf{F}_T = \mathbf{R}_I^b \mathbf{F}_g - \mathbf{F}_n - \mathbf{F}_f. \tag{2.10}$$

Consequently the accelerometer measures

$$\mathbf{a}_m^b = \frac{1}{m}(\mathbf{F}_T - \mathbf{R}_I^b \mathbf{F}_g) = \frac{1}{m}\begin{bmatrix} -F_f \\ 0 \\ -F_n \end{bmatrix}, \tag{2.11}$$

and (2.8) can be used to correctly find the pitch angle.

When the quadrotor is in the air with a similar attitude, the forces are usually assumed to be $\mathbf{F}_g$ and the thrust $\mathbf{F}_t$; all other forces are assumed negligible. The thrust force $\mathbf{F}_t$ and moment $\mathbf{M}_t$ are resolved from the individual thrust forces acting at each propeller. This would give the total force, in the body-fixed reference frame, as

$$\mathbf{F}_T \approx -\mathbf{F}_t + \mathbf{R}_I^b \mathbf{F}_g. \tag{2.12}$$

According to the model, the accelerometers would then measure

$$\mathbf{a}_m^b = \frac{1}{m}(\mathbf{F}_T - \mathbf{R}_I^b \mathbf{F}_g) = \frac{1}{m}\begin{bmatrix} 0 \\ 0 \\ -F_t \end{bmatrix}. \tag{2.13}$$

Clearly in this situation the attitude cannot be determined using (2.7) and (2.8) since, according to the model, the body frame $\vec{i}$ and $\vec{j}$ accelerometers should always measure zero. It would also be invalid to integrate these values to find the vehicle velocities. Still, many researchers make productive use of the traditional attitude and velocity methods within control and estimation schemes. The discrepancy is explained by nontrivial forces that are missing in the model.

2.2 Drag-force-enhanced Quadrotor Model

In this section we present a simplified version of the drag-force-enhanced model originally presented in [103]. We note here that this model applies equally well to any multirotor vehicle, not only quadrotors. We model the quadrotor with the nonlinear equations

$$\dot{\mathbf{x}} = \mathbf{f}(\mathbf{x}, \mathbf{u}) + \xi, \tag{2.14}$$

$$y_i = h_i(\mathbf{x}, \mathbf{u}) + \eta, i = 1, ..., p \tag{2.15}$$

where h_i is the i^{th} measurement function, and the vector $\mathbf{u}$ represents the inputs that drive the evolution of the estimated states. In this paper we will use the input

$$\mathbf{u} = \begin{bmatrix} p & q & r \end{bmatrix}^\top, \tag{2.16}$$

which are the rotation rates about the $\vec{i}_b$, $\vec{j}_b$, and $\vec{k}_b$ axes respectively and correspond to the outputs of the onboard gyroscopes after calibration. ξ and η are zero-mean Gaussian processes with covariance Q and R respectively.

With reference to Figure 2.1, the states we consider are

$$\mathbf{x} = \begin{bmatrix} \phi & \theta & \psi & u & v & w \end{bmatrix}^\top, \tag{2.17}$$

where ϕ, θ, and ψ are the Euler angles that relate the orientation of the body-fixed frame to the inertial frame, u, v, and w represent the components of linear velocity in the $\vec{i}_b$, $\vec{j}_b$, and $\vec{k}_b$ axes, respectively.

The drag-force-enhanced model is obtained from (2.1) by adding a drag force, which is proportional to the body-fixed-frame velocity, to the $\vec{x}$ and $\vec{y}$ body-fixed components

$$\begin{bmatrix} \dot{u} \\ \dot{v} \\ \dot{w} \end{bmatrix} = \mathbf{R}_I^b \begin{bmatrix} 0 \\ 0 \\ g \end{bmatrix} - \begin{bmatrix} 0 \\ 0 \\ \frac{T}{m} \end{bmatrix} - \begin{bmatrix} \frac{\mu}{m} u \\ \frac{\mu}{m} v \\ 0 \end{bmatrix}, \tag{2.18}$$

where μ is the drag force coefficient. The term μ can depend on several factors, but for nominal autonomous flight conditions it can be treated as a constant. We highlight that the changes required to implement the enhanced model are simply two terms added to the body-frame velocity equations. Reference [103] identifies this drag force as the rotor drag, although the detailed derivation somewhat obscures the simplicity and practicality of the model.

Using (2.18), the components of (2.14) can now be expressed as

$$\begin{bmatrix} \dot{\phi} \\ \dot{\theta} \\ \dot{\psi} \end{bmatrix} = \begin{bmatrix} 1 & \sin\phi\tan\theta & \cos\phi\tan\theta \\ 0 & \cos\phi & -\sin\phi \\ 0 & \frac{\sin\phi}{\cos\theta} & \frac{\cos\phi}{\cos\theta} \end{bmatrix} \begin{bmatrix} p \\ q \\ r \end{bmatrix}, \tag{2.19}$$

$$\begin{bmatrix} \dot{u} \\ \dot{v} \\ \dot{w} \end{bmatrix} = \begin{bmatrix} -g\sin\theta + (vr - wq) - \frac{\mu}{m} u \\ g\sin\phi\cos\theta + (wp - ur) - \frac{\mu}{m} v \\ g\cos\phi\cos\theta + (uq - vp) - \frac{T}{m} \end{bmatrix}. \tag{2.20}$$

The other terms in the $\dot{u}$ and $\dot{v}$ portions of (2.20) are due to gravity and the Coriolis acceleration. We discussed in Section 2.1 why gravity is not measured by the accelerometers, and the Coriolis terms can be neglected for a quadrotor that depends on onboard sensors. We can therefore model the $\vec{x}$ and $\vec{y}$ accelerometer outputs a_{mi} and a_{mj} as directly measuring the respective components of the drag force

$$h_1 \triangleq a_{mi} \approx -\frac{\mu}{m} u, \tag{2.21}$$

$$h_2 \triangleq a_{mj} \approx -\frac{\mu}{m} v. \tag{2.22}$$

Equations (2.21) and (2.22) are the two measurement functions included in (2.15).

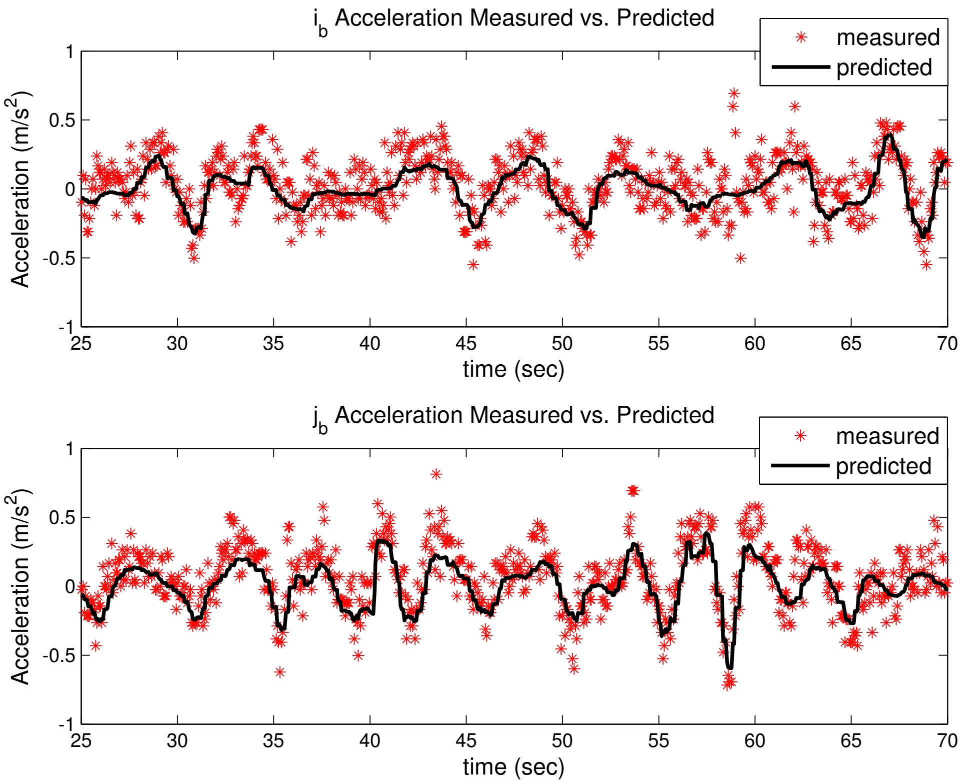

Figure 2.4: The actual accelerometer measurements for a nominal indoor flight are plotted against those predicted by (2.21) and (2.22). We generated this figure using time-stamped accelerometer and pose data recorded during a manually-controlled flight.

Figure 2.4 illustrates the agreement between the actual accelerometer measurements and those predicted by equations (2.21) and (2.22). We generated Figure 2.4 using recorded time-stamped accelerometer and pose data during a manually controlled flight. The accelerometer measurements were from the onboard sensors and the pose measurements from a motion capture system. We used a filtered numerical derivative of the position measurements, expressed in the body-fixed frame of the quadrotor, for the u and v velocities in (2.21) and (2.22). The value for

μ was determined using a least-squares fit of the data from several flights. We found that the error between the predicted and actual accelerometer measurements is modeled well as zero-mean and Gaussian.

2.2.1 Why Traditional Attitude Estimates Provide Some Benefit

Reference [110] offers some additional perspective on accelerometer measurements as they relate to the roll angle ϕ and the pitch angle θ on a quadrotor. Using (2.21) and (2.22) in (2.20) and ignoring the Coriolis forces, the time evolution of u and v from (2.20) can be written as

$$\begin{bmatrix} \dot{u} \\ \dot{v} \end{bmatrix} = \begin{bmatrix} -g\sin\theta - \frac{\mu}{m}u \\ g\sin\phi\cos\theta - \frac{\mu}{m}v \end{bmatrix}. \tag{2.23}$$

Consider for a moment just the first row of (2.23). Using a small angle approximation and taking the Laplace transform gives the transfer function from θ to u as

$$u(s) = \frac{\frac{-gm}{\mu}}{\frac{m}{\mu}s+1}\theta(s). \tag{2.24}$$

Substituting (2.21) in (2.24) we obtain

$$a_{mi}(s) = \frac{g}{\frac{m}{\mu}s+1}\theta(s) \overset{\triangle}{=} H(s)\theta(s). \tag{2.25}$$

By similar arguments, we find that

$$a_{mj}(s) = \frac{-g}{\frac{m}{\mu}s+1}\phi(s) = -H(s)\phi(s). \tag{2.26}$$

Equations (2.25) and (2.26) describe the first-order response relating the changes in attitude to the accelerometer measurements, where $H(s)$ is a low-pass filter.

Figure 2.5 shows the true pitch angle for a quadrotor measured by a motion capture system, as compared to the traditional attitude estimate described in Section 2.1 using actual accelerometer measurements. We have also superimposed the result of filtering the true pitch angle with $H(s)$

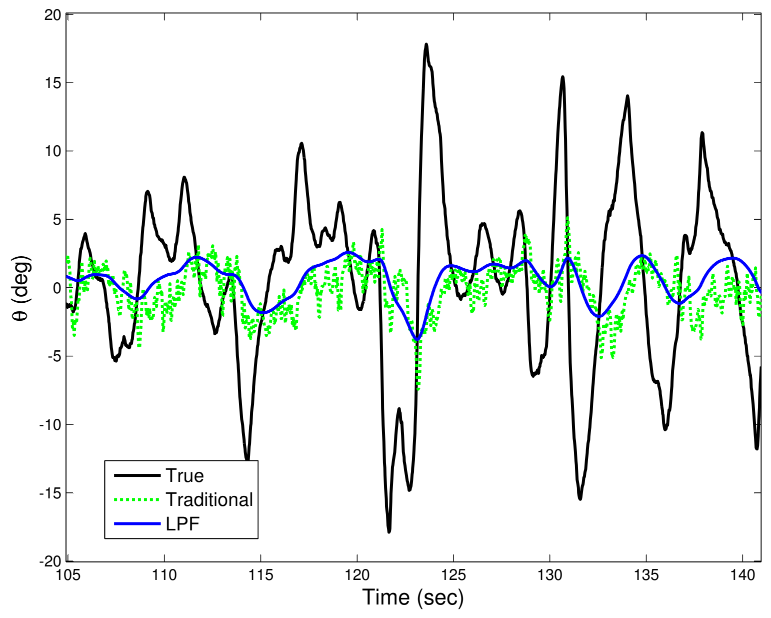

Figure 2.5: Comparison of the true pitch angle θ, the traditional attitude approximation, and a low-pass filtered θ for a small portion of a flight. Notice how closely the traditional attitude estimation method results approach those of the low-pass filtered pitch angle.

as given in (2.25). The traditional attitude estimate based on accelerometer measurements agrees well with the low-pass filtered pitch angle.

The traditional attitude method given in (2.7) and (2.8) is based on the assumption of static equilibrium. The time constant m/μ governs how quickly the accelerometer measurements react to a step change in the roll or pitch angle. For a heavier quadrotor used for onboard vision experiments, $m = 2.75$ kg and $\mu \approx 0.77$. In this case it would take the accelerometer (and therefore the traditional attitude estimate) more than 10 seconds to reach 95% of its steady-state value. Even for the nimble Hummingbird quadrotor by Ascending Technologies, it takes approximately 3 seconds to approach steady state. Long before the accelerometer-based attitude estimate becomes valid, the quadrotor will reach speeds that degrade onboard sensor data or that make collision in a cluttered environment likely.

It is well known that angle estimation using the traditional approach does not describe fast attitude changes, even if the reason behind it is less understood. To compensate, gyroscopes can be used to predict attitude over the short term. Accelerometers are then used to correct the estimates in a measurement update. However, without accounting for the fact that the accelerometers on the quadrotor measure scaled velocities, as shown in (2.21) and (2.22), the measurement update will drag the estimates toward the low-pass filtered attitude and not the true attitude. This estimation approach works reasonably well during flights with gradual attitude changes, but produces less accurate estimates than are possible.

2.3 Observer Design

In this section, we present several filter design options using the drag-force-enhanced model for the case where the states $\mathbf{x}_{imu} = [\phi, \theta, u, v]^\top$ will be estimated using only IMU data. This application is of interest when designing a filter for attitude estimation such as is typically available on quadrotor autopilots.

2.3.1 Linear Fixed-gain Filter

A simple approach to observer design is to make approximations so that the state propagation and measurement equations are linear. In this case, (2.14) and (2.15) are replaced by

$$\dot{\mathbf{x}}_{imu} = \mathbf{A}\mathbf{x}_{imu} + \mathbf{B}\mathbf{u}, \tag{2.27}$$

$$\mathbf{y}_{acc} = \mathbf{C}_{acc}\mathbf{x}_{imu}, \tag{2.28}$$

where $\mathbf{A}$ and $\mathbf{B}$ are the appropriate Jacobians of (2.19) and (2.20) and $\mathbf{C}_{acc}$ is the Jacobian of (2.21) and (2.22). We will assume Coriolis forces are negligible and evaluate the Jacobians at hover. We further simplify the design by choosing a fixed observer gain.

We take $\mathbf{y}_{acc} = [a_{mi}, a_{mj}]^\top$ as modeled by (2.21) and (2.22) as the only elements of (2.15). Using the inputs $\mathbf{u} = [p, q, r]^\top$ gives

$$\mathbf{A} = \begin{bmatrix} 0 & 0 & 0 & 0 \\ 0 & 0 & 0 & 0 \\ 0 & -g & \frac{\mu}{m} & 0 \\ g & 0 & 0 & \frac{\mu}{m} \end{bmatrix}, \tag{2.29}$$

$$\mathbf{B} = \begin{bmatrix} 1 & 0 & 0 \\ 0 & 1 & 0 \\ 0 & 0 & 0 \\ 0 & 0 & 0 \end{bmatrix}, \tag{2.30}$$

$$\mathbf{C}_{acc} = \begin{bmatrix} 0 & 0 & \frac{\mu}{m} & 0 \\ 0 & 0 & 0 & \frac{\mu}{m} \end{bmatrix}. \tag{2.31}$$

State estimates are propagated using

$$\dot{\mathbf{x}}_{imu} = \mathbf{A}\mathbf{x}_{imu} + \mathbf{B}\mathbf{u} + \mathbf{L}_{fg}(\mathbf{y} - \mathbf{C}_{acc}\mathbf{x}_{imu}) \tag{2.32}$$

where the observer gain $\mathbf{L}_{fg}$ is chosen using the *lqr* function in Matlab. Because of its simplicity, this linear fixed-gain filter is the most practical choice for an embedded processor versus the extended Kalman filters presented below. Results using this filter will be presented later in the article.

2.3.2 EKF with Known μ

The Extended Kalman Filter (EKF) offers improved performance over the linear fixed-gain filter at the expense of increased complexity. Appropriate elements of the nonlinear equations (2.19) and (2.20) are used in a separate prediction step to propagate $\mathbf{x}_{imu}$ forward in time, and Jacobians $\mathbf{A}$, $\mathbf{B}$, and $\mathbf{C}_{acc}$ are constantly reevaluated using the current state estimate.

However, most of the increased complexity arises in maintaining the uncertainty of the state estimates, **P**, and calculating the variable filter gain. The uncertainty is propagated using

$$\dot{\mathbf{P}} = \mathbf{A}\mathbf{P} + \mathbf{P}\mathbf{A}^\top + \mathbf{B}\mathbf{R}_{gyro}\mathbf{B}^\top + \mathbf{Q}. \tag{2.33}$$

The process uncertainty in (2.33) is modeled in two parts. Matrix **Q** is a hand-tuned, diagonal matrix that we often use only to model the propagation of bias states. Since the inputs **u** are gyroscope measurements, and **B** is the matrix that specifies how the gyroscopes affect the state evolution, $\mathbf{R}_{gyro}$ is the covariance of the noise on those sensors. Since we can measure the noise characteristics of the gyroscopes, using the $\mathbf{B}\mathbf{R}_{gyro}\mathbf{B}^\top$ term makes the filter easy to tune and more accurate than assuming a generic, diagonal process noise matrix for all of the states.

The accelerometer measurement update is given by

$$\mathbf{P}^+ = (\mathbf{I} - \mathbf{L}\mathbf{C}_{acc})\mathbf{P}^-, \tag{2.34}$$

$$\mathbf{x}_{imu}^+ = \mathbf{x}_{imu}^- + \mathbf{L}\left(\mathbf{y}_{acc} - \mathbf{C}_{acc}\mathbf{x}_{imu}^-\right), \tag{2.35}$$

where

$$\mathbf{L} = \mathbf{P}^-\mathbf{C}_{acc}^\top\left(\mathbf{R}_{accel} + \mathbf{C}_{acc}\mathbf{P}^-\mathbf{C}_{acc}^\top\right)^{-1}. \tag{2.36}$$

The notation Y^- and Y^+ indicates a variable Y before and after the measurement update. We use $\mathbf{R}_{accel}$ to denote the covariance of the accelerometer measurement, and **I** is an appropriate identity matrix. Results for this filter will be given later in the chapter.

2.3.3 EKF with Estimated μ

The filters given in the previous two sections assume that the rotor drag coefficient μ is known. In this section we relax that assumption and add μ to the state vector of the EKF and estimate it simultaneously with the other states. In Section 2.4 we show that μ is observable. In the remainder of the paper, we will designate the filter derived in this section as EKF-μ. We have found that μ can be estimated accurately in a filter using only IMU data. Over several datasets, the filter-estimated μ stays within about 5% of the truth value estimated using a motion capture system. When added to the estimated state the propagation of μ is modeled as a random walk.

Estimating μ as a state of the EKF provides substantial benefit over other approaches that use an improved dynamic model. In reference [103] the authors describe a hand tuning process accomplished by comparing several quantities from the vehicle and their counterparts given by a GPS-driven attitude heading reference system (AHRS). Since the AHRS cannot be collocated with the vehicle's IMU, the parameter μ must be tuned simultaneously with the position and attitude differences that make the accelerometers on the AHRS agree with those of the vehicle. Reference [70] relies on an expensive motion capture system to provide flight data that they compare to control commands; a system identification process is then used to determine the right values for a damping coefficient similar in nature to μ.

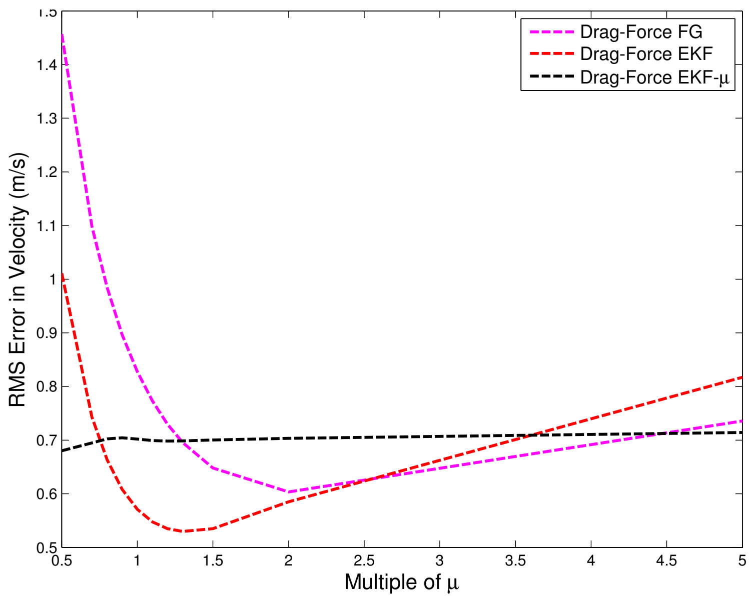

Figure 2.6: RMS errors in velocity for the various filters when the initial estimate of μ is a multiple of the true value. The Fixed Gain filter and EKF use the initial, but incorrect, value of μ throughout the entire flight. With EKF-μ, the value evolves in time. It is interesting to note that the Fixed Gain and EKF still provide low RMS values over a wide range of incorrect estimates of μ.

Because EKF-μ estimates an additional state, its performance suffers as compared to the filters that use a predetermined value for μ. Figure 2.6 illustrates this by comparing the root-mean-squared (RMS) error in body frame velocity over several initial values of μ. As would be expected, EKF-μ is robust to poor initial estimates of μ. If μ cannot be determined beforehand, we suggest EKF-μ be used in an initial manual flight to produce an estimate. In subsequent flights, with perhaps some minor hand tuning of μ, one of the other filters should be used to avoid a decrease in estimation accuracy. Figure 2.6 shows that there is a reasonable range of values for the estimate of μ that enable the Fixed-Gain and EKF filters to perform well.

2.4 Observability of μ

In this section we demonstrate the observability of the drag coefficient μ when it is included in the state. We note that observability is a necessary condition for filter convergence. As a review, we briefly present the theory of nonlinear observability. Afterwards we provide the conditions where the longitudinal quadrotor system is locally observable with μ included in the vehicle state and when the full state is locally observable with μ included. This overview is based on a more detailed presentation of the theory in [111].

2.4.1 Theory

Let $\mathbf{x} \in X$, where X is an open subset of $\mathbb{R}^N$, represent the state of the nonlinear system

$$\dot{\mathbf{x}} = \mathbf{f}(\mathbf{x}) + \sum_{i=1}^{m} \mathbf{g}_i(\mathbf{x}) u_i, \tag{2.37}$$

with n nonlinear outputs of the form

$$y_j = h_j(\mathbf{x}),\ j = 1...n, \tag{2.38}$$

which form the vector output function $\mathbf{y}(y_1(\mathbf{x}), ..., y_n(\mathbf{x}), \mathbf{u})$. Further, let $S(X)$ and $V(X)$ respectively designate the set of all scalar-valued smooth functions and the set of all vector fields (i.e. column vectors on smooth functions) on X. The functions $\mathbf{f}(\mathbf{x})$, $\mathbf{g}_i(\mathbf{x}) \in V(X)$ are the nonlinear functions of the state and the m time-varying scalars u_i are the (known) inputs that drive the system.

Given (2.37) and (2.38), we can say that two states $\mathbf{x}_0$ and $\mathbf{x}_1$ are **distinguishable** if there exists an input function $\mathbf{u}(\cdot)$ such that

$$\mathbf{y}(y_1(\mathbf{x}_0),...,y_n(\mathbf{x}_0),\mathbf{u}) \neq \mathbf{y}(y_1(\mathbf{x}_1),...,y_n(\mathbf{x}_1),\mathbf{u}). \tag{2.39}$$

The system is said to be **locally observable** at a point $\mathbf{x}_0 \in X$ if there exists a neighborhood $\mathcal{N}(\mathbf{x}_0)$ around $\mathbf{x}_0$ such that every $\mathbf{x} \in \mathcal{N}(\mathbf{x}_0)$, other than $\mathbf{x}_0$, is distinguishable from $\mathbf{x}_0$. We can say that the system is **locally observable** if it is locally observable at each point $\mathbf{x}_0 \in X$.

It can be shown that a system is locally observable at a point $\mathbf{x}_0 \in X$ if there are a sufficient number of linearly independent vectors in the gradients of the measurement equations or the gradients of the Lie derivatives evaluated at $\mathbf{x}_0$. Recall that the Lie derivative of a function $\kappa \in S(X)$ with respect to some vector field $\omega \in V(X)$ is defined by the mapping

$$L_\omega \kappa \triangleq \frac{\partial \kappa(\mathbf{x})}{\partial \mathbf{x}} \cdot \omega(\mathbf{x}) : X \to \mathbb{R}. \tag{2.40}$$

2.4.2 Quadrotor Longitudinal States With μ

We will now analyze the observability of the longitudinal quadrotor system with μ included in the state. The longitudinal states of a quadrotor are $\mathbf{x} = [\theta, u, \mu]^\top$. Recall that θ is the pitch angle, u is the body-fixed forward velocity, and μ is drag coefficient. The longitudinal system is

$$\begin{bmatrix} \dot{\theta} \\ \dot{u} \\ \dot{\mu} \end{bmatrix} = \mathbf{f}(\mathbf{x}) + u_1 \mathbf{g}_1(\mathbf{x}) = \begin{bmatrix} 0 \\ -g\sin(\theta) - \frac{\mu}{m}u \\ \varsigma_\mu \end{bmatrix} + u_1 \begin{bmatrix} 1 \\ 0 \\ 0 \end{bmatrix}, \tag{2.41}$$

where we assume that $\dot{\theta} = q$, the rotation rate about the body $\hat{j}$ axis. The acceleration due to gravity is g and m is the mass of the vehicle. The time propagation of μ is modeled as a random walk where ς_μ is a zero-mean, Gaussian random variable. The gyroscope measurement $u_1 = q$ is the input to the system.

The output of the system is the accelerometer measurement in the $\vec{i}_b$ direction, modeled as

$$y_1 = h_1(\mathbf{x}) = -\frac{\mu}{m}u. \tag{2.42}$$

To show that the longitudinal system is locally observable at a point $\mathbf{x}_0 \in X$, we must find three vectors of the observability Grammian that are linearly independent at $\mathbf{x}_0$. We begin with finding the gradient of the output (2.42)

$$\mathbf{d}h_1 = \begin{bmatrix} 0 & -\frac{\mu}{m} & -\frac{u}{m} \end{bmatrix}. \tag{2.43}$$

Next we look among the first-order Lie derivatives. Due to the simplicity of $\mathbf{g}_1(\mathbf{x})$ in (2.41), $\mathbf{d}L_{g1}h_1 = [0\ 0\ 0]$. We then consider

$$\mathbf{d}L_f h_1 = \mathbf{d}\left(\begin{bmatrix} 0 & -\frac{\mu}{m} & -\frac{u}{m} \end{bmatrix} \begin{bmatrix} 0 \\ -g\sin(\theta) - \frac{\mu}{m}u \\ 0 \end{bmatrix} \right) align \tag{2.44}$$

$$= \begin{bmatrix} g\cos(\theta)\frac{\mu}{m} & \left(\frac{\mu}{m}\right)^2 & \left(\frac{g\sin(\theta)}{m} + \frac{2\mu u}{m^2}\right) \end{bmatrix}. \tag{2.45}$$

We use a the second-order Lie derivative of $\mathbf{f}(\mathbf{x})$ to find a final vector

$$\mathbf{d}L_f L_f h_1 = \mathbf{d}\left(\begin{bmatrix} g\cos(\theta)\frac{\mu}{m} & \left(\frac{\mu}{m}\right)^2 & \frac{g\sin(\theta)}{m} + \frac{2\mu u}{m^2} \end{bmatrix} \begin{bmatrix} 0 \\ -g\sin(\theta) - \frac{\mu}{m}u \\ 0 \end{bmatrix} \right) align \tag{2.46}$$

$$= \begin{bmatrix} -g\cos(\theta)\left(\frac{\mu}{m}\right)^2 & -\left(\frac{\mu}{m}\right)^3 & \left(\frac{-2\mu g\sin(\theta)}{m^2} - \frac{3\mu^2 u}{m^3}\right) \end{bmatrix}. \tag{2.47}$$

We can combine the vectors (2.43), (2.45) and (2.47) into an observability matrix $\mathbf{O}_M$ for the longitudinal state

$$\mathbf{O}_M = \begin{bmatrix} 0 & \frac{-\mu}{m} & \frac{-\mu}{m} \\ g\cos(\theta)\frac{\mu}{m} & \left(\frac{\mu}{m}\right)^2 & \frac{g\sin(\theta)}{m} + \frac{2/muu}{m^2} \\ -g\cos(\theta)\left(\frac{\mu}{m}\right)^2 & -\left(\frac{\mu}{m}\right)^3 & \frac{-2/mug\sin(\theta)}{m^2} - \frac{3/mu^2 u}{m^3} \end{bmatrix} \tag{2.48}$$

The determinate of $\mathbf{O}_M$ is

$$|\mathbf{O}_M| = -\frac{u\cos(\theta)\mu^4 g + \sin(\theta) m\mu^3 g^2}{m^5}. \tag{2.49}$$

Since $\mu \neq 0$, the system will be locally observable except when

$$0 = -g\sin(\theta) - \frac{\mu}{m}u. \tag{2.50}$$

We note that 2.50 is exactly equal to (2.23) when $\dot{u} = 0$. Consequently, the condition (2.50) will only be true during unaccelerated flight, as it is then impossible to tell the difference between θ and u, making the state unobservable. Therefore, we may say that the longitudinal state $\mathbf{x}$ is locally observable during accelerated, $\dot{u} \neq 0$, flight. The ability to estimate the drag coefficient μ in the state of the vehicle, using only IMU measurements, makes it straightforward to use the drag-force enhanced model for quadrotor state estimation.

2.4.3 Full State Observability

We can expand the state considered in the nonlinear observability analysis to include $\mathbf{x} = [\phi, \theta, u, v, \mu]^\top$. We do not include the yaw angle ψ or the down velocity w, as they are only observable when a magnetometer and a height sensor are included. Equation 2.41 becomes

$$\begin{bmatrix} \dot{\phi} \\ \dot{\theta} \\ \dot{u} \\ \dot{v} \\ \dot{\mu} \end{bmatrix} = \begin{bmatrix} 0 \\ 0 \\ -g\sin(\theta) - \frac{\mu}{m}u \\ -g\sin(\phi)\cos(\theta) - \frac{\mu}{m}v \\ 0 \end{bmatrix} + u_1 \begin{bmatrix} \sin(\phi)\tan(\theta) \\ \cos(\phi) \\ 0 \\ 0 \\ 0 \end{bmatrix} + u_2 \begin{bmatrix} 1 \\ 0 \\ 0 \\ 0 \\ 0 \end{bmatrix}, \tag{2.51}$$

where the gyroscope measurement $u_2 = p$ is the second input to the system. The second output of the system is the accelerometer measurement in the $\vec{j}_b$ direction

$$y_2 = h_2(\mathbf{x}) = -\frac{\mu}{m}v. \tag{2.52}$$

Following the same procedure as outlined above, we must find five linearly independent equations at $\mathbf{x}_0$ of the observability Grammian. The first two equations are the gradients of (2.42) and (2.52), and the next two are from the first-order Lie derivatives of $\mathbf{f}(\mathbf{x})$ along (2.42) and (2.52). For the last equation, we include the second-order Lie derivative of $\mathbf{f}(\mathbf{x})$ either along (2.42) or (2.52). Taking the determinant of the new five-by-five observability matrix for the former approach results in the same condition as (2.50). Not surprisingly, the latter method provides the condition

$$0 = g\sin(\phi)\cos(\theta) - \frac{\mu}{m}v, \tag{2.53}$$

which is the other equation specified in (2.23). We can therefore claim that the full state $\mathbf{x}$ is locally observable as long as there is acceleration in either the $\vec{i}_b$ or $\vec{j}_b$ directions.

2.5 IMU-only Results

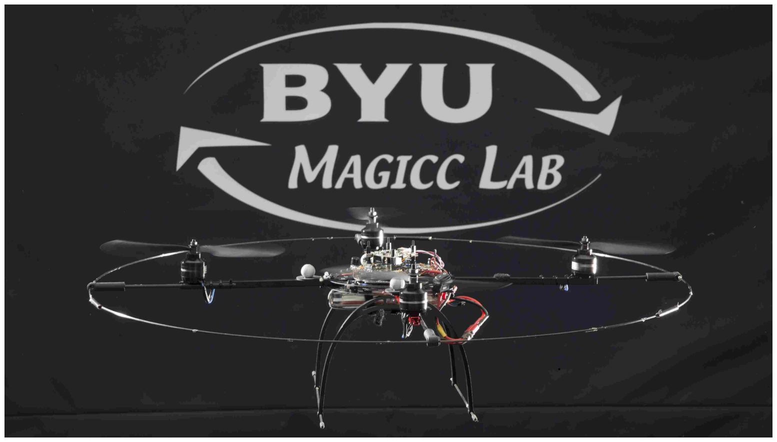

Figure 2.7: A quadrotor from MikroKopter that was used in the experiments. The vehicle is hovering using the measurements from the motion capture system. ©Jaren Wilkey, BYU Photo

We use a MikroKopter [112] quadrotor depicted in Figure 2.7 and a motion capture system, in Figure 2.8, from Motion Analysis [113] to generate the data used in these results. The quadrotor provides accelerometer and gyroscope measurements at 40 Hz. We use this low rate, much lower than is typically used, to highlight the benefits of using the drag-force-enhanced model. We receive pose information for the quadrotor from the motion capture system at 200 Hz. A filtered numerical derivative of the position information is used to estimate the true velocity. All the data was first recorded from a 250 second manually-controlled flight and then processed offline so that comparisons between different filters would be valid.

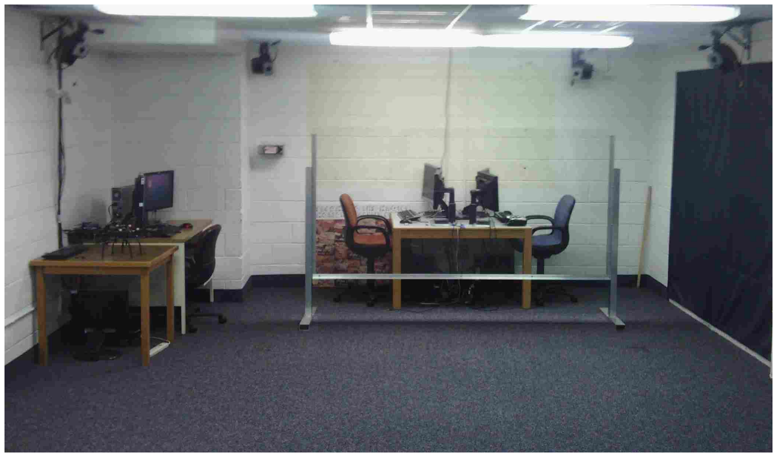

Figure 2.8: Here is the motion capture environment where all the testing was conducted for this research. We use a motion capture system from Motion Analysis. The system provides 6DoF pose information at 200 Hz with sub-degree and sub-millimeter accuracy.

2.5.1 Comparison Filters

As a baseline to compare against, we present results from two filters that rely on the traditional attitude approach. The first is a fixed-gain linear filter described by

$$\dot{\mathbf{x}}_n = \begin{bmatrix} p \\ q \end{bmatrix} + \mathbf{L}_n \left(\mathbf{x}_{accel} - \mathbf{x}_n \right), \tag{2.54}$$

where $\mathbf{x}_n = [\phi, \theta]^\top$. The fixed observer gain $\mathbf{L}_n$ is selected to prevent the estimates from drifting while still tracking fast changes as best as possible. We tuned $\mathbf{L}_n$ to produce results qualitatively similar to those from a popular commercial quadrotor. The vector $\mathbf{x}_{accel}$ is an estimate of ϕ and θ based on the traditional attitude method (2.7) and (2.8). We designate this filter the *Traditional FG* filter.

The second filter is the explicit nonlinear complementary filter developed in [114]. This filter estimates the rotation matrix $\hat{\mathbf{R}}$ between the body-fixed reference frame and the inertial reference frame, as well as the biases on the gyroscopes $\hat{\mathbf{b}}$. The filter is implemented using

$$\dot{\hat{\mathbf{R}}} = \hat{\mathbf{R}}\left(\left(\mathbf{u} - \hat{\mathbf{b}}\right)_\times + k_P\left(\omega_{mes}\right)_\times\right), \quad \hat{\mathbf{R}}(0) = \hat{\mathbf{R}}_0 \tag{2.55}$$

$$\dot{\hat{\mathbf{b}}} = -k_I\omega_{mes} \tag{2.56}$$

$$\omega_{mes} = \sum_{i=1}^{n} k_i v_i \times \hat{v}_i, \quad k_i > 0, \tag{2.57}$$

where the notation $()_\times$ refers to the matrix form of the cross product, and v_i are vectorial measurements. Using only IMU information without a magnetometer, there is only one vectorial measurement: the gravity measurement discussed in the "Accelerometer Tutorial". The implementation completed for this paper was iteratively hand-tuned to provide the minimum RMS error in attitude for the dataset considered. The gains were $k_P = 0.5$ and $k_I = 0.05$ for the results presented below. We designate this as the *Complementary* filter.

2.5.2 Attitude Results

Figure 2.9 plots the error in the estimates for a small portion of the manual flight; results for the pitch angle θ are similar. The figure compares the performance of the *Traditional FG* and *Complementary* filters with the drag-force-enhanced, linear fixed-gain filter we have described in this chapter.

Table 2.1: The combined RMS errors from the various filters for ϕ and θ from a manual flight.

RMS Error for Attitude Estimates	
Filter	**RMS of ϕ and θ (deg)**
Traditional FG	7.27
Complementary	5.66
Drag-Force Linear Fixed-Gain	2.80
Drag-Force EKF	2.16
Drag-Force EKF-μ	2.23

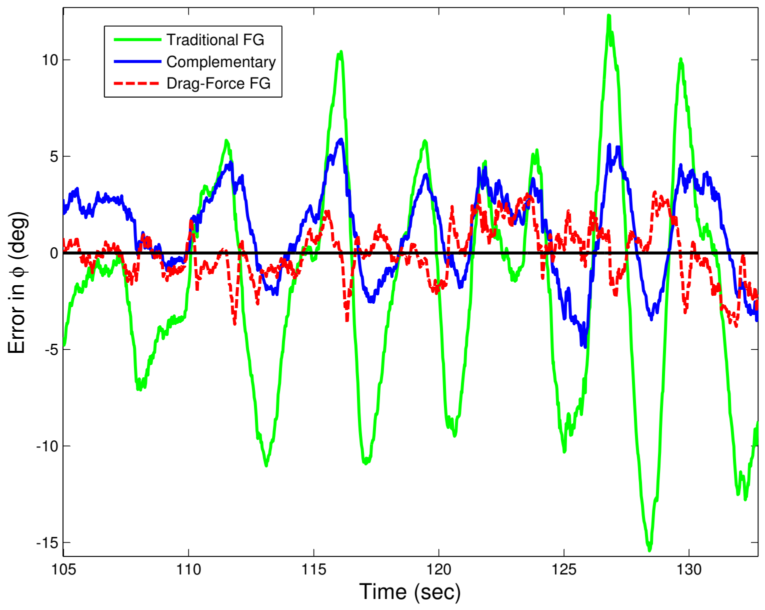

Figure 2.9: The error in the roll angle ϕ over a small window of the manual flight for the Traditional FG, Complementary and drag-force enhanced fixed gain filters. Since the plot is of error, smaller values denote increased performance.

The filters' performance is further described by Table 2.1 which presents the RMS error results for all of the filters over the entire flight. Notice in Table 2.1 that the other filters we have designed using the enhanced model offer improved performance over the linear fixed-gain filter, but at increased computational cost. We chose to represent the results of the linear fixed-gain filter in Figure 2.9 as it is more relevant in an IMU-only, embedded scenario.

2.5.3 Velocity Results

In addition to improving attitude estimates, the enhanced model also provides information on the body-frame velocities u and v that would be otherwise unavailable using only IMU measurements. Table 2.2 documents the RMS errors for velocity estimates using the drag-force-enhanced model in the filters presented above. Figure 2.10 illustrates estimates of u produced using the linear fixed-gain filter and the EKF. Note that velocity estimates for the *Traditional FG* and *Complementary* filters are absent from these results as these filters do not provide velocity estimates.

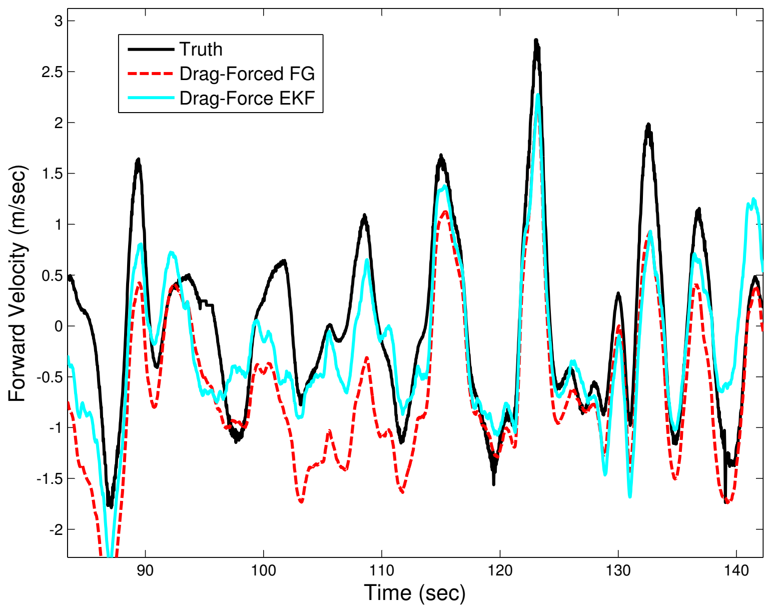

Figure 2.10: Body-frame velocity u truth versus the drag-force fixed-gain filter and EKF estimates for a small portion of a flight. Velocity estimates are not available when using only IMU data with the traditional approaches. Only IMU information is used to produce these estimates. Only the drag-force enhanced filter estimates are shown;

Table 2.2: The combined RMS errors for the velocity estimates u and v using the drag-force enhanced model.

RMS Errors for Velocity Estimates	
Filter	**RMS of u and v (m/s)**
Traditional FG	N/A
Complementary	N/A
Drag-Force Fixed-Gain	0.87
Drag-Force EKF	0.60
Drag-Force EKF-μ	0.67

Although the performance does not appear outstanding in Table 2.2 and Figure 2.10, we note that these results are produced using only inexpensive MEMS accelerometers and gyroscopes at low data rates. The fact that the improved model offers information on velocity along with high-quality attitude estimates is an additional advantage of the proposed approach. The velocity estimates from the drag-force-enhanced model reduce the need for fast position updates that traditional approaches require [109].

2.5.4 Results During Aggressive Maneuvers

We have also found that estimation using the drag-forced enhanced model is robust to aggressive maneuvers despite the near-hover assumption made in [103] to derive the model. Figure 2.11 shows estimates of θ for a segment of aggressive flight. The quadrotor experiences pitch angles in excess of 45 degrees that are estimated well by the method we propose.

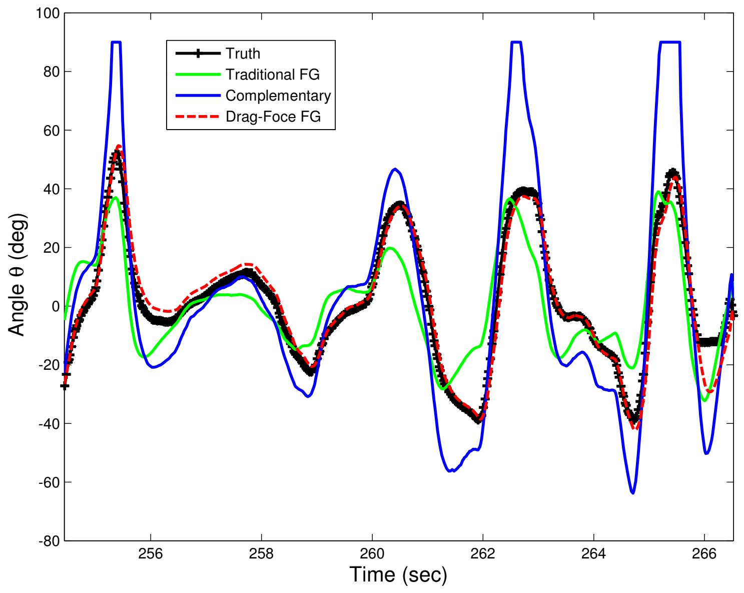

Figure 2.11: The true pitch angle θ and its estimates using the Traditional FG, Complementary, and drag-force-enhanced fixed-gain estimators are shown during an aggressive flight. Notice that the linear fixed-gain filter for the enhanced model provides accurate estimates despite the large angles that are experienced. The filter parameters are identical to those used to produce the previous plots.

The performance improvements shown here are due to the more correct model of the physical system that accounts for the rotor drag. The gyroscope measurements provide information for the fast changes in angle and the accelerometer corrections accurately constrain the drift. In the traditional approach, if the gyroscope measurements are trusted too much in order to track fast angular changes, the attitude estimates drift rapidly. To constrain the drift, the accelerometer measurements must be weighted sufficiently, but using the wrong dynamic model results in inferior performance.

It is important to note that the filter tuning parameters were not modified for this flight segment, which highlights the robustness of the proposed estimators. The *Traditional FG* and *Complementary* filters could be tuned for better performance during aggressive maneuvers but then performance near hover would suffer. As an alternative, an adaptive control or gain scheduling approach could be implemented on those filters to provide improved estimates for a broader flight regime, but at the expense of increased complexity.

2.5.5 Position Dead-reckoning Results

In this section we illustrate the significant results that are possible due to the improved accuracy given by the drag-force-enhanced model. We completed an experiment where the IMU information from the quadrotor was used to dead reckon the global position. We implemented this experiment with two filters: the Traditional EKF and the Drag-Force EKF.

The Traditional EKF is a combination of the two methods presented in the “Accelerometer Tutorial”. The *Traditional FG* filter, explained above, provides the attitude estimates using the accelerometer and gyroscope measurements according to the traditional attitude method. Then the filter estimates the global position by integrating accelerometer measurements using the integrated velocity method.

The Drag-Force EKF is an augmented version of the EKF with Known μ method, which was derived above. The filter uses the gyroscopes and accelerometer measurement updates (2.21) and (2.22) to estimate the attitude and velocity. We augmented this filter to also include north and east position states, which are estimated by integrating the velocity estimates.

In both filters the standard kinematic relationship between velocity and position is used to estimate the position using velocity estimates. Initialization of the position estimates at the starting global location is the only input position information provided to either filter during the whole flight.

Figure 2.12 shows north and east position dead-reckoning estimates obtained using only the IMU information available during the first ten seconds of the quadrotor flight. Note how the Drag-Force EKF estimates trend well with the global position while the estimates from the Traditional

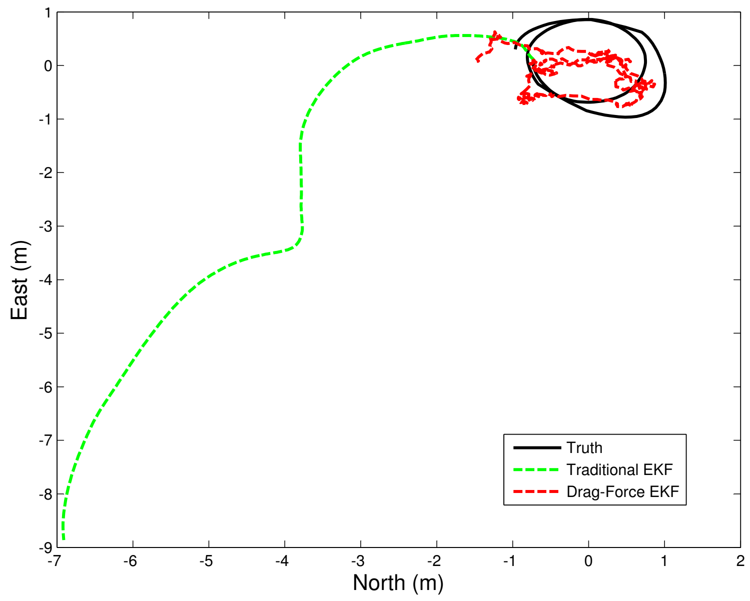

Figure 2.12: These position results were obtained using only the IMU information available from the first 10 seconds of a manual quadrotor flight. Note how the improved EKF estimates trend with the global position while the estimates from the traditional approach walk off the chart. This demonstrates the importance of IMU information to the quadrotor state estimates when a valid model is employed.

EKF do not. Figure 2.13 plots the norm of the north and east error for the Traditional EKF and the Drag-Force EKF. Note the vast difference in drift rate between the two approaches over the whole flight.

The point of these results is to demonstrate how much information the IMU can provide to the quadrotor state estimates when a valid model is employed. The basic position information provided by this approach can contribute to a lower dependence on exteroceptive sensor or GPS information. Correctly modeling the accelerometer measurements has a significant impact on position, velocity, and attitude estimates.

2.6 Conclusion

We have shown that assumptions behind the attitude method for measuring the gravity vector are flawed when applied to a quadrotor, even though the approach provides some benefit. When designing an estimator for a quadrotor it is too restrictive to assume static equilibrium. The

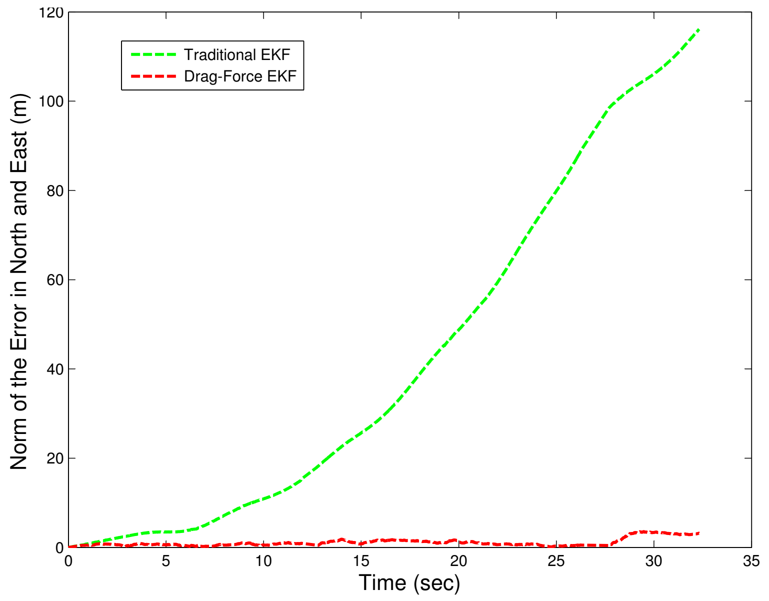

Figure 2.13: The norm of the north and east dead-reckoning position error for the Traditonal EKF and the Drag-Force EKF is shown. IMU measurements are the only sensor information provided to the two filters. Note the differences in the drift rate between the two approaches. Using the Drag-Force EKF will allow less-frequent exteroceptive and/or GPS updates because of the much lower drift rate using only the IMU information.

forces acting on the quadrotor will only sum to zero at hover or after a long period of time at a fixed attitude.

Using only IMU data, the EKF and the linear fixed-gain filter based on the drag-force-enhanced model provide a trade-off between complexity and performance. Each provides an improvement in attitude estimates compared to typical approaches, even during aggressive maneuvers, while also providing significant information about velocity. The velocity estimates, in turn, can be used effectively by the EKF to provide position estimates based on dead reckoning that diverge relatively slowly. If μ is unknown, we have shown that it can be effectively estimated by including it in the state during accelerated flight. The accuracy of the estimates we provided from the IMU-only filters could be further improved by increasing the data rate of the IMU.

We attribute the improvements shown in this article to the correct characterization of accelerometer measurement. This is a noteworthy advantage as IMU measurements are typically available at high rates and are comparatively inexpensive to process.

CHAPTER 3. RELATIVE NAVIGATION APPROACH

Finding solutions to enable GPS-denied aerial flight in a priori unknown environments is currently a popular research focus. The problem is challenging as the robot must discover its own location using only onboard sensors and computational resources. This task requires knowledge of complex elements from many distinct disciplines. Additionally, aerial vehicles which are used in this type of research also present difficult constraints like strict payload capacities and fast vehicle dynamics; constraints which are complicated further by using onboard-generated state estimates in feedback control. Unlike ground robots, these vehicles cannot afford to pause in one place until complex algorithms converge and estimates are sufficiently stable to continue. Only a few researchers have been able to achieve successful flying implementations for autonomous goal-directed flight.

Planar laser scanner-based implementations such as those discussed in [69,71] require strict assumptions regarding the nature of the environment. Six-degree-of-freedom (6DoF) motion estimation using vision is desirable due to a camera sensor's low cost, low power requirements and light weight. Furthermore, machine vision approaches are more flexible in that they require fewer assumptions about the environment.

Some of the earliest examples of vision-based estimation for quadrotor vehicles are [73,74, 76]. Among the first to use vision-based estimates in the control loop was [77]. A few others utilize vision-based estimates in the control loop but must use other aids, such as off-board processing [25, 78], simulated vision using motion capture data [79] or artificial markers [78,81,82] to enable their approaches.

Huang et al. [29] combine work from [69] and [83] to enable a quadrotor that uses an RGB-D sensor for visual odometry (VO) and mapping. They present results for 3D maps in small environments with estimates in the control loop. However the approach requires feedback into the estimation from loop closure and global optimization algorithms due to the use of globally referenced states. They are unable to complete these two tasks onboard.

Weiss et al. [22] describe a system where parallel tracking and mapping (PTAM) [21] is merged with an optical-flow algorithm for a down-pointed camera in an EKF framework. The optical-flow algorithm is necessary to maintain stability of the vehicle when the global navigation fails and needs to be reinitialized. They provide results demonstrating the accuracy of the optical-flow algorithm compared to truth and results for an autonomous hover. However, as the camera points downward, they are unable to do motion planning with obstacle avoidance.

Tomic et al. [26] introduce a quadrotor which utilizes navigation based on either stereo VO or laser scan matching, a combination which provides robustness. They report autonomous flight results moving from indoor to outdoor environments. The authors discuss the difficulties in dealing with relative measurements from the VO and jumps that occur in global position with the recognition of landmarks. The approach does not maintain a metric map but it does keep a topological one containing known landmarks in the environment. The system utilizes constraints set by the IMAV competition[1] for map initialization and landmark recognition which excludes it from use in general a priori unknown environments.

Fraundorfer et al. [88] present a quadrotor capable of autonomous flight and exploration using stereo VO from forward-looking cameras and optical flow from a downward looking camera. Most of the computation is completed onboard. Graph-based global optimization and loop closure are required for global states and these algorithms are computed offboard. The authors present results for exploration and mapping in unknown environments and also localization within a known map. They emphasize that the optical flow of the downward-pointing camera is essential for the system to function.

In this work we propose a new architecture to simplify some of the challenges that constrain GPS-denied aerial flight. In our approach, we combine visual graph-SLAM with a multiplicative extended Kalman filter (MEKF), using for inputs only a front-facing RGB-D camera, IMU and sonar altimeter, as shown in Figure 1.5. The unique aspect about the proposed approach is that we keep the position and yaw states of the MEKF *relative* to the current node in the map, rather than estimate states based in a global reference frame. Requiring global states incurs difficulties like the

[1]www.imav2011.org

need for additional states to incorporate relative position measurements [22, 26], waiting periods for global consistency [77], inclusion of place recognition and map optimization algorithms in the time-critical path [29, 69, 88] and additional logic to accommodate large jumps in pose when loop closures are applied [26].

We demonstrate in this paper that by maintaining relative information in the state, we can directly utilize vision-based measurements, we do not require feedback to the filter from computationally expensive loop closure or SLAM algorithms, and processes that are not essential for real-time estimation and control can be completed in the background. Additionally, the basis for the approach has been shown to scale well to large environments [64] and the images from the RGB-D camera represent a rich source of information for path planning and other high-level tasks.

The remainder of the chapter is outlined as follows. We explain the approach to relative navigation in Section 3.1. The software architecture is described in Section 3.2. We provide hardware results in Section 3.3. Then in Section 3.4, we summarize the work.

3.1 Relative Navigation Approach

Relative navigation refers to navigation with respect to a local reference frame. We propose that the local frame change as the vehicle moves through the environment, establishing a topological representation of the world using a pose graph [91]. The changes in the local frame occur based on the needs of the VO algorithm. The algorithm we use is *keyframe* based. Instead of comparing consecutive images, each current image is compared to a reference image, called a keyframe, to obtain the 6DoF change in pose. New keyframes are declared when the vehicle has moved further than a predetermined threshold from the previous keyframe and the overlap between images becomes too small for reliable matching. The local coordinate frames with respect to which the vehicle navigates are derived from the keyframes.

The map in Figure 3.1 illustrates the relative topological approach. The VO algorithm initializes a keyframe at node 1 and an edge is added between the global frame and the node frame once this information is known. The filter estimates the position and yaw states of the vehicle with respect to the local coordinate frame at node 1 as the vehicle travels. When the VO requires a new keyframe to maintain good performance, a new keyframe and node are declared at pose 2. An edge is added to the map using the relative states and covariance in the MEKF. The navigation then

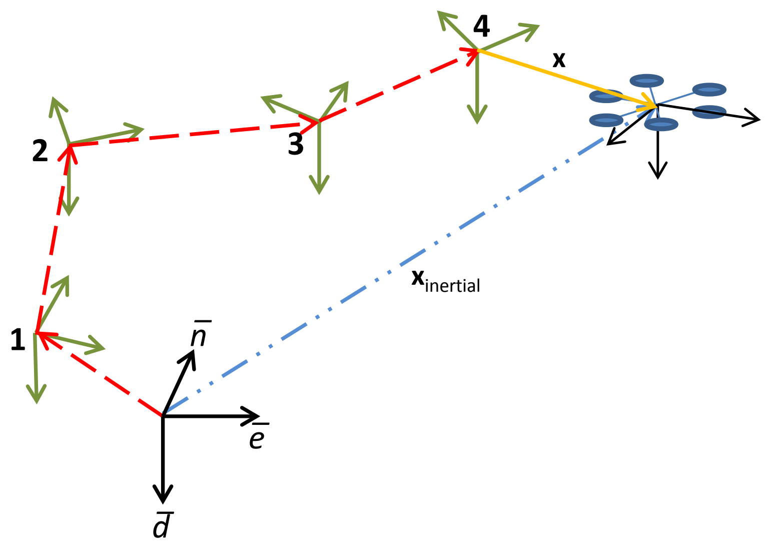

Figure 3.1: Relative navigation using nodes and edges. As the vehicle flies through the environment, nodes are created using the VO keyframes and the edges are defined between them using the relative states of the MEKF. The vehicle state is relative node four in this illustration.

continues with respect to node 2 by marginalizing out the old relative states and augmenting the state vector with new ones. This process continues as the vehicle moves through the environment, with new keyframes and nodes being declared as necessary and the MEKF changing the relative states each time a new keyframe is declared. As current images are compared to a keyframe, the position estimates will not drift when the vehicle is in hover. A vector chain of edges connects the hexacopter to the global reference frame. Global position and yaw for the vehicle can be estimated by first expressing all the constraints in the same coordinate frame and then summing all the edges and the current state.

This relative navigation approach has several key advantages: straightforward use of sensor information for state updates, easy creation of map edges using the filter state and covariance, and flexible use of global information.

Exteroceptive sensors provide relative information. In particular, the VO provides the change in 6DoF pose between the current and keyframe images. By expressing the VO result in the node coordinate frame, the position and attitude are updated directly in the filter. This simplification eliminates needing additional states in the filter or requiring VO measurements to update the velocity states.

Defining edges between consecutive nodes is a simple matter of saving the relative portions of the state and covariance just before a new node is created. The covariance can be used to compute a confidence measure of the current global position. For example, a path planning algorithm might use the combined covariances of the edges to indicate when estimates have drifted sufficiently to warrant a planned loop closure.

Our proposed relative approach offers more flexibility than a globally-based method. The system can fly reliably both with and without loop closure constraints that constrain drift and with and without global optimization. This is possible as the local navigation and control take place regardless of global changes within the map. Without loop closure it is clear that the map will drift and not remain globally consistent. However, the relative relationships between nodes maintain locally consistent topological and metric relationships between saved locations. Therefore, the map could be traversed, even back to the starting location, by using these relative relationships. This is also true when loop closure constraints are available and global optimization is not; we could then pursue a consistent but purely relative topological approach similar to that of [32]. Finally, by enabling both loop closure and global optimization we would be able to mimic the typical SLAM approach that provides globally consistent metric information of the environment.

3.2 Software Architecture

Figure 3.2 provides a visualization of the proposed relative navigation system. The system is divided into two halves, the front end and back end. The whole system is intended to be run onboard a hexacopter, as shown in Figure 1.5.

The front-end subsystem provides the critical processes to keep the hexacopter flying, including the VO, sensor fusion, control, and obstacle avoidance. Consequently, this system is given priority over the back end. All the components in the front end are based in the relative coordinate frame explained above.

The back-end subsystem maintains globally consistent information, including a global map and high-level, global objectives when desired. Notice how the only flow of information from the back end to the front end is from the high-level planner. This is in stark contrast to other solutions that have been developed, which require feedback from the computationally expensive recognition

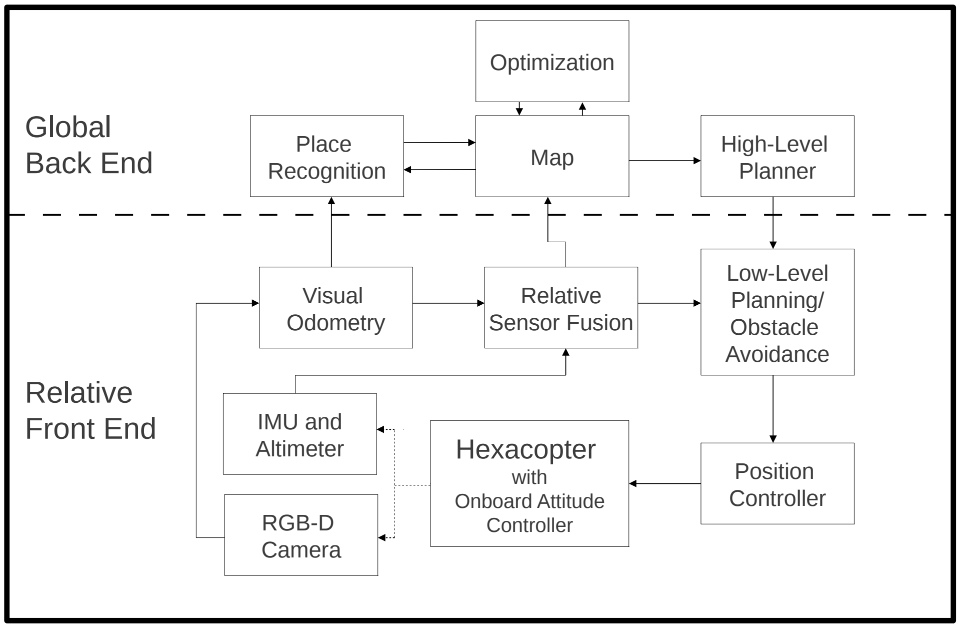

Figure 3.2: Software architecture for the proposed system. The front end provides relative navigation based on keyframes from VO algorithm. The back end provides a globally-consistent navigation solution. Notice that the only flow of information from the back end to the front end is provided by the high-level planner. Most current scenarios require feedback from the costly optimization and place recognition algorithms to the estimation and control. ROS provides the functionality represented by the arrows between components.

and optimization components. The relationship, illustrated in Figure 3.2, between the front and back ends is what allows the flexibility of this approach.

This separation between the front and back ends provides an added level of robustness to any changes in the pose graph. For example, when loops are closed and global optimization is employed, it is possible for large jumps in the global location to occur. These large jumps can cause problems with the real-time control of an air vehicle that employs globally-referenced states. As our vehicle navigates with respect to a local node, global optimization can continually make changes without causing harm to the real-time estimation and control. Another advantage is the potential to utilize other types of constraints in the map between nodes, also without effecting the real-time essential processes. Possibilities include any measurement constraints which aid in the understanding of the global or relative vehicle location, such as intermittent GPS measurements or semantic information [26].

The system is implemented using the Robot Operating System (ROS) [4]. In fact, messages within the ROS framework make up all the arrows in Figure 3.2 and each block is written as a ROS node/package. Below we briefly describe each block that makes up the proposed relative navigation approach.

3.2.1 Visual Odometry

As we discussed above, VO is the process of comparing two images to find the relative change in pose between them. We utilize keyframes in these comparisons, rather than consecutive images, to reduce the amount of drift. Good tutorials on implementing VO are found in [18, 19].

In [97], a robust motion estimation approach using an RGB-D camera is described. It is proposed that an RGB-D sensor provides three modalities that can be used to provide motion estimation solutions: a monocular camera which provides 2D RGB imagery, a range camera that produces 3D point clouds, and the combination of the two sensors giving depth information for each pixel of the image (RGB-D). The advantage of the approach is that information from one sensor may still be useful when it is not available in the other, enabling motion estimates in difficult areas for RGB-D cameras: outdoors, in low light, and in large open spaces. The approach, however, is not sufficiently mature for use on a flying platform. We have developed a VO algorithm that utilizes the 3D information from an RGB-D camera. We provide a quick summary of the algorithm, followed by typical results, below.

3D VO

First, sets of color and depth images are sent to the algorithm. The first image pair sent is designated as the keyframe image pair and all following image pairs are compared to this set until a new keyframe image is assigned. This occurs once the camera has moved 0.25 meters or 10 degrees in yaw from the location where the keyframe image was taken. This metric for changing keyframes was found experimentally by comparing motion estimates to the truth provided by the motion capture system. It is the point where motion estimates began to deteriorate because of the combination of the confined environment and the field of view of the camera. The metric could change in different environments.

On each image FAST features [115] and BRIEF descriptors [116] are extracted and the feature positions are corrected using the distortion information of the camera. A mask is used so that features may only be found in areas with valid depth information. The image is also binned to allow an even dispersion of features across the image. The number of features is currently capped at 750 for the 640 by 480 image sensor. The 3D point location $\mathbf{p} = (X\ Y\ Z)^\top$ for the 2D image feature $\bar{\mathbf{p}} = ()x\,y)^t op$ is found by looking up the depth Z in the depth image and using the projection equations

$$X = \frac{(x - c_x)Z}{f_x} \tag{3.1}$$
$$Y = \frac{(y - c_y)Z}{f_y}, \tag{3.2}$$

where c_x, c_y, f_x, and f_y are the intrinsic camera calibration parameters for the image center and focal points.

Next, correspondence between the current image features and the keyframe features are estimated using forward and backward constrained brute-force searches in a mutual consistency check [18]. The corresponding features are passed into RANSAC [117], which is then employed to find a pose motion estimate while eliminating outliers. We use a three point singular value decomposition (SVD) algorithm based on [118] as the motion model in RANSAC. The solution estimate provides the 6DoF rotation and translation between the keyframe and the current coordinate frames and is of the form

$$\mathbf{p}^c = \mathbf{R}^c_{key}\mathbf{p}^{key} + \mathbf{T}^c. \tag{3.3}$$

Where $\mathbf{p}^c$ and $\mathbf{p}^{key}$ are the current and keyframe 3D feature position vectors, $\mathbf{R}^c_{key}$ rotates points expressed in the keyframe coordinate frame into the current image coordinate frame, and T^c is the origin of the keyframe coordinate frame expressed in the current image coordinate frame.

Inliers are found for the sample solution by re-projecting the 3D keyframe features onto the current image plane using the sample solution and the intrinsic calibration parameters. To be considered a valid inlier, the pixel error distance between the locations of current image feature and the reprojected keyframe feature must be smaller than a threshold. Checking the error on the image provides improved results over solutions evaluated by error in 3D positions. The solution estimate with the highest inlier count is returned by the RANSAC algorithm.

3D VO Results

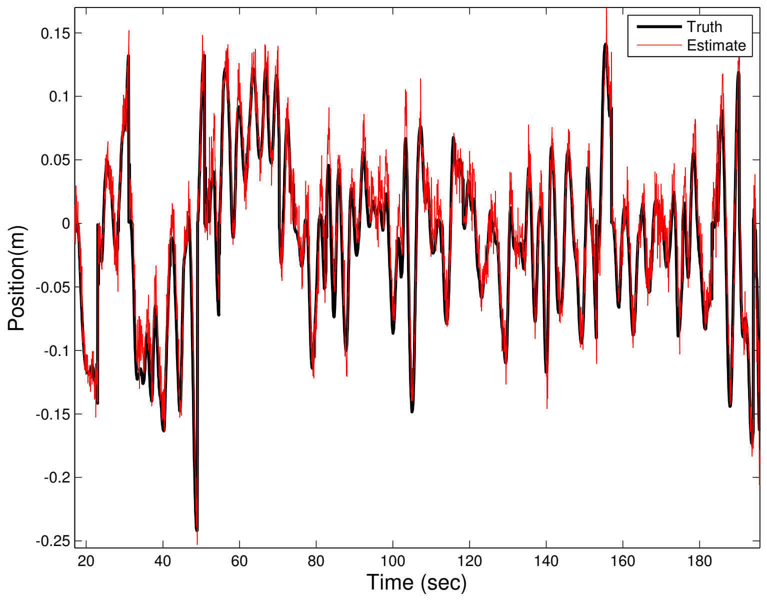

Figure 3.3: Relative camera $\vec{z}$ (out of plane) position comparison between truth and estimates. The discontinuities in the plots are due to new nodes being created, causing the truth and the estimates to jump to the new relative position. We express the global truth from the motion capture in the relative node coordinate frame for the comparison of these results. Results for the camera $\vec{x}$ and $\vec{y}$ positions are similar.

Figure 3.3 presents the camera $\vec{z}$ axis portion of the relative transformations between the current and keyframe images. The discontinuities are due to changing keyframes. Figure 3.4 presents the results for the rotation about the camera $\vec{y}$ axis. This angle corresponds to the vehicle yaw angle because of the change in axes. The algorithm actually outputs the change in rotation as a unit quaternion, which we have converted to Euler angles in this figure for ease of comparison.

Table 3.1 provides the RMS error in the relative transformations over the flight. These values are representative of results that we routinely achieve in the motion capture environment.

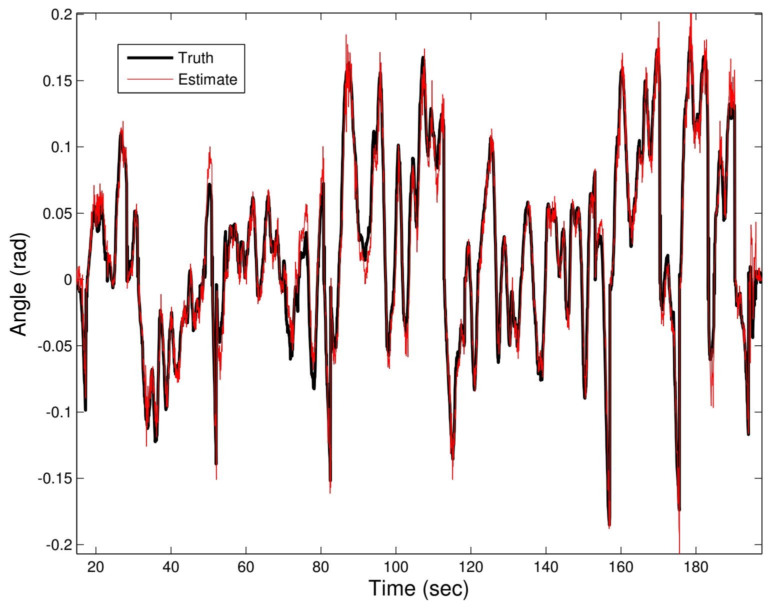

Figure 3.4: Comparison between truth and estimates for the rotation about the camera $\vec{y}$ axis, which is equivalent to the vehicle yaw angle because of the change in axes. Again, the discontinuities in the plots are due to new nodes being created, causing the truth and the estimates to jump to the new relative position. We express the global truth from the motion capture in the relative node coordinate frame for the comparison of these results. Results for the camera roll and yaw positions are similar.

Table 3.1: Presented are the RMS errors of VO estimates produced during a flight. The motion capture data is used as truth for the computation and is expressed in the relative coordinate frame. The estimates were produced in real time during the flight.

RMS Error in Motion Estimates	
Transformation	**RMS Error**
Camera $\vec{x}$ position	0.033 (m)
Camera $\vec{y}$ position	0.041 (m)
Camera $\vec{z}$ position	0.041 (m)
Rotation about camera $\vec{x}$	0.020 (rad)
Rotation about camera $\vec{y}$	0.017 (rad)
Rotation about camera $\vec{z}$	0.013 (rad)

3.2.2 MEKF Sensor Fusion

The sensor fusion is provided by a MEKF that has been designed specifically to function with the relative navigation approach and is detailed in Chapter 4. The MEKF is an indirect EKF, which means that the error in the state $\Delta\mathbf{x}$ and the covariance of the error are maintained in the filter rather than the best estimate $\hat{\mathbf{x}}$ and error covariance.

The true states $\mathbf{x}$ of the rotorcraft are defined as

$$\mathbf{x} = \left[\mathbf{p}^{n\top} \quad \mathbf{q}_n^{b\top} \quad \mathbf{v}^{b\top} \quad \beta^\top \quad \alpha^\top \quad \mathbf{q}_c^{b\top} \quad \mathbf{p}^{b\top}\right]^\top. \tag{3.4}$$

The position vector $\mathbf{p}^n$, relative to the current node, is the displacement of the body in the front f_j, right r_j, and down d_j directions with respect to node j. The quaternion $\mathbf{q}_n^b$ expresses the attitude of the body-fixed frame with respect to the node frame. The component of the quaternion for yaw is relative to the current node. $\mathbf{v}^b$ is the body-fixed frame velocity vector. The gyroscope bias vector is β. So far, we have only needed to estimate the accelerometer biases in the body $\mathbf{x}$ and $\mathbf{y}$ directions in α, the third direction could easily be added. The last two parameters in (3.4) represent the transformation from the body-fixed coordinate frame to the camera coordinate frame and can be optionally included in the state. Once refinements to the transformation are obtained, these estimates can be saved as constants and then removed.

The inputs to the model are the gyroscope measurements and the z accelerometer

$$\mathbf{u} = \left[p_{gyro} \quad q_{gyro} \quad r_{gyro} \quad z_{accel}\right]^\top. \tag{3.5}$$

The nonlinear equations for the states (3.4) are

$$\dot{\mathbf{p}}^n = \mathscr{R}^\top(\mathbf{q}_n^b)\mathbf{v}^b, \tag{3.6}$$

$$\dot{\mathbf{q}}_n^b = \frac{1}{2}\Omega\left(\mathbf{u}_{(1:3)} - \beta - \eta_\omega\right)\mathbf{q}_n^b, \tag{3.7}$$

$$\begin{aligned}\dot{\mathbf{v}}^b =& \mathbf{v}^b \times \left(\mathbf{u}_{(1:3)} - \beta - \eta_\omega\right) + \mathscr{R}(\mathbf{q}_n^b)\mathbf{g} \\ & - \frac{1}{m}\mathbf{M}\mathbf{v}^b + \mathbf{u}_{(4)}\vec{d}_j,\end{aligned} \tag{3.8}$$

$$\dot{\beta} = \eta_\beta \tag{3.9}$$

$$\dot{\alpha} = \eta_\alpha \tag{3.10}$$

$$\dot{\mathbf{q}}_c^b = \eta_{cq} \tag{3.11}$$

$$\dot{\mathbf{p}}^b = \eta_{cp}. \tag{3.12}$$

A rotation matrix $\mathscr{R}(\mathbf{q}_x^y)$ from a quaternion $\mathbf{q}_x^y$ rotates the vector $\mathbf{v}$, expressed in the frame x, into frame y. The operator

$$\Omega(\omega) = \begin{bmatrix} 0 & \omega_3 & -\omega_2 & \omega_1 \\ -\omega_3 & 0 & \omega_1 & \omega_2 \\ \omega_2 & -\omega_1 & 0 & \omega_3 \\ -\omega_1 & -\omega_2 & -\omega_3 & 0 \end{bmatrix} \tag{3.13}$$

assumes that the order of a quaternion it multiplies is of the form $\begin{bmatrix} q_x & q_y & q_z & q_w \end{bmatrix}^\top$. The noise η_ω is the zero-mean Gaussian noise in the measured gyroscopes from the inputs $\mathbf{u}$. The constant matrix $\mathbf{M}$ is

$$\mathbf{M} = \begin{bmatrix} \mu & 0 & 0 \\ 0 & \mu & 0 \\ 0 & 0 & 0 \end{bmatrix}, \tag{3.14}$$

and the constants $\mathbf{g}$ and μ are the gravity and drag coefficient respectively. An improved model of the hexacopter dynamics, contained in (3.8), which accounts for the rotor drag with coefficient μ, provides the ability to fully utilize the information contained in the accelerometer measurements [95]. As a consequence, estimation accuracy improves and the requirements for VO or any other exteroceptive measurement updates are reduced [109].

Error Dynamics

The error dynamics are used to propagate the error covariance matrix $\mathbf{P}$ and are derived from the nonlinear dynamics (3.6) through (3.12). The length of the error state is reduced by one for each quaternion when compared to the length of the true state. The 20-element relative-error state is

$$\Delta\mathbf{x} = \begin{bmatrix} \delta\mathbf{p}^{n\top} & \delta\theta_n^{b\top} & \delta\mathbf{v}^{b\top} & \delta\beta^\top & \delta\alpha^\top & \delta\theta_c^{b\top} & \delta\mathbf{p}^{b\top} \end{bmatrix}^\top. \tag{3.15}$$

The error dynamics can be linearized and result in the following linear model

$$\dot{\Delta\mathbf{x}} = \mathbf{A}\Delta\mathbf{x} + \mathbf{B}\mathbf{u}, \tag{3.16}$$

where $\mathbf{A}$ is the Jacobian of the error dynamics with respect to the error state $\Delta\mathbf{x}$ and $\mathbf{B}$ is the Jacobian of the error dynamics with respect to the input $\mathbf{u}$.

Prediction

We do not maintain a true indirect filter as we require the estimated states for control feedback. Instead, we keep track of the estimated state $\hat{\mathbf{x}}$ and the covariance $\hat{\mathbf{P}}$ of the error state $\Delta\hat{\mathbf{x}}$. During the prediction step, the estimated covariance is propagated forward by numerically integrating the equation

$$\dot{\hat{\mathbf{P}}} = \mathbf{A}\hat{\mathbf{P}} + \hat{\mathbf{P}}\mathbf{A}^\top + \gamma\left(\mathbf{B}\mathbf{G}\mathbf{B}^\top + \mathbf{Q}\right), \tag{3.17}$$

where the matrices $\mathbf{A}$ and $\mathbf{B}$ are in (3.16), γ is a tuning parameter, $\mathbf{G}$ is a diagonal matrix of the measured covariance on the inputs (3.5) and $\mathbf{Q}$ is the process noise covariance which represents modeling error and disturbances. The estimated states are propagated forward by numerically integrating the nonlinear equations of motion, (3.6) through (3.12).

Measurement Updates

We update the filter using altimeter, accelerometer, motion-estimation position, and motion-estimation orientation measurements. The motion estimation measurements can come from the 3D VO, the 2D VO and/or the scan matching. In this implementation we treat the position and orientation motion estimation measurements updates separately. This is possible because we account for the contribution of the rotational uncertainty in the position covariance when the covariances of the measurements are generated.

Delayed View-Matching Updates

An additional challenge with the motion estimation measurement updates is that they are delayed. The stochastic delay is due to the requisite image-processing time. Thus, computing the measurement updates requires a few additional steps. The state and covariance must first be restored to the time the image was taken; the measurement updates are applied; then the state and covariance must be repropagated back to the current time by re-applying the prediction and measurement updates at their respective timesteps. The state, covariance, IMU, and altimeter information are saved at each timestep to accommodate this requirement.

Augment and Marginalize the Relative State

When a new node is created by the view-matching algorithm, the relative portions of the state and covariance must change. The states that change are the positions in $\hat{\mathbf{p}}^n$ and the yaw contained in $\hat{\mathbf{q}}_n^b$. The positions are simply replaced with zeros. The quaternion in the state must maintain the same pitch and roll but the yaw must be zeroed out. We use the relationship between quaternions and Euler angles [119] to adjust the state and covariance appropriately.

3.2.3 Relative Planning/Obstacle Avoidance

The low level planner provides paths for the hexacopter to follow through the environment in the relative frame. The plans are recomputed frequently enough for the vehicle to avoid static and slow-moving obstacles, like a person walking at a casual pace. Point cloud data from the RGB-D sensor is used to create a cost map [120] of the 3D environment that is then projected onto the node $f_j - r_j$ plane, as shown in Figure 3.5. The cost map is expressed in the relative node frame explained above. Given a goal location in the relative coordinate system, a path through the environment is computed using Dijkstra's algorithm [121]. The goal location is the only information received by the front end from the back-end subsystem, shown in Figure 3.2. The path is expressed in the relative coordinate system.

3.2.4 Position Control

We have modified the position controller detailed in [122] to provide control based on waypoints in the relative node coordinate frame and to include integral control for more robustness. The control algorithm utilizes a change of variables on the inputs of the model to eliminate nonlinearities and a linear quadratic regulator (LQR) provides the feedback control. The approach to follow waypoints is based on the procedure outlined in [123], adapted for rotorcraft. The waypoints are expressed in the current node frame when sent by the planning algorithm.

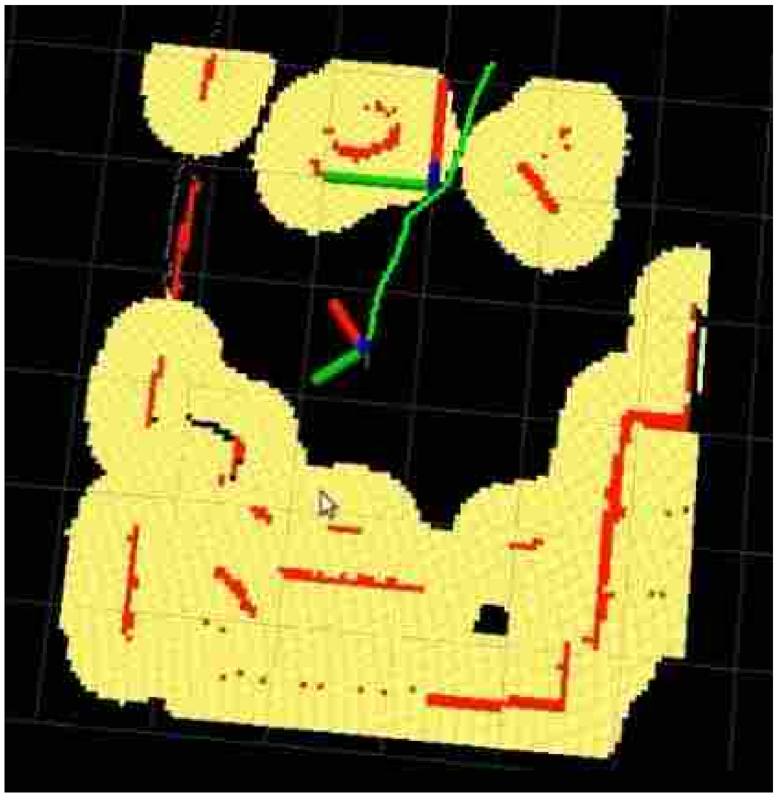

Figure 3.5: Visualization of the cost map and path planner. Point cloud information was provided to the relative path planner as the camera was moved around the CAVE. The red objects are detected obstacles, the yellow areas are inflated costs around the obstacles, and the green line is the computed path, based on a goal location chosen by a user.

One challenge for the control is the change of coordinate systems when a new node is created. The sensor fusion algorithm changes the coordinate system of the relative states as soon as the keyframe image is received. The path planner, however, does not instantly generate a new path for the new reference frame. Consequently the control algorithm must apply the relative transformation provided by the map edge to the old path. Each waypoint $wp[i]$ in the path is transformed using

$$\mathbf{wp}_{new}[i] = \mathscr{L}\left(\mathbf{q}_j^{j+1}\right)\left(\mathbf{wp}[i] - \mathbf{p}^{j+1}\right), \tag{3.18}$$

where $\mathscr{L}\left(\mathbf{q}_j^{j+1}\right)$ is the quaternion rotational operator (equivalent to a rotation matrix) for rotating points in the j-th node frame into the $j+1$-th node frame, and $\mathbf{p}^{j+1}$ is the translation to the $j+1$-th node from the j-th node, expressed in the j-th node frame.

3.2.5 Map

The map used in this work is a collection of nodes and edges in a relative topological pose graph, illustrated in Figure 3.6. The map is flexible as it can be globally referenced through optimization but it is originally based on the relative transformations provided by the motion estimation. New nodes are created with each new keyframe. Edges are added between temporally and spatially consecutive keyframes.

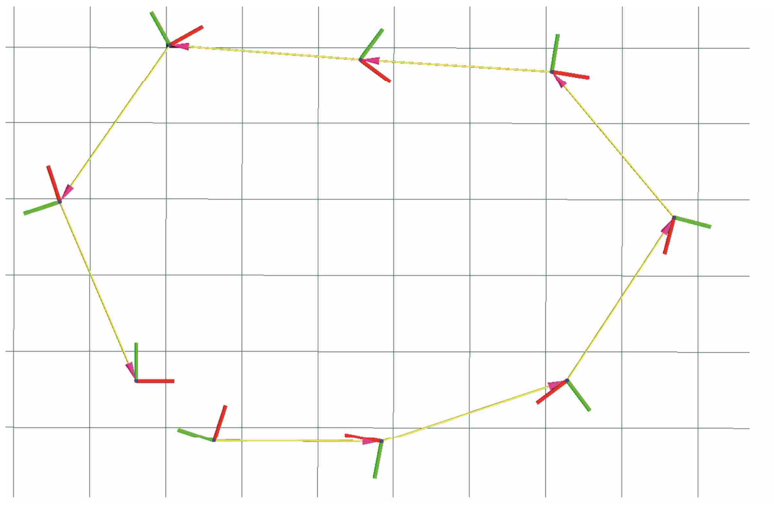

Figure 3.6: A simple pose-graph map representation. Each of the nodes, illustrated by the local frames, contains the RGB-D keyframe images and the global pose estimates. The relative transformations between each of the nodes are provided by the MEKF.

A node is described, ultimately, by a keyframe RGB and depth image pair. Attached to the keyframe pair are the estimates of relative and global position and orientation, yet the image encodes the true instantaneous location of the vehicle. A relative local coordinate frame is defined as part of the node, based on the position and heading of the vehicle when the keyframe is taken, to enable navigation relative to the node.

Edges in the graph represent the estimated relative transformations between nodes. We currently only consider edges from the odometry but we are working to include other constraints, such as those from visual recognition loop closures and intermittent GPS measurements. The odometry edges are created using the MEKF, based on the measurements from the robust motion estimation algorithm. When a new node is received by the estimator, the old relative portions of the state and covariance are marginalized out and saved as the edge between the old and new nodes.

3.2.6 Place Recognition

Place recognition provides the capability to recognize when the current keyframe is already part of the map. Once the algorithm recognizes a match, a loop-closure constraint can be added to the map using one of the motion-estimation algorithms. Loop-closure constraints are essential to providing a topological-consistent map as they constrain the drift in the map caused by odometry errors.

Place recognition is completed by comparing images to one another to find close matches [34, 36]. Each keyframe image in the map is assigned visual words, from a previously calculated visual vocabulary, based on the feature information in the image. Then the map is searched using the words to find images that contain the same information. Once several images are suggested by the algorithm as having a high probability of being the same location, a geometric consistency check is made to eliminate any false positive matches. The 6DoF loop-closure constraint is created by comparing the matching images using a VO algorithm.

3.2.7 Back-end Optimization

The role of nonlinear optimization is to iteratively refine the edges in the map to produce a globally consistent map when it is desired. Because of the flexibility of the relative navigation approach, this can be completed either offline after a flight, or in real time as a background process. In most navigation approaches, the sensor fusion relies on the revised global estimates, causing the computationally heavy optimization to be a part of the time-critical path that enables flight.

In [124], a new optimization approach is introduced, which focuses on the relative transformations between nodes rather than only on the global pose estimates, as is typically done. As a result, the algorithm provides improved estimates of the global poses and relative transformations in less computational time than the state-of-the-art algorithm g2o [63].

3.2.8 High Level Planner

The role of the high-level planner is to provide capabilities such as exploration, target following or other higher-level tasks for the hexacopter system. The algorithm is provided an estimate of the map and the location of the hexacopter, as well as the current relative coordinate system in use by the front-end subsystem. Directions are then provided to the low-level planner in the form of goal locations in the current relative coordinate system. This setup allows the front-end subsystem flexibility. It does not need global information and it is allowed to create its own paths so that obstacles can be avoided. This node is a subject for future work and we plan to leverage prior work of high-level planning for fixed-wing UAVs.

3.3 Experimental Setup and Results

The hardware setup for the results of this chapter was the same as described in Section 1.1. Truth data from a motion-capture system is only used to initialize the global position of the vehicle and in the comparisons made in the figures below. The relative MEKF runs at 100 Hz, the update rate of the IMU. Measurement updates for the altimeter and the visual odometry algorithm are applied at 40 Hz and 15 Hz, respectively. All the processing is performed onboard. During the experiments presented below, the CPU usage averaged at about 40%, measured using the linux "top" command.

We present results for an autonomous hover which demonstrate the performance of the estimator and control algorithms. The estimates are compared to truth and the control maintains the vehicle in a hover about a fixed global location. We also show the true and estimated 3D positions of the vehicle while following a path.

3.3.1 Hover Results

Figures 3.7 through 3.9 demonstrate the performance of some of the state estimates of the filter compared to truth during a flight with the state estimates in the control loop. The vehicle was commanded to hover at a spot 1 m above the take-off location. During this flight 3D information from the camera was consistently available, consequently the 3D VO was used by the motion estimation algorithm.

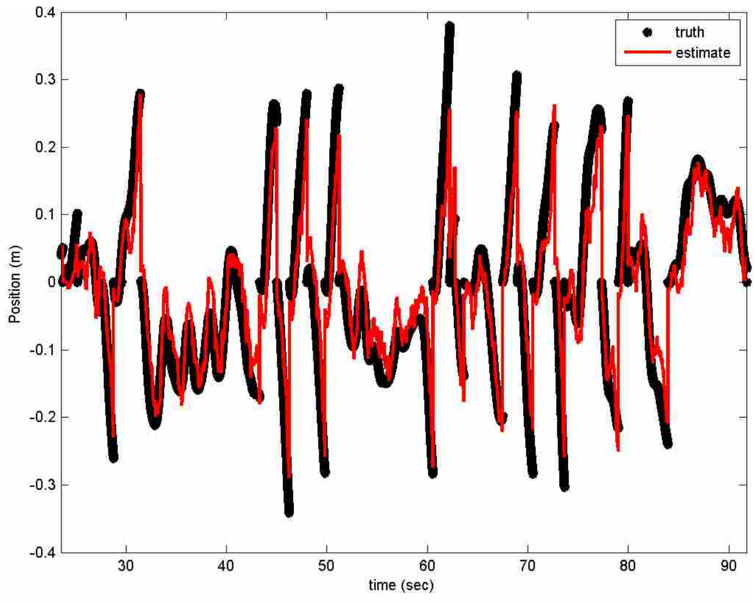

Figure 3.7: Relative right position r truth and estimates comparison. This data is from an estimates-in-the-loop controlled hover flight. The discontinuities in the plots are due to new nodes being created, causing the truth and the estimates to "jump" to the new relative position. We express the global truth from the motion capture in the relative node coordinate frame for the comparison of these results. Results for the relative front and down positions are similar.

In Figure 3.7 we see the results for the relative right position r, with respect to the current node. There were 30 new nodes created during this autonomous flight. We note that all of the state estimates transition between these coordinate frame changes without difficulty. The body-fixed frame side velocity v results are depicted in Figure 3.8. The estimates track the truth, even though the magnitude of the speed is small. The y component of the quaternion $\mathbf{q}_n^b$ is shown in Figure 3.9. The y quaternion roughly corresponds to the pitch angle of the hexacopter for this flight.

Table 3.2: The standard deviations of the hover error in global coordinates.

Standard Deviation of Hover Error	
Direction	**Standard Deviation**
global north (n)	0.070
global east (e)	0.079
global down (d)	0.101

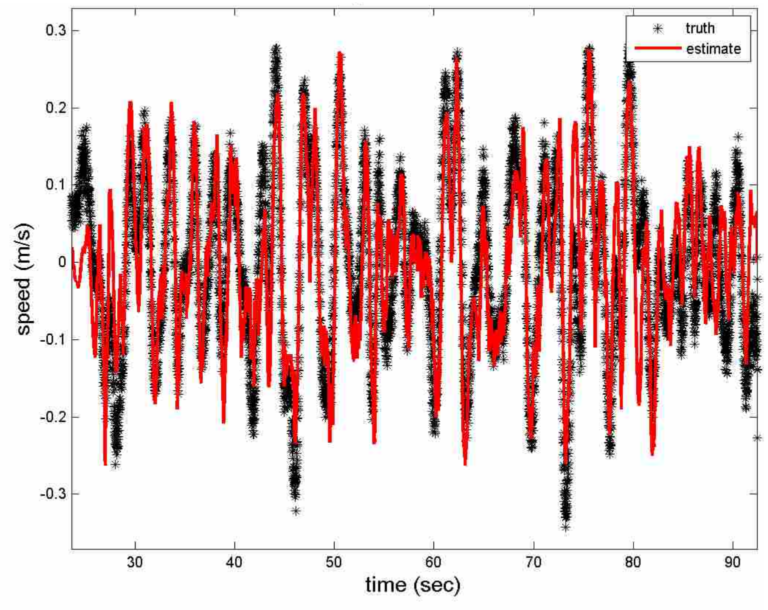

Figure 3.8: Body-fixed frame side velocity v truth and estimate comparison. Notice that there are no discontinuities, as the body-frame velocity is not relative. Results for the front and down body-fixed velocities are similar.

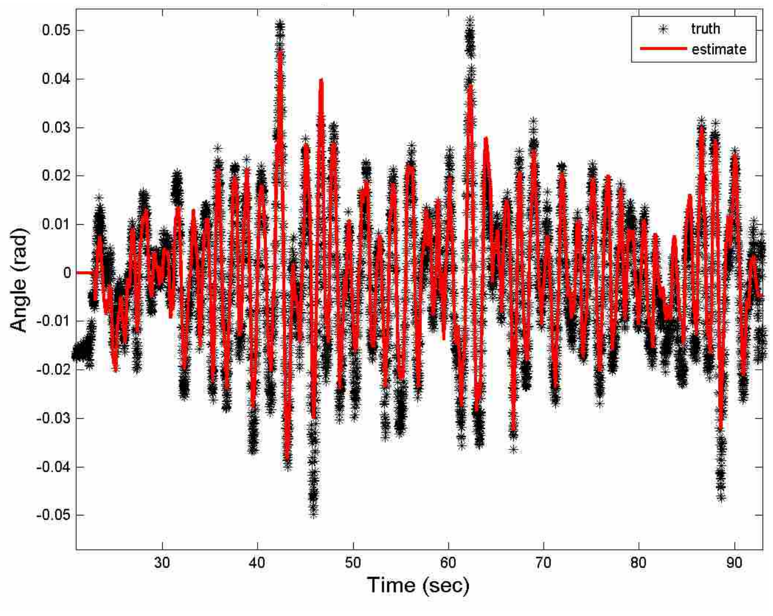

Figure 3.9: The y component of the quaternion $\mathbf{q}_n^b$, which is approximately the pitch angle of the hexacopter for this flight, comparison of truth and estimates. There are not any discontinuities, as this portion of the quaternion is not relative.

Table 3.2 provides the standard deviations of the hover error through the flight. Notice that even though the vehicle is navigating using relative states, it can stabilize around a global location quite well. The down performance is poor because we are close to the payload limit of the hexacopter vehicle. Reducing the weight of the prototype platform should improve these results.

3.3.2 Path Results

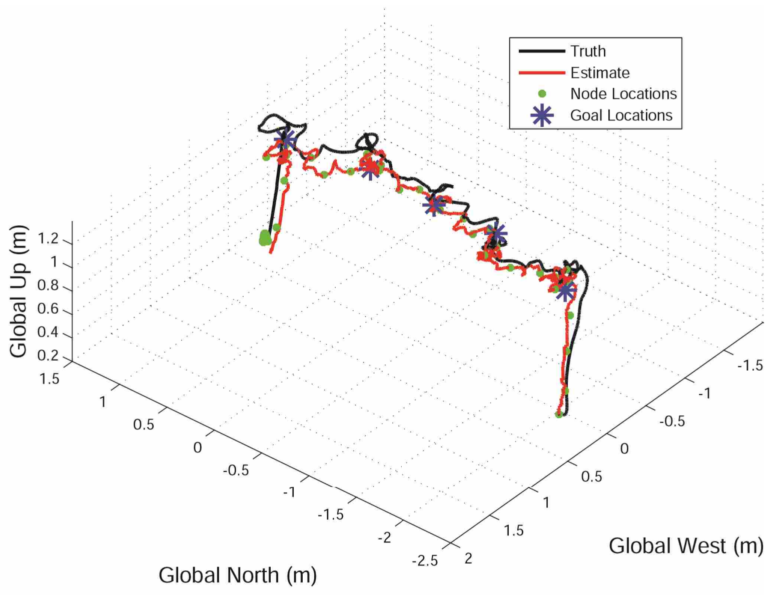

Figure 3.10: The 3D path of a flight within the motion capture environment. We show the true path, the global estimate computed by summing the relative edges and the current state at each timestep, the node locations estimates, and the global positions of the relative goal points. Notice that even though the estimates drift globally, the vehicle arrives at each of the goal locations. This is possible since all the front end functionality is based on the relative system.

Figures 3.10 and 3.11 show results for an autonomous, goal-directed flight of the vehicle. The vehicle is performing all of the tasks of the front end sub-system described in Figure 3.2, with all of the computation being completed onboard. The flight is short to permit the use of the motion capture truth data for comparison. The vehicle was first commanded to hover one meter above the starting location and then it was directed using the goal locations shown in the figures. Recall that the estimates, control, path planning, and goal locations are all originally relative to the current node in the graph. We have converted them into global estimates for display and comparison. The

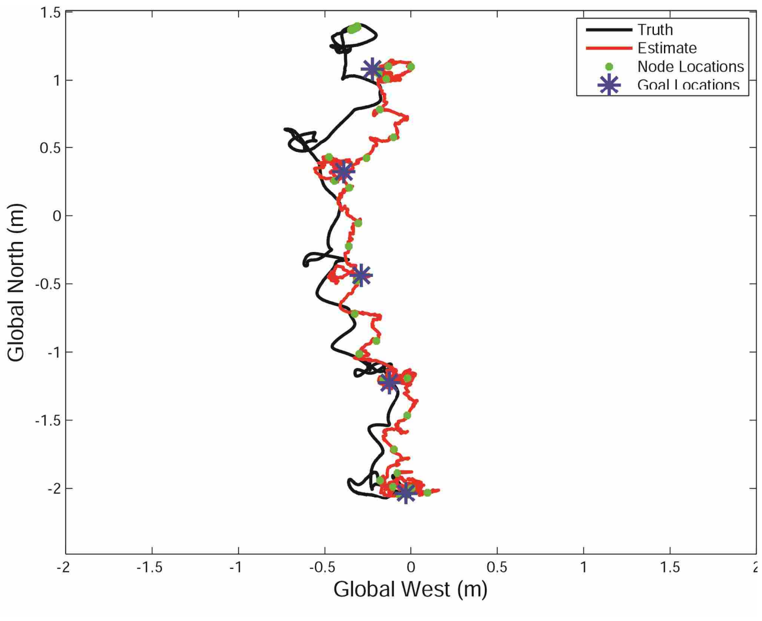

Figure 3.11: The top view of the 3D path of a flight within the motion capture environment. Notice that even though the estimates drift globally, the vehicle arrives at each of the goal locations. This is possible since all the front end functionality is based on the relative system.

estimated global node locations are shown as green points along the estimated path. There is drift in the global locations as we are only conducting relative flights at this point.

3.3.3 Long Hallway Results

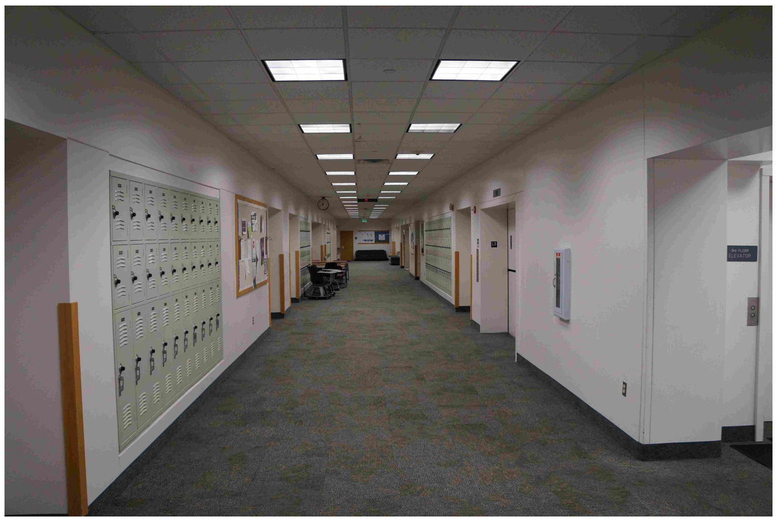

Figure 3.12: The main hallway in the Crabtree building on BYU campus.

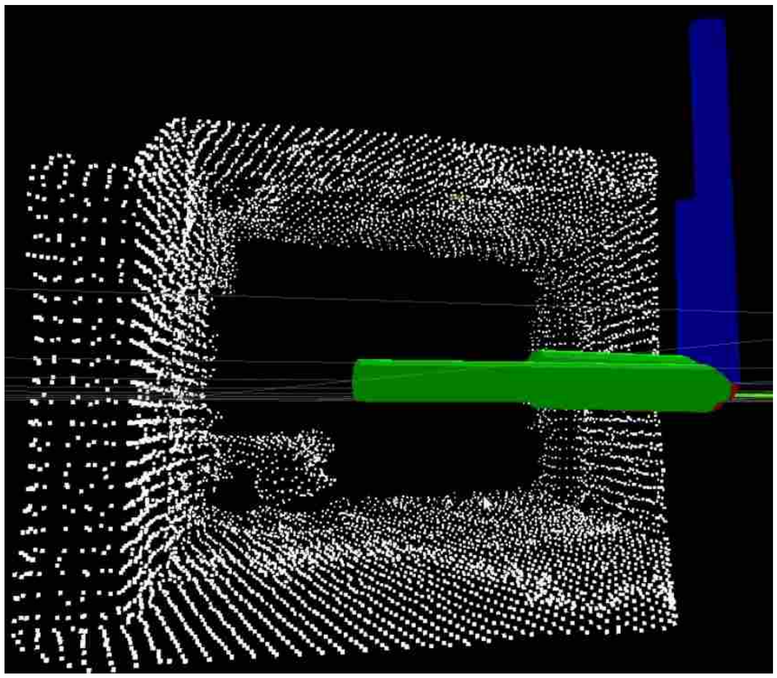

Figure 3.13: The 3D point cloud produced by the RGB-D camera while the hexacopter was moving down the hallway. The valid information for use in the VO is only on the periphery of the image.

After many experiments within the room equipped with the motion capture system to verify the accuracy of the algorithms, we tested the system in a larger environment. We flew the hexacopter down the main hallway of the third floor of the Crabtree building on campus,[2] see Figure 3.12. A large hallway is a challenging environment for the VO, as 3D information is only available on the periphery of the sensor: on the floor, walls and ceiling. Figure 3.13 shows a typical point cloud obtained while the hexacopter was moving down the hallway.

Figure 3.14 shows a top view of the global estimates of the path of the vehicle. The original reference frame was closely aligned with the hallway, so the vehicle drifted between 1 to 2 m laterally. As the vehicle navigates relative to the keyframes, this drift is not concerning.

3.3.4 Flight Down Three Hallways

The next step in testing was to maneuver down hallways that require turns[3]. Results for a flight on the third floor of the Wilkinson Student Center on campus are shown in Figure 3.15. In

[2]A video of this flight can be seen on the MAGICC Lab YouTube channel: `http://youtu.be/PMYwPihUKIM`

[3]I appreciate all the help I received to obtain these results. I could not have done this without my undergraduate "minions": Dallin, David, Dan, Gary, and Craig! Thanks guys!

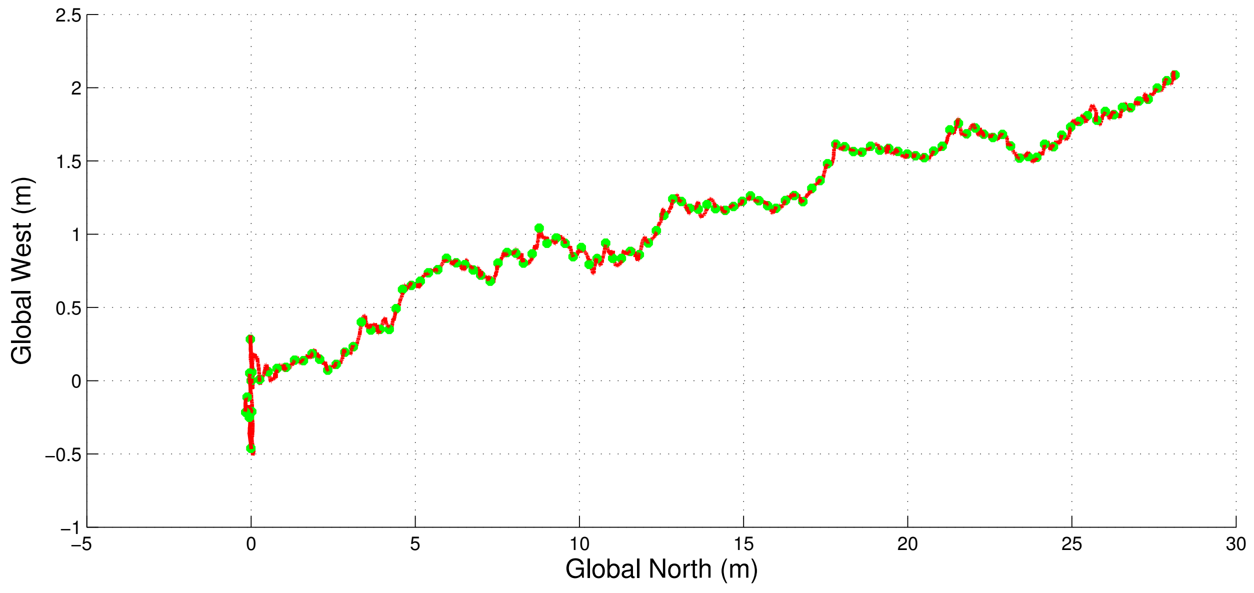

Figure 3.14: The top view of the global position estimates. Note the difference in scale. These are dead-reckoning results, as optimization and loop closure are not enabled. Lateral (west) drift is between one and two meters. The green dots denote the keyframe images created along the way.

black, we show the basic dimensions of the hallways, so that an idea of the drift in the flight can be observed. We set goal locations near the center of the hallways. Notice that there is very little drift in the global yaw, otherwise, the path would not be so square. Recall that again, these are dead-reckoning results.

3.4 Summary

A relative, vision-based framework, like the approach described in this chapter, is an important step in furthering the capabilities of indoor aerial navigation. Current approaches that require globally-referenced states often suffer deficiencies from the need for additional state elements to incorporate relative measurements, waiting periods to process global consistency, inclusion of place recognition and map optimization algorithms in the time-critical path, or schemes to accommodate large jumps in pose when loop closures are applied.

Utilizing a relative approach allows more flexibility as the critical, real-time processes of localization and control do not depend on computationally-demanding optimization and loop-closure processes. Relative exteroceptive measurement updates are supported natively in the proposed MEKF and front-facing keyframes provide a rich source of information for path planning. The

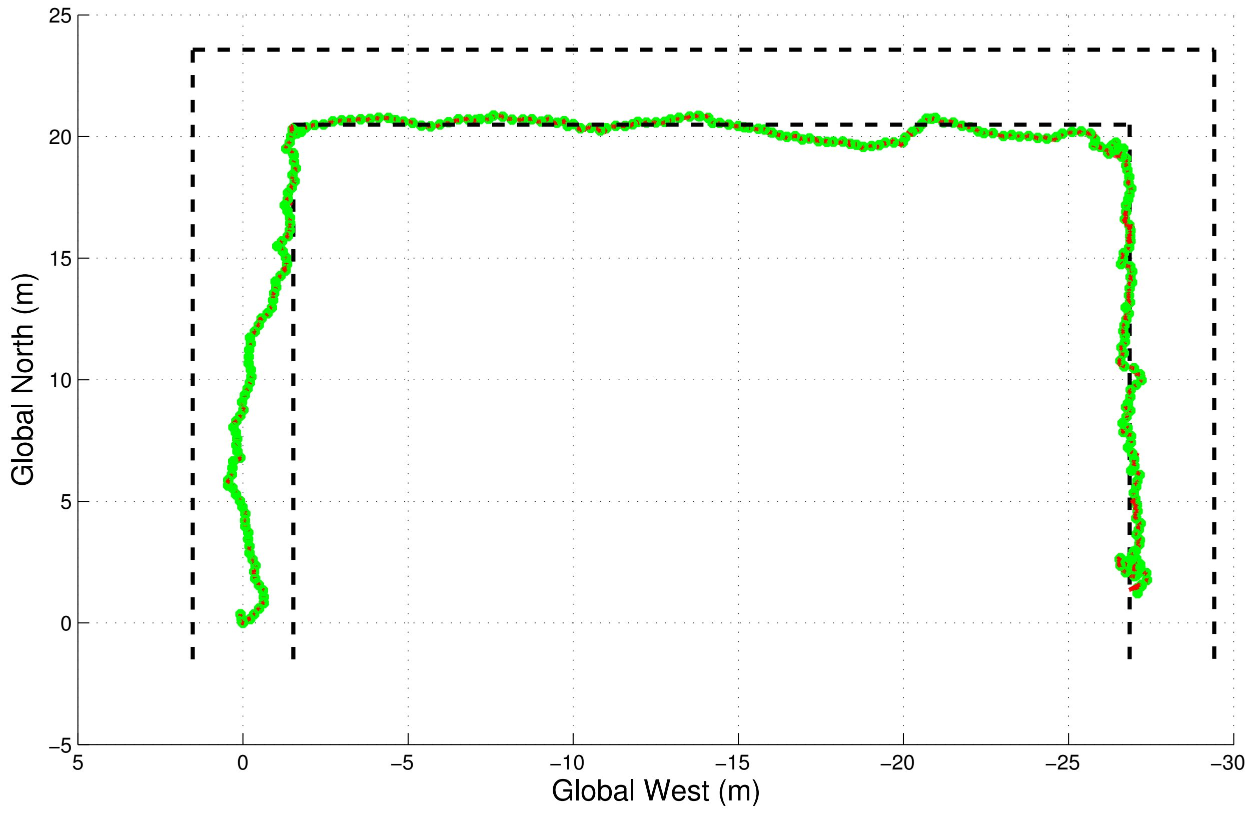

Figure 3.15: The top view of the global position estimates for the flight down three hallways in the Wilkinson Student Center. These are dead-reckoning results, as optimization and loop closure are not enabled. From this figure, we estimate the drift in north and west to be between one and two meters in each direction. The green dots denote the keyframe images created along the way.

graph map also provides potential support for a variety of constraints, such as intermittent GPS and semantic information.

CHAPTER 4. RELATIVE MULTIPLICATIVE EXTENDED KALMAN FILTER

Sensor fusion and localization are important aspects of navigation in unknown, GPS-denied environments, where global measurements and a priori map information are unavailable. Aerial flight using small aircraft like quadrotors is an additional challenge because of limited payload capacity for onboard sensors and computational resources, and the need to control the vehicle's fast dynamics using the state estimates. Unlike ground robots, quadrotors cannot afford to pause in one place until complex algorithms finish processing and state estimates converge for navigation to continue. Accurate and fast state estimates are critical to maintain control of the vehicle and provide quality sensor information.

To date, successful implementations providing GPS-denied autonomous quadrotor flight, like those outlined in [26,29,69–71,77,85,88], typically follow a general approach for localization and mapping. This approach, outlined in [69], enabled the first fully-autonomous quadrotor with the ability to fly using only onboard sensors in a priori unknown environments. Using this original approach the autonomous vehicle must: estimate motion using exteroceptive sensors, fuse motion estimates with other sensor information to form global state estimates, and employ some type of simultaneous localization and mapping (SLAM) to provide loop closure and global consistency.

Incremental motion of the vehicle is computed with an exteroceptive sensor and a motion estimation algorithm. The original approach for GPS-denied autonomous quadrotor flight utilized a planar laser scanner with a fast scan-matching algorithm [69]. A similar method is used in [71]. Planar laser-scanner implementations, however, require strict assumptions regarding the nature of the environment. Six-degree-of-freedom (6DoF) motion estimation using machine vision, as used in the approaches in [22,29,77,88], is advantageous due to a camera sensor's low cost, low power requirements, and light weight. Additionally, high-quality solutions can be obtained using fewer assumptions about the environment. Vision is, however, limited due to requirements for appropriate lighting and a sufficient number of features in the environment. Visual implementations usually employ a version of visual odometry (VO) [17], although a modified version of parallel tracking and mapping (PTAM) [21] is used in [85].

We note that the exteroceptive motion estimates are relative measurements between images or scans. Various approaches are employed to convert the relative measurements into the requisite global form. One method is to sum all previous motion estimates and form a pseudo-global measurement [69]. Another is to treat the measurement as an average velocity over the duration between sensor messages [29,88]. Stochastic cloning [22,26,86] is a more valid option as the state is cloned to more appropriately handle the cross-correlations that are necessary to form a correct global measurement update.

Sensor fusion combines exteroceptive motion measurements with IMU information for improved global state estimates at the fast rate needed for feedback control of the quadrotor. Typically some form of an extended Kalman filter (EKF) is employed and the accelerometer and gyroscope measurements are used in the filter propagation step [29,69,71]. In contrast, the approach detailed in [88] uses a simple low-pass filter for sensor fusion. One difficultly noted in many of these approaches is the lack of a good velocity measurement to aid in the sensor fusion [68,69,71,88]. This issue is discussed in [95] and an improved quadrotor dynamic model is shown to provide body-fixed velocity information using only accelerometer measurements.

Global state estimates of the vehicle will drift if only relative motion and IMU measurements are used. SLAM is a solution that provides the capability to maintain a map and to perform loop closure to eliminate drift and obtain globally consistent state estimates. The use of the pose graph and nonlinear optimization tools from the graph-SLAM paradigm [40,41] and visual place recognition to find possible loop closures [34] is increasingly common. Graph SLAM provides flexibility in working with the sensor fusion and ease with which place recognition loop closures may be applied to the map. Visual place recognition algorithms recognize previously visited locations in the map using only visual information. Because of the need for globally metric information, a majority of the approaches require feedback from the SLAM algorithms as a measurement in the sensor fusion.

One problem with the general approach, which was noted in [70], is the requirement of globally metric information for the maps of the environment and states of the vehicle. Requiring global states incurs difficulties such as the need for additional states to incorporate relative position measurements [22, 26], waiting periods for global consistency [77], dependence on computationally-expensive optimization and loop closure feedback for localization [29,70,88] and logic to accommodate large jumps in pose when loop closures are applied [26].

In [79,96], the authors propose that a vehicle should navigate using a *relative* formulation of the vehicle state, rather than a global one. Similar to the general approach, they use a combination of graph SLAM and an EKF to provide mapping and sensor fusion. The map is a pose graph, with images from the onboard camera as key components of the nodes. The EKF provides estimates at the high rate required for feedback control of the vehicle. The difference is that the position and yaw states of the EKF are defined with respect to the current node in the map, rather than to a global origin. Relative-state information affords many advantages, such as the ability to directly utilize relative exteroceptive measurements, elimination of required feedback to the filter from computationally-expensive SLAM algorithms, easy creation of map edges using the filter state and covariance, and flexible use of global information.

The main contribution of this chapter is the development of a multiplicative extended Kalman filter (MEKF) that uses an improved rotorcraft model [95] to provide relative, rather than global, state estimates. Another contribution is a more detailed development of the relative navigation approach originally described in [96] and the relative filtering approach described in [79]. Additionally, we verify the results of the relative estimation approach with hardware flight tests accompanied by comparisons to motion capture truth data. We also provide flight results with estimates in the control loop.

We utilize a hexacopter vehicle from MikroKopter, pictured in Figure 1.5 and discussed in Section 1.1, as the platform for experiments. An IMU, sonar altimeter, and front-facing RGB-D camera are the only sensors used for the sensor fusion and mapping. All computation is completed onboard using a small computer to perform the necessary processing.

The remainder of the chapter is outlined as follows. We explain the approach to relative navigation in Section 4.1. The model and MEKF are derived in Section 4.2. The results are described in Section 4.3. And we summarize the work in Section 4.4.

4.1 Relative Navigation Approach

Relative navigation refers to navigation with respect to a local reference frame. We propose that the local frame change as the vehicle moves through the environment, establishing a topological representation of the world using a pose graph [91]. The changes in the local frame occur based on the needs of a robust VO algorithm, described in Section 3.2.1. The algorithm is *keyframe* based. Instead of comparing consecutive images, each current image is compared to a reference image, called a keyframe, to obtain the 6DoF change in pose. New keyframes are declared when the vehicle has moved further than a predetermined threshold from the previous keyframe and the overlap between images becomes too small for reliable matching. The local coordinate frames, with respect to which the vehicle navigates, are derived from the keyframes.

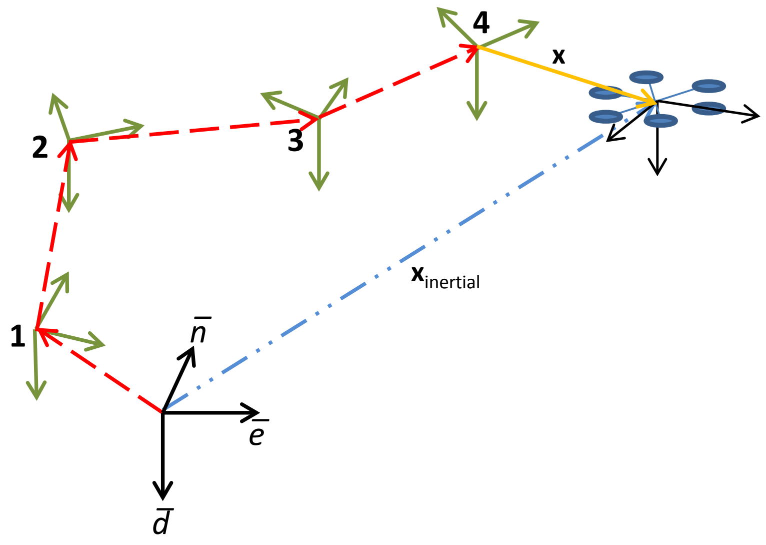

Figure 4.1: Relative navigation using nodes and edges. As the vehicle flies through the environment, nodes are created using the VO keyframes and edges are defined between them using the relative states of the MEKF. The vehicle state is relative to the current node (node 4). The global state of the vehicle can be found by summing the edges and the current state.

The map in Figure 4.1 illustrates the relative topological approach. The VO algorithm initializes a keyframe at node 1 and an edge is added between the global frame and the node frame once this information is known. The filter estimates the position and yaw states of the vehicle with respect to the local coordinate frame at node 1 as the vehicle travels. When the VO requires a new keyframe to maintain good performance, a new keyframe and node are declared at pose 2. An edge is added to the map using the relative states and covariance in the MEKF. The navigation then continues with respect to node 2 by marginalizing out the old relative states and augmenting the state vector with new ones. This process continues as the vehicle moves through the environment, with new keyframes and nodes being declared as necessary and the MEKF changing the relative states each time a new keyframe is declared. As current images are compared to a keyframe, the position estimates will not drift as long as the keyframe does not change. A vector chain of edges connects the hexacopter to the global reference frame. Global position and yaw for the vehicle can be estimated by summing all the edges and the current state.

This relative navigation approach has several key advantages: direct use of sensor information for state updates, straightforward creation of map edges using the filter state and covariance, and flexible use of global information. We discuss below how each of these is helpful for autonomous aerial navigation.

Exteroceptive sensors provide relative information. In particular, the VO provides the change in 6DoF pose between the current and keyframe images. By expressing the VO result in the node coordinate frame, the position and attitude are updated directly in the filter. This simplification eliminates the need for additional states in the filter and does not require VO measurements to update the velocity states.

Defining edges between consecutive nodes is a simple matter of saving the relative portions of the state and covariance just before a new node is created. The covariances from each edge could be combined and then used for a confidence measure of the current global position. For example, a path planning algorithm might use the combined covariances of the edges to indicate when estimates have drifted sufficiently to warrant a planned loop closure.

Our proposed relative approach offers more flexibility than a globally-based method. A front-end subsystem provides the critical processes to keep the hexacopter flying, including the VO, sensor fusion, control, and obstacle avoidance. All the components in the front end are based in the relative coordinate frame explained above. A back-end subsystem will maintain globally consistent information, including a global map and high-level, global objectives when desired. The only information provided by the back end or by human operators are goal locations expressed in the current node reference frame. This is in contrast to other solutions that have been developed that require feedback from the computationally expensive visual place recognition and optimization components. Additional details on the architecture of the approach are found in [96].

This separation between the time-critical front end and globally consistent back end provides the flexibility mentioned. The system can fly reliably either with or without loop-closure constraints that constrain drift, and with or without global optimization. This is possible because the local navigation and control take place regardless of global changes within the map. Without loop closure it is clear that the map will drift and not remain globally consistent. The relative relationships between nodes, however, maintain locally consistent topological and metric relationships between saved locations. Therefore, the map could be traversed, even back to the starting location, by using these relative relationships. This is also true when loop closure constraints are available and global optimization is not; we could then pursue a consistent but purely relative topological approach similar to that of [32]. Finally, by enabling both loop closure and global optimization we could mimic the typical SLAM approach that provides globally consistent metric information of the environment.

4.1.1 Node Frame

The nodes in the pose graph contain keyframe images and they represent locations in the environment where those images are taken. Each node has a local coordinate frame defined, with respect to which the vehicle navigates through the environment. The local frame is formed using the global $\vec{n}$-$\vec{e}$ plane and the body-fixed coordinate frame at the instant in time when the keyframe image is taken. This particular body frame is designated the *reference* body-fixed coordinate frame.

The node frame for the j-th node is defined by unit vectors in the front, right, and down directions: $\vec{f}_j$, $\vec{r}_j$, and $\vec{d}_j$. The $\vec{d}_j$ direction is parallel to the global $\vec{d}$ direction. The $\vec{f}_j$ direction is defined by the projection of the reference body $\vec{b}_x$ axis onto a plane parallel to the global $\vec{n}$-$\vec{e}$ plane, as shown in Figure 4.2. The $\vec{r}_j$ direction is defined to make a right-hand coordinate system. This representation preserves the heading of the vehicle, which corresponds with the direction the image was taken, and maintains a global down direction, which keeps the roll and pitch defined correctly.

The relationship between the keyframe and the local node frame is necessary for correctly expressing VO measurement updates. Points defined in the keyframe's camera coordinate frame

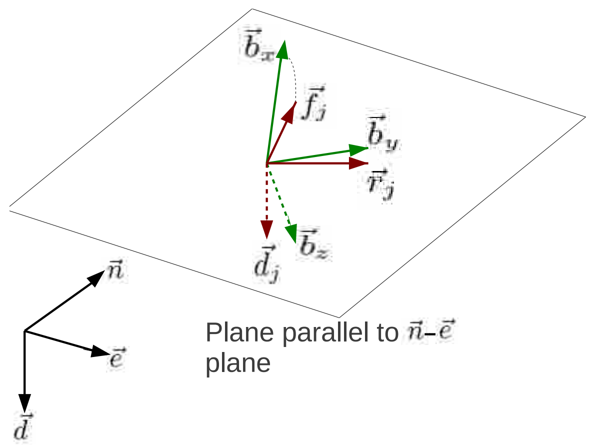

Figure 4.2: Relationship between the global (black), body (green), and node (red) frames at the instant the j-th keyframe image is taken. The $\vec{f}_j$ direction is defined by the projection of the reference body $\vec{b}_x$ axis onto a plane parallel to the global $\vec{n}$-$\vec{e}$ plane.

are expressed in the node coordinate frame using two rotations, $\mathscr{R}_c^b$ and $\mathscr{R}_{br}^n$, and a translation, $\mathbf{p}^b$. The rotation $\mathscr{R}_c^b$ and translation $\mathbf{p}^b$ define the static transformation between the camera frame and the body-fixed frame that is obtained through calibration. $\mathscr{R}_c^b$ rotates points from the camera frame into the body-fixed frame and $\mathbf{p}^b$ are the coordinates of the camera focal point in the body-fixed coordinate frame. These values only change when the location or orientation of the camera or IMU changes on the platform.

The second rotation, $\mathscr{R}_{br}^n$, defines the relationship between the reference body-fixed frame and the node frame. The reference body-fixed frame is created by saving the roll and pitch angles of the vehicle, ϕ_j and θ_j, at the instant in time the image is taken. The matrix $\mathscr{R}_{br}^n\left(\phi_j,\theta_j\right)$ rotates points from the body-fixed frame into the node j coordinate frame. Combined together, the transformation $\mathscr{R}_b^n\left(\mathscr{R}_c^b\mathbf{p}^{cr}+\mathbf{p}^b\right)$ transforms points from the keyframe coordinate frame to the node coordinate frame.

4.1.2 Map

The map used in this work is a collection of nodes and edges in a relative topological pose graph. New nodes are created with each new keyframe. Edges are added between temporally and spatially consecutive keyframes.

A node is described, ultimately, by a keyframe RGB and depth image pair. Attached to the keyframe pair are the estimates of relative and global position and orientation, yet the image encodes the true instantaneous location of the vehicle. The node coordinate frame, defined above, enables navigation relative to the node.

Edges in the graph represent the estimated relative transformations between nodes. We currently only consider edges from the odometry, but we plan to soon include other constraints, such as those from visual recognition loop closures and intermittent GPS measurements. The odometry edges are created using the MEKF, based on the measurements from the VO algorithm. When a new node is received by the estimator from the VO, the old relative portions of the state and covariance are marginalized out and saved as the edge between the old and new nodes.

4.2 Multiplicative Extended Kalman Filter

In contrast to the work in [79], we now use unit-length quaternions to represent the attitude of the vehicle. A quaternion is parameterized by a scalar and a vector. We utilize the standard Hamiltonian form of the quaternion [119]. A unit-length quaternion represents an over-parameterization of the attitude and should not be used directly in an EKF framework because of singularity problems that can occur in the covariance matrix. Many generic estimation methods have been developed to handle this difficulty and are available and commonly used in practice [125]. We chose to derive a MEKF [103, 126] for our relative hexacopter system.

The MEKF is an indirect EKF, which means that the error in the state $\Delta\mathbf{x}$ and the covariance of the error are maintained in the filter rather than the best estimate $\hat{\mathbf{x}}$ and error covariance. The name *multiplicative* was coined as the error in the quaternions is computed using a quaternion product $\otimes$ instead of subtraction (i.e., $\delta q(\theta) = q \otimes \hat{q}^{-1}$) [125]. The three-dimensional error vector θ, obtained from the error quaternion using a small angle approximation, is maintained in the error state. This results in a minimal representation that is safe to use in calculating the covariance. We also use a continuous-discrete representation [123] for our implementation of the MEKF.

In this implementation we do not directly maintain an estimate of the error state in the filter at each timestep and therefore do not maintain a true indirect filter. Instead, we maintain the best estimate $\hat{\mathbf{x}}$ and the covariance $\mathbf{P}$ of the error state $\Delta\mathbf{x}$ in the filter. Details of the implementation are given below.

4.2.1 State Dynamics

We model the hexacopter with the nonlinear equations

$$\dot{\mathbf{x}} = \mathbf{f}(\mathbf{x},\mathbf{u}), \tag{4.1}$$

$$\mathbf{y}_i = \mathbf{h}_i(\mathbf{x},\mathbf{u}),\ \ 1 \leq i \leq p, \tag{4.2}$$

where $\mathbf{h}_i$ is the i^{th} measurement function, and the vector $\mathbf{u}$ represents the inputs that drive the evolution of the estimated states. The inputs to the model are simply the gyroscope measurements and the $\vec{b}_z$ accelerometer

$$\mathbf{u} = \begin{bmatrix} p_{gyro} & q_{gyro} & r_{gyro} & z_{accel} \end{bmatrix}^\top. \tag{4.3}$$

The true states $\mathbf{x}$ of the multirotor are defined with respect to the current node, node j,

$$\mathbf{x} = \begin{bmatrix} \mathbf{p}^{n\top} & \mathbf{q}_n^{b\top} & \mathbf{v}^{b\top} & \beta^\top & \alpha^\top & \mathbf{q}_c^{b\top} & \mathbf{p}^{b\top} \end{bmatrix}^\top. \tag{4.4}$$

The relative position vector $\mathbf{p}^n$, made up of f_j, r_j and d_j, is the displacement of the body with respect to node j. The quaternion $\mathbf{q}_n^b$, with components q_x, q_y, q_z and q_w, expresses the attitude of the body frame with respect to the node frame. The component q_z is relative to the current node. The notation on a quaternion $\mathbf{q}_x^y$ or rotation matrix $\mathscr{R}_x^y$ denotes a rotation that takes points expressed in the x coordinate frame and rotates them into the y coordinate frame. $\mathbf{v}^b$ is the body frame velocity vector, composed of u, v and w. The gyroscope bias vector is β. We only estimate the accelerometer biases in the body $\vec{b}_x$ and $\vec{b}_y$ directions in α. The last two parameters in (4.4) represent the transformation from the body coordinate frame to the camera coordinate frame and can be optionally included in the state. $\mathbf{q}_c^b$ is the quaternion form of the the rotation $\mathscr{R}_c^b$, which rotates points from the camera coordinate frame into the body coordinate frame. $\mathbf{p}^b$ are the coordinates of the camera focal point expressed in the body coordinate frame. Once refinements to the calibration parameters are obtained, these estimates can be saved as constants and then removed from the state vector. The estimated state vector is denoted $\hat{\mathbf{x}}$.

The nonlinear equations in (4.1) are

$$\dot{\mathbf{p}}^n = {\mathbf{q}_n^b}^{-1} \otimes \mathbf{v}^b \otimes \mathbf{q}_n^b = \mathscr{R}^\top(\mathbf{q}_n^b)\mathbf{v}^b, \tag{4.5}$$

$$\dot{\mathbf{q}}_n^b = \frac{1}{2}\Omega\left(\mathbf{u}_{(1:3)} - \beta - \eta_\omega\right)\mathbf{q}_n^b, \tag{4.6}$$

$$\dot{\mathbf{v}}^b = \mathbf{v}^b \times \left(\mathbf{u}_{(1:3)} - \beta - \eta_\omega\right) + \mathbf{q}_n^b \otimes \mathbf{g} \otimes {\mathbf{q}_n^b}^{-1} - \frac{1}{m}\mathbf{M}\mathbf{v}^b + \mathbf{u}_{(4)}\vec{d}_j, \tag{4.7}$$

$$\dot{\beta} = \eta_\beta \tag{4.8}$$

$$\dot{\alpha} = \eta_\alpha \tag{4.9}$$

$$\dot{\mathbf{q}}_c^b = \mathbf{0} \tag{4.10}$$

$$\dot{\mathbf{p}}^b = \mathbf{0}, \tag{4.11}$$

where η_ω, η_β and η_α are zero-mean, Gaussian processes for the noise in the gyroscope measurements and the random-walk models of the gyroscope and the accelerometer biases. A rotation matrix $\mathscr{R}(\mathbf{q}_x^y)$ from a quaternion $\mathbf{q}_x^y$ is equivalent to the quaternion rotational operator $\mathscr{L}\left(q_x^y\right) = \mathbf{q}_x^y \otimes \mathbf{v} \otimes {\mathbf{q}_x^y}^{-1}$, which rotates the vector $\mathbf{v}$, expressed in the frame x, into frame y. The operator

$$\Omega(\omega) = \begin{bmatrix} 0 & \omega_3 & -\omega_2 & \omega_1 \\ -\omega_3 & 0 & \omega_1 & \omega_2 \\ \omega_2 & -\omega_1 & 0 & \omega_3 \\ -\omega_1 & -\omega_2 & -\omega_3 & 0 \end{bmatrix} \tag{4.12}$$

assumes that the order of a quaternion it multiplies is $\begin{bmatrix} q_x & q_y & q_z & q_w \end{bmatrix}^\top$. The constant matrix $\mathbf{M}$ is

$$\mathbf{M} = \begin{bmatrix} \mu & 0 & 0 \\ 0 & \mu & 0 \\ 0 & 0 & 0 \end{bmatrix}, \tag{4.13}$$

and the constants $\mathbf{g}$ and μ are the gravity and drag coefficient respectively. An improved model of the hexacopter dynamics, contained in (4.7), which accounts for the rotor drag with coefficient μ, provides the ability to fully utilize the information contained in the accelerometer measurements. As a consequence, estimation accuracy improves and the requirements for VO or any other exteroceptive measurement updates are reduced [95, 109].

To arrive at the estimated state dynamics, we compute the expectation of (4.5) through (4.11), which yields

$$\dot{\hat{\mathbf{p}}}^n = \hat{\mathbf{q}}_n^{b^{-1}} \otimes \hat{\mathbf{v}}^b \otimes \hat{\mathbf{q}}_n^b = \mathscr{R}^\top(\hat{\mathbf{q}}_n^b)\hat{\mathbf{v}}^b \tag{4.14}$$

$$\dot{\hat{\mathbf{q}}}_n^b = \frac{1}{2}\Omega\left(\mathbf{u} - \hat{\beta}\right)\hat{\mathbf{q}}_n^b \tag{4.15}$$

$$\dot{\hat{\mathbf{v}}}^b = \hat{\mathbf{v}}^b \times \left(\mathbf{u} - \hat{\beta}\right) + \hat{\mathbf{q}}_n^b \otimes \mathbf{g} \otimes \hat{\mathbf{q}}_n^{b^{-1}} - \frac{1}{m}\mathbf{M}\hat{\mathbf{v}}^b - \frac{1}{m}\mathbf{K}\bar{\omega} \tag{4.16}$$

$$\dot{\beta} = \mathbf{0}_{3\times 1} \tag{4.17}$$

$$\dot{\alpha} = \mathbf{0}_{2\times 1} \tag{4.18}$$

$$\dot{\mathbf{q}}_c^b = \mathbf{0}_{4\times 1} \tag{4.19}$$

$$\dot{\mathbf{p}}^b = \mathbf{0}_{3\times 1} \tag{4.20}$$

4.2.2 Error Dynamics

We now derive the error dynamics using the nonlinear dynamics in (4.5) through (4.11). The error dynamics are used to propagate the error covariance matrix **P**.

Error for all the components of the state, except the quaternions, are defined as

$$\delta\mathbf{a} = \mathbf{a} - \hat{\mathbf{a}}. \tag{4.21}$$

Error for a quaternion is defined by

$$\delta\mathbf{q} = \mathbf{q} \otimes \hat{\mathbf{q}}^{-1}. \tag{4.22}$$

Note that for a quaternion or a rotation matrix, the inversion operator is equivalent to the transpose. For a quaternion, this simply involves changing the sign of the vector portion. We then define the attitude error vector $\delta\theta$, using the small angle assumption, as

$$\delta\mathbf{q} \approx \begin{bmatrix} \frac{1}{2}\delta\theta \\ 1 \end{bmatrix} \tag{4.23}$$

The attitude error vector is maintained in the error state and, like the other error components, is assumed to be small. The length of the error state is reduced by one for each quaternion when compared to the length of the true state. The 20-element relative error state is then

$$\Delta\mathbf{x} = \begin{bmatrix} \delta\mathbf{p}^{n\top} & \delta\theta_n^{b\top} & \delta\mathbf{v}^{b\top} & \delta\beta^\top & \delta\alpha^\top & \delta\theta_c^{b\top} & \delta\mathbf{p}^{b\top} \end{bmatrix}^\top. \tag{4.24}$$

The error dynamics, computed using (4.21), (4.22), and (4.5) through (4.11) and (4.14) through (4.20), are as follows. In the derivation we assumed that second-order effects are negligible. We also used the identities $\mathscr{R}(\delta q) \approx \mathbf{I}_{3\times3} - \lfloor\delta\theta\rfloor$, where $\lfloor\ \rfloor$ is the skew-symmetric matrix operator on a vector, and $\lfloor\mathbf{y}\rfloor\mathbf{x} = -\lfloor\mathbf{x}\rfloor\mathbf{y}$ [126].

$$\dot{\delta\mathbf{p}}_n^b = \mathscr{R}^\top(\hat{\mathbf{q}}_n^b)\delta\mathbf{v}^b - \mathscr{R}^\top(\hat{\mathbf{q}}_n^b)\left\lfloor\hat{\mathbf{v}}^b\right\rfloor\delta\theta_n^b \tag{4.25}$$

$$\dot{\delta\theta}_n^b = -\mathbf{u}_{(1:3)} \times \delta\theta_n^b - \beta - \eta_\omega \tag{4.26}$$

$$\dot{\delta\mathbf{v}}^b = \delta\mathbf{v}^b \times \left(\mathbf{u}_{(1:3)} - \hat{\beta} - \eta_\omega\right) + \hat{\mathbf{v}}^b \times (-\delta\beta - \eta_\omega) + \left\lfloor\mathscr{R}^\top\left(\hat{\mathbf{q}}_n^b\right)\mathbf{g}\right\rfloor\delta\theta_n^b - \frac{1}{m}\mathbf{M}\delta\mathbf{v} \tag{4.27}$$

$$\dot{\beta} = \eta_\beta \tag{4.28}$$

$$\dot{\alpha} = \eta_\alpha \tag{4.29}$$

$$\dot{\mathbf{q}}_c^b = \mathbf{0} \tag{4.30}$$

$$\dot{\mathbf{p}}^b = \mathbf{0} \tag{4.31}$$

Equations (4.25) through (4.31) are linearized and result in the following linear model

$$\dot{\Delta\mathbf{x}} = \mathbf{A}\Delta\mathbf{x} + \mathbf{B}\mathbf{u}, \tag{4.32}$$

where $\mathbf{A}$ is the Jacobian of (4.25) through (4.31) with respect to the error state $\Delta\mathbf{x}$ and $\mathbf{B}$ is the Jacobian of (4.25) through (4.31) with respect to the input $\mathbf{u}$. $\mathbf{A}$ and $\mathbf{B}$ are

$$\mathbf{A} = \begin{bmatrix} \mathbf{0}_{3\times3} & -\mathscr{R}^{\top}\left(\hat{\mathbf{q}}_n^b\right)\lfloor\hat{\mathbf{v}}\rfloor & \mathscr{R}^{\top}\left(\hat{\mathbf{q}}_n^b\right) & \mathbf{0}_{3\times3} & \mathbf{0}_{3\times2} & \mathbf{0}_{3\times3} & \mathbf{0}_{3\times3} \\ \mathbf{0}_{3\times3} & -\lfloor\mathbf{u}\rfloor & \mathbf{0}_{3\times3} & -\mathbf{I}_{3\times3} & \mathbf{0}_{3\times2} & \mathbf{0}_{3\times3} & \mathbf{0}_{3\times3} \\ \mathbf{0}_{3\times3} & \lfloor\mathscr{R}^{\top}\left(\hat{\mathbf{q}}_n^b\right)\mathbf{g}\rfloor & \left(-\left\lfloor\mathbf{u}-\hat{\beta}\right\rfloor-\frac{1}{m}\mathbf{M}\right) & \lfloor\hat{\mathbf{v}}^b\rfloor & \mathbf{0}_{3\times2} & \mathbf{0}_{3\times3} & \mathbf{0}_{3\times3} \\ \mathbf{0}_{3\times3} & \mathbf{0}_{3\times3} & \mathbf{0}_{3\times3} & \mathbf{0}_{3\times3} & \mathbf{0}_{3\times2} & \mathbf{0}_{3\times3} & \mathbf{0}_{3\times3} \\ \mathbf{0}_{2\times3} & \mathbf{0}_{2\times3} & \mathbf{0}_{2\times3} & \mathbf{0}_{2\times3} & \mathbf{0}_{2\times2} & \mathbf{0}_{2\times3} & \mathbf{0}_{2\times3} \\ \mathbf{0}_{3\times3} & \mathbf{0}_{3\times3} & \mathbf{0}_{3\times3} & \mathbf{0}_{3\times3} & \mathbf{0}_{3\times2} & \mathbf{0}_{3\times3} & \mathbf{0}_{3\times3} \\ \mathbf{0}_{3\times3} & \mathbf{0}_{3\times3} & \mathbf{0}_{3\times3} & \mathbf{0}_{3\times3} & \mathbf{0}_{3\times2} & \mathbf{0}_{3\times3} & \mathbf{0}_{3\times3} \end{bmatrix} \tag{4.33}$$

$$\mathbf{B} = \begin{bmatrix} \mathbf{0}_{3\times3} \\ \lfloor\delta\theta_n^b\rfloor \\ \lfloor\delta\mathbf{v}^b\rfloor \\ \mathbf{0}_{3\times3} \\ \mathbf{0}_{3\times3} \\ \mathbf{0}_{2\times3} \\ \mathbf{0}_{3\times3} \end{bmatrix} \tag{4.34}$$

4.2.3 Prediction

As was mentioned above, in the filter we maintain the estimated state $\hat{\mathbf{x}}$ and the covariance $\hat{\mathbf{P}}$ of the error state (4.24). The estimated states are propagated forward by numerically integrating the nonlinear equations of motion, (4.5) through (4.11). The estimated covariance is propagated forward by numerically integrating the equation

$$\dot{\hat{\mathbf{P}}} = \mathbf{A}\hat{\mathbf{P}} + \hat{\mathbf{P}}\mathbf{A}^{\top} + \gamma\left(\mathbf{B}\mathbf{G}\mathbf{B}^{\top} + \mathbf{Q}\right), \tag{4.35}$$

where the matrices $\mathbf{A}$ and $\mathbf{B}$ are from (4.32), γ is a tuning parameter, $\mathbf{G}$ is a diagonal matrix of the measured covariance on the inputs (4.3) and $\mathbf{Q}$ is the process noise covariance which represents modeling error and disturbances.

In contrast to [79], some diagonal portions of $\mathbf{Q}$ are non-zero and must be found through hand tuning for the best performance. As we mentioned above, second-order terms were neglected in the derivations of (4.25) through (4.27). This results in not being able to completely account for the error.

4.2.4 Measurement Updates

We update the filter with measurements of the form in (4.2). Specifically, altimeter, accelerometer, and VO position and orientation measurements are used. In this implementation we treat the VO measurements updates separately. This is possible because we account for the contribution of the rotational uncertainty in the position covariance when the covariances of the measurements are generated [97].

Each measurement update follows the same procedure. For generality, we will assume that the measurement is $\mathbf{h}$, the measurement from the model is $\hat{\mathbf{h}}$ and the Jacobian of the analytic residual $\Delta\mathbf{h}$ with respect to the *error state* is $\mathbf{H}$.

First we compute the residual $\Delta\mathbf{h}$

$$\Delta\mathbf{h} = \mathbf{h} - \hat{\mathbf{h}}. \tag{4.36}$$

The covariance of the residual $\mathbf{S}$ is computed using

$$\mathbf{S} = \mathbf{R}_h + \mathbf{H}\hat{\mathbf{P}}\mathbf{H}^\top, \tag{4.37}$$

where $\mathbf{R}_h$ is the covariance on the measurement model $\hat{\mathbf{h}}$. The Kalman gain $\mathbf{L}$ is

$$\mathbf{L} = \hat{\mathbf{P}}\mathbf{H}^\top\mathbf{S}^{-1}. \tag{4.38}$$

The correction (or updated error state) $\Delta\hat{\mathbf{x}}^+$ is computed as

$$\Delta\hat{\mathbf{x}}^+ = \mathbf{L}\Delta\mathbf{h}, \tag{4.39}$$

where the $^+$ notation denotes an updated variable. The covariance is updated using

$$\hat{\mathbf{P}}^+ = (\mathbf{I} - \mathbf{LH})\,\hat{\mathbf{P}}. \tag{4.40}$$

After each measurement update, we need to use the correction (4.39) to update the current state estimate $\hat{\mathbf{x}}$. Using (4.21), we can update a component $\mathbf{a}$ of the state, which is not a quaternion, as

$$\hat{\mathbf{a}}^+ = \hat{\mathbf{a}} + \delta\hat{\mathbf{a}}^+. \tag{4.41}$$

To update the quaternions in the state, we create an error quaternion $\delta\mathbf{q}$ from the quaternion error vector $\delta\boldsymbol{\theta}$ and then use the quaternion product to compute the updated quaternion. First, we change the error vector into the vector portion of the error quaternion using

$$\delta\hat{\mathbf{q}}^+_{vec} = \frac{1}{2}\delta\hat{\boldsymbol{\theta}}^+. \tag{4.42}$$

Next, we create the unit-length error quaternion

$$\delta\hat{\mathbf{q}}^+ = \begin{bmatrix} \delta\hat{\mathbf{q}}^+_{vec} \\ \sqrt{1 - \left(\delta\hat{\mathbf{q}}^+_{vec}\right)^\top \delta\hat{\mathbf{q}}^+_{vec}} \end{bmatrix}; \tag{4.43}$$

or, if $\left(\delta\hat{\mathbf{q}}^+_{vec}{}^\top \delta\hat{\mathbf{q}}^+_{vec}\right) > 1$, using

$$\delta\hat{\mathbf{q}}^+ = \frac{1}{\sqrt{1 + \left(\delta\hat{\mathbf{q}}^+_{vec}\right)^\top \delta\hat{\mathbf{q}}^+_{vec}}} \cdot \begin{bmatrix} \delta\hat{\mathbf{q}}^+_{vec} \\ 1 \end{bmatrix}. \tag{4.44}$$

Finally we can compute the new quaternion for the state estimate using the error quaternion $\delta\mathbf{q}$ and the estimate from the prediction step $\hat{\mathbf{q}}$

$$\hat{\mathbf{q}}^+ = \delta\hat{\mathbf{q}}^+ \otimes \hat{\mathbf{q}} \tag{4.45}$$

The measurement models for each sensor are outlined below. Each of these is used in the generalized method outlined in (4.36) through (4.45) to complete the measurement updates for the filter.

Altimeter Measurement Model

The altimeter provides a global measurement of the altitude of the vehicle, assuming flight near hover and a flat floor. We can obtain an estimate of the global altitude using the position in the current state $\hat{\mathbf{p}}$ and the global position $\hat{\mathbf{p}}_{node}$ of the current node with respect to which we are navigating. No rotational transformation is necessary as the global down and node down directions are parallel. To compute the Jacobian $\mathbf{H}_{alt}$ of the residual with respect to the error state (4.24), we must develop an analytical expression for the residual $\Delta\mathbf{h} = \mathbf{h}_{alt} - \hat{\mathbf{h}}_{alt}$. We have

$$\mathbf{h}_{alt} = \mathbf{p}(3) + \mathbf{p}_{node}(3) + \eta_{alt} \tag{4.46}$$

$$\hat{\mathbf{h}}_{alt} = \hat{\mathbf{p}}(3) + \hat{\mathbf{p}}_{node}(3), \tag{4.47}$$

for which the analytic residual is

$$\begin{aligned} \Delta\mathbf{h}_{alt} &= \mathbf{h}_{alt} - \hat{\mathbf{h}}_{alt} && (4.48) \\ &= \mathbf{p}(3) + \mathbf{p}_{node}(3) + \eta_{alt} - (\hat{\mathbf{p}}(3) + \hat{\mathbf{p}}_{node}(3)) && (4.49) \\ &= \delta\mathbf{p}(3) + \delta\mathbf{p}_{node}(3) + \eta_{alt}. && (4.50) \end{aligned}$$

The Jacobian $\mathbf{H}_{alt}$ of the residual is trivially

$$\mathbf{H}_{alt} = \begin{bmatrix} \begin{bmatrix} 0 & 0 & 1 \end{bmatrix} & \mathbf{0}_{1\times3} & \mathbf{0}_{1\times3} & \mathbf{0}_{1\times3} & \mathbf{0}_{1\times2} & \mathbf{0}_{1\times3} & \mathbf{0}_{1\times3} \end{bmatrix}. \tag{4.51}$$

The covariance of the measurement is $E\left[\eta_{alt}\eta_{alt}^{\top}\right] = \mathbf{R}_{alt}$.

Accelerometer Measurement Model

The body $\vec{b}_x$ and $\vec{b}_y$ axis measurements are modeled as

$$\mathbf{h}_{acc} = -\frac{1}{m}\mathbf{M}\mathbf{v}^b + \alpha + \eta_{acc} \tag{4.52}$$

$$\hat{\mathbf{h}}_{acc} = -\frac{1}{m}\mathbf{M}\hat{\mathbf{v}}^b + \hat{\alpha}. \tag{4.53}$$

The analytic residual is

$$\Delta\mathbf{h}_{acc} = -\frac{1}{m}\mathbf{M}\delta\mathbf{v}^b + \delta\alpha + \eta_{acc}. \tag{4.54}$$

We can then calculate the Jacobian $\mathbf{H}_{acc}$:

$$\mathbf{H}_{acc} = \begin{bmatrix} \mathbf{0}_{3\times3} & -\frac{1}{m}\mathbf{M} & \mathbf{0}_{3\times3} & \mathbf{0}_{3\times3} & \begin{bmatrix} 1 & 0 & 0 \\ 0 & 1 & 0 \\ 0 & 0 & 0 \end{bmatrix} & \mathbf{0}_{3\times3} & \mathbf{0}_{3\times3} \end{bmatrix}. \tag{4.55}$$

Visual Odometry Position Model

Figure 4.3 illustrates the chain of transformations involved in the measurement updates for the VO. The current pose in the state is represented by the transformation from the current node frame to the current body frame, $\mathbf{q}_n^b$ and $\mathbf{p}^n$. The calibration terms, $\mathbf{q}_c^b$ and $\mathbf{p}^b$, are found in two places since there is a reference body frame in between the node frame and the reference camera frame. The VO provides the transformation made up of $\mathbf{q}_{cr}^c$ and $\mathbf{p}^c$.

Using Figure 4.3, we estimate the model of the VO translation $\mathbf{p}^c$ as

$$\mathbf{h}_{vp} = -\mathscr{L}_b^c\left(\mathbf{q}_c^{b\top}\right)\mathbf{p}^b - \mathscr{L}_n^c\left(\mathbf{q}_n^b\mathbf{q}_c^{b\top}\right)\mathbf{p}^n + \mathscr{L}_{br}^c\left(\mathbf{q}_{br}^n\mathbf{q}_n^b\mathbf{q}_c^{b\top}\right)\mathbf{p}^b + \eta_{vp}, \tag{4.56}$$

$$\hat{\mathbf{h}}_{vp} = -\mathscr{L}_b^c\left(\hat{\mathbf{q}}_c^{b\top}\right)\hat{\mathbf{p}}^b - \mathscr{L}_n^c\left(\hat{\mathbf{q}}_n^b\hat{\mathbf{q}}_c^{b\top}\right)\hat{\mathbf{p}}^n + \mathscr{L}_{br}^c\left(\hat{\mathbf{q}}_{br}^n\hat{\mathbf{q}}_n^b\hat{\mathbf{q}}_c^{b\top}\right)\hat{\mathbf{p}}^b. \tag{4.57}$$

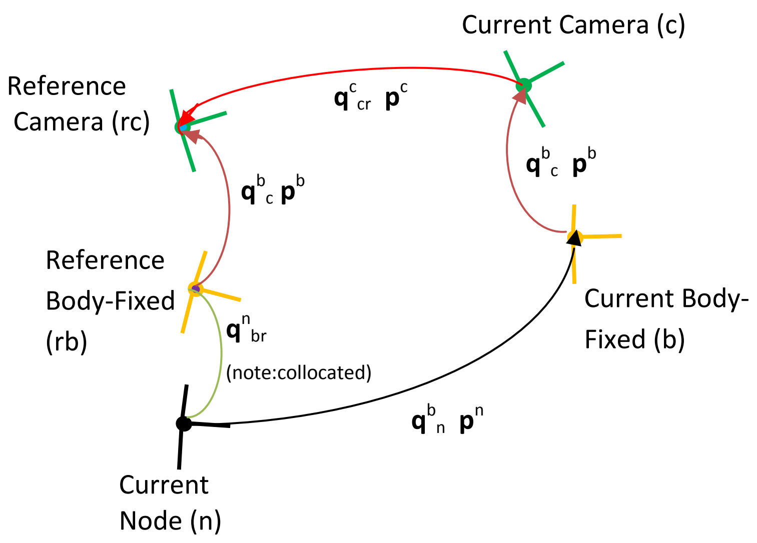
Current Camera (c)
Reference Camera (rc)
$\mathbf{q}^c_{cr}$ $\mathbf{p}^c$
$\mathbf{q}^b_c$ $\mathbf{p}^b$
$\mathbf{q}^b_c$ $\mathbf{p}^b$
Reference Body-Fixed (rb)
$\mathbf{q}^n_{br}$
(note:collocated)
Current Body-Fixed (b)
$\mathbf{q}^b_n$ $\mathbf{p}^n$
Current Node (n)

Figure 4.3: Vector chain for the VO measurement updates. The labels of the frames, as used in the equations below, are noted in parenthesis. Note that the arrows point in the defined direction of the translation vector.

The analytic residual begins as

$$\begin{aligned}\Delta\mathbf{h}_{vp} = &- \mathscr{L}^c_b\left(\mathbf{q}^{b\top}_c\right)\mathbf{p}^b - \mathscr{L}^c_n\left(\mathbf{q}^b_n\mathbf{q}^{b\top}_c\right)\mathbf{p}^n + \mathscr{L}^c_{br}\left(\mathbf{q}^n_{br}\mathbf{q}^b_n\mathbf{q}^{b\top}_c\right)\mathbf{p}^{br} + \eta_{vp} - \\ &\left(-\mathscr{L}^c_b\left(\hat{\mathbf{q}}^{b\top}_c\right)\hat{\mathbf{p}}^b - \mathscr{L}^c_n\left(\hat{\mathbf{q}}^b_n\,\hat{\mathbf{q}}^{b\top}_c\right)\hat{\mathbf{p}}^n + \mathscr{L}^c_{br}\left(\hat{\mathbf{q}}^n_{br}\hat{\mathbf{q}}^b_n\,\hat{\mathbf{q}}^{b\top}_c\right)\hat{\mathbf{p}}^{br}\right). \end{aligned} \tag{4.58}$$

We can switch to a rotation matrix form and replace the truth terms with the estimate and error using (4.21). Note that the order of the rotation terms must switch when going from quaternions to rotation matrices as we are using the Hamilton representation of the quaternion

$$\begin{aligned}\Delta\mathbf{h}_{vp} = &- \mathscr{R}^\top(\delta\mathbf{q}^b_c)\mathscr{R}^\top(\hat{\mathbf{q}}^b_c)\left(\delta\mathbf{p}^b + \hat{\mathbf{p}}^b\right) - \mathscr{R}^\top(\delta\mathbf{q}^b_c)\mathscr{R}^\top(\hat{\mathbf{q}}^b_c)\mathscr{R}(\hat{\mathbf{q}}^b_n)\mathscr{R}(\delta\mathbf{q}^b_n)\left(\delta\mathbf{p}^n + \hat{\mathbf{p}}^n\right) \\ &+ \mathscr{R}^\top(\delta\mathbf{q}^b_c)\mathscr{R}^\top(\hat{\mathbf{q}}^b_c)\mathscr{R}(\hat{\mathbf{q}}^b_n)\mathscr{R}(\delta\mathbf{q}^b_n)\mathscr{R}(\mathbf{q}^n_{br})\left(\delta\mathbf{p}^b + \hat{\mathbf{p}}^b\right) + \eta_{vp} \\ &+ \mathscr{R}^\top(\hat{\mathbf{q}}^b_c)\hat{\mathbf{p}}^b + \mathscr{R}^\top(\hat{\mathbf{q}}^b_c)\mathscr{R}(\hat{\mathbf{q}}^b_n)\hat{\mathbf{p}}^n - \mathscr{R}^\top(\hat{\mathbf{q}}^b_c)\mathscr{R}(\hat{\mathbf{q}}^b_n)\mathscr{R}(\mathbf{q}^n_{br})\hat{\mathbf{p}}^b. \end{aligned} \tag{4.59}$$

Using the approximation $\mathscr{R}(\delta\mathbf{q}_x^y) \approx \left(\mathbf{I} - \lfloor\delta\theta_x^y\rfloor\right)$ or $\mathscr{R}^\top(\delta\mathbf{q}_x^y) \approx \left(\mathbf{I} + \lfloor\delta\theta_x^y\rfloor\right)$, we have

$$\begin{aligned}
\Delta\mathbf{h}_{vp} = &- \left(\mathbf{I} + \left\lfloor\delta\theta_c^b\right\rfloor\right)\mathscr{R}^\top(\hat{\mathbf{q}}_c^b)\left(\delta\mathbf{p}^b + \hat{\mathbf{p}}^b\right) - \left(\mathbf{I} + \left\lfloor\delta\theta_c^b\right\rfloor\right)\mathscr{R}^\top(\hat{\mathbf{q}}_c^b)\mathscr{R}(\hat{\mathbf{q}}_n^b)\left(\mathbf{I} - \left\lfloor\delta\theta_n^b\right\rfloor\right)(\delta\mathbf{p}^n + \hat{\mathbf{p}}^n) \\
&+ (\mathbf{I} + \lfloor\delta\theta_b^c\rfloor)\mathscr{R}^\top(\hat{\mathbf{q}}_c^b)\mathscr{R}(\hat{\mathbf{q}}_n^b)\left(\mathbf{I} - \left\lfloor\delta\theta_n^b\right\rfloor\right)\mathscr{R}(\mathbf{q}_{br}^n)\left(\delta\mathbf{p}^b + \hat{\mathbf{p}}^b\right) \qquad (4.60) \\
&+ \eta_{vp} + \mathscr{R}^\top(\hat{\mathbf{q}}_c^b)\hat{\mathbf{p}}^b + \mathscr{R}^\top(\hat{\mathbf{q}}_c^b)\mathscr{R}(\hat{\mathbf{q}}_n^b)\hat{\mathbf{p}}^n - \mathscr{R}^\top(\hat{\mathbf{q}}_c^b)\mathscr{R}(\hat{\mathbf{q}}_n^b)\mathscr{R}(\mathbf{q}_{br}^n)\hat{\mathbf{p}}^b.
\end{aligned}$$

Finally, after expanding, removing second-order terms, and simplifying using the approximation $\mathscr{R}(\delta\mathbf{q}_x^y) \approx \left(\mathbf{I} - \lfloor\delta\theta_x^y\rfloor\right)$ or $\mathscr{R}^\top(\delta\mathbf{q}_x^y) \approx \left(\mathbf{I} + \lfloor\delta\theta_x^y\rfloor\right)$, we have

$$\begin{aligned}
\Delta\mathbf{h}_{vp} = &- \mathscr{R}^\top(\hat{\mathbf{q}}_c^b)\delta\mathbf{p}^b - \left\lfloor\delta\theta_c^b\right\rfloor\mathscr{R}^\top(\hat{\mathbf{q}}_c^b)\hat{\mathbf{p}}^b - \mathscr{R}^\top(\hat{\mathbf{q}}_c^b)\mathscr{R}(\hat{\mathbf{q}}_n^b)\delta\mathbf{p}^n \\
&+ \mathscr{R}^\top(\hat{\mathbf{q}}_c^b)\mathscr{R}(\hat{\mathbf{q}}_n^b)\left\lfloor\delta\theta_n^b\right\rfloor\hat{\mathbf{p}}^n - \left\lfloor\delta\theta_c^b\right\rfloor\mathscr{R}^\top(\hat{\mathbf{q}}_c^b)\mathscr{R}(\hat{\mathbf{q}}_n^b)\hat{\mathbf{p}}^n \qquad (4.61) \\
&+ \mathscr{R}^\top(\hat{\mathbf{q}}_c^b)\mathscr{R}(\hat{\mathbf{q}}_n^b)\mathscr{R}(\mathbf{q}_{br}^n)\delta\mathbf{p}^b - \mathscr{R}^\top(\hat{\mathbf{q}}_c^b)\mathscr{R}(\hat{\mathbf{q}}_n^b)\left\lfloor\delta\theta_n^b\right\rfloor\mathscr{R}(\mathbf{q}_{br}^n)\hat{\mathbf{p}}^b \\
&+ \left\lfloor\delta\theta_c^b\right\rfloor\mathscr{R}^\top(\hat{\mathbf{q}}_c^b)\mathscr{R}(\hat{\mathbf{q}}_n^b)\mathscr{R}(\mathbf{q}_{br}^n)\hat{\mathbf{p}}^b.
\end{aligned}$$

The Jacobian $\mathbf{H}_{vp}$ of the analytic residual is

$$\mathbf{H}_{vp}^\top = \begin{bmatrix}
-\mathscr{R}^\top(\hat{\mathbf{q}}_c^b)\mathscr{R}(\hat{\mathbf{q}}_n^b) \\
-\mathscr{R}^\top(\hat{\mathbf{q}}_c^b)\mathscr{R}(\hat{\mathbf{q}}_n^b)\lfloor\hat{\mathbf{p}}^n\rfloor + \mathscr{R}^\top(\hat{\mathbf{q}}_c^b)\mathscr{R}(\hat{\mathbf{q}}_n^b)\lfloor\mathscr{R}(\mathbf{q}_{br}^n)\hat{\mathbf{p}}^b\rfloor \\
\mathbf{0}_{3\times3} \\
\mathbf{0}_{3\times3} \\
\mathbf{0}_{3\times2} \\
\lfloor\mathscr{R}^\top(\hat{\mathbf{q}}_c^b)\hat{\mathbf{p}}^b\rfloor + \lfloor\mathscr{R}^\top(\hat{\mathbf{q}}_c^b)\mathscr{R}(\hat{\mathbf{q}}_n^b)\hat{\mathbf{p}}^n\rfloor - \lfloor\mathscr{R}^\top(\hat{\mathbf{q}}_c^b)\mathscr{R}(\hat{\mathbf{q}}_n^b)\mathscr{R}(\mathbf{q}_{br}^n)\hat{\mathbf{p}}^b\rfloor \\
-\mathscr{R}^\top(\hat{\mathbf{q}}_c^b) + \mathscr{R}^\top(\hat{\mathbf{q}}_c^b)\mathscr{R}(\hat{\mathbf{q}}_n^b)\mathscr{R}(\mathbf{q}_{br}^n)
\end{bmatrix}. \qquad (4.62)$$

Visual Odometry Orientation Model

Using Figure 4.3 we can also find a model for the view-matching rotation $\mathbf{q}_{cr}^c$:

$$\mathbf{h}_{vq} = \mathbf{q}_{cr}^c + \eta_{vq} = \mathbf{q}_c^b \otimes \mathbf{q}_{br}^n \otimes \mathbf{q}_n^b \otimes {\mathbf{q}_c^b}^{-1} + \eta_{vq}, \qquad (4.63)$$

$$\hat{\mathbf{h}}_{vq} = \hat{\mathbf{q}}_{cr}^c = \hat{\mathbf{q}}_c^b \otimes \hat{\mathbf{q}}_{br}^n \otimes \hat{\mathbf{q}}_n^b \otimes {\hat{\mathbf{q}}_c^b}^{-1}. \qquad (4.64)$$

The analytic residual can be computed two ways, using subtraction [126] or multiplication [**?**]. The analytic model of the residual using the subtraction method is

$$\Delta\mathbf{h}_{vq} = \mathbf{q}_{cr}^{c} + \eta_{vq} - \hat{\mathbf{q}}_{cr}^{c}. \tag{4.65}$$

If we multiply each side by the matrix $\Xi^{\top}(\hat{\mathbf{q}}_{cr}^{c})$, where

$$\Xi^{\top}(\mathbf{q}_{x}^{y}) = \Xi^{\top}(\mathbf{q}) = \begin{bmatrix} q_4\mathbf{I} + \lfloor q_{vec} \rfloor & -q_{vec} \end{bmatrix}, \tag{4.66}$$

we can eliminate $\hat{\mathbf{q}}_{cr}^{c}$ (note that for any quaternion $\mathbf{q}_{x}^{y}$, $\Xi^{\top}(\mathbf{q}_{x}^{y})\mathbf{q}_{x}^{y} = \mathbf{0}$),

$$\tilde{\Delta\mathbf{h}}_{vq} = \Xi^{\top}(\hat{\mathbf{q}}_{cr}^{c})\Delta\mathbf{h}_{vq} = \Xi^{\top}(\hat{\mathbf{q}}_{cr}^{c})\left(\mathbf{q}_{cr}^{c} + \eta_{vq}\right) - \mathbf{0}. \tag{4.67}$$

Which, expanded out, is

$$\tilde{\Delta\mathbf{h}}_{vq} = \Xi^{\top}(\hat{\mathbf{q}}_{cr}^{c})\left(\delta\mathbf{q}_{c}^{b} \otimes \hat{\mathbf{q}}_{c}^{b} \otimes \hat{\mathbf{q}}_{br}^{n} \otimes \delta\mathbf{q}_{n}^{b} \otimes \hat{\mathbf{q}}_{n}^{b} \otimes \hat{\mathbf{q}}_{c}^{b^{-1}} \otimes \delta\mathbf{q}_{c}^{b^{-1}}\right). \tag{4.68}$$

The analytic Jacobian of (4.68) is much too large to be used in practice. We were able to calculate an approximate Jacobian at nominal values of the quaternions and use that with success. Note that using this method, we calculate the residual in the algorithm using (4.67) instead of (4.36), where $\hat{\mathbf{q}}_{cr}^{c}$ is computed using (4.64) and $\mathbf{q}_{cr}^{c}$ is the measurement from the VO. Additionally, the covariance $\mathbf{R}_{vq}$ must be modified by $\Xi^{\top}(\hat{\mathbf{q}}_{cr}^{c})$, so that

$$\tilde{\mathbf{R}}_{vq} = \Xi^{\top}(\hat{\mathbf{q}}_{cr}^{c})\mathbf{R}_{vq}\Xi(\hat{\mathbf{q}}_{cr}^{c}). \tag{4.69}$$

If instead we use the multiplication approach, the model of the residual is

$$\Delta\mathbf{h}_{vq} = \left(\mathbf{q}_{cr}^{c} + \eta_{vq}\right) \otimes \hat{\mathbf{q}}_{cr}^{c^{-1}}. \tag{4.70}$$

Expanded, we have an unmodified $\mathbf{R}_{vq}$ and

$$\Delta\mathbf{h}_{vq} = \mathbf{q}_c^b \otimes \mathbf{q}_{br}^n \otimes \mathbf{q}_n^b \otimes {\mathbf{q}_c^b}^{-1} \otimes \hat{\mathbf{q}}_c^b \otimes {\hat{\mathbf{q}}_n^b}^{-1} \otimes {\hat{\mathbf{q}}_{br}^n}^{-1} \otimes {\hat{\mathbf{q}}_c^b}^{-1}. \tag{4.71}$$

Replacing the truth using (4.22) we have

$$\Delta\mathbf{h}_{vq} = \delta\mathbf{q}_c^b \otimes \hat{\mathbf{q}}_c^b \otimes \hat{\mathbf{q}}_{br}^n \otimes \delta\mathbf{q}_n^b \otimes \hat{\mathbf{q}}_n^b \otimes {\hat{\mathbf{q}}_c^b}^{-1} \otimes {\delta\mathbf{q}_c^b}^{-1} \otimes \hat{\mathbf{q}}_c^b \otimes {\hat{\mathbf{q}}_n^b}^{-1} \otimes {\hat{\mathbf{q}}_{br}^n}^{-1} \otimes {\hat{\mathbf{q}}_c^b}^{-1}. \tag{4.72}$$

We cannot find a direct Jacobian of (4.72) either, but we can make a few approximations to provide us with an estimate of the Jacobian. First we set $\delta\mathbf{q}_c^b$ to the identity quaternion, which is equivalent to a small angle assumption, and we have

$$\Delta\mathbf{h}_{vq} = \hat{\mathbf{q}}_c^b \otimes \hat{\mathbf{q}}_{br}^n \otimes \delta\mathbf{q}_n^b \otimes {\hat{\mathbf{q}}_{br}^n}^{-1} \otimes {\hat{\mathbf{q}}_c^b}^{-1}. \tag{4.73}$$

We can further reduce the equations by replacing the quaternion $\delta\mathbf{q}_n^b$ with the error vector $\delta\theta_n^b$ using (4.23) and dropping the one, leaving us with

$$\Delta\mathbf{h}_{vq} = \frac{1}{2}\mathscr{R}(\hat{\mathbf{q}}_{br}^n)\mathscr{R}(\hat{\mathbf{q}}_c^b)\ \delta\theta_n^b \tag{4.74}$$

Again, we switch the order of the rotations when going from quaternions to rotation matrices.

The first step to find the Jacobian term for $\delta\theta_c^b$ is to set $\delta\mathbf{q}_n^b$ to the identity quaternion. We then have

$$\Delta\mathbf{h}_{vq} = \delta\mathbf{q}_c^b \otimes \hat{\mathbf{q}}_c^b \otimes \hat{\mathbf{q}}_{br}^n \otimes \hat{\mathbf{q}}_n^b \otimes {\hat{\mathbf{q}}_c^b}^{-1} \otimes {\delta\mathbf{q}_c^b}^{-1} \otimes \hat{\mathbf{q}}_c^b \otimes {\hat{\mathbf{q}}_n^b}^{-1} \otimes {\hat{\mathbf{q}}_{br}^n}^{-1} \otimes {\hat{\mathbf{q}}_c^b}^{-1}. \tag{4.75}$$

With two terms for $\delta\theta_c^b$ in the equation, this component of the Jacobian would be second order. We simply approximate this as zero.

The Jacobian $\mathbf{H}_{vq}$ for the multiplication approach is then

$$\mathbf{H}_{vq} = \begin{bmatrix} \mathbf{0}_{3\times3} & \frac{1}{2}\mathscr{R}(\hat{\mathbf{q}}_{br}^n)\mathscr{R}(\hat{\mathbf{q}}_c^b) & \mathbf{0}_{3\times3} & \mathbf{0}_{3\times3} & \mathbf{0}_{3\times2} & \mathbf{0}_{3\times3} & \mathbf{0}_{3\times3} \end{bmatrix} \tag{4.76}$$

In practice we have found that either the subtraction or multiplication approach works well. Currently we use the subtraction approach.

Delayed View-Matching Updates

An additional challenge with the view-matching measurement updates is that they are delayed. The stochastic delay is due to the requisite image-processing time. Consequently, computing the position and orientation measurement updates requires a few additional steps, which can be visualized in Figure 4.4. First, the state and covariance must first be restored to their values at the time the image was taken. Then the measurement updates can be applied in the normal fashion, (4.36) through (4.45). Finally, the state and covariance must be repropagated back to the current time by re-applying the prediction and measurement updates using the gyroscope, altimeter and accelerometer measurements at their respective timesteps. The state, covariance, IMU, and altimeter information are saved at each timestep to accommodate this requirement.

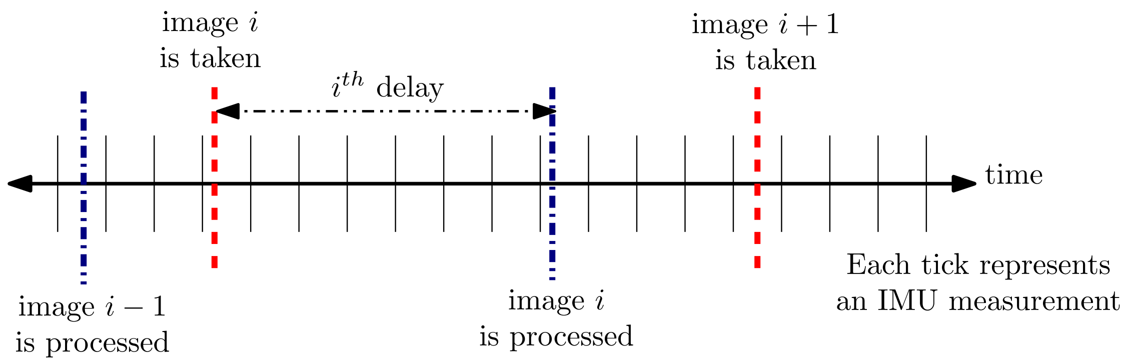

Figure 4.4: This time line illustrates the delayed VO measurement updates. Image i is taken, but the data is not received by the estimator until it is processed, seven timesteps later in this example. We save the state, covariance, and measurements received at each time step. When the i-th measurement is received, we reset the state and covariance back to the time the image was taken, apply the update, and the repropagate the other measurements until we arrive back to the current time.

4.2.5 Augment and Marginalize the Relative State

When a new node is created by the view-matching algorithm, the relative portions of the state and covariance must change. This is slightly more difficult as we now use a quaternion to encode the orientation and the yaw is relative. However, the process is straightforward. We use the relationship between quaternions and Euler angles to adjust the state and covariance appropriately.

We can augment and marginalize simultaneously as we are not augmenting additional states or removing unnecessary ones, but are simply replacing the old relative states with new relative states. The states that change are the positions in $\hat{\mathbf{p}}^n$ and the yaw contained in $\hat{\mathbf{q}}_n^b$. The positions are simply replaced with zeros. The quaternion in the state must maintain the same pitch and roll but the yaw must be zeroed out. This is done using the transformations to and from Euler angles. We use the current quaternion $\hat{\mathbf{q}}_n^b$ to find the current pitch $\hat{\theta}$ and roll $\hat{\phi}$ angles

$$\hat{\phi} = \arctan\left(\frac{2\left(\hat{q}_w\hat{q}_x + \hat{q}_y\hat{q}_z\right)}{\hat{q}_w^2 + \hat{q}_z^2 - \hat{q}_x^2 - \hat{q}_y^2}\right) \tag{4.77}$$

$$\hat{\theta} = \arcsin\left(2\left(\hat{q}_w\hat{q}_y - \hat{q}_x\hat{q}_z\right)\right), \tag{4.78}$$

and then we initialize the new quaternion using

$$\hat{q}_x^+ = \cos\left(\frac{\hat{\theta}}{2}\right)\sin\left(\frac{\hat{\phi}}{2}\right) \tag{4.79}$$

$$\hat{q}_y^+ = \sin\left(\frac{\hat{\theta}}{2}\right)\cos\left(\frac{\hat{\phi}}{2}\right) \tag{4.80}$$

$$\hat{q}_z^+ = -\sin\left(\frac{\hat{\theta}}{2}\right)\sin\left(\frac{\hat{\phi}}{2}\right) \tag{4.81}$$

$$\hat{q}_w^+ = \cos\left(\frac{\hat{\theta}}{2}\right)\cos\left(\frac{\hat{\phi}}{2}\right). \tag{4.82}$$

Initializing the covariance for the quaternion is slightly different, since the covariance of the quaternion is actually the covariance of the attitude error vector.

The error in yaw must be zeroed-out, but the error in roll and pitch must remain unchanged. Thus, we can use (4.77)-(4.82) to accomplish this, but we replace the quaternion $\hat{\mathbf{q}}_n^b$ with the error quaternion $\delta\mathbf{q}_n^b$. Recall that the error quaternion is related to the attitude error vector using (4.23).

The equations for the new error quaternion given the old error quaternion are obtained by substituting (4.77) and (4.78) into (4.79)-(4.82). We can simplify the equations by canceling out all the second-order terms, as the elements of $\delta\theta$ are small:

$$\begin{aligned}
\delta q_x^+ &= \cos\left(\frac{1}{2}\arcsin\left(\delta\theta_y - \frac{1}{2}\delta\theta_x\delta\theta_z\right)\right)\sin\left(\frac{1}{2}\arctan\left(\frac{\delta\theta_x + \frac{1}{2}\delta\theta_y\delta\theta_z}{1+\frac{1}{4}\delta\theta_z{}^2 - \frac{1}{4}\delta\theta_x{}^2 - \frac{1}{4}\delta\theta_y{}^2}\right)\right)\\
\delta q_y^+ &= \sin\left(\frac{1}{2}\arcsin\left(\delta\theta_y - \frac{1}{2}\delta\theta_x\delta\theta_z\right)\right)\cos\left(\frac{1}{2}\arctan\left(\frac{\delta\theta_x + \frac{1}{2}\delta\theta_y\delta\theta_z}{1+\frac{1}{4}\delta\theta_z{}^2 - \frac{1}{4}\delta\theta_x{}^2 - \frac{1}{4}\delta\theta_y{}^2}\right)\right)\\
\delta q_z^+ &= -\sin\left(\frac{1}{2}\arcsin\left(\delta\theta_y - \frac{1}{2}\delta\theta_x\delta\theta_z\right)\right)\sin\left(\frac{1}{2}\arctan\left(\frac{\delta\theta_x + \frac{1}{2}\delta\theta_y\delta\theta_z}{1+\frac{1}{4}\delta\theta_z{}^2 - \frac{1}{4}\delta\theta_x{}^2 - \frac{1}{4}\delta\theta_y{}^2}\right)\right).
\end{aligned}$$

Then we can simplify by canceling out second-order terms, as the elements of $\delta\theta$ are small

$$\delta q_x^+ \approx \cos\left(\frac{1}{2}\arcsin\left(\delta\theta_y\right)\right)\sin\left(\frac{1}{2}\arctan\left(\frac{\delta\theta_x}{1}\right)\right) \tag{4.83}$$

$$\delta q_y^+ \approx \sin\left(\frac{1}{2}\arcsin\left(\delta\theta_y\right)\right)\cos\left(\frac{1}{2}\arctan\left(\frac{\delta\theta_x}{1}\right)\right) \tag{4.84}$$

$$\delta q_z^+ \approx -\sin\left(\frac{1}{2}\arcsin\left(\delta\theta_y\right)\right)\sin\left(\frac{1}{2}\arctan\left(\frac{\delta\theta_x}{1}\right)\right). \tag{4.85}$$

We need the Jacobian $\mathbf{H}_{\delta\theta}$ of the above three equations with respect to $\delta\theta_x$, $\delta\theta_y$ and $\delta\theta_z$

$$\mathbf{H}_{\delta\theta} = \begin{bmatrix} \frac{\partial(\delta q_x^+)}{\partial(\delta\theta_x)} & \frac{\partial(\delta q_x^+)}{\partial(\delta\theta_y)} & \frac{\partial(\delta q_x^+)}{\partial(\delta\theta_z)} \\ \frac{\partial(\delta q_y^+)}{\partial(\delta\theta_x)} & \frac{\partial(\delta q_y^+)}{\partial(\delta\theta_y)} & \frac{\partial(\delta q_y^+)}{\partial(\delta\theta_z)} \\ \frac{\partial(\delta q_z^+)}{\partial(\delta\theta_x)} & \frac{\partial(\delta q_z^+)}{\partial(\delta\theta_y)} & \frac{\partial(\delta q_z^+)}{\partial(\delta\theta_z)} \end{bmatrix}. \tag{4.86}$$

We find the elements of $\mathbf{H}_z$, after canceling out additional second-order terms, to be

$$\frac{\partial\left(\delta q_x^+\right)}{\partial\left(\delta\theta_x\right)} = -\frac{1}{2}\cos\left(\frac{1}{2}\arcsin\left(\delta\theta_y\right)\right)\cos\left(\frac{1}{2}\arctan\left(\frac{\delta\theta_x}{1}\right)\right) \tag{4.87}$$

$$\frac{\partial\left(\delta q_x^+\right)}{\partial\left(\delta\theta_y\right)} = -\frac{1}{2}\sin\left(\frac{1}{2}\arcsin\left(\delta\theta_y\right)\right)\sin\left(\frac{1}{2}\arctan\left(\frac{\delta\theta_x}{1}\right)\right) \tag{4.88}$$

$$\frac{\partial\left(\delta q_x^+\right)}{\partial\left(\delta\theta_z\right)} = 0 \tag{4.89}$$

$$\frac{\partial\left(\delta q_x^+\right)}{\partial\left(\delta\theta_x\right)} = \frac{1}{2}\sin\left(\frac{1}{2}\arcsin\left(\delta\theta_y\right)\right)\sin\left(\frac{1}{2}\arctan\left(\frac{\delta\theta_x}{1}\right)\right) \tag{4.90}$$

$$\frac{\partial\left(\delta q_x^+\right)}{\partial\left(\delta\theta_y\right)} = \frac{1}{2}\cos\left(\frac{1}{2}\arcsin\left(\delta\theta_y\right)\right)\cos\left(\frac{1}{2}\arctan\left(\frac{\delta\theta_x}{1}\right)\right) \tag{4.91}$$

$$\frac{\partial\left(\delta q_x^+\right)}{\partial\left(\delta\theta_z\right)} = 0 \tag{4.92}$$

$$\frac{\partial\left(\delta q_x^+\right)}{\partial\left(\delta\theta_x\right)} = \frac{1}{2}\sin\left(\frac{1}{2}\arcsin\left(\delta\theta_y\right)\right)\cos\left(\frac{1}{2}\arctan\left(\frac{\delta\theta_x}{1}\right)\right) \tag{4.93}$$

$$\frac{\partial\left(\delta q_x^+\right)}{\partial\left(\delta\theta_y\right)} = -\frac{1}{2}\cos\left(\frac{1}{2}\arcsin\left(\delta\theta_y\right)\right)\sin\left(\frac{1}{2}\arctan\left(\frac{\delta\theta_x}{1}\right)\right) \tag{4.94}$$

$$\frac{\partial\left(\delta q_x^+\right)}{\partial\left(\delta\theta_z\right)} = 0 \tag{4.95}$$

We then modify the covariance $\hat{\mathbf{P}}$ using the overall Jacobian of the change in the error state

$$\mathbf{H}_A = \begin{bmatrix} \mathbf{0}_{3\times3} & \mathbf{0}_{3\times3} & \mathbf{0}_{3\times14} \\ \mathbf{0}_{3\times3} & \mathbf{H}_{\delta\theta} & \mathbf{0}_{3\times14} \end{bmatrix}, \tag{4.96}$$

and the following matrices

$$\mathbf{C} = \mathbf{H}_A \mathbf{P} \mathbf{H}_A^\top \tag{4.97}$$

$$\mathbf{T} = \mathbf{H}_A \mathbf{P} \tag{4.98}$$

$$\mathbf{L} = \mathbf{P} \mathbf{H}_A^\top. \tag{4.99}$$

The matrix $\mathbf{T} : 6 \times 20$ is used to overwrite the top six rows of the covariance. The matrix $\mathbf{L} : 20 \times 6$ overwrites the left-most six columns of the covariance, and the matrix $\mathbf{C} : 6 \times 6$ overwrites the top left corner, in that order. The remaining portions of the covariance are left the same as they are not relative.

4.3 Experimental Setup and Results

The results for three scenarios are presented. The first are of the relative MEKF running in real time during a manually-controlled flight. The second set are from an autonomous flight with the estimates in the control loop during a commanded hover. The third set of results are from an autonomous flight in which the hexacopter follows planned paths through an environment. All the flights were completed in a motion capture environment. The motion capture data is only used as the truth for the comparison of the results and to initialize the starting location of the vehicle for easier comparison between truth and estimates.

The hardware setup for the results of this chapter was the same as described in Section 1.1. Truth data from a motion-capture system is only used to initialize the global position of the vehicle and in the comparisons made in the figures below. The relative MEKF runs at 100 Hz, the update rate of the IMU. Measurement updates for the altimeter and the visual odometry algorithm are applied at 40 Hz and 15 Hz, respectively. All the processing is performed onboard. During the experiments presented below, the CPU usage averaged at about 40%.

4.3.1 Real-Time Results

Figures 4.5 through 4.8 illustrate the performance of the relative MEKF estimator compared to truth data from the motion capture system during a manually-controlled flight. Velocity truth was generated using a time difference of the 100 Hz motion capture position information, expressed in the body-fixed coordinate frame. The estimates were computed in real time, onboard the aircraft during the manual flight.

In Figure 4.5 we see the results for the relative state r. Each time a new node is declared by the view-matching algorithm, the relative portions of the state are marginalized out and new relative state elements are augmented in. During this flight, this occurred 85 times. The large discontinuities depicted in Figure 4.5 are the result of this marginalization and augmentation process.

The body-fixed frame velocity v results are depicted in Figure 4.6. Notice the lack of discontinuities, as this state is not relative. This demonstrates the ability of the filter to maintain the quality of the non-relative states during the many changes of reference that occur for the relative

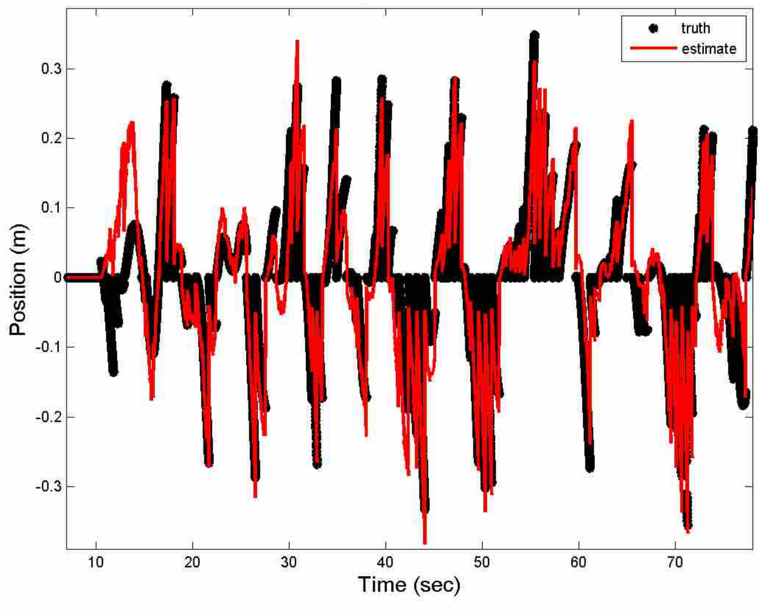

Figure 4.5: Relative right position r truth and estimates. This data is produced using the IMU, altimeter, and VO measurements from data received during a manually-controlled flight. The discontinuities in the plots are due to new nodes being created, causing the truth and the estimates to jump to the new relative position. There are not, however, jumps in the global position estimates. We express the global truth from the motion capture in the relative node coordinate frame for the comparison of these results. Results for the relative front and down positions are similar.

states. Here we also highlight the advantages of using the improved dynamic model and the benefits that the accelerometer measurement updates provide to the velocity states.

The x component of the quaternion $\mathbf{q}_n^b$, depicted in Figure 4.7, roughly corresponds to the roll angle of the hexacopter for this flight. Attitude errors are almost always sub-degree in pitch and roll compared to truth data. We do have the unavoidable drift in yaw as we do not measure a global yaw angle. This could be corrected with the use of loop closures.

The composition of the relative states and the vector chain of edges to the origin provide the global estimate shown in Figure 4.8. At this point, no loop closure and optimization have been applied to these results. These are dead-reckoning results using a MEMs-grade IMU and odometry, through the VO. The estimates trend well with the truth information. The global information is not required in the filter and these results are provided for information only.

Figures 4.5 through 4.8 provide a sample visual confirmation of the quality of the results that the presented filter can produce. Table 4.1 below provides RMS error calculations for the state

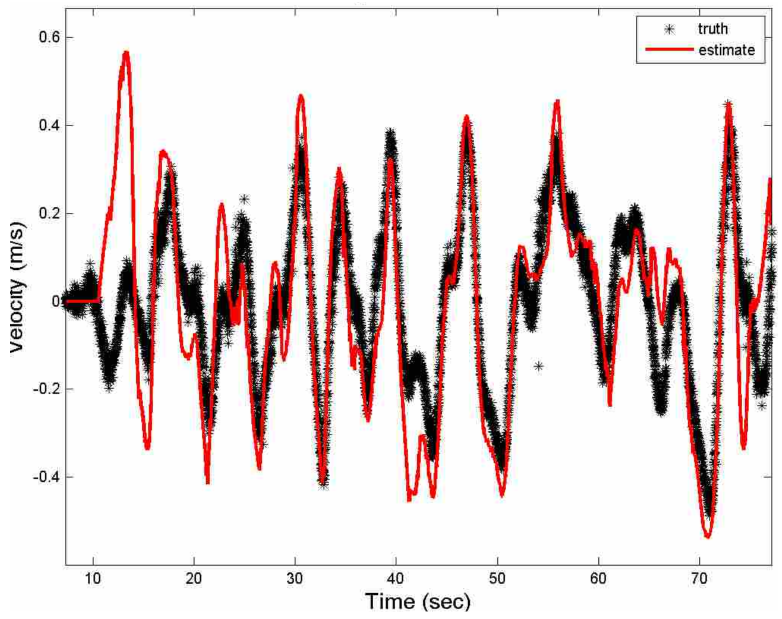

Figure 4.6: Body-fixed frame side velocity v truth and estimate comparison. Notice that there are no discontinuities, as the body-frame velocity is not relative. Results for the front and down body-fixed velocities are similar.

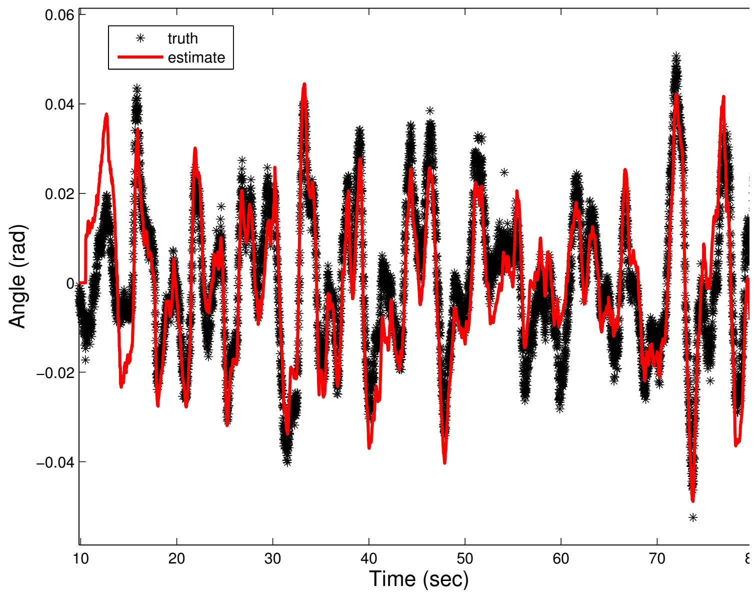

Figure 4.7: The x component of the quaternion $\mathbf{q}_n^b$ truth and estimate comparison. There are not any discontinuities in this figure, as this portion of the quaternion is not relative. Results for the other portions of the quaternion are similar.

estimates over the same manual flight. These results confirm the performance of the states not shown in the figures along with those that are shown.

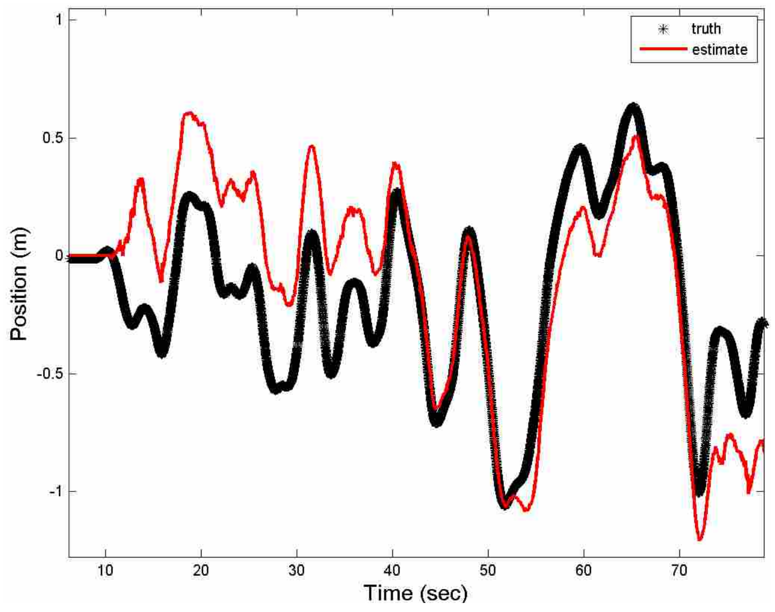

Figure 4.8: A comparison between truth and estimates for the global value for east. This estimate is obtained by a vector sum of the edges and the current state at each time-step. This information is not required for the algorithms, but it is helpful in quantifying the quality of the estimates. Results for global north and down position are similar.

4.3.2 Estimates in Control Feedback: Hover

The vehicle was commanded to hover at a spot one meter above the take-off location for these results. The controller is based off a previous design [122], with some modifications to account for the relative coordinate systems.

Table 4.2 provides the RMS errors of all the states and the global positions during the hover flight. Notice that the performance has improved compared to that of Table 4.1. We believe that this is due to several reasons. The first is that the vehicle was commanded to hover and many more VO measurements compared to the same keyframe were made, which allowed more time for filter convergence. Secondly, since the control is based on the estimates, any deviations affect the control and cause motion excitation that increases the observability of that state; this is especially applicable for the attitude and velocity states. Finally, during the manual flight, faster speeds were obtained which could have contributed to more motion blur or other vision artifacts that could decrease the accuracy of the VO measurements.

Figure 4.9, created based on a similar figure in [67], demonstrates the results from the proposed navigation approach with the current state of the art in estimates-in-the-loop hover per-

Table 4.1: RMS errors in state and global positions from estimates produced during a manually-controlled flight. Motion capture data is used as truth for error computation. State estimates were produced in real time during the flight.

RMS Error in State Estimates	
State	**RMS Error**
front relative position (f)	0.044 (m)
right relative position (r)	0.048 (m)
down relative position (d)	0.029 (m)
x quaternion (q_x)	0.008 (rad)
y quaternion (q_y)	0.010 (rad)
z quaternion (q_z)	0.051 (rad)
w quaternion (q_w)	0.017 (rad)
forward body-fixed velocity (u)	0.086 (m/s)
side body-fixed velocity (v)	0.147 (m/s)
down body-fixed velocity (w)	0.057 (m/s)
global north position (n)	0.548 (m)
global east position (e)	0.325 (m)
global down position (d)	0.061 (m)

Table 4.2: The RMS errors in the state from estimates produced during the estimates-in-the-loop hover flight.

RMS Error for Estimates	
State	**RMS Error**
front relative position (f)	0.019 (m)
right relative position (r)	0.037 (m)
down relative position (d)	0.092 (m)
x quaternion (q_x)	0.006 (rad)
y quaternion (q_y)	0.009 (rad)
z quaternion (q_z)	0.041 (rad)
w quaternion (q_w)	0.017 (rad)
forward body-fixed velocity (u)	0.071 (m/s)
side body-fixed velocity (v)	0.061 (m/s)
down body-fixed velocity (w)	0.051 (m/s)
global north position (n)	0.154 (m)
global east position (e)	0.099 (m)
global down position (d)	0.049 (m)

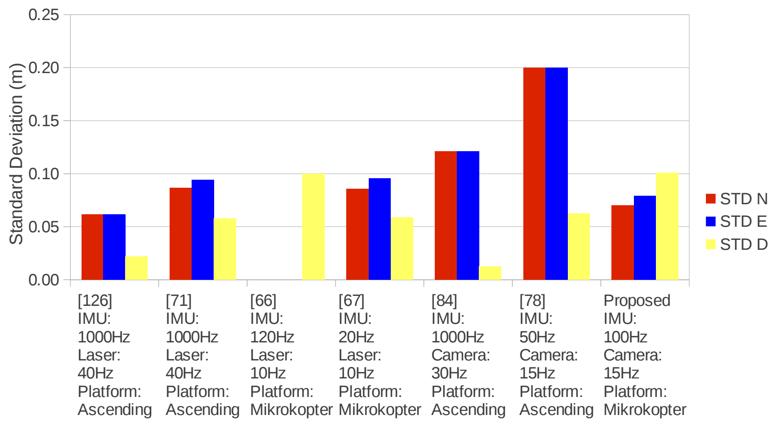

Figure 4.9: A comparison of the standard deviations of estimates-in-the-loop hover performance for several different platforms. This data is based on a similar figure published in [67] using material from [66, 67, 71, 78, 84, 127]. Standard deviations are displayed in the different north, east, and down directions, hence lower numbers denote better performance. Note that the proposed estimation approach is competitive with the state of the art in the north and east directions. Platforms are provided by either MikroKopter or Ascending Technologies.

formance. The figure compares the standard deviations of the positions from the desired hover location, hence a lower number denotes increased performance. Here we see that the proposed approach is close to state-of-the-art performance when compared to any sensor and performs better in north and east by a significant margin than either of the other methods that use a camera. The current weight of the test platform is close to the max weight the motors can support. We expect that the down performance of the hover results would improve with a reduction in weight.

4.3.3 Estimates in Control Feedback: Waypoint Path

Figures 4.10 through 4.14 demonstrate the results of some of the states during a flight with the state estimates in the control loop. The vehicle avoided obstacles by following paths generated by an onboard path-planning algorithm. Goal locations for the vehicle were provided as inputs from a remote operator.

Figure 4.10 provides a comparison of the true and estimated global 3D paths of the vehicle. Recall that global information is not a necessary input to the filter. These dead-reckoning results, created using a MEMs-grade IMU and VO (through the MEKF), are provided for information only. In Figure 4.11 we see the results for the relative forward position state f. There were 26 new

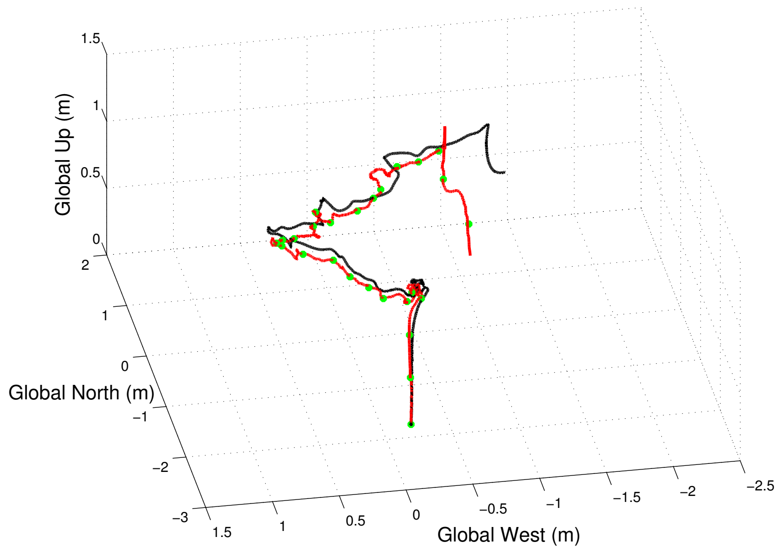

Figure 4.10: A comparison between truth and estimates for the global 3D path. This estimate is obtained by a vector sum of the edges and the current state at each time-step. This information is not required for the algorithms, but it is helpful in quantifying the quality of the estimates. The green dots in the figure denote locations of the keyframes. The hexacopter maneuvers around some obstacles to achieve a goal location selected by a remote operator.

nodes created during this autonomous flight. Notice that each node transition causes jumps in the relative states, but the other states, including the global estimates, transition smoothly through node creation. The body frame velocity u results are depicted in Figure 4.12. The estimates track the truth, even though the magnitude of the speed is small. The y and z components of the quaternion $\mathbf{q}_n^b$ are shown in Figures 4.13 and 4.14. The y quaternion roughly corresponds to the pitch angle of the hexacopter for this flight and the z component is close to the yaw angle of the vehicle. The yaw depicted in Figure 4.14 is the global yaw for ease of comparison.

IMU-Camera Calibration Parameters

Figures 4.15 and 4.16 show the parameters for the transformation between the camera frame and the body-fixed frame. These calibration parameters can be included in the state for refinement. Once adequate estimates of these parameters are found, they are saved as constants and removed from the state. Then anytime either the IMU or camera is physically adjusted on the platform, they can be easily reintroduced by changing two simple pound-define statements and re-estimated.

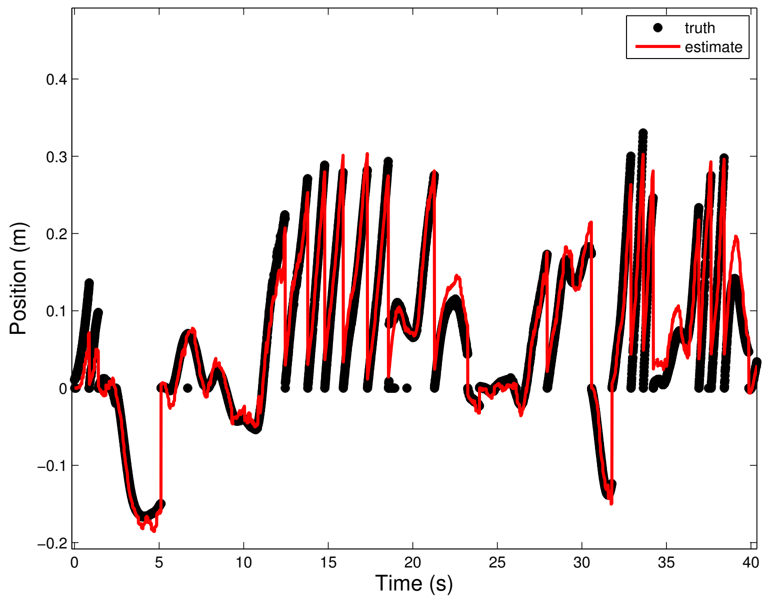

Figure 4.11: Relative front position f truth and estimates. This data is from an autonomous flight with estimates in the control loop. The discontinuities in the plots are due to new nodes being created, causing the truth and the estimates to jump to the new relative position. The global positions do not jump when new nodes are created. We express the truth from the motion capture in the relative node coordinate frame for the comparison of these results. Results for the relative right and down positions are similar.

4.4 Conclusions

A relative, vision-based framework, like the approach described here, is an important step in furthering the capabilities of indoor aerial navigation. Current approaches that require globally referenced states often suffer deficiencies from the need for additional state elements to incorporate relative measurements, waiting periods to process global consistency, computation demands on ground stations, or schemes to accommodate large jumps in pose when loop closures are applied. Utilizing a relative approach allows more flexibility as the critical real-time processes of localization and control do not depend on computationally demanding optimization and loop-closure processes. Additionally, relative exteroceptive measurement updates are supported natively in the proposed MEKF and front-facing keyframes provide a rich source of information for path planning.

We have shown the viability and accuracy of the proposed relative MEKF through hardware results comparing estimates to truth information. We show that we can reliably estimate relative

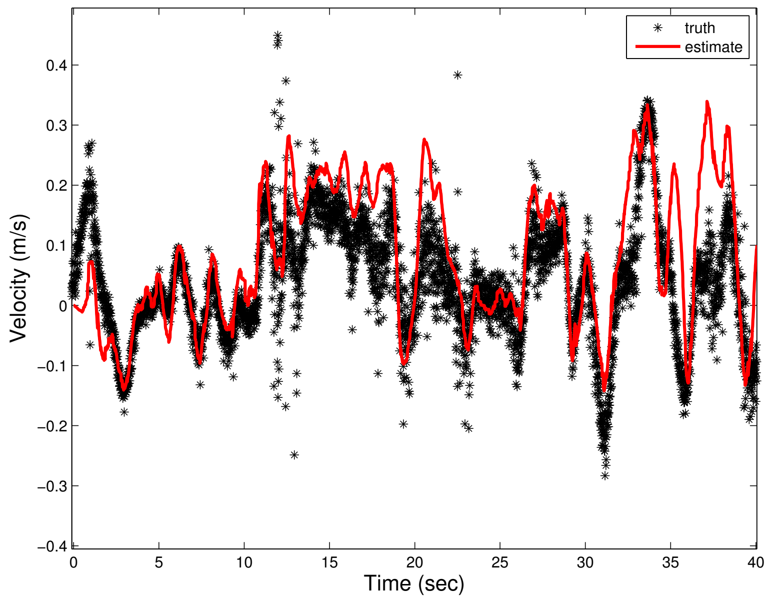

Figure 4.12: Body-fixed frame forward velocity *u* truth and estimate comparison. Notice that there are no discontinuities, as the body-frame velocity is not relative. Results for the right and down body-fixed velocities are similar.

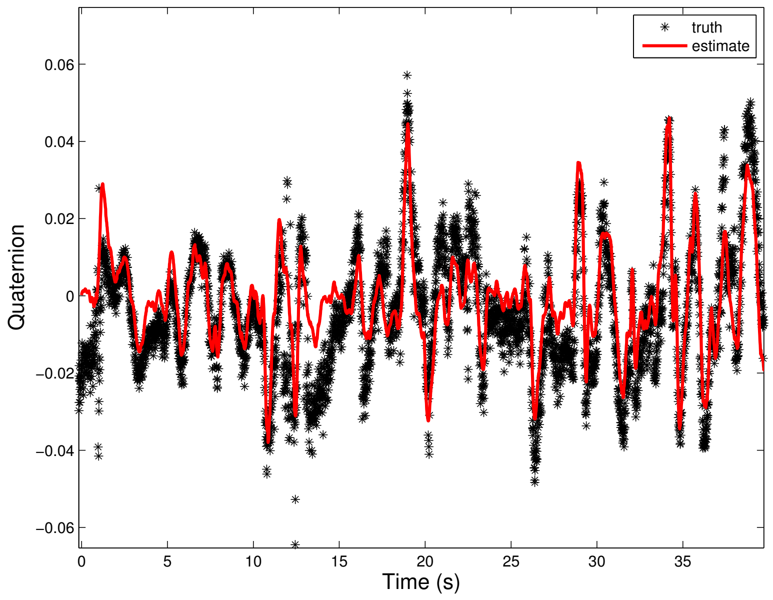

Figure 4.13: The *y* component of the quaternion $\mathbf{q}_n^b$, which is approximately the pitch angle of the hexacopter, comparison of truth and estimates. There are not any discontinuities, as this portion of the quaternion is not relative.

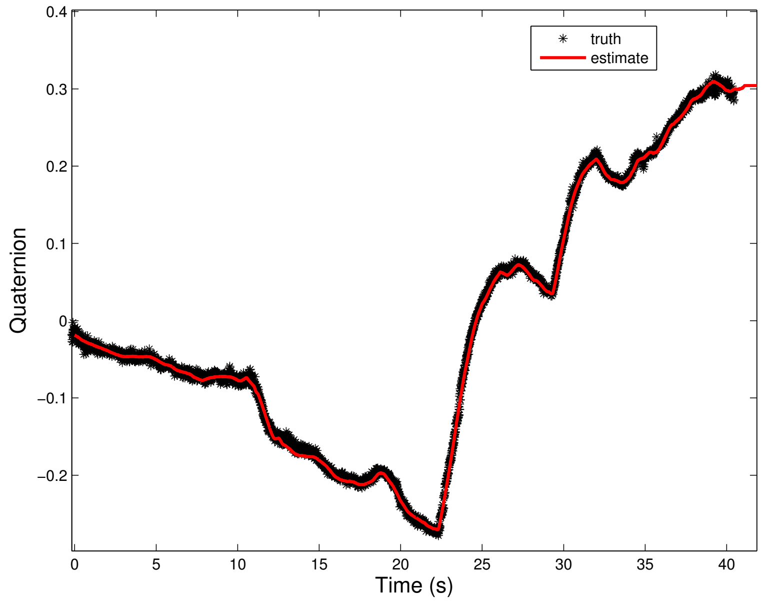

Figure 4.14: The z component of the quaternion $\mathbf{q}_n^b$ truth and estimate comparison. This is approximately equal to the yaw angle for this flight. Note that there is some drift as we cannot measure the global yaw angle. We do not show the relative yaw angle in these results, even though that is what is actually tracked in the state vector.

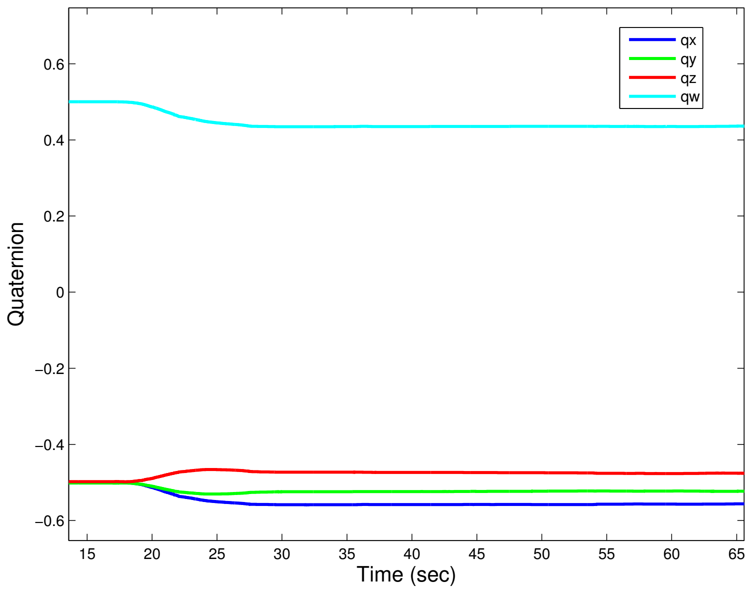

Figure 4.15: The refinement of the rotation between the camera coordinate frame and the body-fixed frame is shown here. These rotation parameters can be included in the state and then saved as constants.

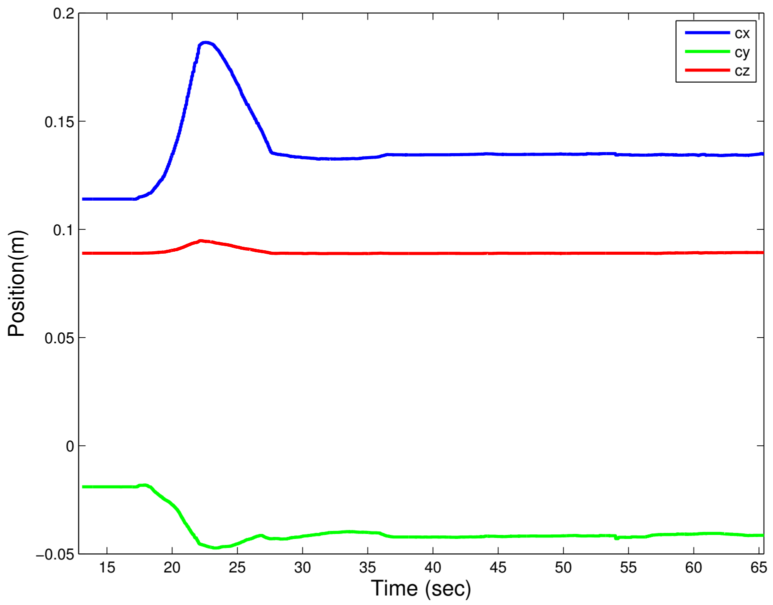

Figure 4.16: The location of the camera focal point, expressed in the body-fixed frame, is shown being refined here. These translation parameters can be included in the state and then saved as constants.

states, marginalize out states, augment new states when a new node is created, and smoothly estimate non-relative states during flights. The estimates are robust enough for utilization in feedback control. The demonstrated feedback-controlled hover results exceed the capabilities of contemporary vision-based approaches.

CHAPTER 5. CONCLUSIONS AND FUTURE WORK

This dissertation presents an architecture and algorithms for vision-based autonomous flight in unstructured, confined, and unknown GPS-denied environments. The novelty of the approach lies in the use of many local coordinate systems instead of a single global frame as the basis for the real-time processes that are critical for flight. This method, which we refer to as relative navigation, eliminates the need for globally-consistent information in the processes critical for flight, which provides a greater amount of flexibility than is available with typical approaches. While we have provided results and experiments to prove the success and viability of the proposed architecture, work remains to be done to have a complete system that is capable of operating in the proposed setting. We briefly discuss this future work below and then conclude with a summary of the contributions of this research.

5.1 Future Work

There is one item of future work for the estimation approach discussed in Chapter 2 that should be considered. We believe that it would be beneficial to implement the developed estimators on a low-level autopilot board and then to compare the performance of the traditional and drag-force-enhanced models on similar (or better yet, the same) multirotor vehicles with estimates in the control loop. A rigorous side-by-side comparison would further demonstrate the benefits of using the drag-force-enhanced model for multirotor estimation and control.

Other work remains that pertains to the relative navigation approach discussed in Chapter 3. In particular, a lot of work is necessary to further develop the back-end subsystem shown in Figure 5.1. Developing the back end was outside the scope of this project. There is also some work that could be done to improve the elements of the front-end subsystem. These improvements are briefly discussed in the following paragraphs.

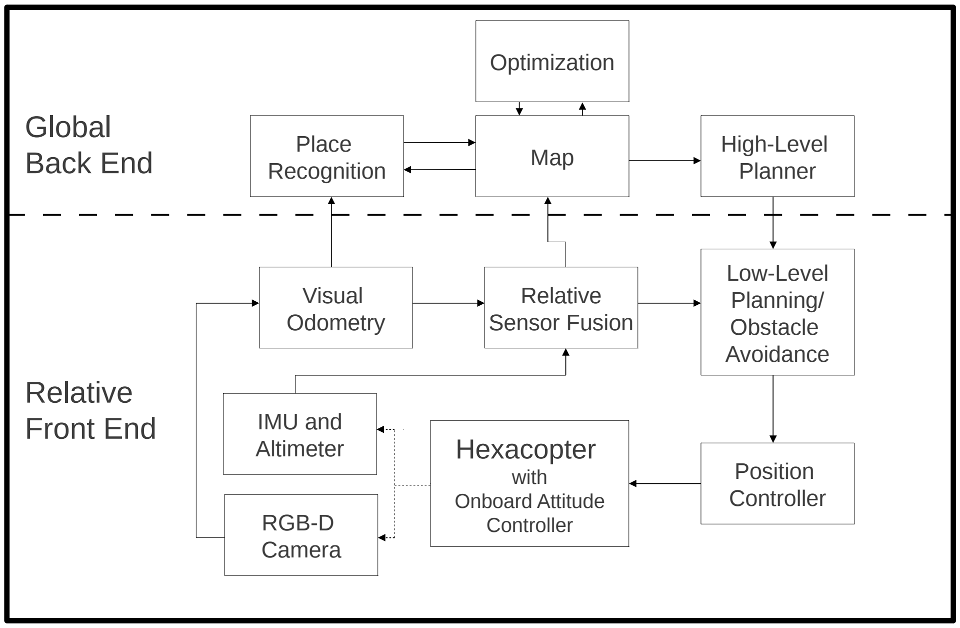

Figure 5.1: Software architecture for the proposed system, as described in Chapter 3. The front end provides relative navigation based on keyframes from the VO algorithm. The back end provides a globally-consistent navigation solution. Notice that the only flow of information from the back end to the front end is provided by the high-level planner. Most current scenarios require feedback from the costly optimization and place recognition algorithms to the estimation and control. ROS provides the functionality represented by the arrows between components.

5.1.1 Back-end Development

The back-end subsystem will require a considerable effort to be transformed from its current state to a working SLAM system. It is helpful that a great deal of research has recently been completed in this area for ground robotics. We will briefly discuss work to refine the relative transformations provided by the front end, provide essential content for the map, begin the place recognition, furnish the needed advancements for the optimization, and develop algorithms for the high-level planner.

One potential improvement for better back-end global information, without needing loop closures, would be to align point clouds produced from the keyframe images. It has been shown that aligning high-density point clouds, using an iterative closest point (ICP) algorithm, can provide more accurate relative transformations than VO [27, 28]. This process is, however, computationally intensive and can be completed only at a rate between 0.5 and 1 Hz. An algorithm could be implemented,[1] and added to the back end, that would align the keyframe point clouds. The relative transformations provided by the MEKF could be used as a starting point for the algorithm. Using only keyframes, the requests would occur at a frequency similar to the requisite slower computational update rate and improved global information would be available to the back end.

The map, in its current state, requires a great deal of work to arrive at the capability necessary to achieve the missions described in Chapter 1. It must be able to be queried, so that other processes, like the place recognition, can request images from specific keyframes. It should be able to be pruned, so that an image that provides essentially the same information as another can be removed to decrease storage space and optimization requirements. Traversal and retrieval of images in the map must also be fast, so that the relative front end could possibly utilize a saved image in navigating through a known map.

We believe that the appropriate way-forward for developing the place recognition block in Figure 5.1 is to begin with one of the open-source libraries that have been recently published online. The seminal work of [34], for example, is now available through Open FabMap.[2] After an open-source system has been implemented and evaluated, improvements essential for a flying implementation could be made based on the observed gaps in capability.

A solid base for an improved optimization system is described in [124]. Improvements are required, however, to provide the full 6DoF optimization solutions necessary for an aerial implementation. Theory must also be developed for the arbitrary placement of loop closures within the pose graph. Finally, code must be written to make the method fast and compatible with the ROS framework that is currently being used.

Developing content for the high-level planner was originally the target we set out to achieve with this research. This block will provide the intelligent behavior the vehicle requires to perform autonomous missions and will open up many possible research avenues in the near future. Two algorithms are essential for basic fully autonomous capability: one for exploration and one to efficiently navigate through a known map. These algorithms should be first priority in the development of the high-level planner. Other possible algorithms include recreating those generated through the

[1] See the off-the-shelf capability provided in the Point Cloud Library: urlhttp://pointclouds.org/

[2] `http://code.google.com/p/openfabmap/`

research the MAGICC Lab has completed using fixed-wing UAVs with GPS information [128], such as target tracking, efficient search of an environment, formation flight, or cooperative control.

5.1.2 Front-end Improvements

Establishing and demonstrating the front-end subsystem was the objective of this research, consequently, there are only a few improvement suggestions for the local navigation and control. We highlight possible upgrades for the path planning, VO, and the estimation and control.

The path planning algorithm that we utilize was originally developed for use on ground robots within a global frame of reference. We use the planner for relative plans on an aerial vehicle. The solution has worked quite well, considering that a few of the main assumptions have been violated. There is, however, one suggestion for improvement. The cost map that forms the basis of the path planner should be adapted to more appropriately handle node transitions. When the relative transformations between nodes are published, the planner should apply the transformations to all of the obstacles in the cost map. This way the motion of the obstacles in the node frame will be appropriately accounted for.

An additional update for the path planning is to compute 3D paths through the environment. Currently the 3D information in the cost map is projected onto the node plane and then 2D path planning methods are employed. For truly autonomous behavior in confined environments, 3D path planning will be essential.

There are two possible improvements to the VO algorithm. The first would be to develop some form of relative sparse bundle adjustment (SBA) to improve the accuracy of the solutions. The other would be to utilize a graphics processing unit (GPU) for speedup.

Originally, we implemented SBA using the open source library in [63] . We were disappointed to find that the relative transformations were not improved. We found that the goal of SBA, in general, is to improve the *global estimate* created by combining the relative transformations, and not to necessarily improve the relative transformations themselves. We believe that work could be done, perhaps using the algorithms discussed in [124] or in [62], to optimize the relative transformations for the VO over a window of frames.

The second improvement for the VO would be to utilize routines available in OpenCV to take advantage of GPU processing. Enabling the GPU to perform some of the work for the VO could save valuable computation time for other processes, and at the same time speed up the frame rate and increase the number of features used in the motion estimation.

A large improvement for the sensor fusion and control algorithms would be to move them onto an embedded-processing board that is running a real-time operating system. Then the current autopilot of the system, which only allows communication over serial at about 20 Hz, could be replaced. Ideally, we would like to process the MEKF and the control, access the IMU and altimeter information, and have direct outputs to the speed controllers for the motors, all on the same board. The measurement updates from the VO would continue to be processed on the computer, but they would be relayed to the MEKF on the smaller board. Currently, we are running everything on the onboard computer. While the MEKF and control do not consume much processing on the computer, it would be preferable to have them running on a real-time system and to be able to compute the control commands at the same rate the estimates are being computed (100 Hz).

The final improvement for the front-end subsystem would be to estimate the ground, so that the vehicle could fly over obstacles or up stairs. Currently we assume a flat floor in the model for the altimeter measurement update in Chapter 4. The approach in [71] is the only one, of which we are aware, that provides results demonstrating this capability. We recommend that a similar approach, which uses the essential elements of relative navigation, be created and implemented.

5.2 Final Summary

A system capable of completing autonomous flight in unstructured, confined, and unknown GPS-denied environments could provide a great deal of value by inspecting and surveying areas that are difficult to access and that present a great amount of risk. Autonomous capability would be critical for enabling the vehicle to intelligently explore an environment and avoid obstacles. The goal of this research has been to develop a new framework that enables improved solutions to this problem and to validate the approach with experiments using a hardware prototype.

In Chapter 2 we examined the consequences and practical aspects of using an improved dynamic model for quadrotor state estimation with only IMU measurements. The improved model correctly explains the measurements available from the accelerometers on a quadrotor. We give a brief tutorial to explain the difference between accelerometer measurements on a ground vehicle and on a quadrotor. We demonstrate several observers based on the improved model to illustrate the estimation improvements available. It is shown that the parameters for this model can be estimated as elements of the state, which makes the model simple to implement and utilize. Finally, we provide hardware results demonstrating that improved attitude, velocity and position estimates can be achieved through the use of the model.

We proposed a new architecture to simplify some of the challenges that constrain GPS-denied aerial flight in Chapter 3. At the core, the approach combines visual graph-SLAM with a MEKF. More importantly, in the front end we depart from the common practice of estimating global states and instead keep the position and yaw states of the MEKF *relative* to the current node in the map. This relative navigation approach provides simple application of sensor measurement updates, intuitive definition of map edges and covariances, and the flexibility of using a globally consistent map when desired. We discuss the architecture of this new system and provide important details for each component. We verify the approach with goal-directed autonomous flight-test results.

The basis of the new relative navigation approach, the MEKF, is detailed in Chapter 4. It employs the drag-force-enhanced model discussed in Chapter 2. We derive the relative filter and demonstrate how the states must be augmented and marginalized each time a new node is declared. We define the transformations necessary to create the node coordinate frame using the current attitude estimate and the keyframe coordinate frame. Measurement updates are derived for use in the indirect form of the MEKF. The relative estimation approach is verified using hardware flight test results accompanied by comparisons to motion capture truth. The ability to refine the transformation between the IMU and camera is also demonstrated. Finally, flight results with estimates in the control loop are shown.

We believe that the relative, vision-based framework described in this work is an important step in furthering the capabilities of indoor aerial navigation in confined, unknown environments. As we discussed, current approaches incur challenging problems by requiring globally-referenced states. Utilizing a relative approach allows more flexibility as the critical, real-time processes of localization and control do not depend on computationally-demanding optimization and loop-closure processes. Relative exteroceptive measurement updates are supported natively in the proposed MEKF and front-facing keyframes provide a rich source of information for path planning. The graph map also provides potential support for a variety of constraints, such as intermittent GPS and semantic information.

REFERENCES

[1] N. Schwartz, “The United States as an Aerospace Nation: Challenges and Opportunities,” in *IFPA Fletcher Conf. on National Security Strategy and Policy*, Medford, MA, Jan. 2010. 1

[2] S. Peterson and P. Faramarzi, “Iran hijacked US drone, says Iranian engineer,” Web, 2011. [Online]. Available: http://www.csmonitor.com/World/Middle-East/2011/1215/Exclusive-Iran-hijacked-US-drone-says-Iranian-engineer-Video 1

[3] A. Moaveni, “Why Iran’s Capture of a “U.S. Drone” Matters - to Iran,” Web, Dec. 2012. [Online]. Available: http://world.time.com/2012/12/05/why-irans-capture-of-a-u-s-drone-matters-to-iran/ 1

[4] M. Quigley, B. Gerkey, K. Conley, J. Faust, T. Foote, J. Leibs, E. Berger, R. Wheeler, and A. Ng, “ROS : an open-source Robot Operating System,” in *IEEE Intl. Conf. on Robotics and Automation Workshop on Open Source Robotics*, Kobe, Japan, 2009. 6, 57

[5] J.-H. Kim and S. Sukkarieh, “Airborne simultaneous localisation and map building,” in *IEEE Int. Conf. Robotics and Automation*, vol. 1, 2003, pp. 406–411. 8

[6] P. Piniés, T. Lupton, S. Sukkarieh, and J. D. Tardós, “Inertial Aiding of Inverse Depth SLAM using a Monocular Camera,” in *IEEE Intl. Conf. on Robotics and Automation*, no. April, 2007, pp. 2797–2802. 8, 10

[7] M. Agrawal, K. Konolige, and R. C. Bolles, “Localization and mapping for autonomous navigation in outdoor terrains: A stereo vision approach,” *IEEE Workshop Applications of Computer Vision*, pp. 7–12, 2007. 8, 11

[8] C. Mei, G. Sibley, M. Cummins, P. Newman, and I. Reid, “A Constant-Time Efficient Stereo SLAM System,” in *Proc. of the British Machine Vision Conference*, 2009, pp. 54.1–54.11. 8, 11, 16

[9] M. Montemerlo, S. Thrun, D. Koller, and B. Wegbreit, “FastSLAM: A factored solution to the simultaneous localization and mapping problem,” in *Proc. of the National Conf. on Artificial Intelligence*, Edmonton, Canada, 2002, pp. 593–598. 8, 11

[10] S. Thrun, L. Y., D. Koller, A. Y. Ng, Z. Ghahramani, and H. Durrant-whyte, “Simultaneous Localization and Mapping with Sparse Extended Information Filters,” *International Journal of Robotics Research*, vol. 23, no. 7-8, pp. 693–716, Aug. 2004. 8

[11] P. Newman, D. Cole, and K. Ho, “Outdoor SLAM using visual appearance and laser ranging,” in *IEEE Intl. Conf. on Robotics and Automation*, 2006, pp. 1180–1187. 8

[12] C. Bibby and I. Reid, "A hybrid SLAM representation for dynamic marine environments," in *IEEE Intl. Conf. on Robotics and Automation*, May 2010, pp. 257–264. 8

[13] M. Dissanayake, P. Newman, S. Clark, H. Durrant-Whyte, and M. Csorba, "A solution to the simultaneous localization and map building (SLAM) problem," *IEEE Trans. on Robotics and Automation*, vol. 17, no. 3, pp. 229–241, June 2001. 8, 10

[14] J. Spletzer, "A new approach to range-only SLAM for wireless sensor networks," Lehigh University, Bethlehem, PA, Tech. Rep., 2003. 8

[15] 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. 8

[16] J. Biswas and M. Veloso, "WiFi localization and navigation for autonomous indoor mobile robots," in *IEEE Intl. Conf. on Robotics and Automation*, May 2010, pp. 4379–4384. 8

[17] D. Nister, O. Naroditsky, and J. Bergen, "Visual odometry," in *Proc. IEEE Conf. on Computer Vision and Pattern Recognition*, vol. 1, 2004. 8, 9, 78

[18] B. D. Scaramuzza and F. Fraundorfer, "Visual Odometry Part I: The First 30 Years and Fundamentals," *Robotics and Automation Magazine*, no. December, pp. 80—-92, 2011. 8, 57, 58

[19] F. Fraundorfer and D. Scaramuzza, "Visual odometry Part 2: Matching, Robustness, Optimization, and Applications," *Robotics and Automation Magazine*, no. June, pp. 78—-90, 2012. 8, 57

[20] A. J. Davison, I. D. Reid, N. D. Molton, and O. Stasse, "MonoSLAM: Real-Time Single Camera SLAM," *IEEE Trans. on Pattern Analysis and Machine Intel.*, vol. 29, no. 6, pp. 1052–1067, June 2007. 8, 10

[21] G. Klein and D. Murray, "Parallel tracking and mapping for small AR workspaces," in *IEEE and ACM Int. Symposium on Mixed and Augmented Reality*, Washington, DC, USA, 2007, pp. 1–10. 8, 11, 14, 52, 78

[22] S. Weiss, M. Achtelik, S. Lynen, M. Chli, and R. Siegwert, "Real-time onboard visual-inertial state estimation and self-calibration of MAVs in unknown environments," in *IEEE Intl. Conf. Robotics and Automation*, 2012. 8, 14, 15, 52, 53, 77, 78, 79

[23] K. Konolige, M. Agrawal, and J. Sola, "Large scale visual odometry for rough terrain," in *Intl. Symposium on Robotics Research,*, 2007. 8, 9

[24] L. M. Paz, P. Pinies, J. D. Tardos, and J. Neira, "Large-Scale 6-DOF SLAM With Stereo-in-Hand," *IEEE Transactions on Robotics*, vol. 24, no. 5, pp. 946–957, 2008. 8, 11

[25] L. R. García Carrillo, A. E. Dzul López, R. Lozano, and C. Pégard, "Combining Stereo Vision and Inertial Navigation System for a Quad-Rotor UAV," *Journal of Intelligent & Robotic Systems*, vol. 65, no. 1-4, pp. 1–15, Aug. 2011. 8, 13, 22, 51

[26] T. Tomic, K. Schmid, P. Lutz, M. Kassecker, E. Mair, I. Grixa, F. Ruess, M. Suppa, and D. Burshka, "Toward a Fully Autonomous UAV: Research Platform for Indoor and Outdoor

Urban Search and Rescue," *Robotics and Automation Magazine*, no. September, 2012. 8, 14, 15, 16, 52, 53, 56, 77, 78, 79

[27] F. Endres, J. Hess, N. Engelhard, J. Sturm, D. Cremers, and W. Burgard, "An evaluation of the RGB-D SLAM system," in *IEEE Intl. Conf. on Robotics and Automation*, May 2012, pp. 1691–1696. 9, 114

[28] W. L. D. Lui, T. J. J. Tang, T. Drummond, and W. H. Li, "Robust egomotion estimation using ICP in inverse depth coordinates," *IEEE Intl. Conf. on Robotics and Automation*, pp. 1671–1678, May 2012. 9, 114

[29] A. S. Huang, A. Bachrach, P. Henry, M. Krainin, D. Maturana, D. Fox, and N. Roy, "Visual Odometry and Mapping for Autonomous Flight Using an RGB-D Camera," in *Int. Symposium on Robotics Research*, Flagstaff, Arizona, USA, 2011. 9, 13, 15, 51, 53, 77, 78, 79

[30] L. Heng, G. H. Lee, F. Fraundorfer, and M. Pollefeys, "Real-Time Photo-Realistic 3D Mapping for Micro Aerial Vehicles," in *IEEE Intl. Conf. on Intelligent Robots and Systems*, San Francisco, CA, 2011, pp. 4012—-4019. 9, 15

[31] M. Fiala and A. Ufkes, "Visual Odometry Using 3-Dimensional Video Input," *2011 Canadian Conference on Computer and Robot Vision*, pp. 86–93, May 2011. 9

[32] G. Sibley, C. Mei, I. Reid, and P. Newman, "Planes, trains and automobiles: autonomy for the modern robot," in *IEEE Int. Conf. on Robotics and Automation*, May 2010, pp. 285–292. 9, 17, 55, 82

[33] H. Durrant-Whyte and T. Bailey, "Simultaneous localisation and mapping (SLAM): Part I the essential algorithms," *Robotics and Automation Magazine*, vol. 13, no. 2, pp. 99–110, 2006. 9, 10

[34] M. Cummins and P. Newman, "FAB-MAP: Probabilistic Localization and Mapping in the Space of Appearance," *The International Journal of Robotics Research*, vol. 27, no. 6, pp. 647–665, June 2008. 10, 11, 16, 67, 78, 115

[35] ——, "Accelerated appearance-only SLAM," in *IEEE Intl. Conf. on Robotics and Automation*, May 2008, pp. 1828–1833. 10

[36] ——, "Highly scalable appearance-only SLAM - FAB-MAP 2.0," in *Proc. of Robotics: Science and Systems*, Seattle, USA, June 2009, pp. 1–8. 10, 67

[37] R. Paul and P. Newman, "FAB-MAP 3D: Topological mapping with spatial and visual appearance," in *IEEE Intl. Conf. on Robotics and Automation*, 2010, pp. 2649–2656. 10

[38] T. Bailey and H. Durrant-Whyte, "Simultaneous Localisation and Mapping (SLAM): Part II State of the Art," *Robotics and Automation Magazine*, vol. 13, no. 3, pp. 1–9, Sept. 2006. 10

[39] S. Thrun, W. Burgard, and D. Fox, *Probabilistic Robotics*. The MIT Press, 2006. 10, 11

[40] S. Thrun, “Simultaneous localization and mapping,” in *Robotics and cognitive approaches to spatial mapping*. Springer, 2008, pp. 871–889. 10, 11, 78

[41] G. Grisetti, R. Kummerle, C. Stachniss, and W. Burgard, “A tutorial on graph-based SLAM,” *IEEE Intelligent Transportation Systems Magazine*, no. Winter, pp. 31–43, Feb. 2010. 10, 11, 78

[42] M. Bryson and S. Sukkarieh, “Active airborne localisation and exploration in unknown environments using inertial SLAM,” in *Proc. IEEE Aerospace Conf*, 2006. 10

[43] M. Veth, J. Raquet, and A. Force, “Fusion of Low-Cost Imaging and Inertial Sensors for Navigation,” Air Force Inst of Tech, Wright-Patterson AFB, OH, Tech. Rep. A469264, 2007. 10

[44] M. Bryson and S. Sukkarieh, “Co-operative Localisation and Mapping for Multiple UAVs in Unknown Environments,” in *Proc. IEEE Aerospace Conf*, 2007, pp. 1–12. 10

[45] ——, “Architectures for Cooperative Airborne Simultaneous Localisation and Mapping,” *Journal of Intelligent and Robotic Systems*, vol. 55, no. 4, pp. 267–297, Jan. 2009. 10

[46] J. Artieda, J. M. Sebastian, P. Campoy, J. F. Correa, I. F. Mondragón, C. Martínez, and M. Olivares, “Visual 3-D SLAM from UAVs,” *Journal of Intelligent and Robotic Systems*, vol. 55, no. 4-5, pp. 299–321, Jan. 2009. 10

[47] A. Nemra and N. Aouf, “Robust Airborne 3D Visual Simultaneous Localization and Mapping with Observability and Consistency Analysis,” *Journal of Intelligent and Robotic Systems*, vol. 55, no. 4, pp. 345–376, Jan. 2009. 10

[48] S. J. Julier and J. K. Uhlmann, “A counter example to the theory of simultaneous localization and map building,” in *IEEE Intl. Conf. on Robotics and Automation*, vol. 4, no. Ic, 2001, pp. 4238 – 4243. 10

[49] J. Castellanos, J. Neira, and J. Tardós, “Limits to the consistency of EKF-based SLAM,” *Robotics*, vol. 45, no. 10, pp. 1878–1881, 2004. 10

[50] T. Bailey, J. Nieto, and E. Nebot, “Consistency of the FastSLAM Algorithm,” in *Proc. IEEE Int. Conf. on Robotics and Automation*, no. May, 2006, pp. 424–429. 10, 11

[51] P. Piniés and J. D. Tardós, “Large-Scale SLAM Building Conditionally Independent Local Maps: Application to Monocular Vision,” *IEEE Transactions on Robotics*, vol. 24, no. 5, pp. 1094–1106, Oct. 2008. 11

[52] D. Hahnel, W. Burgard, D. Fox, and S. Thrun, “An efficient FastSLAM algorithm for generating maps of large-scale cyclic environments from raw laser range measurements,” in *IEEE Intl. Conf. Intelligent Robots and Systems*, vol. 1, 2003, pp. 206–211. 11

[53] K. Murphy and S. Russell, *Sequential Monte Carlo Methods in Practice*. Springer, 2001, pp. 499–516. 11

[54] J. L. Blanco, J. A. Fernandez-Madrigal, and J. Gonzalez, "A Novel Measure of Uncertainty for Mobile Robot SLAM with Rao Blackwellized Particle Filters," *Intl. Journal of Robotics Research*, vol. 27, no. 1, pp. 73–89, 2008. 11

[55] J. Folkesson and H. Christensen, "Graphical SLAM - a self-correcting map," in *IEEE Intl. Conf. on Robotics and Automation*, 2004, pp. 383–390. 11

[56] D. Fox, J. Ko, K. Konolige, B. Limketkai, D. Schulz, and B. Stewart, "Distributed Multi-robot Exploration and Mapping," *Proceedings of the IEEE*, vol. 94, no. 7, pp. 1325–1339, July 2006. 11

[57] K. Konolige and M. Agrawal, "FrameSLAM: From Bundle Adjustment to Real-Time Visual Mapping," *IEEE Trans. on Robotics*, vol. 24, no. 5, pp. 1066–1077, 2008. 11

[58] J.-l. Blanco, S. Member, and J.-a. Fern, "Toward a Unified Bayesian Approach to Hybrid Metric Topological SLAM," *IEEE Trans. on Robotics*, vol. 24, no. 2, pp. 259–270, 2008. 11, 16

[59] K. Konolige, J. Bowman, J. D. Chen, P. Mihelich, W. Garage, M. Park, M. Calonder, V. Lepetit, P. F. Epfl, and P. Fua, "View-based Maps," *Intl. Journal of Robotics Research*, vol. 29, no. 8, pp. 941–957, 2010. 11

[60] B. Williams and I. Reid, "On combining visual SLAM and visual odometry," in *IEEE Intl. Conf. on Robotics and Automation*, May 2010, pp. 3494–3500. 11

[61] M. I. a. Lourakis and A. a. Argyros, "SBA: A software package for generic sparse bundle adjustment," *ACM Trans. Math. Softw.*, vol. 36, no. 1, pp. 1–30, Mar. 2009. 11

[62] G. Sibley, C. Mei, I. Reid, and P. Newman, "Adaptive relative bundle adjustment," in *Robotics Science and Systems Conference*, June 2009, pp. 1–8. 11, 17, 116

[63] R. Kummerle, G. Grisetti, H. Strasdat, K. Konolige, and W. Burgard, "g2o: A General Framework for Graph Optimization," in *IEEE Int. Conf. on Robotics and Automation*, Shanghai, May 2011, pp. 3607–3613. 11, 67, 116

[64] K. Konolige, J. Bowman, J. D. Chen, P. Mihelich, M. Calonder, V. Lepetit, and P. Fua, "View-based Maps," in *Proc. of Robotics: Science and Systems*, vol. 29, no. 8, Seattle, USA, June 2009. 11, 53

[65] G. Chowdhary, D. M. Sobers, C. Pravitra, A. Wu, C. Christman, H. Hashimoto, C. Ong, R. Kalghatgi, E. N. Johnson, J. D. Michael Sobers, C. Christmann, and R. Kalghatg, "Integrated Guidance Navigation and Control for a Fully Autonomous Indoor UAS," in *AIAA Guidance Navigation and Control*, 2011. 12, 22

[66] S. Grzonka, G. Grisetti, and W. Burgard, "Towards a navigation system for autonomous indoor flying," in *IEEE Intl. Conf. Robotics and Automation*, no. Section III, May 2009, pp. 2878–2883. 12, 24, 107

[67] I. Sa and P. Corke, "System Identification, Estimation and Control for a Cost Effective Open-Source Quadcopter," in *IEEE Int. Conf. on Robotics and Automation*, 2012, pp. 2202–2209. 12, 22, 24, 105, 107

[68] M. Achtelik and A. Bachrach, "Autonomous navigation and exploration of a quadrotor helicopter in GPS-denied indoor environments," *Robotics: Science and Systems*, 2009. 12, 78

[69] A. Bachrach, R. He, and N. Roy, "Autonomous Flight in Unstructured and Unknown Indoor Environments," in *Proc. of the EMAV Conference*. European Micro Air Vechicle, Sept. 2009, pp. 2–9. 12, 13, 51, 53, 77, 78

[70] A. Bachrach, S. Prentice, N. Roy, R. He, A. D. Winter, and G. Hemann, "RANGE: Robust autonomous navigation in GPS-denied environments," *Journal of Field Robotics*, vol. 28, no. 5, pp. 644–666, Sept. 2011. 12, 13, 15, 21, 23, 24, 37, 77, 78, 79

[71] S. Shen, N. Michael, and V. Kumar, "Autonomous multi-floor indoor navigation with a computationally constrained MAV," in *IEEE Intl. Conf. on Robotics and Automation*, May 2011, pp. 20–25. 13, 15, 16, 21, 51, 77, 78, 107, 117

[72] ——, "Autonomous indoor 3D exploration with a micro-aerial vehicle," in *IEEE Intl. Conf. on Robotics and Automation*, 2012. 13

[73] E. Altug, J. P. Ostrowski, and C. J. Taylor, "Control of a Quadrotor Helicopter Using Dual Camera Visual Feedback," *The International Journal of Robotics Research*, vol. 24, no. 5, pp. 329–341, May 2005. 13, 51

[74] G. P. Tournier, M. Valenti, J. P. How, and E. Feron, "Estimation and control of a quadrotor vehicle using monocular vision and moire patterns," in *AIAA Guidance, Navigation and Control Conference*, no. August, 2006, pp. 21–24. 13, 51

[75] J. Courbon, Y. Mezouar, N. Guenard, and P. Martinet, "Visual navigation of a quadrotor aerial vehicle," *IEEE Intl. Conf. on Intelligent Robots and Systems*, pp. 5315–5320, Oct. 2009. 13

[76] S. Ahrens, D. Levine, G. Andrews, and J. P. How, "Vision-based guidance and control of a hovering vehicle in unknown, GPS-denied environments," in *IEEE Intl. Conf. on Robotics and Automation*, May 2009, pp. 2643–2648. 13, 51

[77] M. Blosch, S. Weiss, D. Scaramuzza, and R. Siegwart, "Vision based MAV navigation in unknown and unstructured environments," in *Proc. IEEE Int. Conf. on Robotics and Automation*, 2010, pp. 21–28. 13, 14, 15, 22, 51, 53, 77, 79

[78] F. Bourgeois, L. Kneip, S. Weiss, and R. Siegwart, "Delay and Dropout Tolerant State Estimation for MAVs," in *Proc. Intl. Symposium on Experimental Robotics*, Berlin, 2010, pp. 1–14. 13, 14, 15, 16, 51, 107

[79] R. Leishman, J. Macdonald, R. W. Beard, and T. McLain, "Relative navigation and control of a hexacopter," in *IEEE Intl. Conf. on Robotics and Automation*, St. Paul, MN, USA, May 2012, pp. 4937–4942. 13, 19, 51, 79, 84, 90

[80] S. Weiss, M. W. Achtelik, M. Chli, and R. Siegwart, "Versatile distributed pose estimation and sensor self-calibration for an autonomous MAV," in *IEEE Intl. Conf. on Robotics and Automation*, no. May, May 2012, pp. 31–38. 13, 14, 15

[81] L. Meier, P. Tanskanen, F. Fraundorfer, and M. Pollefeys, “PIXHAWK: A system for autonomous flight using onboard computer vision,” in *EEE Int. Conf. on Robotics and Automation*, May 2011, pp. 2992–2997. 13, 21, 24, 51

[82] D. Bohdanov and H. Liu, “Vision-based Quadrotor Micro-UAV Position and Yaw Estimation and Control,” in *Proc. AIAA Conf. on Guidance, Navigation, and Control*, 2012. 13, 51

[83] P. Henry, M. Krainin, E. Herbst, X. Ren, and D. Fox, “RGB-D Mapping: Using Depth Cameras for Dense 3D Modeling of Indoor Environments,” in *Int. Symposium on Experimental Robotics*, 2010. 13, 51

[84] M. Achtelik, M. Achtelik, S. Weiss, and R. Siegwart, “Onboard IMU and monocular vision based control for MAVs in unknown in- and outdoor environments,” *IEEE Intl. Conf. on Robotics and Automation*, pp. 3056–3063, May 2011. 14, 15, 107

[85] M. W. Achtelik, S. Lynen, S. Weiss, L. Kneip, M. Chli, and R. Siegwart, “Visual-inertial SLAM for a small helicopter in large outdoor environments,” *IEEE International Conference on Intelligent Robots and Systems*, pp. 2651–2652, Oct. 2012. 14, 15, 77, 78

[86] S. Roumeliotis and J. Burdick, “Stochastic cloning: a generalized framework for processing relative state measurements,” in *IEEE Intl. Conf. on Robotics and Automation*, vol. 2, no. May, 2002, pp. 1788–1795. 14, 16, 78

[87] K. Schmid, F. Ruess, M. Suppa, and D. Burschka, “State estimation for highly dynamic flying systems using key frame odometry with varying time delays,” in *IEEE Intl. Conf. on Intelligent Robots and Systems*, Oct. 2012, pp. 2997–3004. 14

[88] F. Fraundorfer, L. Heng, D. Honegger, G. H. Lee, P. Tanskanen, and M. Pollefeys, “Vision-Based Autonomous Mapping and Exploration Using a Quadrotor MAV,” in *IEEE Intl. Conf. on Intelligent Robots and Systems*, 2012. 15, 16, 52, 53, 77, 78, 79

[89] R. Voigt, J. Nikolic, C. Hurzeler, S. Weiss, L. Kneip, and R. Siegwart, “Robust embedded egomotion estimation,” in *IEEE Intl. Conf. on Intelligent Robots and Systems*, Sept. 2011, pp. 2694–2699. 15

[90] B. J. Kuipers, “Modeling human knowledge of routes: partial knowledge and individual variation,” in *Proc. of the National Conference on Artificial Intelligence*, Washington D.C., 1983. 16

[91] B. Kuipers and Y. Byun, “A robust, qualitative method for robot spatial learning,” *Proc. of the AAAI*, 1988. 16, 53, 80

[92] H. Choset and K. Nagatani, “Topological simultaneous localization and mapping (SLAM): toward exact localization without explicit localization,” *IEEE Trans. on Robotics and Automation*, vol. 17, no. 2, pp. 125–137, Apr. 2001. 16

[93] A. Martinelli, V. Nguyen, N. Tomatis, and R. Siegwart, “A relative map approach to SLAM based on shift and rotation invariants,” *Robotics and Autonomous Systems*, vol. 55, no. 1, pp. 50–61, Jan. 2007. 16

[94] R. C. Leishman, J. Macdonald, S. Quebe, J. Ferrin, T. W. Mclain, and R. W. Beard, "Utilizing an Improved Rotorcraft Dynamic Model in State Estimation," in *IEEE Intl. Conf. on Intelligent Robots and Systems*, San Fransisco, Sept. 2011. 19, 24

[95] R. C. Leishman, J. Macdonald, R. W. Beard, and T. W. McLain, "Quadrotors & Accelerometers State Estimation with an Improved Dynamic Model," *Control Systems Magazine (Submitted; preprint available from the authors)*, 2013. 19, 62, 78, 79, 87

[96] R. C. Leishman, T. W. McLain, and R. W. Beard, "Relative Navigation Approach for Vision-Based Aerial GPS-Denied Navigation," in *Intl. Conference on Unmanned Aircraft Systems (Accepted)*, Atlanta, GA, 2013. 19, 79, 82

[97] R. C. Leishman, D. Koch, and T. W. McLain, "Robust Motion Estimation Using an RBG-D Camera," in *AIAA Infotech @ Aerospace Conference (Submitted; preprint available from the authors)*, Boston, MA, USA, 2013. 19, 57, 90

[98] R. C. Leishman and T. W. McLain, "A Multiplicative Extended Kalman Filter for Relative Rotorcraft Navigation," *IEEE Trans. on Robotics (Submitted; preprint available from the authors)*, 2013. 19

[99] J. P. How, B. Bethke, A. Frank, D. Dale, and J. Vian, "Real-Time Indoor Autonomous Vehicle Test Environment," *IEEE Control Systems Magazine*, pp. 51–64, Apr. 2008. 22

[100] V. Kumar and N. Michael, "Opportunities and Challenges with Autonomous Micro Aerial Vehicles," in *Intl. Sympossium on Robotics Research*, 2011. 22

[101] M. Hehn and R. D. Andrea, "Quadrocopter Trajectory Generation and Control," in *International Federation of Automatic Control World Congress*, 2011. 22

[102] N. Guenard, T. Hamel, and R. Mahony, "A Practical Visual Servo Control for an Unmanned Aerial Vehicle," *IEEE Transactions on Robotics*, vol. 24, no. 2, pp. 331–340, Apr. 2008. 22

[103] P. Martin and E. Salaün, "Generalized multiplicative extended Kalman filter for aided attitude and heading reference system," in *AIAA Guidance, Navigation, and Control*, no. August, 2010. 23, 24, 29, 30, 37, 47, 84

[104] T. Madani and A. Benallegue, "Backstepping Control for a Quadrotor Helicopter," *IEEE Intl. Conf on Intelligent Robots and Systems*, pp. 3255–3260, Oct. 2006. 23

[105] R. Xu and U. Ozguner, "Sliding Mode Control of a Quadrotor Helicopter," *IEEE Conf. on Decision and Control*, pp. 4957–4962, 2006. 23

[106] T. Madani and A. Benallegue, "Control of a Quadrotor Mini-Helicopter via Full State Backstepping Technique," *Proc. IEEE Conf. on Decision and Control*, pp. 1515–1520, 2006. 23

[107] S. Bouabdallah, "Design and control of quadrotors with application to autonomous flying," Ph.D. dissertation, ETH Zurich, 2007. 23

[108] A. A. Mian and W. Daobo, “Nonlinear Flight Control Strategy for an Underactuated Quadrotor Aerial Robot,” in *IEEE Intl. Conf. on Networking, Sensing and Control*, 2008, pp. 938–942. 23

[109] J. Macdonald, R. C. Leishman, R. W. Beard, and T. W. McLain, “Analysis of an Improved IMU-Based Observer for Multirotor Helicopters,” *Journal of Intelligent & Robotic Systems (in review; preprint available from the authors)*, 2013. 24, 47, 62, 87

[110] C. Chaimberlain, “System Identification, State Estimation, and Control of Unmanned Aerial Robots,” Master’s Thesis, Brigham Young University, Provo, UT, Apr. 2011. 32

[111] M. Vidyasagar, *Nonlinear Systems Analysis.* Prentice-Hall, 1993. 38

[112] “Mikrokopter Home Page.” [Online]. Available: http://www.mikrokopter.de 42

[113] “Motion Analysis Home Page.” [Online]. Available: http://www.motionanalysis.com/ 42

[114] R. Mahony, T. Hamel, and J.-M. Pflimlin, “Nonlinear Complementary Filters on the Special Orthogonal Group,” *IEEE Transactions on Automatic Control*, vol. 53, no. 5, pp. 1203–1218, 2008. 43

[115] T. D. Edward Rosten, “Machine learning for high-speed corner detection,” in *European Conference on Computer Vision*, 2006, pp. 430—-443. 58

[116] M. Calonder, V. Lepetit, C. Strecha, and P. Fua, “BRIEF: Binary Robust Independent Elementary Features,” in *Computer Vision ECCV 2010*, ser. Lecture Notes in Computer Science, K. Daniilidis, P. Maragos, and N. Paragios, Eds. Springer Berlin Heidelberg, 2010, vol. 6314, pp. 778–792. 58

[117] M. A. Fischler and R. C. Bolles, “Random sample consensus: a paradigm for model fitting with applications to image analysis and automated cartography,” *Communications of the ACM*, vol. 24, no. 6, pp. 381–395, 1981. 58

[118] K. S. Arun, T. S. Huang, and S. D. Blostein, “Least-squares fitting of two 3-D point sets,” *IEEE Trans. on Pattern Analysis and Machine Intelligence*, vol. 9, no. 5, pp. 698–700, May 1987. 58

[119] J. Kuipers, *Quaternions and Rotation Sequences.* Princeton University Press, 1999. 64, 84

[120] E. Marder-Eppstein, “costmap_2d,” 2013. [Online]. Available: http://www.ros.org/wiki/costmap_2d 64

[121] K. Konolige and E. Marder-Eppstein, “navfn,” 2013. [Online]. Available: www.ros.org/wiki/navfn 64

[122] J. Ferrin, R. Leishman, R. Beard, and T. McLain, “Differential Flatness Based Control of a Rotorcraft For Aggressive Maneuvers,” in *IEEE Int. Conf. Intelligent Robots and Systems*, 2011. 64, 105

[123] R. W. Beard and T. W. McLain, *Small Unmanned Aircraft.* Princeton University Press, 2012. 64, 84

[124] J. Macdonald, “Efficient Estimation for Autonomous Multi-Rotor Helicopters Operating in Unknown, Indoor Environments,” Ph.D. dissertation, Brigham Young University, Nov. 2012. 67, 115, 116

[125] J. L. Crassidis, F. L. Markley, and Y. Cheng, “Survey of Nonlinear Attitude Estimation Methods,” *Journal of Guidance, Control, and Dynamics*, vol. 30, no. 1, pp. 12–28, Jan. 2007. 84

[126] N. Trawny and S. I. Roumeliotis, “Indirect Kalman Filter for 3D Attitude Estimation A Tutorial for Quaternion Algebra,” Multiple Autonomous Robotic Systems Lab, Tech. Rep. 612, 2005. [Online]. Available: http://www.cs.umn.edu/~trawny 84, 88, 96

[127] A. Bachrach, “Autonomous Flight in Unstructured and Unknown Indoor Environments,” Ph.D. dissertation, MIT, 2009. 107

[128] T. W. McLain, R. W. Beard, D. B. Barber, and N. B. Knoebel, “An Overview of MAV Research at Brigham Young University,” in *Proc. of the NATO Research and Technology Organization Symposium on Platform Innovations and System Integration for Air, Land, and Sea Vehicles*, Florence, Italy, 2007. 116